Data-Driven Recommender Systems

Jérémie Mary

Mémoire présenté pour obtenir

le diplôme d'Habilitation à Diriger des Recherches

de l'Université de Lille

Spécialité : Informatique

par

Jérémie MARY

Data-Driven Recommender Systems

Sequences of recommendations

Directeur de Recherches : Philippe Preux - PR Univ. Lille

Soutenu le 24 Novembre 2015 devant le jury composé de :

Jean-Yves	Audibert	Capital Fund Management	**Examinateur**
Francis	Bach	DR Inria	**Rapporteur**
Patrick	Gallinari	PRU Paris vi	**Rapporteur**
Rémi	Gilleron	PRU Lille	**Examinateur**
Philippe	Preux	PRU Lille	**Garant**
Louis	Wehenkel	PRU Liège	**Rapporteur**

Contents

Remerciements

Les travaux de recherche présentés dans ce mémoire ont été réalisés à l'Université de Lille et dans le centre de l'Inria Lille - Nord Europe. Ils sont le fruit de collaborations nombreuses et d'un terreau propice. Je tiens donc à remercier l'ensemble des personnes qui ont permis le développement de ma réflexion.

Je voudrais remercier tout particulièrement Philippe Preux pour m'avoir toujours encouragé et soutenu dans ma progression tout au long de ces dernières années ainsi que pour avoir été le garant de cette Habilitation.

Rémi Gilleron m'a fait l'honneur et le plaisir de présider le jury ; je lui suis reconnaissant de l'intérêt qu'il a manifesté pour mes travaux. Le machine learning à Lille lui doit beaucoup.

J'adresse également de vifs remerciements à Francis Bach, Patrick Gallinari et Louis Wehenkel pour avoir accepté de relire ce mémoire et d'en être rapporteurs, ainsi qu'à Jean-Yves Audibert pour avoir accepté de faire parti du jury.

Depuis le début de mes travaux de recherches, j'ai eu la chance d'avoir de multiples collaborations avec des chercheurs d'un large horizon, du plus fondamental au plus appliqué ainsi qu'avec des partenaires industriels. Ces échanges m'ont considérablement enrichi ; j'en remercie tous les protagonistes. J'ai évidemment une pensée particulière pour tous les membres passés et présents de l'équipe SequeL. Ce fût un honneur que de participer à ce projet.

J'ai eu par ailleurs l'occasion de participer activement à l'encadrement de doctorants, ce qui est un élément particulièrement satisfaisant de la vie de chercheur et m'a énormément apporté. Merci donc à Olivier Nicol, Frédéric Guigou, Florian Strub, Romain Warlop ainsi qu'à leurs co-encadrants.

Merci également à mes collègues et amis du Grappa, d'Inria, du CNRS, de l'université de Lille pour les nombreuses discussion professionnelles ou amicales que nous avons pu avoir. Une pensée particulière pour l'efficacité des secrétaires et personnels de support -assistantes de projet au premier rang- qui nous facilitent considérablement la tâche.

Évidemment le soutien de mes proches a été sans failles et je rend donc hommage à Céline, Clémence, Constantin et mes parents pour m'avoir offert -à la maison- des conditions propices à la conduite de ces travaux ; en particulier l'été de la soutenance.

Enfin puisque les grandes décisions sont souvent liées à des rencontres, je tiens à profondément remercier celui qui m'a présenté au monde le la recherche et m'a convaincu de suivre le chemin des études doctorales : Éric Cancès.

Last but not least, thanks for reading.

Part I

Overview

Introduction

All models are wrong, but some are useful. George Box.

This document presents the research I have undertaken since the end of my PhD thesis. This work was done as assistant professor at the University of Lille and the Inria Lille Nord Europe research center. Most of it was done in the SequeL team which was created in 2006 both in the Laboratoire d'Informatique Fondamentale de Lille (LIFL) which is now know as CRIStAL and in the Inria Lille under the direction of Philippe Preux.

My main research directions are statistical learning and machine learning with the goal of sequential prediction or decision making under uncertainty. This field is at the intersection of statistics, computer science, applied mathematics and signal processing. In my opinion its main objective is to build tools able to face situations where vanilla statistics does not apply well because of some constraints carried by the application. As an example this can be due to the lack of a good model, highly non linear data, size of the data, non independent sampling. With the fast decay of the storage cost and the exponential grow of the number of sensors -including smartphones- some tremendous amounts of data are available to build some new applications. As the growth rate of available data -including some public one with the open access initiative- is higher than the growth rate of the number of brains available to take care of it, there is a strong industrial interest in automated methods with good performances. We also have to keep in mind that the data size has grown faster than the speed of processors and that an important part of the complete system performance is directly related to the ability to handle all the available data. Thus in some situations, even a quadratic complexity in the size of the data is too much which strongly advocates for algorithms that scale at most linearly and are anytime.

Even if in some games and some dedicated applications machines, became better than humans, as soon as the model is not fully available or reasonably sized the human is still undefeated. Nevertheless machine learning can be used to handle efficiently at low cost some large scale tasks such as recommender systems, Ads personalization, time series predictions, planning, image annotation and segmentation.

Many tools and techniques of machine learning are now deployed at large scale. Nevertheless the techniques still require some deep understanding to be applied to a new domain. My PhD thesis was mostly oriented towards a theoretical analysis of some classes of algorithms, in these years I tried to pay much more attention to the final application and to help to bridge the gap between the practice and theory. I always tried to do it in a principled way and I feel that

grounding the studied system with a real problem is a key ingredient of the Machine Learning as a field.

The most common formulation in statistics is the following: we assume a set of n (observation, label) tuples, after these training examples a new observation comes and we want to be able to predict the label. This setting can nicely fit a wide variety of problems as shown in these few examples:

- Observations can be explicit ratings given to some movies, music or items and the label would be the next item to be interested in.
- Observation can be some network traffic and label would be an expert labeling about the detection of an fraudulent behavior.
- Observation can be emails or forum posts and the label can be the sentiment of the user or a spam tag.
- Observation can be visited webpages or Facebook like or friends and label would be probability of credit default or more controversially political orientations and customer value.
- ...

The link is usually discovered by the minimization of a well chosen regularized loss function. [Bottou, 2012] provides some common losses such as quadratic, Hinge, Quantile, Logistic with their common uses. In machine learning it is common to deal with a large number of features or discrete values. This requires to estimate a large number of parameters which is prone to overfitting. To avoid this we often penalize -aka regularize- the loss function with a term which constraints the model. Common regularizers are norms -$L_2, L_1, Frobenius, \ldots$- on the parameters of the model. Turning a problem into the minimization of a well chosen and understood function is the main trick of the Machine Learning.

The rest of this document is organized as follow: first I'll highlight the very core of the recommender systems. This is not designed to be an exhaustive recall of all the existing work around this, but rather what I feel to be the most central after years of industrial tests and challenges over real data. Then I'll present one of the first work we had on recommender systems which had the particularity to work under budget/time constraints. I feel like it was a step towards the complete description on a recommender system which is not only about guessing the tastes of customers but also on how to make an efficient use of this guess. Secondly I'll move to my contributions to one of the most important problems for these systems which is the offline evaluation of new policies. This is central because being able to estimate with guarantees enables confident optimization, and we will see that even for contextual bandits some difficulties occurs. Then I'll present some of our work on the clustering problem in the case of sequences. This part is much more theoretical than the rest of the document, but as the number of available unsupervised data grows incredibly fast over the internet, we think there is a practical need to shift from supervised settings to unsupervised learning and this will require some grounding. Finally, I'll conclude this document by some thoughts on the future of the developed questions and select some publications to provide a supplementary material.

Whenever possible I'll mention the industrial problem which motivated this study and the contacts I developed working on it. I omitted my work on games -more specifically poker- and in bioinformatics because this does not fit well in the main line of this document.

Chapter 1

Large scale recommendation algorithms

The work described in this chapter is based on these publications and on the PhD of Olivier Nicol [Nicol, 2014]:

- O. Nicol, J. Mary & Ph. Preux *ICML Exploration ans Exploitation challenge: Keep it Simple!* Journal of Machine Learning Research (JMLR), 2012

- S. Girgin, J. Mary, Ph. Preux & O. Nicol *Managing Advertising Campaigns - an Approximate Planning approach* Frontiers of Computer Science, 2011

- S. Girgin, J. Mary, Ph. Preux & O. Nicol *Advertising Campaigns Management: Should We Be Greedy?* The 10th IEEE International Conference on Data Mining (ICDM), 2010

- F. Guillou, G. Romaric, J. Mary and Ph. Preux *User Engagement as Evaluation: a Ranking or a Regression Problem?*, RecSys'14 Challenge Workshop.

- S. Girgin, J. Mary, Ph. Preux & O. Nicol *Large-Scale Online Learning and Decision-Making Workshop, Windsor, 2012 Planning-based Approach for Optimizing the Display of Online Advertising Campaigns* MLOAD 2010 - NIPS 2010 Workshop on online advertising, 2010

Recommender Systems (RSs) collect information on the preferences of its users for a set of items (e.g., movies, courses, songs, books, jokes, items, applications,...). They target to answers questions that are hard to formulate or not formulated questions such as *Propose me something interesting for this afternoon, or What are good birthday gift ideas for my mom?* .

Facing such questions, the idea of Collaborative Filtering (CF) is to mimic the way in which humans build their own decisions: besides our own experiences, we also integrate the experience and knowledge that reach each of us from a group of relatives.

1.1 Core algorithms

There is a huge amount of work in recommender systems and my goal is not here to provide a complete survey which is already done in [Bobadilla et al., 2013]. In the following I am going to emphasize two flavors of well known algorithms - namely the logistic regression and the matrix factorization - which received an intense attention from the community and are the core of many deployed systems. I'll try to present some knowledge developed by taking part into challenges before introducing my actual interest which is recommendations performed in a sequential and possibly evolving environment.

1.1.1 Logistic Regression

The logistic regression is a linear regression model which tries to fit the probability for an observation to belong to one class. A reformulation as a L_2 regularized loss to minimize can be written:

$$\mathscr{L}(w) = \frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^{n} \log\left(1 + e^{-y_i \cdot w^T x_i}\right)$$

Where $(x_i, y_i), i \in 1 \dots n$ a set of n examples (description, class) with the description in dimension d (in order to add an intercept we often append a 1 in front of all the description vectors). w is the vector of parameters to optimize (dimension of w is the same than the description space) and $\lambda \in \mathbb{R}$. The minimization is done by an iterative algorithm: gradient descent, Newton's method, stochastic gradient,...

Remark 1. *For more complex objective functions the gradient strategies are still very effective and one can cite in particular Nesterov accelerated gradient [Nesterov, 2004] which by a clever "sliding" is an optimal method (in terms of oracle complexity) for smooth convex optimization and Frank-Wolfe methods which considers a linear approximation of the objective function, and moves slightly towards a minimizer of this linear function [Frank and Wolfe, 1956].*

At first glance working with a large number of features can be a problem. Besides the problems of overfitting, it is intractable to conduct a systematically search through a high-dimensional space, but this is not only a curse:

Linear separation

For a fixed number of independent observations sampling the whole space while the dimensionality of the data increase the probability of finding a linear separator -for any affectation of classes- increases. This is a remarkable advocate for the use of linear methods as the logistic regression when the dimensionality is large. In a noisy setting adding some new dimensions -with non redundant information- helps because the distance between two different centroids is increasing as a square root of the dimensionality.

It can even be interesting to build new features from the data. This direction has been intensively studied through the use of kernels [Schölkopf and Smola, 2002]. As a bonus the

kernel trick allows to turn any linear machine learning algorithm based which only involves scalars products computations into a non linear method (in the initial space), by providing efficient method to minimize the objective function over the higher dimensional space. This trick is extensively used in SVMs. Though these methods are very useful their scaling is problematic even if there exist some linear approximations to the Gaussian kernel as done in [Jose et al., 2013].

Fast approximated linear algebra

It is surprisingly easy to perform some approximated linear algebra computations in high dimension. One example of this is the squared-length sampling technique which consists in subsampling the rows and/or columns of a matrix according to the square of their L_2 norm. An excellent version is given Chapter 6 of [Kannan and Vempala, 2009] but the first result is in [Frieze et al., 1998].

Let $p_1, p_2, \ldots, p_d$ be nonnegative reals adding up to 1. Pick $j \in \{1 \ldots n\}$ with probability p_j.

For any vector $\vec{v}$, consider the vector valued random variable

$$X = \frac{M^{(j)}\vec{v}_j}{p_j}$$

Then $\mathbb{E}(X) = M\vec{v}$. So X is an unbiased estimator of $M\vec{v}$ and

$$Var(X) = \sum_{j=1}^{n} \frac{\|M^{(j)}\|^2 \vec{v}_j^2}{p_j} - \|M\vec{v}\|^2$$

So if $p_j = \frac{\|M^{(j)}\|_2^2}{\|M\|_F}$, after s samples if we note Y the random variable which averages the realizations, then $\mathbb{E}(Y)$ is an estimator of $M\vec{v}$ with

$$Var(Y) = \frac{1}{s}\|M\|_F^2\|v\|^2$$

Let $j_1, \ldots, j_s$ be s independant identitcally distributed (i.i.d.) random variables taking values in $\{1 \ldots n\}$ such as the probability that j_1 is equal to $i \in [n]$ is proportional to $\|M^{(i)}\|_2^2$.

Let B a matrix such that its i^{th} column is $\frac{1}{\sqrt{sp_{j_i}}} M_{j_i}$. Let $u_1, \ldots, u_k$ be the k top left singular vectors of B. A low rank approximation to M is

$$\tilde{M} = \sum_{i=1}^{k} u_i u_i^\top M.$$

One can prove that for $s = 4k/\varepsilon^2$ this approximation satisfies:

$$\mathbb{E}\|M - \tilde{M}\|_F^2 \leq \|M - M^*\|_F^2 + \varepsilon\|M\|_F^2$$

where M^* is the best rank k approximation to M.

The most surprising part of this algorithm is that the complexity of computing the projection matrix $\sum_{i=1}^{k} u_i u_i^\top$ is independent of n.

On the regularization

The goal of a regularizer is to promote some simplification into the choice of the model. This is related to the Occam's razzor which is a principle that promotes simplicity. Here a natural choice would be to add a penalty to our loss that would be the number of non zeros parameters in the model. This penalty is often referred as L_0 despite this is not a norm. This is called sparsity promotion.

Unfortunately L_0 is not convex and this leads to computationally intractable optimization - which is NP-Hard even in the noiseless setting.

In such situation a natural idea is to relax. The convex norm closest to L_0 is L_1 which allows to optimize efficiently with guarantees using linear gradient methods. This replacement can be justified in the noiseless setting, because if coefficients of X are i.i.d subgaussian entries the probability of exact recovery of the coefficients replacing L_0 by L_1 is $\Omega(s \log \frac{d}{s})$ [Candes and Tao, 2005].

In application, this is common to test a set of regularizers including L_2 norms in order to minimize an estimator of the loss built by cross validation or bootstrap.

Heterogenous Data handling

Probability prediction is one of the most important problems in ML. When trying to apply these methods on real world datasets, we often face some heterogenous data mixing numerical and categorial features. There has been many attempts to override this point including the tedious handcrafting of features.

Nevertheless a strategy succeeded in several Kaggle/KDD challenges [all, 2014, 2012], even outperforming random forests and in my opinion should now be used as a baseline for any probability prediction task:

- Split any categorical feature with k different values in k boolean features,
- Discretize any numerical features individually using your favorite clustering method. Quantiles, k-means or latent Dirichlet allocation are usually good choices [Boullé, 2006].
- (Optional) Add some other binary features built upon some gradient booster decision trees GBDT [Friedman, 2000]. This step sometimes does not improve a lot the final performance but can be crucial to win a competition such as the Yandex one [Serdyukov et al., 2014] and the RecSys 2014 one [Said et al., 2014b].
- Now perform a variant of a regularized logistic regression trained by gradient descent on the binary data. The missing values are replaced by 0.

From a practical point of view there are several remarks on the last step of this approach.

- This is useless to do a gradient descent if you can afford the cost of the quadractic solving but this methods is designed to work with large volumes of data. As with deep learning this is common to start with some stochastic gradient descent and then after some passes to switch to more sophisticated methods as Adagrad [Duchi et al., 2010] - see [Bottou, 2012] for a lot of clever considerations on SGD.

- It can be useful to perform a second order -or higher- logistic regression. This leads to a great increase in the number of parameters to estimate. To avoid this problem an interesting regularization is to push towards a low rank constraint on the matrix of second order terms. This is the core of the idea behind factorizing machines [Rendle, 2012] which is used in many state of the art softwares for large data as libFM and Vowpal Wabbit (lqr option).

Implementation details

The gradient descent described above only updates parameters associated to features with a value set to 1. Because of the binarization, for each example the number of features with value 1 is usually low with respect to the total number of features. Thus it is very efficient to code the model using a hash tables to store the value of the parameters using the names of the features to generate a key. If you are running in trouble with the memory because of the number of possible features a common practice is to use the hashing trick [Weinberger et al., 2009] which can be interpreted as a regularization and simply consists in allowing collisions.

It is possible to keep several low rank matrixes instead of one by affecting a "field" to each feature. Then a low rank matrix is estimated for each couple of field of features. This additional step can cause severe overfitting but it is useful to reach the very top of the performance. It is also recommended to take advantage of SSE operations to speed up the computation of scalar products. A BSD implementation of this strategy is available in the LibFFM library.

An other very popular approach is the random forest but their classical implementation does not scale very well on online data. A possibility here is to use extremely randomized forest [Geurts et al., 2006] which are very interesting to handle heterogenous data. Nevertheless we never succeeded in recommender challenge using these methods though they are competitive.

1.1.2 Matrix Factorization

Logistic regression and more generally supervised learning require some descriptors of the situation to predict, also known as features. As an example it is as answering the question: *What is the probability that a male of 17 years who owns an iPhone is going to buy this product with tags A,B,C?*

But sometimes we do not have these descriptors of the context. As an example in the Netflix dataset we only have some ratings of movies - with a date - for some users and movies. A natural idea to predict which movies could be interesting for a user is to evaluate this probability using the movies he already rated as a context. Though effective, this approach is outperformed by the matrix factorization formulation when the available data is sparse.

First the known ratings are presented as a matrix $\mathbf{M}$ with one row per user and one column by movie (also called item of size $m \times n$. Of course this matrix has a lot of unknown values and our objective is to fill these ones.

A classic -though not perfect- measure of performance is to evaluate the L_2 norm also known as Mean Square Error (MSE) of the reconstructed matrix using some cross validation on the known terms. In other words we want to minimize the:

$$MSE(\hat{\mathbf{M}}) = \frac{1}{nm} \sum_{i,j \in \mathbf{M}} (m_{i,j} - \hat{m}_{i,j})^2$$

If we add the constraint on $\hat{\mathbf{M}}$ to be of rank k then the optimal solution is the singular value decomposition of $\mathbf{M}$ truncated to the k highest singular values.

Of course in practice we are going to perform this minimization using only the known values of $\mathbf{M}$ so we will need some additional regularization. A natural class of estimator for $\hat{m}_{i,j}$ is to express it as a scalar product $< \mathbf{u}_i, \mathbf{v}_j >$ from points of a latent space of dimension k.

Thus for a fixed k which is a parameter of the approach we aim to minimize:

$$\zeta(\mathbf{U}, \mathbf{V}) \stackrel{def}{=} \sum_{i,j \in \mathbf{M}} \left(m_{i,j} - \mathbf{U}_i, \mathbf{V}_j^t\right)^2 + \Omega(U, V) \tag{1.1}$$

in which $\lambda \in \mathbb{R}^+$ and the usual regularization term is:

$$\Omega(\mathbf{U}, \mathbf{V}) \stackrel{def}{=} ||\mathbf{U}||^2 + ||\mathbf{V}||^2 = \sum_i ||\mathbf{U}_i||^2 + \sum_j ||\mathbf{V}_j||^2.$$

ζ is not convex. The minimization is usually performed either by stochastic gradient descent (SGD), or by alternate least squares (ALS). ALS-WR [Zhou et al., 2008] a Tikhonov regularization which weights users and items according to their respective importance in the matrix of ratings. This is the default setting for Mahout[1] and is very hard to defeat consistently even with some more recent approaches as PMF [Salakhutdinov and Mnih, 2007].

$$\Omega(\mathbf{U}, \mathbf{V}) \stackrel{def}{=} \sum_i \#\mathcal{J}(i)||\mathbf{U}_i||^2 + \sum_j \#\mathcal{I}(j)||\mathbf{V}_j||^2.$$

Guaranties

This formulation as a recovery of a low rank matrix from a fraction of observed values has attracted the attention of mathematicians, [Fazel, 2002] is possibly the first work in this field.

A key idea [Cai et al., 2010] is the minimization of a convex surrogate of the rank under constraints on the L_2 norm of the reconstruction which is a convex optimization problem tractable by standard algorithms. This method has the advantage of exactly, rather than approximately, recovering the entries of the matrix when a suitable low rank assumption is satisfied. This path leads to the SOFT-IMPUTE algorithm [Mazumder et al., 2010] which iteratively replace

[1] https://mahout.apache.org/

the missing elements with those obtained from a soft-thresholded SVD. The classically studied quantity is the Lagrange form of the minimization of the nuclear norm -as a convex surrogate of the rank- of $\hat{\mathbf{M}}$ under constraint of L_2. This is equivalent to minimize equation 1.1 with

$$\Omega(\mathbf{U}, \mathbf{V}) \stackrel{def}{=} \lambda \cdot \|UV\|_*.$$

where $\|\mathbf{M}\|_*$ is the nuclear norm, that is the sum of the singular values of $\mathbf{M}$ and $\lambda \geq 0$ a regularization parameter.

This estimator is asymptotically consistent with a rate of convergence of in O(1/numbers of iterations), though there is no guarantee for a fixed number of known ratings.

The minimization of the sole nuclear norm allows to get some interesting properties. If the rank of $\mathbf{M}$ of dimension $n_1 \times n_2$ is k and has a "strong incoherence property" then with high probability nuclear norm minimization will recover $\mathbf{M}$ from a random uniform sample provided that the number of known rating of $m \gg nk \log^{O(1)}(n)$, where $n = \max(n_1, n_2)$ [Candès and Tao, 2010]. The strong incoherence property is a condition on the matrix coefficients of U and V basically saying that they behave in magnitude as if the singular vectors were distributed randomly. [Candès and Recht, 2009] had a different incoherence property - ℓ^∞ norm of singular vector should be close to the minimal possible value, namely $O(1/\sqrt{n})$- and they obtain a $n^{1.2}k \log(n)$ number of observations requirement.

Nevertheless these methods are not competitive on real data with state of the art matrix factorization: as an example on Netflix the RMSE of these methods is 0.94 while an ALS-WR reaches 0.86 and runs much faster: less than 20 seconds on an Intel i7 using with 8 cores against several hours.

The main limit to these works for RS comes from the uniform and independent sampling hypothesis. This is possible to get rid of the uniform assumption at a cost of increased variance over the reconstruction but the goal of a RS is to not sample independently from the previous recommendation: we want to take into account the preferences of the user which are expressed through its own ratings.

Remark 2. *In order to keep the temporal information it is also possible to work with a tensor instead of a matrix, see [Rendle et al., 2010] for more details.*

Remark 3. *A factorization machine can mimic the behavior of a matrix factorization by using the right input data (e.g. feature vector) [Rendle, 2010].*

Remark 4. *One should pay more attention to the fact that the sampling of known values in* $\mathbf{M}$ *is not uniform. This impact the evaluation of the* L_2 *norm of the error but this effect is usually neglected.*

1.1.3 Bandits algorithms

Let us consider a bandit machine with m independent arms. When pulling arm j, the player receives a reward drawn from $[0, 1]$ which follows a probability distribution ν_j. Let μ_j denote the mean of ν_j and $j^* \stackrel{def}{=} \text{argmax}_j \, \mu_j$ be the best arm. The parameters $\{\nu_j\}$, $\{\mu_j\}$, j^* and μ^* are unknown.

A player aims at maximizing its cumulative reward after T consecutive pulls. More specifically, by denoting j_t the arm pulled at time t and r_t the reward obtained at time t, the player wants to maximize the quantity $\text{CumRew}_T = \sum_{t=1}^{T} r_t$. As the parameters are unknown, at each time-step (except the last one), the player faces the dilemma:

- either *exploit* by pulling the arm which seems the best according to the estimated values of the parameters;
- or *explore* to improve the estimation of the parameters of the probability distribution of an arm by pulling it;

A well-known approach to handle the exploration vs. exploitation trade-off is the Upper Confidence Bound strategy (UCB) [Auer et al., 2002] which consists in playing the arm j_t:

$$j_t = \underset{j}{\text{argmax}}\, \hat{\mu}_j + \sqrt{\frac{2 \ln t}{t_j}}, \tag{1.2}$$

where $\hat{\mu}_j$ denotes an estimation of μ_j at time-step t and t_j corresponds to the number of pulls of arm j since $t = 1$. UCB is optimal up to a constant. This equation clearly expresses the exploration-exploitation trade-off: while the first term of the sum ($\hat{\mu}_j$) tends to exploit the seemingly optimal arm, the second term of the sum tends to explore less pulled arms.

Work by [Li et al., 2010] extend the bandit setting to contextual arms. They assume that a context is associated to each arm j in the form of a vector of real features $\mathbf{v}_j \in \mathbb{R}^k$ and that the expectation of the reward associated to an arm is $\mathbf{u}^* \cdot \mathbf{v}_j$, where $\mathbf{u}^*$ is an unknown vector. The algorithm handling this setting is known as LinUCB. LinUCB follows the same scheme as UCB in the sense that it consists in playing the arm with the largest upper confidence bound on the expected reward:

$$j_t = \underset{j}{\text{argmax}}\, \hat{\mathbf{u}} \cdot \mathbf{v}_j^T + \alpha \sqrt{\mathbf{v}_j \mathbf{A}^{-1} \mathbf{v}_j^T},$$

where $\hat{\mathbf{u}}$ is an estimate of $\mathbf{u}^*$, α is a parameter and $\mathbf{A} = \sum_{t'=1}^{t-1} \mathbf{v}_{j_{t'}} . \mathbf{v}_{j_{t'}}^T + \mathbf{Id}$, where $\mathbf{Id}$ is the identity matrix. Note that $\hat{\mathbf{u}} . \mathbf{v}_j^T$ corresponds to an estimate of the expected reward, while $\sqrt{\mathbf{v}_j \mathbf{A}^{-1} \mathbf{v}_j^T}$ is an optimistic correction of that estimate.

While the objective of UCB and LinUCB is to maximize the cumulative reward, theoretical results [Li et al., 2010; Abbasi-yadkori et al., 2011] are expressed in term of *cumulative regret* (or regret for short)

$$\text{Regret}_T \overset{def}{=} \sum_{t=1}^{T} r_t^* - r_t,$$

where r_t^* stands for the best expected reward at time t (either μ^* in the UCB setting or $\max_j \mathbf{u}^* . \mathbf{v}_j^T$ in the LinUCB setting). Hence, the regret measures how much the player looses (in expectation), in comparison to playing the optimal strategy. Standard results prove regrets of order $\tilde{O}(\sqrt{T})$ or $O(\ln T)$, depending on the assumptions on the distributions and depending on the precise analysis [2].

[2] $\tilde{O}$ means O up to a logarithmic term on T.

During preliminary time-steps, the value of the eq. (1.2) is dominated by the second term. Afterwards, the first term becomes predominant, and the one that influences the most the choice of the arm to pull. Hence, the behavior of UCB is very similar to the strategy `Explore-Exploit` which consists in uniformly exploring the arms during a few steps, then focussing on the arm with the best empirical mean. On the other hand, UCB never stops exploring seemingly sub-optimal arms. This continuous exploration is usually confined to ε-greedy strategy which plays the seemingly-best arm with probability $(1-\varepsilon)$ and explores otherwise.

The strength of UCB lies in the following property: the number of pulls of a sub-optimal arm j is in the order $\frac{\ln T}{(\mu^*-\mu_j)^2}$. Hence, the continuous exploration of arm j only costs a regret of the order $\frac{\ln T}{\mu^*-\mu_j}$. As a corollary to that property, the exploration budget is non-uniformly spread among the set of arms: the larger the regret of arm j, the less frequently arm j is played. UCB does not lose time, hence rewards, playing arms which are likely to be non-optimal.

In short, while (i) UCB-like algorithms continuously explore to avoid focussing on a sub-optimal arm, (ii) the number of exploration steps is kept reasonable, and (iii) the exploration budget is non-uniformly shared among the set of arms (most promising arms are more explored). This (automatically) finely tuned exploration grants UCB algorithm with an optimal regret (up to a constant), while `Explore-Exploit` suffers a regret of $O\left(T^{2/3}\right)$ and ε-greedy needs the knowledge of $\min_{j\neq j^*}(\mu^*-\mu_j)$ to reach a $O(\ln T)$ regret [Auer et al., 2002].

This strategy can be extended to the choice of arm with respect to a context described by some features. The intuition of the extension is illustrated in figure 1.1 when the reward depends linearly on features -this is known as LinUCB. Some work extends the methodology using a kernel [Valko et al., 2013]. Of course LinUCB, and more generally contextual bandits requires the context (values of features) to be provided. In real applications this is done using side information over the items and the users [Shivaswamy and Joachims, 2011] -*i.e.* expert knowledge, categorization of items, Facebook profiles of users, implicit feedback ...

As highlighted by [Chapelle and Li, 2011] in some applications, an old Bayesian version of this strategy performs extremely well: the Thompson sampling. The principle is to keep a distribution law for each parameter and to update it with respect to the new observations. When these parameters are required to take an action, they are sampled from their estimated distribution rather than picked at the maximum of likelihood. This ensures a level of exploration because parameters which have been rarely updated will have higher variance. This idea was initially an heuristic, but there is now some non asymptotic guarantees [May et al., 2012]. Sadly the proof does not explain the reasons of the fast convergence because it analyses the regret in a very similar way to UCB.

Of course, it is possible to include some knowledge in the prior distribution of parameter or in the update rule. The Xbox live matchmaking system -TrueSkill [Herbrich et al., 2007]- and AdPredictor from Microsoft [Graepel et al., 2010] are using these strategies.

1.1.4 Common limits and improvement margin

Yet these methods are not totally new, they needed some time to be widely understood and used in real systems. Moreover to build an actual system we often need to pay some attention

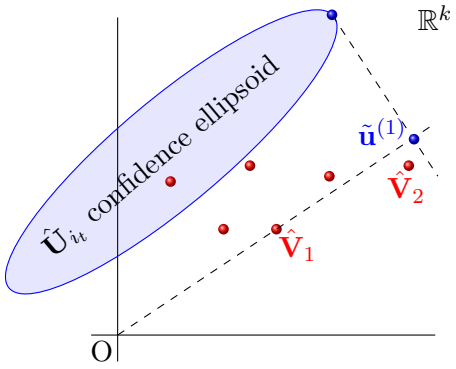


Figure 1.1: This figure illustrates the use of the upper confidence ellipsoid for item selection in the context of a new user. As explained in the text, items and users are represented as vectors in $\mathbb{R}^k$. In the figure, the red dots correspond to the known items vectors. The blue area indicates the confidence ellipsoid on the unknown vector associated to the user. The optimistic rating of the user for item 1 is the maximum scalar product between $\hat{\mathbf{V}}_1$ and any point in this ellipsoid. By a simple geometrical argument based on iso-contours of the scalar product, this maximum scalar product is equal to the scalar product between $\hat{\mathbf{V}}_1$ and $\tilde{\mathbf{u}}^{(1)}$. The optimistic recommendation system recommends the item maximizing the scalar product $\langle \tilde{\mathbf{u}}^{(j)}, \hat{\mathbf{V}}_j \rangle$.

to some important side effects. I mention here some of the questions we tried to contribute:

Hybrid systems Most of the time there are different ways to attack a problem, and as for challenges, one does not rely just on one tool, but on a wide variety of methods as well, combined together in some empirically optimized fashion. This was the case in our winning submission to RecSys'14 as well as in the work with Deezer - more info in section 1.2.

Coldstart Most of the time we have no data is some regions of the space. With RS it corresponds to new users and new items. Unfortunately they are possibly the most important ones because you want to fidelize your new users and when you accept a new item this is because you feel it's a better one. One could rely on the hybrid approach but we believe in optimization of the recommender policy to embed some exploration/exploitation dilemma. More information in sections 1.2 and 1.3.

Evaluations of systems You can always compute the classification error on model predicting the probability of a click. But is this really related to the online performance of a new model? The real world evolves and can even be non stationary. In this setting, offline evaluation of algorithms is problematic. Developed in section 2.

Unsupervised learning for sequences More and more data is available but labeling is not always possible or too expensive. This field is close to pattern identification and is at the limit of non parametric statistics. Our main contribution was to provide some theoretical ground to the clustering of processes. See section 3.

1.2 Challenges and collaborations

In my opinion a core difference between statistics and machine learning is to put more emphasis on some task from real world. This means to take some care to design some algorithms able to be computed in reasonable time with large amounts of data, but it also means to attack some usage cases in a setting transferable to the industrial level. I feel that challenges organized in conjunction with industrial parters cast an interesting light on the development of machine learning. I took part and I was involved in the organization of 3 of them.

1.2.1 ICML'12

This competition was the second Exploration vs. Exploitation challenge organized by the University College of London jointly with ICML 2011. The goal was to learn online a link between the description of some users and a recommendation taken by Adobe. To avoid to perform the evaluation online a task was designed as follow:

1. Six tuples (values of features, taken action) where presented to the algorithm.
2. The algorithm selects one
3. Result of the action aka reward (0 or 1) is revealed,
4. Loop to 1 until no more available data. The score is the sum of the reward.

To win this challenge we modified a state-of-the-art algorithm: Ad Predictor which we also provided with additional possibilities. This algorithm makes assumptions that are quite similar to the ones made by LinUCB. LinUCB is possibly the most famous contextual bandit algorithm but as the linear assumption does not work extremely well there we use some versions made to take care of discrete features (although we show how to relax this constraint). After deriving a new algorithm, LAP, able to deal with the classical contextual bandit problem, we show that this new algorithm exhibits a performance that is comparable with that of the state-of-the-art solution LinUCB while being much more computationally efficient: linear *versus* cubic time complexity. We also derive a generalized version of this algorithm, GAP, which is able to deal with a much wider range of types of data that can be met in real applications than LinUCB.

More details can be found in [Nicol et al., 2012]. Nevertheless the most surprising part of this work was the fact that we were able to build an even slightly better algorithm by ignoring the "action" taken by Adobe. This means that the difference between the users -in terms of click probability- is much stronger than the lift which can be obtained using some personalization. Thus we felt like the measure of performance of this challenge was not fully relevant for building an actual RS and we proposed a different metric for the next challenge.

1.2.2 ICML'13

As a follow up to the ICML'12 challenge we organized a new one [Mary et al., 2012] based on some Yahoo! news data using the offline evaluation as described in[Langford et al., 2008; Li

et al., 2011]. The task was to optimize the click probability on a recommended news given a description of a visit: some features are about the news and some others about the user. This was the first challenge with data collection done using a random uniform allocation of the news, which allows to test any policy without requiring to bias the estimators to avoid variances issues because all the regions of space are explored in the same way.

This challenge was a great success with more than 5000 submitted algorithms from more than 20 teams - including MIT and Princeton. Different methodologies were tested and some conclusions where shared among teams:

- Thompson sampling is actually effective an requires almost no tuning while UCB bases methods were more sensitives
- It was hard but possible to perform well using the features. *i.e.* there is some information hidden in but hard to spot online.
- Some news are so important that any exploration and personalization should be turned off when they occur. This is because the click probability on a news decrease with time. Moreover this news will be replaced by a new one after some delay. So it was very important to get as more click as possible with them.

The winner of phase 1 managed to handle all these parts and built a truly personalized news recommender system. Yet the final result was surprising: the winner -a master student from Peter Auer- used a UCB-v variant [Audibert et al., 2009] which consist in selecting the arm (active news) with the highest score $\hat{\mu}$ that takes care of variance estimation:

$$\hat{\mu} = \mu + \sqrt{\frac{c \cdot \mu \cdot (1-\mu) \cdot \log(t)}{n}} + c \cdot \left(\frac{0.5 - \mu}{n}\right) \log(t)$$

Where t is the current time step, c a constant to tune -winner used 1 as a value, μ empirical mean of the arm (news), n number of plays with this arm. This entry did not performed so well in the first part of the challenge but had no overfitting. Latter studies on the data showed that:

- $C = 1$ was close the optimal choice on the test set,
- The performance was due to the presence of a high number of news with a low variance on the click rate. As the variance of a Bernoulli of parameter p is $p(1-p)$, in UCB-v this leads to a great penalty of optimism for news with a click rate close to 0 while best news benefits both from a higher mean estimate and higher variance.

More details can be found in the chapter 4 of the Olivier Nicol's PhD thesis [Nicol, 2014].

1.2.3 RecSys'14

This challenge is described in [Said et al., 2014a] and was about identifying tweets with high probability of retweet. despite the fact that the task is inherently sequential, it was turned into

a ranking problem, which is a common surrogate in recommender systems because users prefers RS to be able to discover to a good ordering over their tastes rather than being able to precisely estimate their ratings in terms of RMSE.

With Frédéric Guillou, Romaric Gaudel and Philippe Preux we won this challenge by developing a variant of LambdaMART [Burges, 2010] which is the boosted tree version of LambdaRank, which is based on RankNet which is a way to learn to rank using stochastic gradient algorithm. Our variant was more sensitive to features strongly correlated to the result (namely retweet was definitely the most important feature).

So again we emphasized the importance of data analysis when working on an offline task, but also demonstrated that the achieved result is strongly biased by the evaluation metric which should as close as possible to the actual use which is online.

More details can be found in [Guillou et al., 2014].

1.2.4 Orange

This collaboration was at the origin of my interest in recommender systems and click optimization. The idea is not only to build some estimators of the click probability but also to decide how to use them when one needs to include some constraints from the application. As an example Ads are usually sold with a budget and a duration and this is not considered as a good practice to spend all the customer's budget on the first day. I provide more focus to this work in the section 1.3.

1.2.5 Deezer

This work is not fully disclosable, but the idea was to overcome the drawbacks from vanilla matrix factorization in the case of music recommendation. Common drawbacks are the presence of "Hubs" which are points which tend to be too close to many other ones and the fact that music recommendation is inherently a sequential task within a context. Most of people do not appreciate the brutal mixing between two different types of music -even if they do like both- and have preferences which fluctuates with the period of the day. There is also a particular attention to pay to the popularity of the artists: the most popular ones are very useful to explore the taste of a user but good music recommender system should avoid only focussing on them. Finally an other problem is the building of satisfaction indicators and to be able to modify the recommendation flow under negative feedbacks from the user.

1.2.6 Nuukik

Nuukik is a start-up created in 2012 which targets the market of the sales recommendation form mid-sized online stores. In order to give as much as possible flexibility to their customers in their marketing strategy, they developed an easy to configure engine to plug over any recommender system. Our first collaboration was in the field of the cold start in order to be able to address smaller businesses. When actually very few data are available, matrix recommendation based systems usually requires to be completed with some additional features (this is known as hybrid

systems), Francis Bach showed how the use any kernel on these features to conduct a generalized low rank matrix completion [Abernethy et al., 2006]. But on some situations, including the new products, the amount of data is not big enough and one has to rely on the exploration strategy to collect the required amount of information. We used an adaptation of the ideas from the linear bandits to do it [Mary et al., 2014a] leading to a patent deposit. A new contract is ongoing about the clever use of the seasonality of sales and the mixing of data from different stores.

1.2.7 TBS

This work was about the prediction of affluence on newspaper websites. This question is important for several reasons: First, hardware and network bandwidth need to be provisioned if a site is growing. Secondly, sites that sell premium advertising space need to estimate how many page views will be available within the 6 to 8 next weeks. If vanilla time series are applied they are trapped mostly by two phenomenons:

- Strong impulsive and occasional changes because of some breaking news. The impulses cause large prediction errors, long after their occurrences.
- Complex seasonality: there is a superposition of many seasonalities: day in the week, holidays, weather, hour of the day...

Based on data from several major French news websites including some TVs and radios, we built a robust prediction algorithm based on wavelet decomposition of the seasonal trend outperforming the state of the art by more than 20% in terms of mean square error at 6 weeks.

1.3 Approximate planing

In this section we consider a particular case of recommendation with some constraints on the advertising campaign. Usual constraints on campaigns can be limits on the number of displays as well as lifetimes with the requirement to not spend too fast a fixed budget. This problem is essentially sequential in time, having virtually no end. At a given time t, there is a pool of K^t running ad campaigns $Ad_1, Ad_2, \ldots, Ad_{K^t}$. At time t, each ad campaign Ad_j has its remaining click budget B_j^t which is the number of clicks that this ad has to receive during a certain amount of time, its remaining lifetime L_j^t. The B_j^t and L_j^t are known (these are specified by the contract between the announcer, and the website manager). Finally, at each time, some ad campaigns reach the end of their lifetime, and new ad campaigns, along with their particular budget and lifetime, may appear with probability u. For this purpose, we may assume that there exists an unknown generative model $\mathcal{M}$ that generates ads at each time step.

Each time a visitor requests a certain page (url) on the website, we assume some side information may be available about the visitor (for instance because of the presence of a cookie on the visitor's computer, or because he/she logged in on the website and some information has been kept about him/her); this information may or may not be accurate with regards to the current visitor, that is, the real person being behind the computer, and navigating

this website. Based on this information and the requested webpage itself, or more likely, the category of this requested webpage, each visitor is assigned to one among N profiles, denoted $Profile_i, i \in 1, \ldots, N$. To each profile is associated its daily probability of visit on the website $R_1, \ldots, R_N$ ($\sum_1^N R_i = 1$). The requested page is part of the profile so that two requests of the same visitor of two different webpages are associated to two different profiles. We also assume that $\forall i$, R_i is known: indeed, these quantities are easily estimated from weblogs: the number of visits is so large that the confidence on these estimates is very high.

Now, when a visitor requests a certain page of the website, one ad is selected to be displayed on the page. The goal is that during its lifetime, each ad is clicked according to its budget; at the end of a given ad campaign, an inferior amount of clicks decreases the income of the web operator, the decrease being rather sharp. For ease of presentation, we suppose that the pay-off associated to a successful ad campaign (the budget of clicks has been obtained), as well as the cost of not fulfilling it is the same for all campaigns. Moreover, the ads of a given campaign are expected to be clicked at a quite uniform rate along its lifetime.

Here, we assume that the problem is stationary, this means that the profiles, the probability of visits belonging to each profile, the probability that a visit of a profile produces a click on an ad, and the ad campaign generative model are constant in time.

To be precise, we define the following words: a "visitor" is a real person who is currently viewing web pages; a "request" is a request to a web server to open an url along with the available (if any) side information; an "announcer" is an entity who is responsible for an ad campaign; the announcer contracts the "website manager" to buy a certain amount of clicks on some ads.

1.3.1 Towards our solution

In order to maximize the expected total number of clicks, a very valuable piece of information is the probability of click (usually called *Click-Through Rate* – CTR) of any profile, on any ad. Let us denote $p_{i,j}$, the probability that a request belonging to $Profile_i$ results in a click on ad Ad_j. The probabilities $p_{i,j}$ are unknown. An accurate prediction of these $p_{i,j}$ results in the display of attractive commercial banners to the website visitors. Still, as this learning is performed online, the main issue is to balance the estimation (exploration) of the unknown parameters, with the exploitation of their current estimates. This problem can be formulated as a multi-armed bandit problem (with the ads in the role of the arms).

However, the existence of finite click budgets, along with ad mortality after a finite and rather small lifetime, requires non asymptotic optimal, or to the least good, solution: indeed, we can not satisfy ourselves with an optimal policy that would be reached after a very large numbers of page visits, and clicks, waiting for the law of large numbers to be acceptable. This finiteness of click budgets and lifetimes makes the problem a combinatorial one, along with stochasticity, uncertainty, and an evolution in time. Finiteness creates dependencies between the allocation of the visits to the different ads. The optimal solution is thus no longer to display its favorite ad to each profile that is, the ad which is the most likely to be clicked by this visitor. The pure greedy approach to satisfy each profile would fail to balance the odds of click with the necessary fulfillment of contracts, which is the real goal.

A workable technique to obtain an ad allocation policy is linear programing (LP), based on the current estimators of the probabilities $p_{i,j}$. However, as the LP needs to be specified a finite total number of visits, this method is designed for finite time problems. Though at first sight this finite number of visits seems to lose its meaning in a dynamic environment, we observe that it actually balances policies between greedy behaviors and too far-sighted ones on the set of current ads. Finally, one has to choose the best behavior with respect to the arrival of new ads. Hence in this part, we endeavor the use of a linear programming approach in the context of a never ending traffic of website visitors and uncertain upcoming arrival of new ads.

When we started to work on this problem, the most relevant works may be split into two groups: those based on a bandit approach and those based on a linear programming approach.

Then, a flurry of variations of the basic bandit problem has been studied in different contexts. "Contextual bandits", also known as "side information bandits", is a development in which extra information is available, hopefully to help choosing the best arm to pull [Pandey et al., 2007; Langford and Zhang, 2008; Wang et al., 2005; Kakade et al., 2008]. In the on-line advertisement problem, these information would typically be the information available on the Internet user, the page he/she is visiting, the day, the time of the day, the season, ... Among these works, Pandey *et al.* [Pandey et al., 2007] considered clusters on the website pages and on the ads and built a two-stage bandit method: select the cluster first, then select the ad in this cluster. For Kakade *et al.* [Kakade et al., 2008], the side information is a vector $x \in \mathbb{R}^d$. Inspired by the perceptron, their "Banditron" is an algorithm which goal is to classify those vectors into K labels, *i.e.* the K bandit arms. Here, the ad serving problem is reduced to an on-line multiclass prediction problem with bandit feedback.

Chakrabarti *et al.* [Chakrabarti et al., 2008] analyzed a version of MAB where ads lifetime could be budgeted and revealed to the player, or even be stochastic to model uncertain events which are beyond the control of the ad system. Their results are based on the knowledge of a prior on the arm reward distribution, and they focus on finding rapidly a good arm to stick with among a very large set of ads.

Pandey and Olston [Pandey and Olston, 2006] proposed an adaptation of the MAB framework to tackle an instance of the problem with click budgets on the ads in the multiple ad display context.

These methods could be useful in our model if they helped us to infer the click probabilities. But, most of the time, they only select the best ads in each context. Yet, to be optimal, the allocation of the different clusters of profile on limited resources has to be scheduled. This planning can be computed with a linear program. Moreover, and most importantly, these works do not consider the limitation on budgets, and the mortality of ads in their planning.

Linear Programming

[Abe and Nakamura, 1999] mixed the multi-armed bandit setting with a linear program resolution for on-line advertising with constraints on the impressions proportions. The relevance of the model was demonstrated through simulations. Still, they only considered a context with unlimited resources and a static set of ads.

[Mehta et al., 2005] treated this problem as an on-line bipartite matching problem with daily budget constraints. However, it assumed that we have no knowledge of the sequence of appearance of the profile, whereas in practice we often have a good estimate of it. [Mahdian and Nazerzadeh, 2007] tried then to take advantage of such estimates while still maintaining a reasonable competitive ratio, in case of inaccurate estimates. Extensions to click budget were discussed in the case of extra estimates about the click probabilities. Nevertheless, the daily maximization of the income is not equivalent to a global maximization.

Hybridizing Bandits and Linear Programming

To sum-up, we wish to optimize a combinatorial problem in order to determine the probability of allocation $s_{i,j}$ of a certain $profile_i$ on a certain Ad_j. However, a key ingredient for such a traditional approach is missing which is the probability of click $p_{i,j}$ of any $profile_i$ on any Ad_j. These probabilities have to be estimated. This two-fold problem calls for a combination of combinatorial optimization method, along with a learning algorithm. Linear programming and multi-armed bandits provide such tools to be used in sequence: first MAB to estimate the probabilities, and then LP to obtain the optimal solution; however, the word "optimal" may be misleading since we mean here, optimal up to the accuracy of the estimation. However, we are not yet solving the real problem in which:

1. requests observed during a certain time span are not distributed according to the estimated probability of visits R_i,
2. real visitors do not click according to estimated probabilities,
3. new ad campaigns will appear in a near future. The way they appear, their frequency and quality, influences the way we should allocate our ads in the present.

These three issues create discrepancies between the "optimal" solution computed at a certain time, and what may be considered optimal later on, when visitors, and clicks are actually observed (even not mentioning the upcoming ads).

To cope with these observed events, a first solution is to continuously adapt the solution of the problem to new observed conditions. Basically, this is done by merely iterating the process of estimation of probabilities, and solving the combinatorial optimization problem with those current estimates on the current ads.

Before delving into the details of our algorithm, let us define precisely the following:

- $p_{i,j}$ is the probability of click of any $profile_i$ on a displayed Ad_j. For the sake of simplicity, we assume that these probabilities are stationary.
- $\hat{p}^t_{i,j}$ is the estimated probability of click of any $profile_i$ on a displayed Ad_j, thus $\hat{p}$ is an estimate of p. Even though p is stationary, $\hat{p}$ depends on time t because this estimate relies on the series of events that have been observed up to time t.
- $x^t_{i,j}$ is the "optimal" probability of allocation of $profile_i$ to a displayed Ad_j. This would provide the optimal policy if we were relying on the real p, not on this estimate $\hat{p}^t$ to compute x. Henceforth, x depends on time as $\hat{p}$ depends on time.

Sort the set of ads so that Ad_1 dies first and Ad_{K^t} dies last.

$$\textbf{Maximize} \quad \sum_{\substack{1\leq i\leq N\\ 1\leq j\leq K^t}} x_{i,j}\, p_{i,j} \tag{1.3}$$

$$\textbf{Subject to} \quad \sum_{1\leq i\leq N} x_{i,j}\, p_{i,j} \leq B_j^t \qquad \forall j \in \{1,\dots,K^t\} \tag{1.4}$$

$$\sum_{1\leq j\leq K^t} x_{i,j} \leq R_i * H \qquad \forall i \in \{1,\dots,N\} \tag{1.5}$$

$$\sum_{1\leq k\leq j} x_{i,k} \leq L_j^t * R_i \qquad \forall i \in \{1,\dots,N\}, \forall j \in \{1,\dots,K^t\} \tag{1.6}$$

$$x_{i,j} \geq 0 \qquad \forall i \in \{1,\dots,N\}, \forall j \in \{1,\dots,K^t\} \tag{1.7}$$

- $s_{i,j}^t$ is the effective probability of allocation of $profile_i$ to a displayed Ad_j. If p was known, x would be optimal, and s would be equal to x; however, since we rely on $\hat{p}$ to compute x, we should not stick to (exploit) the current allocation x, but also explore alternate allocations. s is thus a combination of exploitation (rely on x) and exploration.

1.3.2 Our hybrid algorithm: MAB+LP

Our idea is to cast this problem into the linear program framework paying some attention to the building of reliable estimators. We note H the horizon, that is, an expected number of ad requests. x, the result of the LP is a matrix where $x_{i,j}$ is the number of visits from profile i to allocate on ad j. Given $(N, K^t) \in \mathbb{N}^2$, the number of profiles and the number of currently running ad campaigns respectively, $p \in [0,1]^{N\times K^t}$, the click probabilities, $(B^t, L^t) \in \mathbb{N}^{K^t} \times \mathbb{N}^{K^t}$ the budgets, and $R \in [0,1]^N$, the profiles appearance proportions, find $x^t \in \mathbb{R}^{N\times K^t}$, the allocation policy:

So we have to solve a linear problem with $N \times K^t$ variables and $N \times K^t + N + K^t$ constraints. The objective (eq. (1.3)) is to maximize the number of clicks. Inequalities (1.4) express the constraints on the click budget of the ads. Inequalities (1.5) express the constraints on the number of visits in the profiles. Inequalities (1.6) express the constraints on the lifetime of the ads. For the latter inequalities, the idea is to constrain, for each profile, the allocation on an ad to be less than the number of people that the ad will see during its life. But, recursively, the allocation on ad A has also to take into account the number of persons already taken by the ads which will die before A. Note that the L_j may be considered as a number of ad requests, as we will assume the number of visits is constant over days (which is false in the general case, but do not significantly alter the solution).

If the click probabilities $p_{i,j}$ were known, the planning induced by x would be, in expectation, the best non adaptive policy to follow. Still, the CTRs have to be learned on-line. A simple idea, already used in [Abe and Nakamura, 1999], is to use the current estimate of p to compute the planning. For example, after having observed a certain amount of visits, a straightforward

Table 1.1: The iterated routine.

.

Loop at time t:

Allocation policy:

Current estimators:	$\widehat{p}_{i,j}^t$
Compute the linear program:	$x^t = LP(\widehat{p}^t, B^t, L^t, R, H)$
Compute the probability allocations:	$s_{i,j}^t = \frac{x_{i,j}^t}{\|x_i^t\|}\ \forall i \in \{1 \dots N\}, \forall j \in \{1 \dots K^t\}$ with $\|x_i^t\| = \sum_{j=1}^{K^t} x_{i,j}^t$

Visit: A visit, the t^{th}, occurs from $Profile_i$.

Display: One ad, Ad_l, is chosen, each Ad_j being displayed with probability $s_{i,j}^t$.

Click: A click occurs or not.

Update: $\widehat{p}_{i,l}^t$ is updated.

$L_k^{t+1} \leftarrow L_k^t - 1 \quad \forall\, k \in \{1, \dots, K^t\}$

$B_l^{t+1} \leftarrow B_l^t - 1$ if a click has occurred.

A new Ad appears with probability u (update K^t if so).

estimate for $p_{i,j}$ is $\widehat{p}_{i,j}^t = \frac{NC_{i,j}^t}{NI_{i,j}^t}$ with, for each profile i and ad j, $NC_{i,j}^t$ being the number of clicks observed up to time t and $NI_{i,j}^t$ the number of displays performed up to time t. This involves that each time a new visit is observed, $\widehat{p}_{i,j}^t$ can be updated because there is a pair (i,j) for which $NI_{i,j}$ has been incremented (and perhaps even the associated $NC_{i,j}$). Then one should immediately take into account this improved estimator and recompute the associated allocation policy. This leads to the algorithm outlined in table Tab. 1.1.

Yet, in practice, repeatedly solving the LP after every web page request is not reasonable when facing very large amounts of website visitors per day. Hence, to make it effective, we can set a time period T ($T \gg 1$) and carry out this computation every T visits.

Moreover, this is an on-line learning problem where one has to learn the $p_{i,j}$ which are used to compute the LP solution, from which the probability of display of ads to the different visit profiles is deduced (the $s_{i,j}^t$ in the algorithm 1.1). Then, we have to balance between exploiting the current estimates and exploring the set of possible actions to improve this estimate. This exploration-exploitation trade-off is naturally addressed in the multi-armed bandit setting. Based on the MAB framework, different ways to introduce exploration in the allocation policy are possible, among which we mention the following two:

LP-ε: Deflect from LP solution ε-greedy methods can be applied. This means following the LP solution with a (high) probability $1 - \varepsilon$ and with probability ε, choose an ad at random.

Exp-LP: Modify the LP the $\widehat{p}_{i,j}^t$ can be modified to compel the planning to include exploration. For this purpose, Abe and Nakamura used Gittins' indices [Abe and Nakamura, 1999]. We may also substitute the $\widehat{p}_{i,j}^t$ for the associated UCB indices, or with a value sampled from the posterior Beta distribution over the probability of click (See [Granmo,

H	R_i		Budget	100	100
				Ad 1	Ad 2
20	1/2 →	Profile 1		**10**	0
	1/2 →	Profile 2		**10**	0

Planning with $H = 20 \rightarrow$ **Greedy**
Profile 1: 100% on Ad_1
Profile 2: 100% on Ad_1

H	R_i		Budget	100	100
				Ad 1	Ad 2
300	1/2 →	Profile 1		**125**	**25**
	1/2 →	Profile 2		0	**150**

Planning with $H = 300 \rightarrow$ **Far-sighted**
Profile 1: 73% on Ad_1, 17% on Ad_2
Profile 2: 100%on Ad_2

2008]).

1.3.3 The Horizon of allocation

In [Girgin et al., 2011], we report the performance of this first method on a 4 days simulation. For this finite time simulation, the horizon parameter H can be precisely set to the remaining number of visits to be treated. Then H becomes H^t and is decreased by one after every ad request. Still, in the real problem, time does not stop after a fixed number of days. The flow of website visitors is infinite and new ads are constantly created, and others are stopped. Moreover, we wish to maximize the total number of clicks, rather than a daily amount of clicks. Therefore, the performance of our policies is dependent on the way new ad campaigns are created, and also on their quality (whether the ad is attractive or not). At this point, an important question is raised: should we be greedy with our current ads and just give to every visitor his/her favorite ad because really soon other interesting ads will enrich our pool of ads? Or should we be more far-sighted and allocate as if a long time will pass before a new ad campaigns appears?

Actually, tuning the horizon parameter, *i.e.* the number of visits we are planning to process, let us balance between a greedy and a far-sighted strategies on the current set of ads. Indeed, if we allocate the next visits as if there were just a little number of visitors yet to come, then the budget constraints would not prevent any visitor from getting its favorite ad, and the algorithm would play greedily. On the contrary, forecasting the arrival of people on a long term basis would render MAB+LP more far-sighted on how to manage the current set of ads. H will now be a fixed parameter with which the MAB+LP algorithm will compute the policy applied to next visit. To illustrate this point, let us consider the following problem displayed in a tabular format that provides the $p_{i,j}$ (Note that they are set here to unrealistic values for the sake of comprehension):

	Ad 1	Ad 2
Profile 1	0.8	0.1
Profile 2	0.8	0.5

Now assume both ad campaigns have a budget of 100 clicks, and 2 visit profiles, each with H expected visits. In the two tables below, the allocation computed by the LP is given for two horizons, $H = 20$, and $H = 300$:

The figures inside these 2 arrays are the numbers of visits allocated to each (profile, ad)

pair. To obtain the amount of clicks for each pair, these figures have to be multiplied by the $p_{i,j}$. With $H = 20$, the allocation is greedy whereas when $H = 300$, the algorithm tries to take into account the constraints because this big horizon makes it think that a large number of visits will have to share the limited budgets. However, the setting of H strongly depends of the ad campaigns that will come into play in the future. But as two different values of H results in distinct policies, one has to carefully choose the right horizon for its problem.

Chapter 2

Offline Evaluation of Sequential Decision Making Algorithms

The work described in this chapter is based on these publications:

- [Mary et al., 2014b]J. Mary, O. Nicol & P. Preux *Improving offline evaluation of contextual bandit algorithms via bootstrapping techniques* International Conference on Machine Learning, ICML 2014

- J. Mary *Exploration and Exploitation, Lessons from challenges*, Large-Scale Online Learning and Decision-Making Workshop, Windsor, 2012

In machine learning being able to measure the performance is usually the first step to solve a real world problem. Most of the time cross validated losses can be used for this task. But this is not enough when we enter the full loop of decision making: we want to use an algorithm not only to predict a click probability but in fact we aim to maximize this probability. So the algorithm has to take some actions, observes the results and loop. This is why we are interested in formalizing our measure in term of regret or cumulated rewards. For short we are optimizing a policy rather than a predictor.

But in reinforcement learning -which includes multi-armed bandits- this task of policy evaluation is possibly the most fundamental problem. Of course to estimate the average reward obtained by running a given policy to select actions can be done by running the policy and measure the average reward collected online. However in many applications, running a new policy in the actual system can be expensive or even impossible. For example, controlling a vehicle with a new policy can be risky and deploying a new ad display policy on a website may lead to bad user experience and loss of customers...

Being able to evaluate offline (also known as off-policy evaluation [Precup et al., 2000] or counterfactual reasoning [Bottou et al., 2012] or covariante shift [Quionero-Candela et al., 2009]) is an important problem of any algorithm aiming to learn a policy.

2.1 Replay

This can be formalized as follows: we collect some data using a policy π_{LOG} to make the decision online. As we are online, it is possible to gather the rewards and to build a dataset. $D = (x_t, \pi_{LOG}(x_t), r_t), t = 1 \dots T$ where x_t is the description of the context of the visit- *i.e.* the feature you want to use to make a decision, $\pi_{LOG}(x_t)$ is the chosen action -*i.e.*. the recommendation- and r_t the reward - *i.e.* click action, buying,...

Our goal is to use the collected data to evaluate a new policy π. Sadly there is some impossibility results. First obviously if the probability of π_{LOG} to take action a in context x_t has never been explored then it is impossible to build a consistent estimator.

Secondly [Langford et al., 2008] shows that if π_{LOG} is context dependent it is impossible to assert that a given statistic about a given policy can be computed using a dataset, even if we know that the logging policy plays all the actions in equal proportion. Such an example can be constructed as follows. Let $\mathcal{D}$ be a contextual bandits distribution with two arms (or actions) and a context set $\mathcal{X}$ composed of only two distinct elements 0 and 1. For all t, $\vec{r_t} = (0, 0)$ - which means that both actions yield a reward of 0 - if the context is 0 and $\vec{r_t} = (0, 1)$ - which means that action 0 yields a reward of 0 whereas action 1 yields a reward of 1 - if the context is 1.

Provided that the contexts are equiprobable, a logging policy defined by $\pi_{LOG}(i) = i$ performs both actions in equal proportions. A dataset logged by this policy would look like that:

context	0	0	1	0	0	1	1
action	0	0	1	0	0	1	1
reward	0	0	1	0	0	1	1

...

Let another policy be defined as $\pi(i) = 1 - i$. It is easy to see that π would have a CTR of 0 on $\mathcal{D}$. However it would be impossible to use a dataset logged by π_{LOG} to have the slightest idea of any characteristic of π's CTR. Indeed it only contains context-action couples of the form $(0, 0)$ or $(1, 1)$ whereas π performs action 1 in context 0 and action 0 in context 1.

When π_{LOG} is random uniform, it is possible to use the replay strategy: 1 and to analyze it. The idea is to present the contexts to the new policy in the same order than during the data acquisition phase. For each context we ask to the new policy π to choose an action. If the chosen action is the same than π_{LOG} then the corresponding reward is revealed and the policy π is updated, else we just go to the next available context without updating not rewarding π.

Yet simple this strategy is very effective when a limited number of actions are possible. Which is usually the case with problems like news recommendation or best seller identification. Nevertheless the problem has some interesting extensions.

2.1.1 Bias and variance

First even if not actually stated in the initial analysis which was only asymptotic, the algorithm presents a bias that was identified in [Nicol, 2014].

Algorithm 1 *Replay method*(introduced by Langford *et al.* [Langford et al., 2008], analyzed by Li *et al.* [Li et al., 2011]) on a dataset S acquired via uniformly random interactions -*i.e.* π_{LOG} is random uniform- with a context distributed following a distribution $\mathcal{D}$.
Remark: In order to be rigorous, we use a history of events h_t which is the list of triplets (x, a, r) from S that the method did not reject before time t.

Input: A contextual bandit algorithm A and a dataset S of T triplets (x, a, r)
Output: An estimate of g_A

$h_1 \leftarrow \emptyset$
$\widehat{G}_A \leftarrow 0$
$V \leftarrow 0$ ▷ Counts the number of evaluations.
for each $t \in \{1 \,..\, T\}$ **do**
  $\pi \leftarrow A(h_t)$
  if $\pi(x_t) = a_t$ **then** ▷ $\approx choosefunction$
    $h_{t+1} \leftarrow h_t + \{(x_t, a_t, r_t)\}$ ▷ $\approx updateprocedure$
    $\widehat{G}_A \leftarrow \widehat{G}_A + r$
    $V \leftarrow V + 1$
  else
    $h_{t+1} \leftarrow h_t$ ▷ We do nothing, the history does not change.
  end if
end for
return $\frac{\widehat{G}_A}{V}$

Theorem 1. *Given a contextual bandit problem $\mathcal{D}$, a static policy π, a dataset S of size T containing i.i.d. recommendations acquired using a random uniform logging policy on $\mathcal{D}$, the estimator defined as follows:*

$$\hat{g}_\pi^{(replay)}(S) = \begin{cases} \frac{\sum_{t=1}^T V_t.r_t}{\sum_{t=1}^T V_t} & when \ \sum_{t=1}^T V_t > 0 \\ 0 & when \ \sum_{t=1}^T V_t = 0 \end{cases}$$

which is outputted by the replay method is a biased estimator of $g_\pi(\mathcal{D})$. This bias, which can be quantified as follows:

$$\mathbb{E}\hat{g}_\pi^{(replay)} - g_\pi = -g_\pi . \left(\frac{K-1}{K}\right)^T \ ,$$

goes to zero exponentially fast as T grows.

This bias comes from the observation initial sampling policy π_LOG where all actions are not present the exact same number of times. We also provided an unbiased algorithm to estimate the next policy π *Replay**. This algorithm uses our knowledge of the initial sampling policy (random uniform) to build a estimator which can be interpreted as the "importance sampling" estimate.

This bias reduction comes at cost of an increased variance on the estimator. In fact this variance increase is enough to make the unbiassed estimator not admissible in terms of means square error. We had some early results but [Li et al., 2015] where the first to provide a complete analysis. For short, it is better to the estimate of π_{LOG} from the sampled data rather to use the theoretical values. This can be interpreted as a variant of the James Stein's paradox.

Remark 5. *The James Stein's paradox states that it is possible to reduce the quadratic risk -with respect to "ordinary" least square estimate- for the estimator of a mean of observation of gaussian n random vectors of known variance σ^2 using estimates of the form of*

$$\hat{\mu} = \left(1 - \frac{n}{n + \bar{\boldsymbol{x}}^2 \boldsymbol{x}}\right)$$

This obviously comes at the cost of a bias on the estimator.

2.1.2 Non uniform sampling

In a running system, the cost of uniform sampling can be too high in terms of revenue or user experience. If the system is performing some epsilon greedy exploration, it is possible to use this data as in section 2.2. But there is a strong desire to use all the data and not only a subpart of it.

If we keep the assumption of a π_{LOG} independent from the context - which is acceptable when recommendations are based on best sales - then it is possible to reweigh the estimates of Replay to build some estimators low bias or unbiased estimators. Again here the importance sampling is not an admissible estimator as a reweighed replay has a lower MSE. But even in this setting, the learning of a new policy π can be very problematic as some regions of the (context,action) space are more sampled. It is easier to learn (and not only to evaluate) a policy close to π_{LOG} than a different one. This is a core problem because all the guarantees only hold when both π and π_{LOG} are fixed. If we do not pay attention to this problem it is very likely that we are going to be trapped in a local optima because we are not able to train and evaluate poorly policies different from π_{LOG}. Of course the analysis holds only with an independence assumption on the context with respect to the past actions which is often false.

The solution to overcome this is to pay more attention to confidence intervals and to use the causality framework. Of course these can be difficult to build and the problem is equivalent to the batch reinforcement learning which is not fully solved. One interesting step in this direction is done by Leon Bottou [Bottou et al., 2012] in the case of Ad placement optimization together with some bidding strategies. The core idea is to model explicitly some of the dependences and to assume they are the only one which are relevant to objective function. Then using some concentration theorem it is possible to compute confidence intervals of the performance of the policy with respect to some variations around π_{LOG}. Then the choice of the new policy to use in production is left to expert choice and will involve a tradeoff between risk and performance.

2.2 Bootstraped Replay

When replay-like algorithms are used to evaluate the performance of policies, one issue is that they uses only a small part of the initial data to evaluate the policy. A large part of the initially collected dataset is discarded to ensure unbiased evaluation asymptotically. This is what we called the time acceleration issue. Now, we propose an adaptation of the method: each line of the dataset is going to be presented to the policy several times. This repetition increases the number of lines used to update the policy at the cost of a possible overfitting. To tackle this bias we also introduce a cross-validation step.

We first tag some lines of the dataset S collected under π_{LOG} as "validation lines". Then we build a bigger dataset B duplicating a fixed number of times each non tagged lines (number of possible actions on each choices seems to be a good empirical value for the number of duplications). We now use the replay methodology on this bigger dataset -so the policy is allowed to have more time to explore interesting areas - but the CTR estimate will be built only on lines tagged for validation (which are not duplicated). We also proposed some heuristics to fit the case of non constant click rates. It leads to a new algorithm BRED 2 which can be interpreted as the use of the bootstrap to build estimates for contextual bandits algorithms.

Algorithm 2 *Bootstrapped Replay on Expanded Data* *BRED.*

Input

- A (contextual) bandit algorithm A
- A set $\mathcal{S}$ of T triplets (x, a, r)
- An integer B

Output: An estimate of g_A

$h^{(b)} \leftarrow \emptyset$, $\forall b \in \{1..B\}$ */*empty history*/*
$\widehat{G}_A^{(b)} \leftarrow 0$, $\forall b \in \{1..B\}$
$T^{(b)} \leftarrow 0$, $\forall b \in \{1..B\}$
/ Bootstrap loop*/*
for $b \in \{1..B\}$ **do**
  / estimation of* $CTR_A^{(b)}(T)$ **/*
  for $i \in \{1..T * K\}$ **do**
    Sample with replacement an element (x, a, r) of S
    $x \leftarrow$ JITTER(x) */*optional - adds some noise*/*
    $\pi \leftarrow A(h^{(b)})$
    if $\pi(x) = a$ **then**
      add (x, a, r) to $h^{(b)}$
      $\widehat{G}_A^{(b)} \leftarrow \widehat{G}_A^{(b)} + r$
      $T^{(b)} \leftarrow T^{(b)} + 1$
    else
      /* Do nothing. */
    end if
  end for
end for

return $\frac{1}{B} \sum_{b=1}^{B} \frac{\widehat{G}_A^{(b)}}{T^{(b)}}$ **OR** **return** $\frac{\sum_{b=1}^{B} \widehat{G}_A^{(b)}}{\sum_{b=1}^{B} T^{(b)}}$

/* The first one is the bagged estimate as it is usually defined in bootstrapping. The second comes with the nice property of having a bias in $O(a^{TB})$ instead of $O(a^T)$, with $a = \frac{K-1}{K}$ which is slightly less than one. This may be significant if T is small.*/

The algorithm has provable concentration rates around the estimates of CTRs using the

nicest property of the bootstrap: we are asymptotically more accurate than the standard intervals obtained using sample variance and assumptions of normality.

Theorem 2. *Assuming that*

$$\xi\left(CTR_A(T)\right) = z + \frac{p_1}{\sqrt{T}} + \ldots + \frac{p_\alpha}{T^{\alpha/2}} + o\left(\frac{1}{T^{\alpha/2}}\right)$$

Then for algorithm A producing a fixed policy over time, BRED applied on a dataset of size T evaluates the expectation of the CTR_A with no bias and with high probability for B and T large enough:

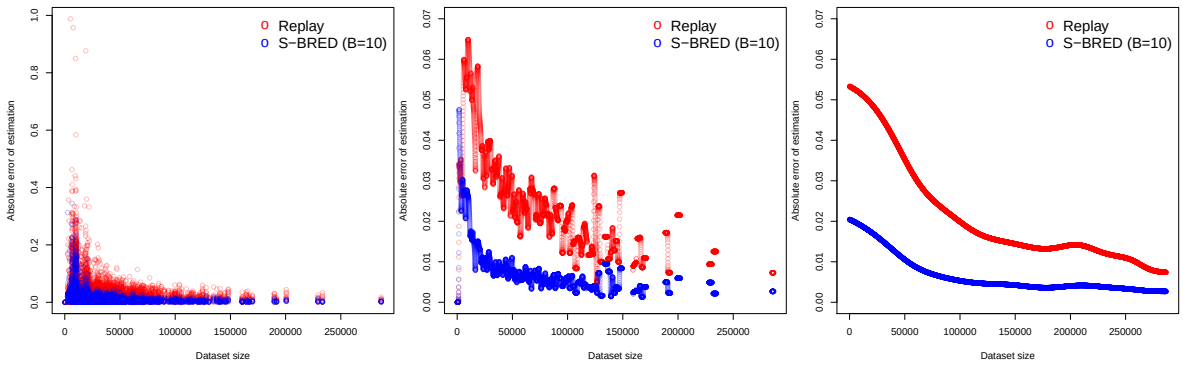
$$\left|\xi\left(CTR_A(T)\right) - \xi\left(\widehat{C}TR_A(T)\right)\right| = O\left(\frac{1}{T}\right)$$

This means that the convergence of the estimator of $\xi\left(CTR_A(T)\right)$ is much faster than the convergence of the estimator of $g_A(T)$ (which is in $O(1/\sqrt{T})$. This will allow a nice control of the risk that $\hat{g}_A(T)$ may be badly evaluated.

The sketch of the proof of theorem 2 is the following: first we prove that the replay strategy is able to estimate the moments of the distribution of CTR_A fast enough with respect to T. The second step consists in using classical results from bootstrap theory to guarantee the unbiased convergence of the aggregation $\widehat{CTR}_A(T)$ to the true distribution with an $O(\frac{1}{T})$ speed. The rational behind this is that the gap introduced by the subsampling will be of the order of $O(\frac{1}{\sqrt{TB}})$.

Adding some noise in the duplicates (aka Jittering which can be interpreted as a regularization or some smoothing) we are also able to build better empirical estimates for context dependent policies. We also introduced a variant of the bootstrap S-BRED which exhibits better performance in this setting. Instead of performing B bootstrap resamples of size KT, S-BRED gets the B resamples as follows: it expands the dataset by simply appending $K-1$ copies of S to itself. S-BRED then applies a random permutation to all the S_b (this is called shuffling, hence the S in front of the acronym). This is illustrated on 2.1.

More details can be found in [Mary et al., 2014b] and in chapter 5 of [Nicol, 2014].



(a) No convolution.

(b) Convolution with a Gaussian Kernel of bandwidth 1,000.

(c) Convolution with a Gaussian Kernel of bandwidth 50,000.

Figure 2.1: Absolute error of S-BRED (B=10) and replay on the various zones of the Yahoo! R6B dataset [Research, 2012]. The results are averaged (convoluted) locally using a Gaussian Kernel. Lower is better. The curves speak for themselves but one little additional detail is worth discussing. When the bandwidth is equal to 1,000, we see that there is no error with very small datasets. Then the error made by S-BRED increases faster than the one made by replay which sounds surprising. This is because when the dataset is very small, the CTR found by a replay method including the one found by the ground truth replay is small or even zero. Therefore an evaluation method that always outputs zero has low error. At some point when there is a little bit more data, the method becomes very variate as one click found completely changes the value of the estimate. Then where more data starts becoming available, the error rate drops again progressively. S-BRED only follows this process much faster than replay (see that the error then goes down faster too) as it tries to make the most with the data at hand.

Chapter 3

Ergodic time series

The work described in this chapter is based on these publications:

- [Ryabko and Mary, 2012] D. Ryabko & J. Mary *Reducing statistical time-series problems to binary classification* Neural Information Processing Systems (NIPS), 2012

- [Khaleghi et al., 2012] A. Khaleghi, D. Ryabko, J. Mary, and P. Preux. *Online clustering of processes.* In *AISTATS*, JMLR W&CP 22, pages 601–609, 2012.

- [Ryabko and Mary, 2013] D. Ryabko & J. Mary *A Binary-Classification-Based Metric between Time-Series Distributions and Its Use in Statistical and Learning Problems* Journal of Machine Learning Research, 2013

- A. Khaleghi, D. Ryabko, J. Mary & P. Preux *Consistent algorithms for clustering time series*, Journal of Machine Learning Research, 2015 (accepted to appear)

This chapter is also about sequential machine learning, but focusses on more theoretical aspect of what is a reasonable task if we have only some weak hypothesis on our data. In particular we are interested in grounding the clustering of time series. This is an important in the field of recommendation because most of the marketing department are eager to cluster their customer in meaningful categories to be able to focus their actions.

Here we choose the stationary ergodic assumption. This implies that the process will not change its statistical properties with time and that its statistical properties can be deduced from a single, sufficiently long sample (realization) of the process. This means that we can estimate the frequencies of occurrence of any "pattern" on the data by collecting data for a sufficient time and that this data collection can start at anytime. In particular the mean of the variance of the observed process cannot shift with time. This assumption is one of the weakest in statistics and it is already impossible to test the ergodicity of a signal. For intuition it can be useful to recall that ergodic Markov chains which are the chains where it is possible to go from every state to every state (not necessarily in one move). seasonality are not stationary but can be ergodic.

3.1 From time-series to classification

Binary classification is one of the most well-understood problems of machine learning and statistics: a wealth of efficient classification algorithms has been developed and applied to a wide range of applications. Perhaps one of the reasons for this is that binary classification is conceptually one of the simplest statistical learning problems. It is thus natural to try and use it as a building block for solving other, more complex, newer or just different problems; in other words, one can try to obtain efficient algorithms for different learning problems by reducing them to binary classification. This approach as described in chapter 1 has been applied to many different problems, starting with multi-class classification, and including regression and ranking [Balcan et al., 2007; Langford et al., 2006], to recall just a few examples. However, all of these problems are formulated in terms of independent and identically distributed (i.i.d.) samples. This is also the assumption underlying the theoretical analysis of most of the classification algorithms.

In this work we consider learning problems that concern time-series data for which independence assumptions do not hold. The series can exhibit arbitrary long-range dependence, and different time-series samples may be interdependent as well. Moreover, the learning problems that we consider — the three-sample problem, time-series clustering, and homogeneity testing — at first glance seem completely unrelated to classification.

We show how the considered problems can be reduced to binary classification methods, via a new metric between time-series distributions. The results include asymptotically consistent algorithms, as well as finite-sample analysis. To establish the consistency of the suggested methods, for clustering and the three-sample problem the only assumption that we make on the data is that the distributions generating the samples are stationary ergodic; this is one of the weakest assumptions used in statistics. For homogeneity testing we have to make some mixing assumptions in order to obtain consistency results (this is indeed unavoidable, as shown by [Ryabko, 2010b]). Mixing conditions are also used to obtain finite-sample performance guarantees for the first two problems.

The proposed approach is based on a new distance between time-series distributions (that is, between probability distributions on the space of infinite sequences), which we call *telescope distance*. This distance can be evaluated using binary classification methods, and its finite-sample estimates are shown to be asymptotically consistent. Three main building blocks are used to construct the telescope distance. The first one is a distance on finite-dimensional marginal distributions. The distance we use for this is the following well-known metric: $d_{\mathcal{H}}(P,Q) := \sup_{h\in\mathcal{H}} |\mathbb{E}_P h - \mathbb{E}_Q h|$ where P, Q are distributions and $\mathcal{H}$ is a set of functions. This distance can be estimated using binary classification methods, and thus can be used to reduce various statistical problems to the classification problem. This distance was previously applied to such statistical problems as homogeneity testing and change-point estimation [Kifer et al., 2004]. However, these applications so far have only concerned i.i.d. data, whereas we want to work with highly-dependent time series. Thus, the second building block are the recent results of [Adams and Nobel, 2012], that show that empirical estimates of $d_{\mathcal{H}}$ are consistent (under certain conditions on $\mathcal{H}$) for arbitrary stationary ergodic distributions. This, however, is not enough: evaluating $d_{\mathcal{H}}$ for (stationary ergodic) time-series distributions means measuring the distance between their finite-dimensional marginals, and not the distributions themselves. Finally, the third step to construct the distance is what we call *telescoping*. It consists in summing the distances for

all the (infinitely many) finite-dimensional marginals with decreasing weights. The resulting distance can "automatically" select the marginal distribution of the right order: marginals which cannot distinguish between the distributions give distance estimates that converge to zero, while marginals whose orders are too high to have converged have very small weights. Thus, the estimate is dominated by the marginals which can distinguish between the time-series distributions, or converges to zero if the distributions are the same. It is worth noting that a similar telescoping trick is used in different problems, most notably, in sequence prediction [Solomonoff, 1978; Ryako, 1988; Ryabko, 2011].

We show that the resulting distance (telescope distance) indeed can be consistently estimated based on sampling, for arbitrary stationary ergodic distributions. Further, we show how this fact can be used to construct consistent algorithms for the considered problems on time series. Thus we can harness binary classification methods to solve statistical learning problems concerning time series. A remarkable feature of the resulting methods is that the performance guarantees obtained do not depend on the approximation error of the binary classification methods used, they only depends on their estimation error.

Moreover, we analyze some other distances between time-series distributions, the possibility of their use for solving the statistical problems considered, and the relation of these distances to the telescope distance introduced in this work.

To illustrate the theoretical results in an experimental setting, we chose the problem of time-series clustering, since it is a difficult unsupervised problem which seems most different from the problem of binary classification. Experiments on both synthetic and real-world data are provided. The real-world setting concerns brain-computer interface (BCI) data, which is a notoriously challenging application, and on which the presented algorithm demonstrates competitive performance.

A related approach to address the problems considered here, as well as some related problems about stationary ergodic time series, is based on (consistent) empirical estimates of the distributional distance, see [Ryabko and Ryabko, 2010; Ryabko, 2010a; Khaleghi et al., 2012], as well as [Gray, 1990] about the distributional distance. The empirical distance is based on counting frequencies of bins of decreasing sizes and "telescoping."

3.2 Notation and Definitions

Let $(\mathcal{X}, \mathcal{F}_1)$ be a measurable space (the domain), and denote $(\mathcal{X}^k, \mathcal{F}_k)$ and $(\mathcal{X}^{\mathbb{N}}, \mathcal{F})$ the product probability space over $\mathcal{X}^k$ and the induced probability space over the one-way infinite sequences taking values in $\mathcal{X}$. Time-series (or process) distributions are probability measures on the space $(X^{\mathbb{N}}, \mathcal{F})$. We use the abbreviation $X_{1..k}$ for $X_1, \dots, X_k$. A set $\mathcal{H}$ of functions is called *separable* if there is a countable set $\mathcal{H}'$ of functions such that any function in $\mathcal{H}$ is a pointwise limit of a sequence of elements of $\mathcal{H}'$.

A distribution ρ is called stationary if $\rho(X_{1..k} \in A) = \rho(X_{n+1..n+k} \in A)$ for all $A \in \mathcal{F}_k$, $k, n \in \mathbb{N}$. A stationary distribution is called (stationary) ergodic if

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1..n-k+1} \mathbb{I}_{X_{i..i+k} \in A} = \rho(A) \quad \rho - \text{a.s.}$$

for every $A \in \mathcal{F}_k$, $k \in \mathbb{N}$. (This definition, which is more suited for the purposes of this work, is equivalent to the usual one expressed in terms of invariant sets, see, e.g., [Gray, 1990].)

3.3 A Distance between Time-Series Distributions

We start with a distance between distributions on $\mathcal{X}$, and then we extend it to distributions on $\mathcal{X}^{\mathbb{N}}$. For two probability distributions P and Q on $(\mathcal{X}, \mathcal{F}_1)$ and a set $\mathcal{H}$ of measurable functions on $\mathcal{X}$, one can define the distance

$$d_{\mathcal{H}}(P, Q) := \sup_{h \in \mathcal{H}} |\mathbb{E}_P h - \mathbb{E}_Q h|. \tag{3.1}$$

This metric in its general form has been studied at least since the 80's [Zolotarev, 1983]; its special cases include Kolmogorov-Smirnov [Kolmogorov, 1933], Kantorovich-Rubinstein [Kantorovich and Rubinstein, 1957] and Fortet-Mourier [Fournie, 1992] metrics. Note that the distance function so defined may not be measurable; however, it is measurable under mild conditions which we assume whenever necessary. In particular, separability of $\mathcal{H}$ is a sufficient condition (separability is required in most of the results below).

We are interested in the cases where $d_{\mathcal{H}}(P, Q) = 0$ implies $P = Q$. Note that in this case $d_{\mathcal{H}}$ is a metric (the rest of the properties are easy to see). For reasons that will become apparent shortly (see Remark below), we are mainly interested in the sets $\mathcal{H}$ that consist of indicator functions. In this case we can identify each $f \in \mathcal{H}$ with the indicator set $\{x : f(x) = 1\} \subset \mathcal{X}$ and (by a slight abuse of notation) write $d_{\mathcal{H}}(P, Q) := \sup_{h \in \mathcal{H}} |P(h) - Q(h)|$. In this case it is easy to check that the following statement holds true.

Lemma 1. *$d_{\mathcal{H}}$ is a metric on the space of probability distributions over $\mathcal{X}$ if and only if $\mathcal{H}$ generates $\mathcal{F}_1$.*

The property that $\mathcal{H}$ generates $\mathcal{F}_1$ is often easy to verify directly. First of all, it trivially holds for the case where $\mathcal{H}$ is the set of halfspaces in a Euclidean $\mathcal{X}$. It is also easy to check that it holds if $\mathcal{H}$ is the set of halfspaces in the feature space of most commonly used kernels (provided the feature space is of the same or higher dimension than the input space), such as polynomial and Gaussian kernels.

Based on $d_{\mathcal{H}}$ we can construct a distance between time-series probability distributions. For two time-series distributions ρ_1, ρ_2 we take the $d_{\mathcal{H}}$ between k-dimensional marginal distributions of ρ_1 and ρ_2 for each $k \in \mathbb{N}$, and sum them all up with decreasing weights.

Definition 1 (telescope distance $D_{\mathbf{H}}$)**.** *For two time series distributions ρ_1 and ρ_2 on the space $(\mathcal{X}^{\mathbb{N}}, \mathcal{F})$ and a sequence of sets of functions $\mathbf{H} = (\mathcal{H}_1, \mathcal{H}_2, \dots)$ define the* telescope distance

$$D_{\mathbf{H}}(\rho_1, \rho_2) := \sum_{k=1}^{\infty} w_k \sup_{h \in \mathcal{H}_k} |\mathbb{E}_{\rho_1} h(X_1, \dots, X_k) - \mathbb{E}_{\rho_2} h(Y_1, \dots, Y_k)|, \tag{3.2}$$

where w_k, $k \in \mathbb{N}$ is a sequence of positive summable real weights (e.g., $w_k = 1/k^2$ or $w_k = 2^{-k}$).

Lemma 2. *$D_{\mathbf{H}}$ is a metric if and only if $d_{\mathcal{H}_k}$ is a metric for every $k \in \mathbb{N}$.*

Proof. The statement follows from the fact that two process distributions are the same if and only if all their finite-dimensional marginals coincide. □

Definition 2 (empirical telescope distance $\hat{D}$)**.** *For a pair of samples $X_{1..n}$ and $Y_{1..m}$ define the* empirical telescope distance *as*

$$\hat{D}_{\mathbf{H}}(X_{1..n}, Y_{1..m}) := \sum_{k=1}^{\min\{m,n\}} w_k \sup_{h\in\mathcal{H}_k} \left| \frac{1}{n-k+1} \sum_{i=1}^{n-k+1} h(X_{i..i+k-1}) - \frac{1}{m-k+1} \sum_{i=1}^{m-k+1} h(Y_{i..i+k-1}) \right|. \quad (3.3)$$

All the methods presented in this work are based on the empirical telescope distance. The key fact is that it is an asymptotically consistent estimate of the telescope distance, that is, the latter can be consistently estimated based on sampling.

Theorem 3. *Let $\mathbf{H} = (\mathcal{H}_k)_{k\in\mathbb{N}}$ be a sequence of separable sets $\mathcal{H}_k$ of indicator functions (over $\mathcal{X}^k$) of finite VC dimension such that $\mathcal{H}_k$ generates $\mathcal{F}_k$. Then, for every stationary ergodic time series distributions ρ_X and ρ_Y generating samples $X_{1..n}$ and $Y_{1..m}$ we have*

$$\lim_{n,m\to\infty} \hat{D}_{\mathbf{H}}(X_{1..n}, Y_{1..m}) = D_{\mathbf{H}}(\rho_X, \rho_Y) \quad (3.4)$$

Note that $\hat{D}_{\mathbf{H}}$ is a biased estimate of $D_{\mathbf{H}}$, and, unlike in the i.i.d. case, the bias may depend on the distributions; however, the bias is $o(n)$.

Remark 6. *The condition that the sets $\mathcal{H}_k$ are sets of indicator function of finite VC dimension comes from the results of [Adams and Nobel, 2012], who show that for any stationary ergodic distribution ρ, under these conditions, $\sup_{h\in\mathcal{H}_k} \frac{1}{n-k+1} \sum_{i=1}^{n-k+1} h(X_{i..i+k-1})$ is an asymptotically consistent estimate of $\sup_{h\in\mathcal{H}_k} \mathbb{E}_\rho h(X_1, \ldots, X_k)$. This fact implies that $d_{\mathcal{H}_k}$ can be consistently estimated, from which the theorem is derived.*

An identical but possibly easier to catch version of the telescope distance it to split it's computation in small steps: Say we have samples $X = (X_1 \ldots X_n)$ and $Y = (Y_1 \ldots Y_m)$ and we want to compute the telescope distance between them.

First, we just run a classifier (as an SVM) on the two samples, considering each $X_i, i = 1, \ldots, n$ as a class-0 example and each $Y_i, i = 1, \ldots, m$ as class-1 example. Train the classifier and measure the number of (training) examples of class 0 (X_i) classified as 0. call this T_x^1

Then measure the number of (training) examples of class 1 (Y_i) classified also (!) as 0. call this T_y^1 then take

$$d_1 = |T_x^1/n - T_y^1/m|$$

Now we'll construct d_k in the same way, for each $k = 2, \ldots, \sqrt{n}$: we take the k-tuples

$$(X_1, \ldots, X_k), (X_2, \ldots, X_{k+1}), \ldots (X_{n-k+1}, \ldots, X_n)$$

and call them class 0 examples, and the same with k-tuples for Y, we take

$$(Y_1, \ldots, Y_k), (Y_2, \ldots, Y_{k+1}), \ldots, (Y_{m-k+1}, \ldots, Y_m)$$

and call them class 1 examples.

We train our favorite classifier, get T_x^k and T_y^k in the same way, and obtain

$$d_k = |T_x^k/(n-k+1) - T_y^k/(m-k+1)|$$

Finally the empirical estimate of the distance is:

$$d = \sum_{k=1}^{\sqrt{n}} w_k d_k$$

where $w_k = 1/k^2$ but can be any summable sequence.

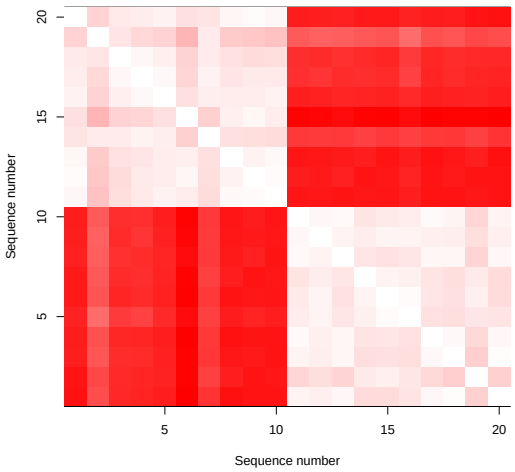


Figure 3.1: Toy example of telescope distance computed on 20 randomly generated sequences of length 200. The 10 first sequences were from an uniform law on [0,1] and the 10 next sequences are from N(0,1) distribution. Red means the distance is bigger, white that it is 0. The classifier was an SVM with RBF kernel with default parameters.

More informations and experiments on real data in [Ryabko and Mary, 2013, 2012]

3.4 Consistent Clustering of processes

Clustering is a widely studied problem in machine learning, and is key to many applications in almost all fields of science. The goal is to partition a given dataset into a set of non-overlapping clusters in some natural way, thus hopefully revealing an underlying structure in the data. In particular, this problem for time series data is motivated by many research problems from a variety of disciplines, such as marketing and finance, biological and medical research, video/audio analysis, etc., with the common feature that the data are abundant while little is known about the nature of the processes that generate them.

Nevertheless clustering is an ill defined problem if not related to a supervised task. In this work we decided to focus on clustering of stationary ergodic processes generating sequences of number (in any dimension). The assumption that a given time series is stationary ergodic is one of the most general assumptions used in statistics; in particular, it allows for arbitrary long-range serial dependence, and subsumes most of the non-parametric as well as modeling assumptions used in the literature on clustering time series, such as i.i.d., (Hidden) Markov, or mixing time series.

This allows us to define the following clustering objective: group a pair of time series into *the same cluster if and only if the distribution that generates them is the same.*

To perform this task we use properties of the distributional distance between two temporal processes ρ_1 and ρ_2:

$$d(\rho_1, \rho_2) = \sum_{m,l=1}^{\infty} w_{m,l} \sum_{B \in B^{m,l}} |\rho_1(B) - \rho_2(B)|$$

Where $w_{m,l} = w_m \cdot w_l$ and $w_i = 2^{-i}, i \in \mathbb{N}$ and the sets $B^{m,l}, m, l \in N$ are obtained via the partitioning of $\mathbb{R}$ into cubes of dimension m and volume 2^{-ml} (starting at the origin). and show it is possible to estimate it efficiently. This naturally leads to a asymptotically consistent clustering algorithm. The intuition is the following: for a fixed stationary ergodic process the average frequency of occurrence of any pattern converges to the same value if the process is observed long enough. Then we just have to pay some attention on how to be sure to be able to grasp any pattern from a sequence of numbers - this is done using a set of discretizations and to take care of all the patterns with where the length of the process is long enough to estimate the frequency of occurrence.

More information and experiments in [Khaleghi et al., 2012]

Chapter 4

Future of this work

This section is about closely related topics or directions that I wish to investigate in the future.

4.1 Multitask Learning and Recommendation

An old goal of AI is to mimic the learning behavior of biological systems. One of the idea is that for some tasks it seems easier to first learn a simplified version and then move to more sophisticated and complex goals. What we feel here is that there is more to learn in a task than being able to complete it, something about representation. To some extents convolutional neural networks [Lecun et al., 1998] trained on large datasets of labelled pictures achieve this goal: it is possible to train a new network for a different classification just by a replacement of the last layer of the network [Karayev et al., 2013]. A statistical machine learning variant of this approach is to use tensors of order 3 or more to represent data as videos or multiples relations between the same items (aka adjacency tensor of hypergraphs).

Applied to recommender systems [Rendle et al., 2010], the idea is to use the spectral structure of a tensor rather than the spectral decomposition of a matrix. The extra dimensions are used to store the time and/or to separate observations as using one layer for each category of movies or navigations/basket/buy. Here we aim to transfer knowledge between the layers of the tensor. Some recent work studied transfer learning techniques for recommender systems with the main objective of alleviating the sparsity problem of collaborative filtering. For example, [Pan et al., 2010] introduce a solution called Coordinate System Transfer in which they assume the existence of two dense source domains (user-item affinity matrix) from which the knowledge was going to be transferred to the sparse target domain, assuming that one dense source shared the same user as the target domain, and the other source the same item. [Li et al., 2009] also works on transfer learning from one dense source to a sparse target but did not that the domains shared neither user or item. Their solution based on orthogonal nonnegative matrix tri-factorization [Ding et al., 2006] copy the core matrix (called the codebook) learned from the source domain to the core matrix of the target domain. [Moreno et al., 2012] go further and transfer multiple source domains to the target domain.

One problem is that the spectral theory of third order tensor is not as nice as the matrix one. In particular there is no unicity of the decomposition without adding some additional

constraints (KKT) and the tensor is of course much more sparse than the matrices we usually work with (because of the high dimensionality of tensors). So for practical use much of the performance comes from the regularization/factorization schemes being used. The two dominant ones are CANDECOMP/PARAFAC (decomposition as a sum of rank one tensors [Harshman and Lundy, 1994]) and Tucker3/HOSVD [Lathauwer et al., 2000] (decomposition as a core tensor of constrained rank expanded to the right dimension using 3 orthogonal matrices). This approach, which induces inherently some sharing of parameters between both different terms and different relations also inspired some probabilistic formulations [Chu and Ghahramani, 2009].

This kind of strategy has been successfully applied [Jenatton et al., 2012] on large multi-relational datasets with thousands of relations. The proposed model does not perform an actual tensor decomposition but captures various orders of interaction of the data by extracting and factorizing several matrix extracted from the data tensor. This outperforms classical tensor decomposition on some tensor test datasets and present a good scaling to a huge number of classes. The method is purely statistically driven and can compete with a good wordnet [Miller, 1995] for verb detection.

Many of the ideas developed in RS to improve the results of a pure SVD can be adapted to the tensor case. As an example, it is possible to think to preparations of the data matrix (the bias removal on rows and columns, together with some renormalization across the layers), the weighted boolean representation for navigation data (work on implicit data by [Hu et al., 2008]), the use of Tikhonov regularization (ALS-WR by [Zhou et al., 2008]) to handle missing data, the factorization machine point of view...

Moreover little attention has been paid to efficient implementations of tensor decompositions. Percy Liang [Kuleshov et al., 2015] proposes tensor factorization using random projections to reduce the problem to simultaneous matrix diagonalization with guarantees on the spectral information, and we guess that L_2^2 subsampling could be also used. Finally [Jain and Oh, 2014] provides some guarantees of recovery of a tensor from partial observation in the noiseless setting. The guarantee is for three-mode $n \times ntimesn$ dimensional rank-r tensor which can be recovered exactly from $O(n^{3/2}r^5log^4n)$ independently randomly sampled entries. This also work advocates for the use of alternate least square schemes which are shown to be able to recover tensor of real data with high probability. Moreover for matrix factorization ALS schemes can be implemented in a very parallelized way because on each step of the ALS we are required to solve as many independent linear systems as the number of rows (for left step) and columns (for right step).

4.2 Sequential Recommendation and Offline evaluation

Mean square error, nDCG, logistic loss,... are convenient measures intensively used as a surrogate to evaluate the performances of a recommender system. There is some work on more exotic measures such as "diversity" [Lathia et al., 2010] -a good recommender system should present different alternatives- or "hubness" -in order to avoid the presence of items recommended for anyone- [Flexer et al., 2012]. But the nature of a recommender systems is to be online, and its performance should be expressed in terms of the collection of some sort of rewards (clicks,

buying, loved item,...). This is why many Internet companies like Netflix are running some A/B testing of different versions of their websites and some surprises can occur. As an example Netflix reported in 2012 on its forum that the increase of the size of the picture of the movies had more positive impact on the average user engagement than a diminution of the MSE of the taste matrix recovery by 5%. Of course it is possible to use bandit variants of A/B testing in order to reduce their cost (but it comes at a cost of a decreased statistical confidence from the identification of the best arm). The important point here is that recommender systems are online algorithms that take decisions.

We saw that a simplified framework for to study this setting is the contextual bandit problem. But when this framework is mixed with matrix factorization, many questions are open. A straightforward way to cast the matrix recovery problem into the bandit framework is to draw a row -a user- under a distribution and to give him a recommendation -*i.e.* select a column- then a reward is given accordingly to a distribution parametrized by the corresponding entry in the matrix.

Assuming that we are allowed to select the same position in the matrix several times, the question is: *would it be possible to get a logarithmic regret bound with respect the time?* The setting is quite similar to the linear bandit framework except that we have to compute the context (which is the V part of the SVD) from our observations. The core issue is that most of guarantees on matrix recovery are for independently sampled entries which is not the case here. We have the hope to adapt the proof of ε_n greedy here.

But then a new question arises: *what is the impact of the latent space size d?*. Of couse the smaller the latent space, the easier the problem should be.

To answer this question we first need to revisit the linear bandit analysis. Here the goal is to estimate an unknown vector $\mathbf{u}$ of dimension d. At each round we are provided k arms described by a known context vector $\mathbf{x}_i, i \in 1 \dots k$ of dimension d. The algorithm has to select an arm i and receive a reward $\langle \mathbf{u}, \mathbf{x}_i \rangle$. There are several slightly different ones but our current favorite is OFUL [Abbasi-yadkori et al., 2011] which provides a strategy with a regret bound in $O(d \log^2 T)$ where T is the number of arms pulled.

Now let us assume that the arms context vector lies in a lower space of dimension d' with $d' \ll d$ and $d' \ll k$. Then there exists an unknown matrix $\mathbf{A}$ such that the reward for arm i can be expressed as $\langle \mathbf{Ax}, \mathbf{Ax}_i \rangle$ which is a linear bandit problem of dimension d'. So we hope to prove a regret bound like $O(d' \log^2 T)$ plus the cost of the estimate of A with $O(k \log^2(T))$ as a very first guess. But it could turn in a negative result if the estimation of $\mathbf{A}$ has a high cost in terms of regret. Of course we also need to take care of the non independent sampling here.

Now if we go back to the matrix factorized bandit problem we should consider the fact that in real systems, some new items and users appear on a regular basis. In the non contextual framework this part is handled by assuming some knowledge on the probability of apparition of a new better arm (UCB-air [Wang et al., 2009]), but we also have the problem that some arms are discarded with time (and in many applications you cannot recommend the same arm forever so you need to focus on new ones). The problem is that if we assume a constant rate of arrival of new arms/items, for each arm we are going to spend a few pulls in order to evaluate their description in the latent space. This also means that it will lead us to a *linear growing* of

the regret. This mandatory linear growing of the regret can be regarded as a negative result here. Nevertheless one could study the dependency of growth with respect to d'. We also have some hope for the tensor case. More precisely we could pay the cost of exploration of some cheap layers and transfert it to the most valuable layers.

We also remarked in some experiments that in terms of regret using the "right" value for the latent space size is not always optimal. It is often better to start with a small value of d' and to increase it progressively. Intuitively this is reasonable: first focus on popular items, then popular categories -*i.e.* highest singular values. But we need to build here a finer understanding of the aggregated/clustered strategies of contextual bandits.

Besides theses aspects that would enable a better understanding of online "low rank" recommendation, there is also the hope to cast the problem inside a more general framework: the reinforcement learning problem [Sutton and Barto, 1998]. For example people from sales usually use their expert knowledge to the predicted personalized sales probabilities to build some cross-sales, up-sales or promotions. In music recommender systems it seems a good idea to include more diversity in the recommendations periodically to avoid to bore the listener. Leon Bottou also showed during his invited talk at ICML'2015 that the display position and the number of displayed ads is of crucial importance when using a complete system and can totally spoil the collected data if done improperly. The spoil comes from variations around the CTR due to an external factor not recorded: all the emplacements over a webpage are not equivalents and if the information is not recorded, you could favorise too much the elements recommended by your current system, which will lead to reinforce your confidence in the fact that you are doing well. A NIPS'15 workshop *Machine Learning for (e-)Commerce* is aiming to portray the main challenges of recommendation as a reinforcement learning problem and to propose an industry-academia agreed collection of benchmarks problems for theoretical study and experimental work on theses questions, but we can already list a few of them:

- *How to collect the explorative data?* The choice of the sampling distribution (or the need for rollouts) is a hard problem of RL. The question presents some obvious links with offline evaluation problems, but a random uniform exploration can lead to some very bad user experience. Here we expect to limit this effect using an initially good policy and to try to use only some small modifications around this policy.

- *What should be the degrees of freedom of such algorithms?* In RL The representation space is crucial and cannot be too large if we want to avoid convergences issues. In RS it is often possible to discretize many of the actions (size of recommendations, number,...). But it is much harder to also include some online optimizations of real valued parameters. One could try approximate reinforcement learning or direct policy search [Busoniu et al., 2011], but the feature constructed by neural network should also be considered.

- When the rewarding process is only partially observable (*i.e.* the observable space contains important variables that influes on the behavior), most of the RL methods fail. When the value of latent variables only depends on the actions, if the dependency is not too long we can brute force the problem by considering sequences of actions rather than single actions. Nevertheless if the order of the corresponding Markov Decision Process is too large, brute force is out of reach. The idea would be to use some transfer between tasks, starting by

easy ones and adding layers of complexity as in a tutorial.

These lines of work will probably require online access or enough data to perform some offline evaluation of the optimization strategy. Hopefully we have this access thanks to some industrial collaborations as the CIFRE PhD thesis of Romain Warlop and with the collaboration with Nuukik and Orange. The mid/long term goal is to build end-to-end systems for recommender systems. A key element of the success of deep learning is the increased availability for computational power and data but this is not enough in an evolving environment.

Bibliography

Abbasi-yadkori, Y., Pal, D., and Szepesvari, C. (2011). Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems 24th (NIPS)*, pages 2312–2320.

Abe, N. and Nakamura, A. (1999). Learning to Optimally Schedule Internet Banner Advertisements. In *Proc. of the 16th International Conference on Machine Learning (ICML)*, pages 12–21, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Abernethy, J., Bach, F., Evgeniou, T., and Vert, J.-P. (2006). Low-rank matrix factorization with attributes. *Inria tech Report*.

Adams, M. and Nobel, A. B. (2012). Uniform approximation of Vapnik-Chervonenkis classes. *Bernoulli*, pages 1310–1319.

all, G. S. . (2012). Kdd cup 2012. http://www.kddcup2012.org/.

all, O. C. . (2014). Kaggle criteo ad display challenge. http://www.kaggle.com/c/criteo-display-ad-challenge.

Audibert, J.-Y., Munos, R., and Szepesvári, C. (2009). Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theor. Comput. Sci.*, 410(19):1876–1902.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256.

Balcan, F., Bansal, N., Beygelzimer, A., D. Coppersmith, J. L., and Sorkin, G. (2007). Robust reductions from ranking to classification. *Lecture Notes in Computer Science*, 4539:604–619.

Bobadilla, J., Ortega, F., Hernando, A., and GutiéRrez, A. (2013). Recommender systems survey. *Know.-Based Syst.*, 46:109–132.

Bottou, L. (2012). Stochastic gradient tricks. In Montavon, G., Orr, G. B., and Müller, K.-R., editors, *Neural Networks, Tricks of the Trade, Reloaded*, Lecture Notes in Computer Science (LNCS 7700), pages 430–445. Springer.

Bottou, L., Peters, J., Quiñonero Candela, J., Charles, D. X., Chickering, D. M., Portugualy, E., Ray, D., Simard, P., and Snelson, E. (2012). Counterfactual reasoning and learning systems. Technical report, arXiv:1209.2355.

Boullé, M. (2006). Modl: A bayes optimal discretization method for continuous attributes. *Machine Learning*, 65(1):131–165.

Burges, C. J. (2010). From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82, Microsoft Research.

Busoniu, L., Ernst, D., De Schutter, B., and Babuska, R. (2011). Approximate reinforcement learning: An overview. In *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*, pages 1–8.

Cai, J.-F., Candès, E. J., and Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM J. on Optimization*, 20(4):1956–1982.

Candès, E. J. and Recht, B. (2009). Exact matrix completion via convex optimization. *Found. Comput. Math.*, 9(6):717–772.

Candes, E. J. and Tao, T. (2005). Decoding by linear programming. *IEEE Trans. Inf. Theor.*, 51(12):4203–4215.

Candès, E. J. and Tao, T. (2010). The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Inf. Theor.*, 56(5):2053–2080.

Chakrabarti, D., Kumar, R., Radlinski, F., and Upfal, E. (2008). Mortal Multi-Armed Bandits. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *20^{th} Advances in Neural Information Processing Systems (NIPS)*, pages 273–280. MIT Press.

Chapelle, O. and Li, L. (2011). An empirical evaluation of thompson sampling. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F. C. N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24: Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NIPS-2011), Granada, Spain.*, pages 2249–2257. Curran Associates, Inc.

Chu, W. and Ghahramani, Z. (2009). Probabilistic models for incomplete multi-dimensional arrays. In Dyk, D. V. and Welling, M., editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS-09)*, volume 5, pages 89–96. Journal of Machine Learning Research - Proceedings Track.

Ding, C., Li, T., Peng, W., and Park, H. (2006). Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 126–135, New York, NY, USA. ACM.

Duchi, J., Hazan, E., and Singer, Y. (2010). Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley.

Fazel, M. (2002). *Matrix Rank Minimization with Applications*. PhD thesis, Stanford University.

Flexer, A., Schnitzer, D., and Schlüter, J. (2012). A mirex metaanalysis of hubness in audio music similarity. In *In Proc. of the 13th Int. Soc. for Music Information Retrieval Conf. (ISMIR.*

Fournie, E. (1992). Un Test de type Kolmogorov-smirnov pour processus de diffusion ergodiques. Research Report RR-1696, Inria.

Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110.

Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

Frieze, A. M., Kannan, R., and Vempala, S. (1998). Fast monte-carlo algorithms for finding low-rank approximations. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 370–378. IEEE Computer Society.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Mach. Learn.*, 63(1):3–42.

Girgin, S., Mary, J., Preux, P., and Nicol, O. (2011). Managing advertising campaigns - an approximate planning approach. In *Frontiers of Computer Science*.

Graepel, T., Candela, J. Q., Borchert, T., and Herbrich, R. (2010). Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoftâs bing search engine. In *Proceedings of the 27th International Conference on Machine Learning ICML 2010, Invited Applications Track (unreviewed, to appear)*. Invited Applications Track.

Granmo, O.-C. (2008). A Bayesian Learning Automaton for Solving Two-Armed Bernoulli Bandit Problems. In *Proc. of the 7^{th} Internaitonal Conference on Machine Learning and Applications (ICML-A)*, pages 23–30. IEEE Computer Society.

Gray, R. M. (1990). *Probability, Random Processes, and Ergodic Properties*. Springer.

Guillou, F., Gaudel, R., Mary, J., and Preux, P. (2014). User Engagement as Evaluation: a Ranking or a Regression Problem?

Harshman, R. A. and Lundy, M. E. (1994). PARAFAC: Parallel factor analysis. *Computational Statistics and Data Analysis*, 18:39–72.

Herbrich, R., Minka, T., and Graepel, T. (2007). Trueskill$^{\mathrm{TM}}$: A bayesian skill rating system. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 569–576. MIT Press.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA. IEEE Computer Society.

Jain, P. and Oh, S. (2014). Provable tensor factorization with missing data. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 27*, pages 1431–1439. Curran Associates, Inc.

Jenatton, R., Roux, N. L., Bordes, A., and Obozinski, G. R. (2012). A latent factor model for highly multi-relational data. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 3167–3175. Curran Associates, Inc.

Jose, C., Goyal, P., Aggrwal, P., and Varma, M. (2013). Local deep kernel learning for efficient non-linear svm prediction. In *Proceedings of the International Conference on Machine Learning*.

Kakade, S. M., Shalev-Shwartz, S., and Tewari, A. (2008). Efficient Bandit Algorithms for Online Multiclass Prediction. In *Proc. of the 25th International Conference on Machine Learning (ICML)*, pages 440–447, New York, NY, USA. ACM.

Kannan, R. and Vempala, S. (2009). Spectral algorithms. *Foundations and Trends in Theoretical Computer Science*, 4(3-4):157–288.

Kantorovich, L. V. and Rubinstein, G. S. (1957). On a function space in certain extremal problems. *Dokl. Akad. Nauk USSR*, 115:1058–1061.

Karayev, S., Hertzmann, A., Winnemoeller, H., Agarwala, A., and Darrell, T. (2013). Recognizing image style. *CoRR*, abs/1311.3715.

Khaleghi, A., Ryabko, D., Mary, J., and Preux, P. (2012). Online Clustering of Processes. In *AISTATS 2012*, volume 22 of *JMLR W&CP*, pages 601–609, La Palma, Spain.

Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 180–191. VLDB Endowment.

Kolmogorov, A. N. (1933). Sulla determinazione empirica di una legge di distribuzione. *Giornale dell'Istituto Italiano degli Attuari*, 4(1):83–91.

Kuleshov, V., Chaganty, A., and Liang, P. (2015). Tensor factorization via matrix factorization. In *Artificial Intelligence and Statistics (AISTATS)*.

Langford, J., Oliveira, R., and Zadrozny, B. (2006). Predicting conditional quantiles via reduction to classification. In *UAI '06, Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence, Cambridge, MA, USA, July 13-16, 2006*. AUAI Press.

Langford, J., Strehl, A., and Wortman, J. (2008). Exploration scavenging. In *International Conference on Machine Learning (ICML)*.

Langford, J. and Zhang, T. (2008). The Epoch-Greedy Algorithm for Multi-armed Bandits with Side Information. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *20th Advances in Neural Information Processing Systems (NIPS)*, pages 817–824. MIT Press.

Lathauwer, L. D., Moor, B. D., and Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278.

Lathia, N., Hailes, S., Capra, L., and Amatriain, X. (2010). Temporal diversity in recommender systems. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 210–217, New York, NY, USA. ACM.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, B., Yang, Q., and Xue, X. (2009). Can movies and books collaborate?: Cross-domain collaborative filtering for sparsity reduction. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 2052–2057, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proc. of the 19th international conference on World wide web (WWW)*, pages 661–670, New York, NY, USA. ACM.

Li, L., Chu, W., Langford, J., and Wang, X. (2011). Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proc. Web Search and Data Mining (WSDM)*, pages 297–306. ACM.

Li, L., Munos, R., and Szepesvári, C. (2015). Toward minimax off-policy value estimation. In Lebanon, G. and Vishwanathan, S. V. N., editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, volume 38 of *JMLR Proceedings*. JMLR.org.

Mahdian, M. and Nazerzadeh, H. (2007). Allocating Online Advertisement Space with Unreliable Estimates. In *ACM Conference on Electronic Commerce*, pages 288–294.

Mary, J., Garivier, A., Li, L., Munos, R., Nicol, O., Ortner, R., and Preux, P. (2012). Icml exploration and exploitation 3 - new challenges.

Mary, J., Gaudel, R., and Preux, P. (2014a). Bandits Warm-up Cold Recommender Systems. Research Report RR-8563, INRIA Lille ; INRIA.

Mary, J., Nicol, O., and Preux, P. (2014b). Improving offline evaluation of contextual bandit algorithms via bootstrapping techniques. In *Proc. ICML, JMLR WCP, Beijing - China*.

May, B. C., Korda, N., Lee, A., and Leslie, D. S. (2012). Optimistic bayesian sampling in contextual-bandit problems. *J. Mach. Learn. Res.*, 13:2069–2106.

Mazumder, R., Hastie, T., and Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *J. Mach. Learn. Res.*, 11:2287–2322.

Mehta, A., Saberi, A., Vazirani, U., and Vazirani, V. (2005). Adwords and Generalized Online Matching. In *Proc. of the 46^{th} Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 264–273. IEEE Computer Society.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Moreno, O., Shapira, B., Rokach, L., and Shani, G. (2012). Talmud: Transfer learning for multiple domains. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 425–434, New York, NY, USA. ACM.

Nesterov, Y. (2004). *Introductory lectures on convex optimization : a basic course*. Applied optimization. Kluwer Academic Publ., Boston, Dordrecht, London.

Nicol, O. (2014). *Data-driven evaluation of Contextual Bandit algorithms and applications to Dynamic Recommendation*. PhD thesis, Université Lille 1, Cité Scientifique, Villeneuve d'Ascq, France.

Nicol, O., Mary, J., and Preux, P. (2012). Icml exploration and exploitation challenge: Keep it simple ! In *Journal of Machine Learning Research (JMLR)*.

Pan, W., Xiang, E. W., Liu, N. N., and 0001, Q. Y. (2010). Transfer learning in collaborative filtering for sparsity reduction. In Fox, M. and Poole, D., editors, *AAAI*. AAAI Press.

Pandey, S., Agarwal, D., Chakrabarti, D., and Josifovski, V. (2007). Bandits for Taxonomies: A Model-based A pproach. In *Proc. of the* 7^{th} *SIAM International Conference on Data Mining*.

Pandey, S. and Olston, C. (2006). Handling Advertisements of Unknown Quality in Search Advertising. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *18*th *Advances in Neural Information Processing Systems (NIPS)*, pages 1065–1072. MIT Press.

Precup, D., Sutton, R. S., and Singh, S. P. (2000). Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 759–766, San Francisco, CA, USA. Morgan Kauffman.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (2009). *Dataset Shift in Machine Learning*. The MIT Press.

Rendle, S. (2010). Factorization machines. In Webb, G. I., Liu, B., Zhang, C., Gunopulos, D., and Wu, X., editors, *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 995–1000. IEEE Computer Society.

Rendle, S. (2012). Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22.

Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2010). Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 811–820, New York, NY, USA. ACM.

Research, Y. (2012). R6b - yahoo! front page today module user click log dataset. `http://webscope.sandbox.yahoo.com/`.

Ryabko, D. (2010a). Clustering processes. In *Proc. the 27th International Conference on Machine Learning (ICML 2010)*, pages 919–926, Haifa, Israel.

Ryabko, D. (2010b). Discrimination between B-processes is impossible. *Journal of Theoretical Probability*, 23(2):565–575.

Ryabko, D. (2011). On the relation between realizable and non-realizable cases of the sequence prediction problem. *Journal of Machine Learning Research*, 12:2161–2180.

Ryabko, D. and Mary, J. (2012). Reducing statistical time-series problems to binary classification. In *Neural Information Processing Systems (NIPS)*.

Ryabko, D. and Mary, J. (2013). A binary-classification-based metric between time-series distributions and its use in statistical and learning problems. *Journal of Machine Learning Research*.

Ryabko, D. and Ryabko, B. (2010). Nonparametric statistical inference for ergodic processes. *IEEE Transactions on Information Theory*, 56(3):1430–1435.

Ryako, B. (1988). Prediction of random sequences and universal coding. *Problems of Information Transmission, 24.*, pages 87–96.

Said, A., Dooms, S., Loni, B., and Tikk, D. (2014a). Recommender systems challenge 2014. In *Proceedings of the eighth ACM conference on Recommender systems*, RecSys '14, New York, NY, USA. ACM.

Said, A., Dooms, S., Loni, B., and Tikk, D. (2014b). Recsys challenge 2014. http://2014.recsyschallenge.com/.

Salakhutdinov, R. and Mnih, A. (2007). Probabilistic matrix factorization. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1257–1264. Curran Associates, Inc.

Schölkopf, B. and Smola, A. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive computation and machine learning. MIT Press.

Serdyukov, P., Dupret, G., and Craswell, N. (2014). Yandex challenge. http://www.kaggle.com/c/yandex-personalized-web-search-challenge.

Shivaswamy, P. and Joachims, T. (2011). Online learning with preference feedback. NIPS workshop on choice models and preference learning.

Solomonoff, R. (1978). Complexity-based induction systems: Comparisons and convergence theorems. *Information Theory, IEEE Transactions on*, 24(4):422–432.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series. Mit Press.

Valko, M., Korda, N., Munos, R., Flaounas, I. N., and Cristianini, N. (2013). Finite-time analysis of kernelised contextual bandits. *CoRR*, abs/1309.6869.

Wang, C.-C., Kulkarni, S., and Poor, H. (2005). Bandit Problems With Side Observations. *IEEE Transactions on Automatic Control*, 50(3):338–355.

Wang, Y., yves Audibert, J., and Munos, R. (2009). Algorithms for infinitely many-armed bandits. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1729–1736. Curran Associates, Inc.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A., and Attenberg, J. (2009). Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1113–1120, New York, NY, USA. ACM.

Zhou, Y., Wilkinson, D., Schreiber, R., and Pan, R. (2008). Large-scale parallel collaborative filtering for the netflix prize. In *Proceedings of the 4th international conference on Algorithmic Aspects in Information and Management (AAIM)*, pages 337–348, Berlin, Heidelberg. Springer-Verlag.

Zolotarev, V. M. (1983). Probability metrics. *j-THEORY-PROBAB-APPL*, 28(2):278–302.

Part II

Annexes

Chapter 5

Offline evaluation of contextual bandits

Offline evaluation of recommendation systems

Abstract

The evaluation of recommendation algorithms is a critical issue. Live evaluation is often avoided due to the potential loss of revenue. Live evaluation on a part of the audience is possible for people having access to a working recommendation system. However, the accuracy of existing offline evaluation is not satisfactory. We precisely address this issue in this paper. After reminding some recent work on the use of contextual bandits to solve exploration/exploitation dilemma in online recommendation, we show how to perform offline evaluation of a recommendation algorithm using a batch of data, give a methodology, an algorithm, as well as investigate the theoretical properties of this evaluation procedure. We demonstrate the efficiency of this procedure experimentally on a large publicly available dataset.

1. Introduction

This paper is about the offline evaluation of recommendation algorithms. Offline evaluation is realized using data collected on a live recommendation system. We wish to be able to evaluate offline the performance of algorithms on such a dataset. We also wish this performance evaluated offline to be close to the performance of the algorithm played live. Our contribution is based on a combination of bandit theory and bootstrap theory. We provide a theoretical analysis of our evaluation method and an experimental assessment of its performance on a large publicly available dataset.

The paper is organized as follows. Sec. 2 introduces and details of the problem we tackle. Readers familiar with the evaluation of recommendation algorithms may skip it. Section 3 provides the necessary background in bandit theory, non contextual bandit algorithms such as UCB, and contextual bandit algorithm such as LinUCB; it also relates bandit problems with the recommendation problem; this section may be skipped by readers acquainted with bandit theory. Readers who skipped sections are invited to board at section 4 which sets the stage: building on previous experimentation, issues related to current method for offline evaluation are detailed. This section closes with the main contribution of this paper which is a new method for offline evaluation of recommendation algorithms we named BRED. Sec. 5 analyzes the accuracy of the estimation of performance provided by BRED which is conveyed in theorem 2. Sec. 6 demonstrates experimentally that BRED is effective and is much more accurate than the current state of the art replay method.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. The evaluation of recommendation algorithms

Under various forms, and under various names, recommendation is a very common activity over the web. One can think of the Netflix challenge type of applications, news systems, Digg, Amazon, online advertising,... The key idea is always to take advantage of a user profile in order to identify the most attractive content in a given context (here, the context contains all the information we have on the user: objective features like the requested url and timestamp as well as subjective features provided by the user: date of birth, gender, ..., and features that may be obtained from various sources of information, such as social networks). Moreover, we want to be able to track changes in user preferences, and quickly adapt to events such as breaking news or product releases. A common issue with this kind of data is the *partial labeling*: the user feedback (click or not on a recommended item) is observed only for the displayed items, and often only for a single item, the one that is clicked. So we face a classical exploration/exploitation dilemma: on the one hand, we want to explore in order to collect precise enough information to be able to perform the best recommendation in all the possible contexts while on the other hand, we want to exploit the collected information in order to maximise the actual revenue on an online system. A natural way to model this situation is as a reinforcement learning problem (Sutton & Barto, 1998), and more precisely using the contextual bandit framework (Lu et al., 2010). For such systems, the best way to compare the performance of two algorithms is to perform A/B testing on a subset of the web audience (Kohavi et al., 2009). Of course, doing this raises a lot of difficulties: high cost because of the engineering effort to do it

live, degradation of the user experience, problems with natural variations of performance over time ("special" periods of time such as pre-Christmas sales). So it is highly desirable to have an accurate *offline* methodology of evaluation of recommendation algorithms.

A lot of evaluation methods for recommendation systems have been proposed (Shani & Gunawardana, 2011). One possibility is to ignore the partial labeling problem. By so doing, the problem turns into a supervised classification task, or a regression task. In this case we do not learn an online policy of recommendation but rather we learn to predict the probability of click of a user, or the most likely outcome (click or not), or a rating. If the world can be assumed to be stationary, it is reasonnable to use a large amount of data collected on the web-server in order to learn the relation between the user and the probability of click on the items. However, for swiftly evolving set of items (such as ads, news, ...) it gets much trickier. We need simpler systems, that are able to learn fast and to be reactive in face of changes.

A second possibility is to simulate an *online* behavior: this requires a "behavior" simulator. But creating such a simulator requires a lot of effort and is hardly reliable in the end. The simulator will not behave like a real user and designing a well working algorithm is usually easy once we know how the simulator works. Then, chances are high that algorithms overfit the simulator, rather than performing well in the real online setting.

The last possibility is to use web-server logs and perform some sort of rejection sampling to obtain an unbiased evaluator of a recommendation policy. The idea has been proposed by (Langford et al., 2008; Li et al., 2011); it is presented later in the paper as the "replay method". A well-known issue with this approach is the usual high variance of the estimator of the performance. Studies have been conducted to understand and to reduce this variance (Strehl et al., 2010; Dudík et al., 2011), or to collect more data when the variance is too high (Bottou et al., 2012). An other less studied issue is the *time acceleration*: due to the rejection mechanism at the core of the evaluation methodology, only a fraction of the data is used to compute the performance of a policy, this fraction being (1/number of possible recommendations at the current time step) if the logged data were acquired by a random uniform policy. As long as only a fraction of the visits is used for evaluation, this is exactly as if we had a smaller number of visits per unit of time, a situation which can be problematic in our non stationary world.

In this paper, we focus on the problem of the offline evaluation of a recommendation policy: our aim is to get a more accurate estimation of the performance of a recommendation policy while minimizing the amount of data required to do so. To achieve this goal, we propose a new evaluation protocol. A theoretical analysis of this and we provide some background material in the next section.

3. Background on bandits

We formalize the news recommendation problem as a contextual bandit problem. Before dealing with contextual bandits, we introduce basic material about the non contextual bandit problems, along with algorithms to solve them.

3.1. (Non contextual) bandits

The bandit problem is also known in the litterature as the multi-arm bandit problem and other variations. This problem can be traced back to Robbins and Munro in 1952 (Robbins, 1952) and even Thompson in 1933 (Thompson, 1933). There are many variations in the definition of the problem: a basic setting is as follows.

Let us consider a bandit machine with K independent arms. At each time step, the player performs one out of K actions: we denote a_j the action of pulling arm j. When performing action a_j, the player receives a reward drawn from $[0,1]$ according to a probability distribution ν_j. Let μ_j denote the mean of ν_j and j^* be the arm with maximum expected reward $\mu^* = \mu_{j^*}$. ν_j, μ_j, j^* and μ^* are unknown. A common objective is to maximize the cumulative reward after T consecutive pulls. More specifically, by denoting j_t the arm pulled at time t and r_t the reward obtained at time t, the player aims at maximizing the quantity $\mathrm{CumRew}_T = \sum_{t=1}^{T} r_t$. At each time-step (except the last one), the player faces the exploration vs. exploitation dilemma. An optimal strategy has to balance exploration and exploitation. A well-known approach to handle this trade-off is the Upper Confidence Bound strategy (UCB) (Auer et al., 2002). At time t, the UCB strategy consists in playing the arm j_t with maximum upper confidence bound on its expectation

$$j_t = \operatorname*{argmax}_j \hat{\mu}_j + \sqrt{\frac{2\ln t}{t_j}}, \qquad (1)$$

where $\hat{\mu}_j$ denotes an estimation of μ_j, and t_j is the number of pulls of arm j since $t=1$.

Let us discuss the behavior of UCB qualitatively. During preliminary time-steps, the value in eq. (1) is dominated by the rightmost term: indeed, for each arm, this term quantifies the uncertainty about the expected reward, and during the first pulls, we have no knowledge at all, hence very large uncertainty. After a while, the leftmost term becomes predominant and the one that influences the most the choice of the arm to pull. Hence, the behavior of UCB is very similar to the strategy `Explore-Exploit` which consists in uniformly exploring the arms during a few steps,

then focussing on the arm with the best empirical mean. On the other hand, UCB never stops exploring seemingly sub-optimal arms.

The strength of UCB lies in the following property: the number of pulls of a sub-optimal arm j is in the order $\frac{\ln T}{(\mu^* - \mu_j)^2}$. Hence, the continuous exploration of arm j only costs a loss of the order $\frac{\ln T}{\mu^* - \mu_j}$. As a corollary to that property, the exploration budget is non-uniformly spread among the set of arms. UCB does not lose time, hence rewards, playing arms which are likely to be non-optimal.

3.2. The recommendation problem as a bandit problem

In the recommendation problem, one has a set of items to recommend (that is a set of arms to pull). In general, each item is associated to a set of features; likewise, when recommendation are made for a particular user who may also be characterized by a set of features. Hence, when a recommendation has to be made, there exists a context composed of both sets of features. When arms have features, the problem becomes a contextual bandit problem. We now briefly introduce the contextual bandit setting.

3.3. Contextual bandits

We follow (Langford & Zhang, 2007) to define the contextual bandit problem.

Let $\mathcal{X}$ be an arbitrary input space and $\mathcal{A} = \{1..K\}$ be a set of K actions. Let $\mathcal{D}$ be a distribution over tuples $(x, \vec{r})$ with $x \in \mathcal{X}$ and $\vec{r} \in \{0,1\}^K$ is a vector of rewards: in the $(x, \vec{r})$ pair, the j^{th} component of $\vec{r}$ if the reward associated to action a_j, that is pulling arm j in the context x. In the recommendation problem, this reward corresponds to whether the item k is clicked or not in a given context.

A contextual bandit problem is an iterated game in which at each round t:

- $(x_t, \vec{r_t})$ is drawn from $\mathcal{D}$.
- x_t is provided to the player.
- The player chooses an action $a_t \in \mathcal{A}$ based on x_t and on the knowledge it gathered from the previous rounds of the game.
- The reward $\vec{r_t}[a_t]$ is revealed to the player whose score is updated.
- The player updates his knowledge based on this new experience.

It is important to note the partial observability of this game: the reward is known only for the action performed by the player, not for others. In the recommendation setting, this simply means that we know whether the displayed item has been clicked or not, but we have no information about any other item.

We define a recommendation algorithm A as taking as input the ordered list of (x, a, r) triplets and output a policy π. A policy π_t maps $\mathcal{X}$ to $\mathcal{A}$, that is chooses an action given a context.

We measure the performance of a recommendation algorithm A with its average score obtained after T rounds, according to the distribution $\mathcal{D}$. We note this quantity $CTR_A(T, \mathcal{D})$. To simplify notations, we will also note that quantity $CTR_A(T)$ by dropping $\mathcal{D}$.

The objective is to design a recommendation algorithm A that maximizes the expectation of the cumulated rewards over T time steps, *i.e.*

$$G_A(T) \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{D}} \sum_{t=1}^{T} \vec{r_t}[A(x_t)]$$

For convenience, we define the per-trial payoff as the average click rate after T time steps:

$$g_A \stackrel{\text{def}}{=} \frac{G_A(T)}{T} = \mathbb{E}_{\mathcal{D}}\left(CTR_A(T)\right)$$

g_A is the quantity we wish to estimate as the measure of performance of a recommendation algorithm.

The contextual bandit problem is theoretically more challenging than the non contextual bandit problem since the kind of performance bound that can be derived is highly dependent on the nature of $\mathcal{X}$. However one very popular contextual bandit algorithm is LinUCB (Li et al., 2010) which can be considered as optimal under some very specific assumptions on $\mathcal{X}$. We can also mention one of the first works on the topic: the epoch-greedy algorithm (Langford & Zhang, 2007) that is similar to ε-greedy with an ε decreasing along time.

4. Exhibiting time acceleration

4.1. Lessons drawn from the ICML E&E Challenge 3

There has been a series of ICML challenges about Exploration & Exploitation. The latest is detailed in the supplementary material. The main conlusions drawn from the ICML Exploration & Exploitation Challenge held in 2012 have been:

- Non-contextual algorithms such as UCB or ε-greedy performed surprisingly well. The very best algorithm was a UCB-V with normality asumption (Audibert et al., 2009) .

- LinUCB — which had been tested using the same methodology on the same kind of data from Yahoo! (Li et al., 2010) — performed surprisingly bad, whichever effort was done to tune the exploration parameter. Adding a regularization term improved its performance but not enough to catch up with UCB-V.

The intuitive reason is actually very simple: as everyone knows, the real world is not stationary, really not! In particular, news are aging and being constantly replaced by new ones; each news only lives a fraction of a day. So, basically, non contextual algorithms have to learn whether a news is likely to be clicked or not whereas a contextual algorithm has to learn if a combination (news, user profile) is likely to produce a click or not. Clearly in the second case, the space to learn from is much larger than in the first case; to learn in this much larger space requires much longer spans of time, spans of time that are longer than the actual lifespan of news.

Algorithm 1 *Replay method* (Langford et al., 2008; Li et al., 2011).
Remark: for the sake of the precision of the specification of the algorithm, we use a history h which is the list of triplets (x, a, r) that have yet been used to estimate the performance of the algorithm A. The goal is to avoid hiding internal information maintenance in A; a real implementation may be significantly different for the sake of efficiency, by learning incrementally.

Input:

- A contextual bandit algorithm A
- A set S of L triplets (x, a, r)

Output: An estimate of g_A

$h \leftarrow \emptyset$
$\widehat{G}_A \leftarrow 0$
$T \leftarrow 0$
for $t \in \{1..L\}$ **do**
  Get the t-th element (x, a, r) of S
  $\pi \leftarrow A(h)$
  if $\pi(x) = a$ **then**
    add (x, a, r) to h
    $\widehat{G}_A \leftarrow \widehat{G}_A + r$
    $T \leftarrow T + 1$
  end if
end for
return $\frac{\widehat{G}_A}{T}$

There is a more subtle reason that makes thing much worse. This is related to the evaluation protocol which resulted from a lot of smart thinking, but unfortunately merely yield such deceptive conclusions. Basically, to estimate the performance of a recommendation algorithm, the replay method uses the dataset $\mathcal{S}$ and iteratively selects one data, queries the algorithm to get its recommendation for this context and checks whether this recommendation matches the logged one. In case of a match, the algorithm may learn from this data, and its score is accordingly updated (whether a click was observed or not). Otherwise, the data is simply discarded. When there are K items to choose from, the probability that the chosen action and the action in $\mathcal{S}$ do not match is $\frac{K-1}{K}$, which is then the probability that the data is discarded, and is not used for the evaluation of the recommendation algorithm. Thus only a small fraction of the collected data are actually used for the evaluation. This is as if time was accelerated, using only 1 data out of K, hence the "time acceleration" phenomenon.

4.2. Intuition of the time acceleration

While the evaluation of $CTR_A(T)$ is meant over T time steps, the replay method tends to really evaluate $CTR_A(T/K)$ because of time acceleration. As long as T is large enough, these two values are the same since their estimators have converged. However, when T is smaller, the estimator obtained by the replay method obviously converges slower so that after T rounds, the estimated quantity is really $CTR_A(T/K)$ rather than $CTR_A(T)$. This fact has severe consequences for learning algorithms: they basically learn with K times less data than expected. So, to be fair, one should use K times more data to evaluate a recommendation algorithm with the replay method. This has a cost obviously, and this is simply not possible in a changing environment: one can not slow down the pace of time.

The situation gets even worse for contextual learning algorithms[1]. Indeed, in this situation (illustrated in example 3 of (Li et al., 2011)), the replay method can not even converge to the correct CTR estimate.

4.3. Offline evaluation with BRED

Now that the shortcoming of the replay method has been understood, we look for an other offline evaluation protocol that does not suffer from the time acceleration issue. The idea we propose stems from the idea of bootstrap.

Let us remind the standard bootstrap approach and apply it to the present situation. The general idea of bootstrap is to build B datasets $\mathcal{S}_{b\in\{1,...B\}}$ each of size L out of a single dataset $\mathcal{S}$ of size L by drawing L examples randomly with replacement. In each of the resulting datasets $\mathcal{S}_b$, there are data that occur more than once, and other data that do not appear at all. Then the $CTR_A(T)$ of the recommendation

[1] Please, note that the proof of unbiased convergence of the replay method only holds for algorithms producing fixed policies, that is not adaptive algorithms

algorithm A is estimated on each of the B datasets. Each of the B evaluations provides an estimator of $CTR_A^{(b)}(T)$. The estimated $CTR_A(T)$ is simply the average of these B estimates. From a theoretical point of view, and under mild assumptions, the bootstrap estimator converges with no bias to the actual error rate at a speed in $O(1/L)$.

The core idea of the evaluation protocol we propose in this paper is inspired by the bootstrap, and somehow inherits its theoretical properties.

Algorithm 2 *Bootstrapped Replay on Expanded Data BRED.*

We sketch this algorithm so that it looks very much like the replay method in Alg. 1. The same remark may be done regarding the histories $h^{(b)}$.

Input

- A (contextual) bandit algorithm A
- A set $\mathcal{S}$ of L triplets (x, a, r)
- An integer B

Output: An estimate of g_A

$h^{(b)} \leftarrow \emptyset, \forall b \in \{1..B\}$ */*empty history*/*
$\widehat{G}_A^{(b)} \leftarrow 0, \forall b \in \{1..B\}$
$T^{(b)} \leftarrow 0, \forall b \in \{1..B\}$
/ Bootstrap loop*/*
for $b \in \{1..B\}$ **do**
  / estimation of* $CTR_A^{(b)}(T)$ **/*
  for $i \in \{1..L \times K\}$ **do**
    Sample with replacement (x, a, r) of S
    $x \leftarrow \text{JITTER}(x)$ */*optional*/*
    $\pi \leftarrow A(h^{(b)})$
    if $\pi(x) = a$ **then**
      add (x, a, r) to $h^{(b)}$
      $\widehat{G}_A^{(b)} \leftarrow \widehat{G}_A^{(b)} + r$
      $T^{(b)} \leftarrow T^{(b)} + 1$
    end if
  end for
end for
return $\frac{1}{B} \sum_{b=1}^{B} \frac{\widehat{G}_A^{(b)}}{T^{(b)}}$

From a dataset of size L with K possibles choices at each time step, we generate B datasets of size $K \times L$ in the same way as bootstrap does. This implies that each data of the initial set of data appears K times on average. In order to avoid the duplication of any data, we may introduce a random perturbation to the context, a technique known as jittering. Then we use the replay method (Alg. 1) to evaluate the $CTR_A^{(b)}$'s on each of the datasets $\mathcal{S}_b$. As explained above, the replay method evaluates the CTR of an algorithm using $1/K$ of the data on average. So on any of the $\mathcal{S}_b$ datasets, the algorithm is evaluated using $K \times L/K = L$ data on average. The algorithm is thus evaluated on L data, and performs L learning steps. Hence, the bias is reduced for contextual algorithms. Each of the dataset $\mathcal{S}_b$ is used to produce an estimate of $CTR_A(L)$. The complete algorithm is summarized in Alg. 2. We call this method: "Bootstrapped Replay on Expanded Data", or BRED for short.

This BRED approach may seem very unlikely to succeed, and prone to overfitting. However, its close relationship with the celebrated bootstrap method in statistics yields a straightforward theoretical analysis, showing that the estimated CTR_A is unbiased, and converging rather quickly to its true value. In particular, it converges much faster than the estimator provided by the replay method. The next section provides the theoretical analysis of BRED.

5. Theoretical analysis

In this section, we make a theoretical analysis of our evaluation method BRED. The core loop in BRED is a bootstrap loop; henceforth, to complete this analysis, we first restate the theorem 1 which is a standard result of the bootstrap asymptotic analysis (Kleiner et al., 2012).

Bootstrap is used to estimate an unbiased expected value of a random variable based on a sample of data. Bootstrap computes the empirical value on a set of B samples of the dataset. Each sample b is obtained by sampling with replacement the dataset. Here, the quantity we wishes to estimate is $\mathbb{E}(CTR_A(T))$, that is the average performance of an algorithm A after T recommendations. Each of the B bootstrap step estimates a realization of $CTR_A(T)$. Their mean estimates $\mathbb{E}(CTR_A(T))$. The estimation of $CTR_A^{(b)}$ simulates $T^{(b)}$ time steps (this is a random variable when A is stochastic).

Theorem 1. *Suppose that:*

- *A is a recommendation algorithm which generates a fixed policy fixed over time (this hypothesis can be weakened as discussed in remark 2),*
- *K items may be recommended at each time step,*
- *$CTR_A(T)$ admits an expansion as an asymptotic series*

$$CTR_A(T) = z + \frac{p_1}{\sqrt{T}} + \ldots + \frac{p_\alpha}{T^{\alpha/2}} + o\left(\frac{1}{T^{\alpha/2}}\right)$$

where z is a constant independent of the distribution $\mathcal{D}$ (as defined in Sec. 3.3), and the p_i are polynomials in the moments of the distribution of $CTR_A(T)$ under $\mathcal{D}$ (this hypothesis is discussed and explained in remark 1),

- *For any b, the empirical estimator* $\widehat{CTR}_A^{(b)}(T^{(b)})$ *admits a similar expansion:*

$$z + \frac{\widehat{p}_1^{(b)}}{\sqrt{T^{(b)}}} + \ldots + \frac{\widehat{p}_\alpha^{(b)}}{(T^{(b)})^{\alpha/2}} + o\left(\frac{1}{(T^{(b)})^{\alpha/2}}\right). \tag{2}$$

Then, for $L \leq T^{(b)} \times K$ *and assuming finite first and second moments of* $CTR_A^{(b)}(T^{(b)})$*, with high probability:*

$$\left|\frac{1}{B}\sum_{b=1}^{B} CTR_A(T^{(b)}) - \widehat{CTR}_A^{(b)}(T^{(b)})\right| = O\left(\frac{Var(\widehat{p}_\alpha^{(1)} - p_\alpha | D_L)}{\sqrt{T^{(b)} \cdot B}}\right) + O\left(\frac{1}{T^{(b)}}\right) + O\left(\frac{1}{L\sqrt{T^{(b)}}}\right) \tag{3}$$

where $\mathcal{D}_L$ *is the resampled distribution of* $\mathcal{D}$ *using* L *realizations.*

Proof. it is actually a straightforward adaptation of the proof of theorem 3 of (Kleiner et al., 2012). Indeed, we only have to take care of L. Also note that this theorem is a reformulation of the bootstrap main convergence result as introduced by (Efron, 1979). □

Now, we use theorem 1 to bound the error made by BRED in the theorem 2.

Theorem 2. *Assuming that*

$$CTR_A(T) = z + \frac{p_1}{\sqrt{T}} + \ldots + \frac{p_\alpha}{T^{\alpha/2}} + o\left(\frac{1}{T^{\alpha/2}}\right)$$

Then for a algorithm A *producing a fixed policy over time, BRED applied on a dataset of size* L *evaluates the expectation of the* CTR_A *with no bias and with high probability for* B *and* L *large enough:*

$$\left|\frac{1}{B}\sum_{b=1}^{B} CTR_A(T^{(b)}) - \widehat{CTR}_A^{(b)}(T^{(b)})\right| = O\left(\frac{1}{L}\right)$$

This means that the convergence of the estimator of $CTR_A(L)$ *is much faster than the convergence of* g_A *(which is in* $O(1/\sqrt{L})$*. This will allow us to make a very good estimator of* g_A *by replacing* $CTR_A(L)$ *by our estimate.*

The sketch of the proof of theorem 2 is the following: first we prove that the replay strategy is able to estimate the moments of the distribution of CTR_A fast enough with respect to the size of $\mathcal{S}$. The second step consists in using classical results from bootstrap theory to guarantee the unbiased convergence of the average $\widehat{CTR}_A(T^{(b)})$ to the true distribution with an $O(\frac{1}{T^{(b)}})$ speed. The rational behind this is that the gap introduced by the subsampling averaging will be of the order of $O(\frac{1}{\sqrt{T^{(b)}B}})$. The proof of 2 is available in the supplementary material.

This work being meant to be of practical use, we make a series of remarks about this theorem. These remarks should be considered as conveying important qualitative understanding of the theorem and thus, properties of the proposed algorithm, BRED.

Remark 1: The key point of the theorems is the existence of an asymptotic expansion of $CTR_A(T^{(b)})$ in polynoms of $1/\sqrt{L}$. This is a natural hypothesis for $CTR_A(T^{(b)})$ because the CTR is an average of bounded variables (probabilities of click). For a recommendation algorithm A producing a fixed policy, the mean is going to concentrate according to the central limit theorem (CLT).

Remark 2 Let us consider algorithms that produce a policy which changes along time (a learning algorithm or an algorithm which has some parameters which vary along time). After a sufficient amount of recommendations, such algorithm which is reasonable enough will produce a policy that will not change any longer (if the world is stationary). Thus again, the CLT will apply and we will observe a convergence of the $CTR_A(L)$ to its limit in $1/\sqrt{L}$. Of course if the algorithm does not converges to a fixed policy — or can converge to policies with different CTR — then all the guarantees are lost. Section 6 shows that empirically this point is not a problem.

Remark 3: BRED can be adapted to the case in which data have not been collected uniformly but following a known policy $\pi_{collect}$. This can be done by renormalization. This would lead to a dependency of the confidence intervals on the algorithm A. Indeed, if A tends to play often like the collecting strategy $\pi_{collect}$, the confidence interval around $\mathbb{E}(CTR_A(T))$ will be tighter than if the policy produced by A is very different from $\pi_{collect}$. We can easily deal with this situation by using the fact that BRED provides more than an estimate of $CTR_A(L)$: BRED also provides an estimation of its distribution, so that we do have confidence intervals.

Remark 4: The proof holds thanks to the assumption that the evaluated recommendation algorithm A produces a policy that is fixed along time. As such, this theorem does not hold for general bandit algorithms. (Li et al., 2011) provides an example showing that a similar proof is impossible in the general case. This is due to the fact that we lose the independence between the choices of A because of the context, and the Chernoff's bounds no longer applies. This undesired behavior is likely to appear because the same example is presented several times to the algorithm, leading

to strong risks of overfitting. That is why we have to introduce some kind of independence between the data. This is done using the JITTER function.

In the neural network field, the technique of jittering consists in deliberately adding artificial noise to the inputs (the attributes of data) during training[2]. Jittering has been intensively studied in this field (Koistinen & Holmström, 1992; Bishop, 1995). It is strongly related to kernel regression and regularization methods. Training with jitter is an approximation to training with the kernel regression estimator as target (Scott, 1992) — the amount of noise is the bandwidth of the kernel. Empirically in such a context, jitter works because the associations to learn are usually smooth. This means that for two similar users, the objective function makes similar choices. This point is clear with LinUCB which makes a linear combination of the features to infer the appeal of an item to a user. As long as the jitter is small, it does not change too much the objective function and it provides "new" examples for free (noisy versions of the examples contained in the training set). Obviously, if the jitter is too strong, the noise is predominant, and hides the information contained in the dataset. Also note than jitter is a convenient way to avoid overfitting between the recommendation algorithm and the context. Indeed, as each tuple (context,action,reward) of the dataset is presented several times to the recommendation, it is possible to design a recommendation algorithm that tries to take an unfair advantage of this behavior (assuming the recommendation algorithm has been designed knowing the evaluation methodology, and taking advantage of it instead of trying only to solve the live recommendation system problem). By adding some noise to the context, we avoid this unwanted behavior by making each example unique. While the theoretical analysis of the algorithm does not hold in this setting, we can think of jittering as a way to make examples more independent from one another; this helps the Chernoff bound being respected even if the independence assumption is still false — In fact Chernoff-like bounds are true even for non identically distributed variables if these variables are bounded and independent (Levchenko, 2005).

6. Experiments in realistic settings

As we proved that BRED has promising theoretical guarantees in the setting introduced in (Li et al., 2011), let us now compare its empirical performance to that of the replay method

[2]Please, keep in mind that a typical neural network iterates several times over the whole training set, a procedure looking like the bootstrap iterations in BRED.

6.1. Synthetic data and discussing Jittering

The first set of experiments was run on synthetic data. Indeed, we needed to be able to compare the errors of estimation of the two methods on various fixed size datasets relatively to the ground truth: an evaluation against the model itself.

Before going any further, let us describe the model we used. It is a linear model with Gaussian noise (as in (Li et al., 2010)) and was built as follows:

- a fixed action set (or news set) of size $K = 10$.
- The context space $\mathcal{X}$ has F dimensions (with $F = 15$ here). A given context is a real valued vector x of dimension F such that $x = c + n$ with the informative part c sampled from $\mathcal{N}(0, 1)$ and the noise n sampled from $\mathcal{N}(0, \frac{1}{2})$.
- Each news i has two hidden components: a real number q_i which characterizes its overall quality and a real valued vector w_i of length F which characterizes its affinity with $\mathcal{X}$.
- The CTR of a news i displayed in a context x is given linearly by $q_i + w_i^T x$.
- Finally there are two kinds of news:
  - news that are interesting in general like *Obama is re-elected* for which q_i is high (sampled from $\mathcal{U}(0.4, 0.5)$) and $w_i = 0_F$
  - specific news like *New Linux distribution released* for which q_i is low (sampled from $\mathcal{U}(0.1, 0.2)$) and w_i is composed of mostly zeros and 1,2 or 3 relevant weights sampled from $\mathcal{N}(0, \frac{1}{5})$.

A reasonable but coarse approach consists in always selecting a news which is appealing to anyone for which a non-contextual algorithm such as UCB is enough. A finer algorithm like LinUCB (contextual) could perform better in the long run by trying to learn how to efficiently display specific news.

Figure 1 display the results and interpretation of experiments which consisted in evaluating both UCB and LinUCB using the different methods. It is clear that BRED converges much faster than the replay method.

Remark 5: as it can be seen on Figure 1, jittering is very important to obtain good performance when evaluating a learning algorithm. Empirically, a good choice for the level of jitter seems to be a function in $O\left(\frac{1}{\sqrt{L}}\right)$, with L the size of the dataset. This confirms our intuition and what is shown by the *Jitter-free* curve: jittering is very important

when the dataset is small but gets less and less necessary as the dataset grows.

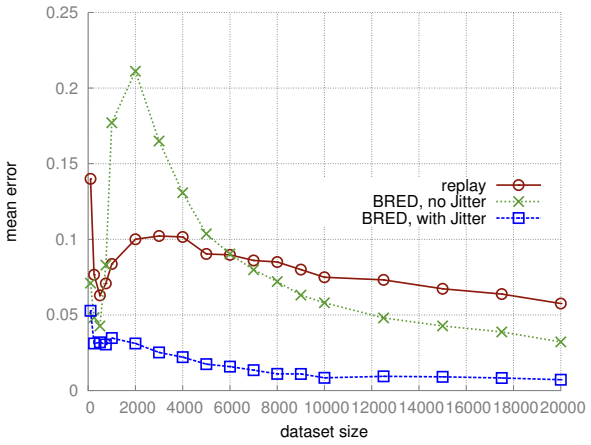


Figure 1. Mean of the absolute value of the difference between the true CTR of a LinUCB and the estimated one for different methodologies. Conducted on artificial dataset as described by section 6.1. The lower, the better. Jittering is actually efficient to avoid overfitting issues. The rather small error rates for very small datasets are due to the fact that on too small datasets all the recommendation algorithm tend to make random choices which are not very hard to evaluate.

6.2. Real data

On a real dataset, it is not reasonable to assume the world to be static. Moreover, as shown in (Agarwal et al., 2009), the click probability of a given news depends on the hour of the day and of the elapsed time since its release. A possible approach is to build a model of this evolution and to use it in oder to correct the bias from the CTR. But on really large datasets (such as the Yahoo! news dataset), it is possible to split the data in small batches, that is small periods of time during which the world may assumed to be static. Of course the recommendation algorithm is not reset at each batch; it has to be able to handle appearances and disappearances of news. This adaptation is fairly easy for bandit algorithms: we simply remove from the history h each pull of an arm corresponding to an outdated news, and we add new news as an unplayed arm. This will result in a strong exploration of new ones.

Remark 6: the adaption of BRED to this non static setting is also straightforward. It is enough to run it on the whole dataset of size L. The convergence results hold as long as the length of the batch where the static assumption holds grows linearly in L.

Finally we ran the very same experiment as one that was run by (Li et al., 2011): they measured the error of the estimated CTR of UCB ($\alpha = 1$) by the replay method on datasets of various sizes relatively to what they call the ground truth: an evaluation of the same algorithm on a real fraction of the audience. As we obviously cannot do that, we used a simple trick: we divided the Yahoo! Today dataset into the minimum number of batches such that within a batch, the action set is static. For each batch, we computed a ground truth by averaging the estimated CTR of the algorithm using the replay method on a number of random permutations of the data of the batch. Note that the CTR of an algorithm estimated via the replay method on a dataset of size L is an unbiased estimate of the CTR of this algorithm over L/K real time steps (K being the size of the action set). This is why for each batch, we subsampled L/K events and evaluated the algorithm using the replay method and BRED on this smaller dataset — this means that reported points for the replay method uses an average of L/K^2. In this dataset $K = 30$ and we used $B = 10$.

7. Conclusion and perspectives

In this paper, we studied the problem of recommendation system evaluation, sticking to a realistic setting: we focused on obtaining a methodology for practical offline evaluation, providing a good estimate using a reasonable amount of data. Previous methods are prooved to be asymptotically unbiased with a low speed of convergence on a static dataset, but yield counter-intuitive estimates of performance on real datasets. Here, we introduce BRED, a method with a much faster speed of convergence on static datasets (at the cost of loosing unbiasedness) which allows it to be much more accurate on dynamic data. Experiments demonstrated our point; they were performed on a publicly available dataset made from Yahoo! server logs and on synthetic data presenting the time acceleration issue. This paper was also meant to highlight the time acceleration issue and the misleading results given by a careless evaluation of an algorithm.

A possible extension of this work is to use BRED to build a "safe controller". Indeed, when a company uses a recommendation system that behaves according to a certain policy π that reaches a certain level of performance, the hope is that when changing the recommendation algorithm, the performance will not be drop. As an extension of the work presented here, it is possible to collect some data using the current policy π, compute small variations of π with tight confidence intervals over their CTR and then replace the current policy π with the improved one. This may be seen as a kind of "gradient" ascent of the CTR in the space of policies.

An other important extension for practical use is to handle the situations where we are allowed to present $k > 1$ items to the user. This can be done using the variance reduction methodology presented in (Langford et al., 2008).

References

Agarwal, Deepak, Chen, Bee-Chung, and Elango, Pradheep. Spatio-temporal models for estimating click-through rate. In *Proceedings of the 18th international conference on World wide web(WWW)*, pp. 21–30, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4. doi: 10.1145/1526709.1526713.

Audibert, Jean-Yves, Munos, Rémi, and Szepesvári, Csaba. Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theor. Comput. Sci.*, 410(19):1876–1902, April 2009. ISSN 0304-3975. doi: 10.1016/j.tcs.2009.01.016.

Auer, Peter, Cesa-Bianchi, Nicolò, and Fischer, Paul. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, May 2002. ISSN 0885-6125.

Bishop, C.M. *Neural Networks for Pattern Recognition*. Neural Networks for Pattern Recognition. Oxford University Press, Incorporated, 1995. ISBN 9780198538646.

Bottou, Léon, Peters, Jonas, Quiñonero Candela, Joaquin, Charles, Denis X., Chickering, D. Max, Portugualy, Elon, Ray, Dipankar, Simard, Patrice, and Snelson, Ed. Counterfactual reasoning and learning systems. Technical report, arXiv:1209.2355, September 2012.

Dudík, Miroslav, Langford, John, and Li, Lihong. Doubly robust policy evaluation and learning. *CoRR*, abs/1103.4601, 2011.

Efron, B. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. ISSN 00905364. doi: 10.2307/2958830.

Kleiner, Ariel, Talwalkar, Ameet, Sarkar, Purnamrita, and Jordan, Michael. The big data bootstrap. In Langford, John and Pineau, Joelle (eds.), *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pp. 1759–1766, New York, NY, USA, July 2012. Omnipress. ISBN 978-1-4503-1285-1.

Kohavi, Ron, Longbotham, Roger, Sommerfield, Dan, and Henne, Randal M. Controlled experiments on the web: Survey and practical guide. *Journal of Data Mining and Knowledge Discovery*, 18:140–181, 2009.

Koistinen, Petri and Holmström, Lasse. Kernel regression and backpropagation training with noise. In Moody, John E., Hanson, Steve J., and Lippmann, Richard P. (eds.), *Advances in Neural Information Processing Systems 4*, pp. 1033–1039. San Francisco, CA: Morgan Kaufmann, 1992.

Langford, John and Zhang, Tong. The epoch-greedy algorithm for multi-armed bandits with side information. In *Proc. NIPS*, 2007.

Langford, John, Strehl, Alexander, and Wortman, Jennifer. Exploration scavenging. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 528–535, 2008.

Levchenko, K. Notes from a lecture of van vu at university of california, san diego. In `http://www.cs.ucsd.edu/~klevchen/techniques/index.html`, 2005.

Li, Lihong, Chu, Wei, Langford, John, and Schapire, Robert E. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web (WWW)*, pp. 661–670, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772758.

Li, Lihong, Chu, Wei, Langford, John, and Wang, Xuanhui. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In King, Irwin, Nejdl, Wolfgang, and Li, Hang (eds.), *Proc. Web Search and Data Mining (WSDM)*, pp. 297–306. ACM, 2011. ISBN 978-1-4503-0493-1.

Lu, T., Pál, D., and Pál, M. Contextual multi-armed bandits. In *Proc. of the 13th Artificial Intelligence and Statistics (AI & Stats), JMLR: W&CP 9*, May 13-15 2010.

Robbins, Herbert. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

Scott, D.W. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley Series in Probability and Statistics. Wiley, 1992. ISBN 9780471547709.

Shani, Guy and Gunawardana, Asela. Evaluating recommendation systems. In *Recommender systems handbook*, chapter 8, pp. 257–297. Springer, 2011.

Strehl, Alexander L., Langford, John, Li, Lihong, and Kakade, Sham. Learning from logged implicit exploration data. In *Proc. NIPS*, pp. 2217–2225, 2010.

Sutton, R.S. and Barto, A.G. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series. MIT Press, 1998. ISBN 9780262193986.

Thompson, W.R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3–4):285—294, 1933.

Offline evaluation of recommendation systems. Supplementary material

Abstract

These notes contain supplementary material for the paper entitled "Offline evaluation of recommendaiton systems" submitted to ICML 2014.

1. The ICML Exploration&Exploitation Challenge 3

1.1. The ICML E&E Challenge 3

The ICML Exploration vs. Exploitation challenge was organized in 2012 as a follow-up to a similar in spirit challenge held at ICML 2011. The challenge was based on a dataset associated to Yahoo! front page, corresponding to 30 millions news displays. We note $\mathcal{S}$ this dataset. The collection of data was done carefully, so that an unbiased evaluation of recommendation algorithms may be done offline on this dataset, according to (**?**). To meet this goal, news were served uniformly at random on Yahoo! front page to a fraction of the audience also sampled uniformly at random, and contextual information about the served news, the visitor of the webpage and whether he/she clicked on the news was logged to make up the dataset. The dataset was collected over a month. This involved that the set of items to recommend was not fixed (each item lives only during a few hours on average). Subsequently, the whole dataset was made publicly available on Yahoo! Labs webscope as the R6 dataset. To be specific, each data in $\mathcal{S}$ is a tuple (x, a, r) where:

- x is a context (including a timestamp),
- a is the news that was served,
- r is the user feedback, either click or not.

To be precise, it should be pointed out that the set of displayable news is changing along time. This set of displayable news is part of the context x, and a refers to one of the elements of this set of displayable news. Though we do not mention it in the sequel and drop time indices to make notations lighter, the reader should keep in mind that the set of possible actions changes over time, and the number of possible actions also changes over time. So, the entities such as K or D are really K_t and D_t.

The goal of the challenge was to successfully demonstrate the superiority of contextual algorithms over non contextual algorithms. It is noteworthy that a set of 135 features associated to the users (the x in the tuple of data) was carefully crafted by Yahoo! engineers to be useful and improve the quality of recommendations. The evaluation of algorithms was done following the replay method described by Alg. 1 introduced in (**??**). The core idea of this method is to use rejection sampling. That is, data of the dataset $\mathcal{S}$ are considered iteratively; for each data, if the algorithm chooses an action that is different from the one logged in $\mathcal{S}$ (which mean we are not able to know what the reward would be), we simply skip this data. As actions of $\mathcal{S}$ have been chose randomly uniform the stored action and the choice of A matches with probability $1/K$. The interest of this strategy on this kind of data is discussed by (**?**).

2. Proof of theorem 2

We remind theorem 2 of the main paper:

Theorem 1. *Assuming that*

$$CTR_A(T) = z + \frac{p_1}{\sqrt{T}} + \ldots + \frac{p_\alpha}{T^{\alpha/2}} + o\left(\frac{1}{T^{\alpha/2}}\right)$$

Then for a algorithm A producing a fixed policy over time, BRED applied on a dataset of size L evaluates the expectation of the CTR_A with no bias and with high probability for B and L large enough:

$$\left| \frac{1}{B} \sum_{b=1}^{B} CTR_A(T^{(b)}) - \widehat{CTR}_A^{(b)}(T^{(b)}) \right| = O\left(\frac{1}{L}\right)$$

This means that the convergence of the estimator of $CTR_A(L)$ is much faster than the convergence of g_A (which is in $O(1/\sqrt{L})$. This will allow us to make a very good estimator of g_A by replacing $CTR_A(L)$ by our estimate.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Algorithm 1 *Replay method* (**??**).
Remark: for the sake of the precision of the specification of the algorithm, we use a history h which is the list of triplets (x, a, r) that have yet been used to estimate the performance of the algorithm A. The goal is to avoid hiding internal information maintenance in A; a real implementation may be significantly different for the sake of efficiency, by learning incrementally.

Input:

- A contextual bandit algorithm A
- A set S of L triplets (x, a, r)

Output: An estimate of g_A

```
h ← ∅
Ĝ_A ← 0
T ← 0
for t ∈ {1..L} do
    Get the t-th element (x, a, r) of S
    π ← A(h)
    if π(x) = a then
        add (x, a, r) to h
        Ĝ_A ← Ĝ_A + r
        T ← T + 1
    end if
end for
return Ĝ_A / T
```

The sketch of the proof of theorem 2 is the following: first we prove that the replay strategy is able to estimate the moments of the distribution of CTR_A fast enough with respect to the size of $\mathcal{S}$. The second step consists in using classical results from bootstrap theory to guarantee the unbiased convergence of the average $\widehat{CTR}_A(T^{(b)})$ to the true distribution with an $O(\frac{1}{T^{(b)}})$ speed. The rational behind this is that the gap introduced by the subsampling averaging will be of the order of $O(\frac{1}{\sqrt{T^{(b)}B}})$. Now, we turn to the real proof of theorem 2.

Proof. For the sake of simplicity, let us forget about the JITTER(x) step in BRED. This point is discussed in remark 4 below.

At each iteration of the bootstrap loop (indexed by b), BRED is estimating the CTR using the replay method on a dataset of size $L' = K \times L$. As the actions in $\mathcal{S}$ are randomly uniformly chosen, we have $\mathbb{E}(T^{(b)}) = L'/K = L$.

As the policy is fixed, we can use the multiplicative Chernoff's bound as in (**?**) to obtain for all bootstrap step b:

$$\mathbb{P}r\left(\left|T^{(b)} - L\right| \geq \gamma_1 L\right) \leq \exp\left(-\frac{L\gamma_1^2}{3}\right)$$

for any $\gamma_1 > 0$ (where $\mathbb{P}r(e)$ denotes the probability of event e). A similar inequality can be obtained with $\mathbb{E}(\widehat{G}_A) = Lg_A$:

$$\mathbb{P}r\left(\left|\widehat{G}_A - Lg_A\right| \geq \gamma_2 Lg_A\right) \leq \exp\left(-\frac{Lg_A\gamma_2^2}{3}\right)$$

Thus with $\gamma_1 = \sqrt{\frac{3}{L}\ln\frac{4}{\delta}}$ and $\gamma_2 = \sqrt{\frac{3}{Lg_A}\ln\frac{4}{\delta}}$ using a union bound over probabilities, we have with probability at least $1-\delta$:

$$1 - \gamma_1 \leq \frac{T^{(b)}}{L} \leq 1 + \gamma_1$$

$$g_A 1 - \gamma_2 \leq \frac{\widehat{G}_A}{L} \leq g_A 1 + \gamma_2$$

which implies

$$\left|\frac{\widehat{G}_A}{T^{(b)}} - g_A\right| \leq \frac{(\gamma_1 + \gamma_2)g_A}{1 - \gamma_1} = O\left(\sqrt{\frac{g_a}{L}}\ln\frac{1}{\delta}\right)$$

So with high probability the first moment of $\widehat{CTR}_A(T^{(b)})$ as estimated by the replay method admits an asymptotic expansion in $1/\sqrt{\mathbb{E}(T^{(b)})} = 1/L$.

Now we need to focus on higher order terms. All the moments are finite because the reward distribution over $\mathcal{S}$ is

bounded. Recall that by hypothesis $CTR_A(T)$ admits a α^{th} order term:

$$p_\alpha = \mathbb{E}_D\left(CTR_A(T^{(b)})^\alpha\right)$$

The Chernoff's bound can be applied to $|(T^{(b)})^\alpha - L^\alpha|$ and $|\widehat{G}_A^\alpha - L^\alpha g_A^\alpha|$ leading to

$$\left|\frac{\widehat{G}_A^\alpha}{(T^{(b)})^\alpha} - g_A^\alpha\right| = O\left(\left(\frac{g_a}{L}\right)^{\frac{\alpha}{2}} \ln\frac{1}{\delta}\right)$$

With probability at least $1-\delta$. So for a large enough $T^{(b)}$, $\widehat{CTR}_A(T^{(b)})$ admits a expansion in polynoms of $1/\sqrt{L}$. Thus theorem 1 applies. For a large enough number of bootstrap iterations (the value of B in BRED), we obtain a convergence speed in $O(1/L)$ with high probability, which concludes the proof.

□

3. Some more experimental results

3.1. Artificial data

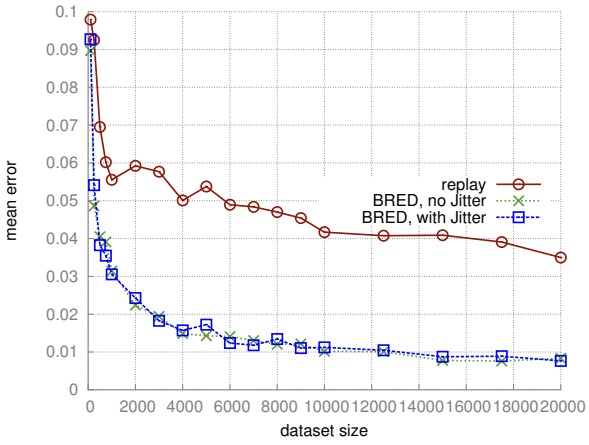


Figure 1. Mean of the absolute value of the difference between the true CTR of a UCB and the estimated one for different methodologies. Conducted on artificial dataset as described in the section 6.1 of the main paper. The lower, the better. Jittering is useless here because UCB does not use the context.

3.2. Real data

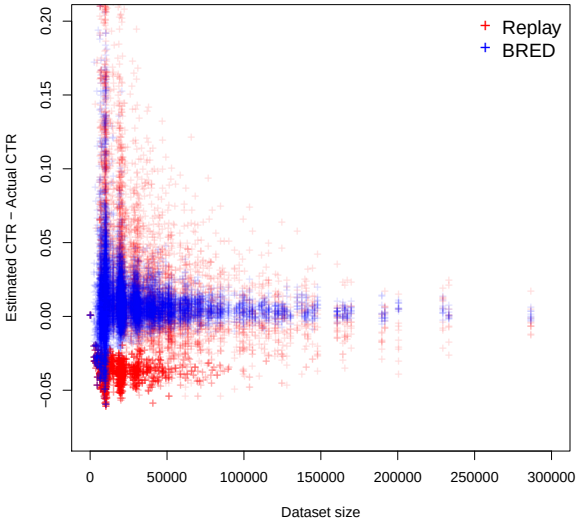


Figure 2. The difference between the estimated CTR and the actual one on some batches extracted from the Yahoo! R6B dataset for a UCB. Batches are build as explained in section 6.2 of the main paper. The closer to 0, the better. Please note that the replay method tends to under-estimate the true CTR for small batches. This is due to the fact that UCB does not have enough time to reach its actual CTR.

Chapter 6

Recommendation systems

Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

ICML Exploration & Exploitation challenge: Keep it simple!

Olivier Nicol OLIVIER.NICOL@INRIA.FR
Jérémie Mary JEREMIE.MARY@INRIA.FR
Philippe Preux PHILIPPE.PREUX@INRIA.FR
LIFL (UMR CNRS 8022) & INRIA Lille Nord Europe
Université de Lille
59650 Villeneuve d'Ascq
France

Editor: X

Abstract

Recommendation has become a key feature in the economy of a lot of companies (online shopping, search engines...). There is a lot of work going on regarding recommender systems and there is still a lot to do to improve them. Indeed nowadays in many companies most of the job is done by hand. Moreover even when a supposedly smart recommender system is designed, it is hard to evaluate it without using real audience which obviously involves economic issues. The ICML Exploration & Exploitation challenge is an attempt to make people propose efficient recommendation techniques and particularly focuses on limited computational resources. The challenge also proposes a framework to address the problem of evaluating a recommendation algorithm with real data. We took part in this challenge and achieved the best performances; this paper aims at reporting on this achievement; we also discuss the evaluation process and propose a better one for future challenges of the same kind.

Keywords: Recommendation - Bayesian - Exploration - Exploitation - Evaluation - Click prediction

1. Introduction

The ICML Exploration & Exploitation challenge considers the problem of predicting clicks of visitors on items presented on a website, based on website logs. The challenge relies on data provided by Adobe. The dataset represents a website activity regarding visitors' appeal towards a set of options. The options correspond to a set of news, one being displayed in a small box on the visited web page. The aim is to propose interesting news to the visitor; their interest for a given news is asserted by a click on it, which provides the visitor further information on the subject.

Let us list a few characteristics of this challenge:

- This is an online process, that is, the data in the dataset are ordered by time.

- Participants do not have access to the data set: the evaluation of proposed algorithm is performed by the organisers. This lack of availability of data also limits the optimisation of the hyper parameters.
- Name and purpose of the attributes are unknown; so it is impossible to rely on some expert knowledge. We only know that all the attributes are somehow characterising the visitor except one: the id of the option.
- We do not have access to the domain of definition of the discrete attributes.
- We do not know the distribution of the value of the continuous attributes. Some of the values are very small, while some others are very large.
- There are missing values in the data. Some of the attributes are never available.
- We have no way to know if we see a visitor for the first time or not (no id accessible during the evaluation process).

In a nutshell, our approach is based on an additive model updated at after each data handling in a Bayesian way. After an overview of the algorithm, we present how we progressively improved our performances. This improvement was mostly obtained thanks to our continuous work on the feature construction issue, and, more surprisingly, by a continuous simplification of the model. Indeed, we ended up making our predictions ignoring completely the displayed option. More importantly for an "Exploration & Exploitation" challenge, all our attempts to take into account any underlying dynamics, or to explore carefully also led to a decrease of the performance. We then exhibit a few reasons why we think simplicity was the best option in this challenge, but that it might not be the case in real recommendation systems. Actually, we think that the bias towards simple algorithms comes from, at least partially, the evaluation process of the challenge. Finally, we conclude by proposing a better procedure to evaluate recommendation systems on that kind of data set.

2. Formalisation of the task and organisation of the challenge

We consider the following problem. A display d is a point in a space $C \times D \times O$ characterising an option presented to a visitor where:

- C is a compact subspace of $\mathbb{R}^{99}$.
- D is a space of dimension 20 of discrete values.
- a point $u \in C \times D$ characterises a visitor.
- O is the option space: $O = \{1, \ldots, K\}$, $K = 6$ in the challenge.

We have a data set $\mathcal{D}$ made of $N = 18 \cdot 10^6$ records ordered by time. To each display d_i is associated a reward $r_i \in \{0, 1\}$, meaning whether the option was clicked or not. As already mentioned, there are lots of missing values for the discrete and continuous attributes, but the option and the reward are known for all data.

A batch b is a group of 6 consecutive tuples (display, reward). During the evaluation process of a program $\mathcal{P}$, the batches are given one at a time in chronological order. At iteration t, $\mathcal{P}$ receives b_t without the associated rewards. $\mathcal{P}$ chooses the element of b_t which it considers the most likely to be associated with a reward of 1. The reward of this element (and only this one) is then revealed to $\mathcal{P}$ and added to the current score. The goal is obviously to score as much as possible.

Computational resources are limited: approximately 1GB of memory for the process of the whole data set, and 100 ms on a 2GHz processor to process each batch.

For debugging purposes, we were initially given 60 lines of the data set. Those 60 data should not be assumed to be representative of the whole data set in terms of attribute values, and their distribution. In particular, the number of successful displays is grossly over-represented in these 60 lines with regards to the whole data set. Thus, it is clear that one should neither train his/her algorithm with these 60 data, nor even learn precise information about attribute values with these 60 data.

In the first part of the challenge, each participant was evaluated on the same set of data, made of the first $3 \cdot 10^6$ lines (so on $5 \cdot 10^5$ batches) of the data set. This set of data is unknown: the evaluation is performed on a server managed by the organisers of the challenge. Each participant is allowed to repeatedly submit his/her program to get his/her score. There are computational limits so that the frequency at which each participant can submit a code is not very high; it depends on the server load, that is, the number of participants having submitted their code; in our experience, the time to perform one evaluation of one program ranged from less than 1 hour to more than a whole day. As our algorithm, and probably most of submitted algorithms, is stochastic, the score obtained is varying at each submission: chance plays a role.

At the end of the first round of the challenge, the $3 \cdot 10^6$ data were revealed to let the participants improve their algorithm. The resulting program was then run only once on the whole data set made of N data, just before the workshop was held. Our algorithm was designed using only the data revealed after the first part of the challenge. After the second part, as the whole data set was revealed, we made use of it to better understand our results.

3. Our approach

We investigated the use of ideas based on adPredictor to predict click rates in online advertisement by Graepel et al. (2010). This solution is inspired from the TrueSkill$^{\text{TM}}$ algorithm by Herbrich et al. (2007) used to rank online players on the Xbox gaming network. The main idea is to consider the probability of click of a visitor u on an option o as the sum of some contributors having different levels of uncertainty and which are updated in a Bayesian way.

3.1 The model of the clicks

The model for the click probability of a visitor on an option is made up of k discrete features built on $C \times D \times O$. The construction of these features will be discussed in section 4. Before that, let us precise that for each possible value of each feature, we estimate a Gaussian distribution $\mathcal{N}(m, \sigma)$. Subsequently, values from this distribution are drawn on demand, acting as weights; each value of a given feature has a weight which is drawn from a Gaussian distribution each time this is required; we name such weights "Gaussian weights", and our algorithm estimates their parameters (mean, and variance). For a given display d, each feature of $C \times D \times O$ takes a value (which may be NA). We say that these values are *active* for d. The Gaussian weight associated to this value is called a *contributor* and reflects its contribution to the click probability of d. This contribution is more or less uncertain depending on σ. We note $active(d)$ the *active contributors* of a display d. Given a *contributor* c, m_c (resp. σ_c) is the mean (resp. the standard deviation) of the associated Gaussian weight.

3.2 Making the decision on a batch

When a batch b is received, a score $s(d)$ is computed for each display $d \in b$ as follows:

$$s(d) = \sum_{a \in active(d)} X(m_a, \alpha \cdot \sigma_a) \tag{1}$$

where $X(m, \sigma)$ is a realisation of a $\mathcal{N}(m, \sigma)$ and α a real parameter. Then the display with the maximum score is chosen.

3.3 Online learning

After an option has been chosen for a display d, the algorithm is notified whether d led to a click or not, by way of the binary reward. The active contributors of d are updated in a positive way if a click occurred, and in a negative way otherwise. This comes close to the update of the weights of a perceptron, but in the present case, the weights are not scalar but Gaussian, and this necessitates smarter updates. We use the following notations:

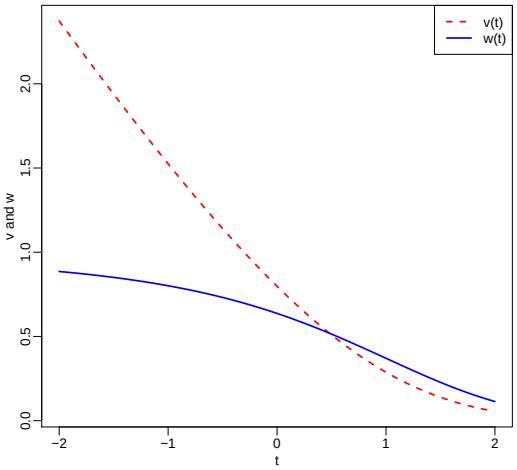
- $y = 1$ if there is a click, -1 otherwise.
- $\Sigma^2 = \beta^2 + \sum_{a \in active(d)} \sigma_a^2$, with β a real parameter
- $\Lambda = \frac{y \cdot \sum_{a \in active(d)} m_a}{\Sigma}$

Σ quantifies the uncertainty on the score and Λ its correctness. The numerator of the fraction defining Λ may be understood as follows: if a score is positive we expect a click and if it is negative we do not; so this numerator is positive if the prediction was correct, and the larger this product, the more correct the prediction. Then, this product is divided by the uncertainty, which means that the more uncertain the prediction, the less it can be considered as really meaningful since we did know that it was uncertain. The update rules for the parameters of the active contributors are as follows:

$$\tilde{m}_a = m_a + y \cdot \frac{\sigma_a}{\Sigma} \cdot v(\Lambda)$$

$$\tilde{\sigma}_a^2 = \sigma_a^2 \cdot \left(1 - \frac{\sigma_a^2}{\Sigma^2} \cdot w(\Lambda)\right),$$

where v and w are real valued functions. Many different functions can be used for v and w. With respect to our tests, as long as the shape is similar, the results are quite robust to variations on v and w. So we decided to use the same functions as Graepel et al. (2010) (see Fig. 1).



$$v(t) = \frac{d\Phi(t)}{dt} \cdot \frac{1}{\Phi(t)}$$

$$w(t) = -\frac{dv(t)}{dt}$$

with $\Phi(t)$ the cumulative distribution function of $\mathcal{N}(0, 1)$

Figure 1: The functions v and w.

The main idea of these update equations is that the update is small if the outcome is not surprising (*i.e.*, the prediction was correct). This can easily be seen from the shape of v and w (see Fig. 1). Indeed when Λ (the correctness) is negative, $v(\Lambda)$ is large and so is the correction to m whereas if Λ is positive, then almost no update is performed. The same thing is true for the multiplicative update of σ as when Λ grows, $w(\Lambda)$ gets closer to 1. To see why this is important, let us consider a bad contributor c active along with a lot of good contributors in a display. If we observe a click which is not surprising given the number of good contributors, should we change a lot our estimation of c? Obviously the answer is no since it is very likely that the click has resulted from the contributions of the good ones and not from a bad estimation of c.

This update strategy offers some similarities with TD-learning by Sutton (1988): the more we are surprised by the consequences of a decision, the larger the correction. This strategy usually leads to faster convergence rates of learning.

Another interesting idea is the use of the uncertainty. It makes the learning of the model much more robust than the learning strategy of a simple perceptron for example. Indeed, let us consider the case in which we are sure that a display is bad (low uncertainty on the contributors), and in which unexpectedly, we get a click. This is very surprising, but since the update also depends on the uncertainty, we will not make a big update because of an

event that is most certainly noise.

Note that the update equations were computed through the approximate message passing algorithm. The reader who would like to have a deeper understanding of these equations and of the functions v and w in addition to the intuition we gave here should refer to Herbrich et al. (2007) and Graepel et al. (2010).

4. Our performances along the course of the challenge

The goal of the challenge is to score as much as possible on the 5.10^5 first batches (3.10^6 displays). The final score is the only output we get when an algorithm is submitted; so it is the only criterion we will use to compare different approaches in this section. This section describes how we improved our program along the course of the challenge. For a pseudo-code implementation of our final algorithm, see Appendix B.

4.1 Early results

A purely uniform random strategy scores slightly lower than 1200 on 5.10^5 batches; a strategy that always chooses the same option scores equally. If we model the number of clicks with a Poisson law of parameter $\lambda = 1200$, a significant difference (with risk 5%) is 57 points. This order of magnitude should be kept in mind when trying to compare two different scores.

The algorithm presented in section 3 needs discrete features; so during the first trials, we just ignored the continuous ones. Moreover as the main goal of the challenge was to do recommendation, it is quite natural to try to catch the preferences of the visitors in the features. So we simply built 20 features, each one of them being the Cartesian product of a discrete feature $f_d \in D$ and the option feature. We handled the missing values by just assigning them to a specific contributor.

Note that running the algorithm on $D \times O$ is equivalent as having 6 models, one for each option, using only D. Indeed, two contributors of two different features associated with different options can never be active together. This is actually the approach we took in the beginning but we changed a bit later to be able to handle both features resulting from a Cartesian product with O and features built otherwise without having to duplicate them.

The first attempts scored equally as random. Our first improvement occurred when we set the value of β to 100 instead of 1 (as it was arbitrary set when first executed). Our score was approximately 1500 (25% more than random). In that setting, our score went up to 1610 (34% more than random) after optimising β and the value of the priors on the contributors. It was a good start as we performed significantly better than the random strategy using only 21 out of the 120 features of (C, D, O) (in fact we were using (D, O)).

4.2 Discretisation

4.2.1 Cartesian products

From this point, the most natural idea to get better performances out of this model was to include the continuous features. Since we need discrete ones, we began with a discretisation of the continuous values in 5 buckets. Cutting values were fixed using the first seen values using an EM algorithm (Mc Lachlan and Krishnan, 1996) to cluster the 64 first values in 5 Gaussian weights. As with the discrete features, the *null* value (for missing values) has its own bucket. Each bucket of each feature was then associated to a different contributor.

We note C_d the discretised version of C. Using this, we tried to run the algorithm with different versions of the model:

- Still with the idea of trying to find the favourite option of each visitor, we first tried to do a Cartesian product of each feature with the option. So with $C_d \times O, D \times O$, our algorithm scored around 1550. It was very disappointing since it is worse than with only $D \times O$.
- As adding $C_d \times O$ made the performances decrease, we tried to add the continuous features without any Cartesian product. With $C_d, D \times O$, our algorithm scored around 1900 (58% more than random).
- As removing the Cartesian product with O for C_d drastically improved the performances, we also tried C_d, D but our algorithm only scored around 1600.

We tried to combine (with a Cartesian product) only a few features from C_d with the option but no matter which ones we picked, we found no contribution scoring more than 1900. We also tried to identify online which features from C_d were actually correlated with the option using ANOVA and then to dynamically combine them with the option. It did not work either. According to these results, it seems clear that the features from D are characterising the visitor preferences whereas the features from C are characterising the general behaviour of the visitor facing an option.

4.2.2 A better discretisation

We were trying different approaches in parallel to the one we present in this paper and we had noticed that using only one cutting value per feature was pretty efficient (so two buckets per feature). Nevertheless trying to find the good cutting value using only the 60 given lines was pretty hopeless. That is why we decided to build several 2-buckets discretisations for each continuous feature and then choose the best one.

More formally, for each feature $C_i \in C$ we have a set S_i of possible cutting values each of which corresponds to a 2-buckets discretisation. The possible cutting values are chosen before running the algorithm looking at the 60 lines of the data set we have been given ($\forall C_i \in C \quad |S_i| \approx 20$ in order to cover C_i quite exhaustively). For each cutting value v_{ij} of S_i we have 2 Gaussian weights $w_{ij}^{(1)}$ and $w_{ij}^{(2)}$ (one for values lower than v_{ij} and one for greater values).

In principle, for a given display, each feature has one *active* Gaussian weight (see section 3.1). To compute the score of this display, we sum these *active* Gaussian weights (see section 3.2). Here for each feature $C_i \in C$ we have several active weights, one per discretisation/cutting value. So for each $C_i \in C$ we only consider as *active* the one weight associated to the cutting value which maximises the following criterion:

$$booleanToInt\left(\sigma_{ij}^{(1)} \leq T(t) \quad AND \quad \sigma_{ij}^{(2)} \leq T(t)\right) \times d(w_{ij}^{(1)}, w_{ij}^{(2)})$$

where:

- $\sigma_{ij}^{(k)}$ is the standard deviation of the weight $w_{ij}^{(k)}$.
- $booleanToInt(x)$ is equal to 1 if x is true and 0 otherwise.
- $d(w_1, w_2)$ is the probability that the Gaussian weight with the greatest mean between w_1 and w_2 is actually greater than the other. It is a metric of how far apart the two weights are.
- $T(t)$ is a threshold function depending on time. The one we used is just a linearly decreasing function of time $T(t) = \gamma \cdot t$ with a minimum value (time is actually the batch index).

During the update, we update the *active* weights of all the discretisations as in section 3.3.

To sum up, for each feature $C_i \in C$ instead of building only one complex discretisation we build $|S_i|$ discretisations with 2 discrete values (so with only on cutting value). We then choose the cutting value with the two most separated weights if they are both accurately enough estimated. This approach slightly outperformed the EM based discretisation mentioned above (see Sec. 4.2.1). Indeed, it scored around 1940 (62% more than random).

Note that while running, the algorithm could add new values to some S_i if it observed too much values outside of the minimum and maximum observed in the 60 given lines. However when we were able to run the algorithm by ourselves after the end of phase 1, we noticed that it was almost never the case, hence the good results we observed building discretisations only observing these data.

4.2.3 A simpler discretisation

To check whether we were learning interesting things with our two previous approaches, we tried to build an offline fixed discretisation. To do so, we looked at the density plot of the continuous features for the 60 lines of the data set given during phase 1 (see Fig. 2). We did not use the frequency of clicks as there were too few of them. To build the discretisation, we just assigned one interval per peak on the plot. We identified 12 features as different from the others and from each other like the two ones in the left part of Fig. 2. Their discretisations were built one by one. For the 87 others, we made three groups:

- A group where all the values in the 60 lines were missing. We just kept the approach that tries to find only one cutting value. We found out during phase 2 that these features were actually *null* in all data.

- A group where all the features looked almost exactly like the one on the right of Fig. 2. They represent one third of all the continuous features and we only built a unique discretisation for all of them.
- A group where the value of the features is 0 most of the time but not always. They represent half of all the continuous features and as the previous groups we only built a unique discretisation for all of them.

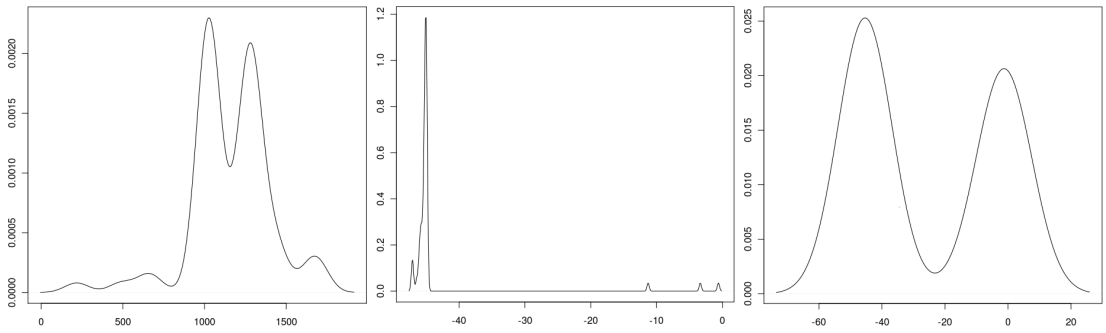


Figure 2: The density plot of 3 continuous features for the 60 lines of the data set given during phase 1. The two on the right are on a logarithmic scale.

With this simple method of discretisation, our algorithm scored 1950 (63% more than random) which was already more than the previous approach. We then focused on the two largest groups of features and after merging and splitting a few intervals, we were able to bring our score up to 2000 (67% more than random) and sometimes a bit more because of the variance of the algorithm. This has remained our best score for a while.

4.3 Dynamics

The batches were served in chronological order during the challenge to let us identify the dynamics of the system. When we observe the update equation for σ we notice that whatever happens, σ always decreases making the model unable to handle the fact that some weights may vary over time. None of the approaches we tried allowed us to improve our performances. We try to explain why in section 5. However we present here two of these approaches.

4.3.1 Microsoft's idea

In Graepel et al. (2010), the following update rule is proposed to be used at each time step and for each weight of each feature:

$$\tilde{\sigma}_i^2 = \frac{\sigma_{i(p)}^2 \cdot \sigma_i^2}{(1-\varepsilon) \cdot \sigma_{i(p)}^2 + \varepsilon \cdot \sigma_i^2}$$

$$\tilde{m}_i = \sigma_i^2 \cdot \left((1-\varepsilon) \frac{m_i}{\sigma_i^2} + \varepsilon \cdot \frac{m_{i(p)}}{\sigma_{i(p)}^2} \right)$$

The idea behind these update equations is to let the weights evolve back to their prior in order to slowly forget the influence of past data. This allows the model to adapt to the dynamics. In the equations, the bigger ε, the faster the algorithm forgets. Unfortunately the smaller ε, the better the algorithm performed in the challenge. The better performances were achieved with $\varepsilon = 0$.

4.3.2 Two or more models

The previous approach is not fully satisfying as it cannot take into account the fact that some weights may evolve at different speeds. Some other may also not evolve at all. Moreover, the convergence of the weights towards their prior makes the model learn slower as we are always forgetting a little.

To deal with these issues, we propose to use m models. At each time step only the oldest model is used to make a prediction and all of them are updated. Every τ time steps, we destroy the oldest model and replace it by a new one. This new model can be initialised in several ways. The simplest one would be to always give to each weight the same prior. Then we would handle the dynamics by making prediction with a model which has been learning during between $\tau \cdot (m-1)$ and $\tau \cdot m$ time steps. We could also try to build the priors using the history of their values. Are these values very stable? unstable? evolving following some kind of trend? We tried such estimators but as nothing worked, we will not go into the details. However we will see in section 5 that we have good reasons to think that this failure was due to the challenge itself. We will study this idea more carefully in some future work.

4.4 Exploration

Throughout the challenge we had been using the update equation (1) with $\alpha = 1$. Actually this parameter did not even exist. As none of our approaches seemed to do better than 2000 we were starting to explore different paths. Just to see what happened we submitted an algorithm which was not taking into account the standard deviations in the prediction phase (it was actually summing up the means of the weights). Surprisingly this approach scored 2130 (130 points more than with exploration and 78% more than random). Then, we tried to optimize the value of α but the best value was 0 (no exploration at all). A value up to 0.05 was not making any difference though.

We obviously tried decreasing exploration approach as $\varepsilon_n - greedy$ but it did not improve the performances. The only one that seemed to slightly improve the algorithm (in terms of performance and variance) is the following: for the first 5000 batches (1% of the total) instead of choosing the display with the best score, we choose the display d maximising:

$$v(d) = \sum_{a \in active(d)} \sigma_a^2$$

which is the variance of the sum of the active Gaussian weights. During phase 2, by running the two approaches 100 times, we were able to check whether the performances were really improved or not. Here are the results: using $s(d)$ for the 5000 first batches: $mean = 2130$

$variance = 1086$, while using $v(d)$ for the 5000 first batches: $mean = 2140$ $variance = 587$. That is how our final score 2170 for phase 1 was achieved.

4.5 Further score improvements

For phase 2 we got access to the data of phase 1. Then we could run a lot more simulations to optimise the algorithm. Trying to optimise α, β and the prior on the weights led to minor improvements. However to be able to present the results, we did again a few experiments. In one of them, we tried to use C_d, D as a model which is the raw feature space with our discretisation over C and without the option. We ran the algorithm 100 times with these features and it achieved a mean score of 2215 and a variance of 190. The best score we got during this experiment is 2240 (85% more than random) which is much better than our previous approach. This is the algorithm that won the second phase of the challenge. In summary, the simplest version of the model ended up having the better performances and being the more stable.

We had tried this approach before and it had scored badly on the server of the challenge. We can only try to guess why: a bug on the server's side, a mistake in the file submission on our side... We could have avoided that kind of thing by submitting each algorithm more than once but we would have lost a lot of time as one submission usually returned after several hours. This is an interesting idea to think about in case of a future challenge. The simplest thing would be to run the algorithm 5 or 10 times at each submissions. However if computational resources are limited, it becomes an issue. To address this problem, more data could be given to the challengers so that they can test their algorithm by themselves before submitting it. Restrictions on the number of submissions per day could even be imposed in that case to easily allow a sharing of resources between challengers.

5. Understanding the results

The purpose of the challenge was to design an algorithm capable of doing three main things:

- identify the preferences of a visitor to recommend something to him/her
- balance exploitation and exploration
- adapt to the dynamics of system (some options may for example become less popular for some visitors)

However, in the challenge we had to choose between 6 couples $(visitor, option)$ and what we did is basically visitor selection. Indeed our best approach only used C and D — the visitor features — to make its decisions. Moreover as presented in the last section the approach which performs the best neither explores, nor adapts.

5.1 Why did visitor selection work so well?

What we did is identifying the general behaviour of visitors in front of an option without paying attention to evolutions or exploring anything. Then when a batch comes we select

the visitor who clicks the most in general regardless to the displayed option. We can try to explain why that works so well in two ways.

5.1.1 Intuition

Trying to find the click probability of a visitor in general is a lot easier than trying to find it for each option. Indeed each time a visitor is seen, the model can be updated whereas to find the preferences, no information is obtained for non displayed options. When working on the logs of a big french company we had also noticed that some variables are very important as far as clicks are concerned. For instance some pages have higher click probabilities than others and the time of the day matters (visitors do not click at night). C, D might have contained that kind of features making our task even easier. Moreover half of the visitors in the company's logs never click. Identifying all of them in the challenge already doubles the click probability (which is basically the maximum score we achieved).

Intuitively the general behaviour of someone on the Internet is very unlikely to change dramatically as opposed to his preferences. That is why trying to find some kind of dynamics was hopeless. So having to identify something pretty easy to characterise and very stable, the reason why almost no exploration was needed is rather clear. To picture a bit more that the task was not that hard see Fig. 3 and 4 where we see that we very quickly improve the performances and stabilise them. Note that the low click rate around batch $170,000$ must be due to the data as it is also there for the random strategy.

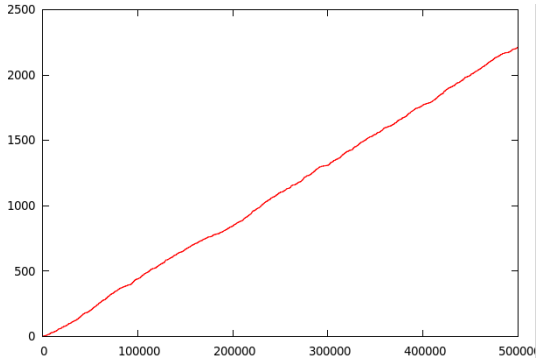


Figure 3: Evolution of our score during phase 1. The x-axis is the number of the current batch.

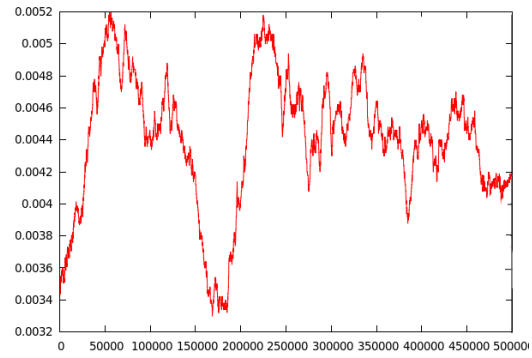


Figure 4: Evolution of the click rate during phase 1.

5.1.2 Features

In a more pragmatic way we can explain why we performed so well without using the options (or any other Cartesian product) by looking at the click frequency plot of some features. Looking at Fig. 5, we can clearly identify patterns both for the discrete and the continuous features. The one in the middle of Fig. 5 is particularly informative. We tried to run a very simple UCB strategy as described by Auer and Kivinen (2002) with one arm per value

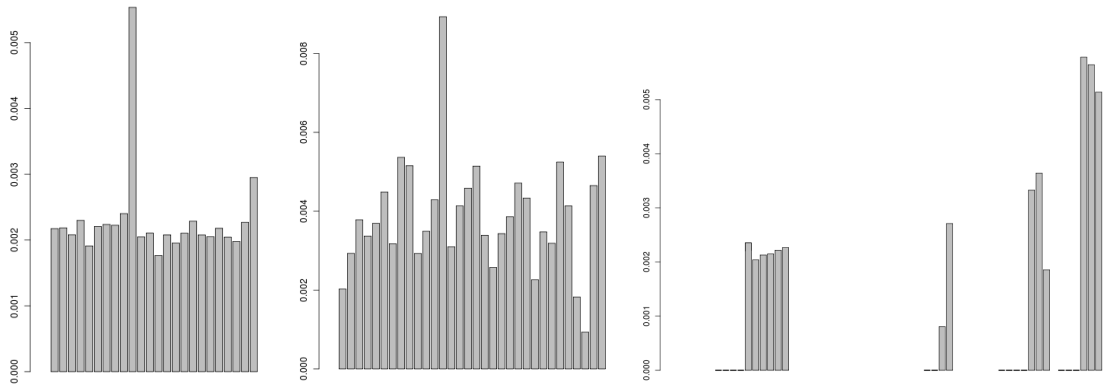


Figure 5: Click frequency against the value of 3 different features. The two features on the left are discrete and the values appearing less than 5000 times in the data set have been removed. The one on the right is continuous and plotted on a log scale. For that latter feature, please note that the high frequencies on the right only represents 0.5% of the values.

of this feature and scored 1450. It means that using only one feature from D allows us to perform 21% better than a purely uniform random strategy!

5.2 Crossed effects

Does this mean that there are not any crossed effects between features in this data set? The answer is no. We performed an analysis of variance (ANOVA) and found a lot of crossed effects between O and some other features. We also found crossed effects between features of D, C. During phase 1 we had tried to learn them online and during phase 2, we knew about the Cartesian products. However we still have not been able to exploit them. We can only try to give an intuition to explain that. The general behaviour of visitors seems to be something very stable and very well characterised by the set of features we have been given. Trying to enhance the model with things like preferences which are much more unstable and much harder to identify is very tricky. In our experiments we have even noticed that it adds disturbance to the model making it less efficient in learning and using the behaviour of visitors.

6. More experiments

6.1 Click rate prediction

In Graepel et al. (2010) the following way to infer click probability from the model is proposed:

$$p(d) = \Phi(\frac{s(d)}{\sqrt{\beta^2 + \sigma^2}})$$

with the same notations as in section 3.3.

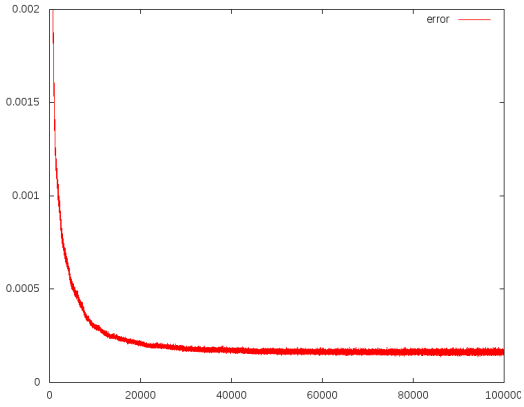


Figure 6: Mean squared error on the probability over time (computed on 100 randomly generated displays).

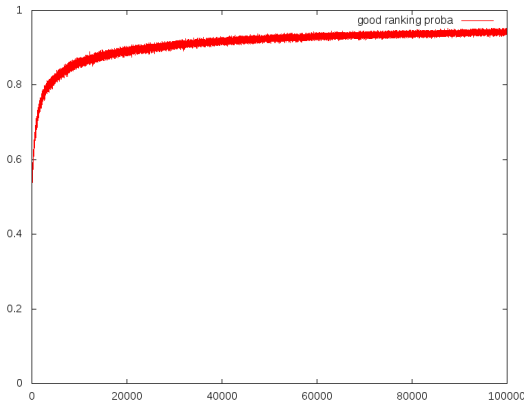


Figure 7: Probability that two predictions are correctly ranked over time (computed on 100 couples of randomly generated displays).

We experimented this equation on a toy example. In this example, we assume that reality is as assumed by the model: each click probability is the result of a sum of contributors. In the following experiment we use 10 features which can take 5 possible values. For each feature f_i, the values of the 5 contributors are as follows:

$$\{10^{-4}.p_i^0,\ 10^{-4}.p_i^1,\ 10^{-4}.p_i^2,\ 10^{-4}.p_i^3,\ 10^{-4}.p_i^4\}$$

The real parameters p_i are uniformly drawn in $[1, 5]$ at the beginning of the experiment.

At each time step a random display is presented with its reward to update the model (the values of the features are drawn uniformly). The squared error (Fig. 6) on the predictions converges but its value remains very high when it stabilises. Indeed the squared error stabilises around 0.002. The average over the click probabilities is around 0.04 and $\sqrt{0.0002} \approx 0.014$ (35% of 0.04).

However as it can be seen on Fig. 7 the order of the probabilities is very well learnt and it is learnt very fast. The model reaches a rate of 80% of well ranked probabilities after less than $1,000$ steps and then goes up to 95%. Hence the success of the algorithm when it comes to choose between displays during the challenge. Also note that when the error on the predictions stops improving after $30,000$ time steps, the probability of good ranking keeps improving until the end of the experiment.

6.2 Influence of the parameters

The model has a few parameters and we propose here to study their influence on the performance of our algorithm. We will not talk about α since the best option in the challenge was to do no exploration at all ($\alpha = 0$).

6.2.1 Beta

β impacts the learning speed. Fig. 8 shows that the algorithm is not very sensitive to its value. Any value between 300 and 600 achieves almost the best performances. Then when β grows we do not learn fast enough and the performances decrease until stabilising around 1630. If β gets closer to 0 we try to learn too fast and performances dramatically decrease until 1200 (random strategy).

6.2.2 Priors

Another important parameter is the prior we take for the Gaussian weights. As we had no prior knowledge about the data, we had to give default values. As far as the mean is concerned, if it is between -1 and 1 it does not change anything. For larger values, we eventually learn correctly but it takes more time, hence a decrease in the performances.

Optimising the standard deviation (σ) was more interesting. As we can see on Fig. 9 we can find an optimal value to give to each weight which is around 20. However Fig. 10 and 11 (the latter being a zoom on the best scores of the former) show that we can do better. If we initialise with different values the standard deviation of the discrete features (σ_d) and the standard deviation of the continuous features (σ_c), we can win something like 30 points. It seems to mean that learning from D is easier than learning from C. We tried to split C and give different values of σ to each resulting group but nothing significant came out. Note that on Fig. 10 even though we can find optimal values for σ_d and σ_c the algorithm remains very stable. Only very low values for σ (less than 2) or very high values (more than 80) leads to scores inferior to 2000.

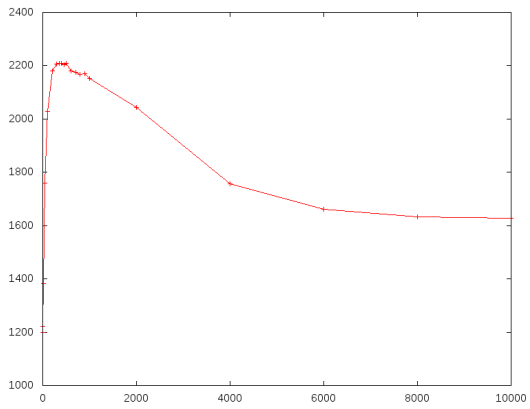


Figure 8: Score in the challenge against β.

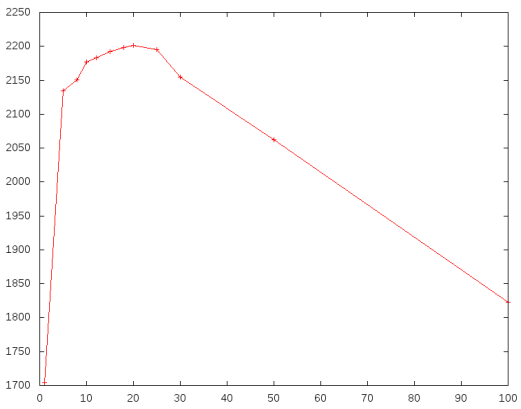


Figure 9: Score in the challenge against σ.

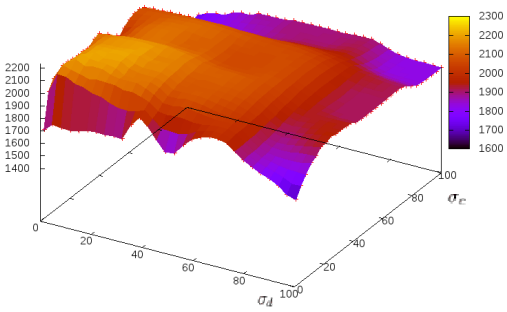


Figure 10: Score in the challenge against σ_d and σ_c.

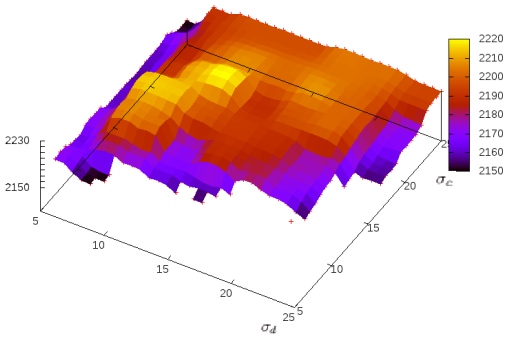


Figure 11: Zoom on the best scores in the challenge against σ_d and σ_c.

7. Evaluation protocol

As already mentioned, the evaluation protocol used during the challenge is problematic because there exists a much stronger link between the visitor and the click probability, than between the option $\times$ visitor and the click rate. So if we are given a batch with the possibility to choose a couple (visitor, option), we should focus only on the visitor.

Of course it is impossible in this situation to know what would have been the performance of an algorithm because we can not go backward in time or find an exactly identical situation. One good solution to compare two algorithms would be to use them online with real visitors at the same time but it would be very expensive (in terms of missed clicks) and would necessitate a big visitor flow to conduct experiments.

An other possibility would have been to use rejection sampling on the option. The process would be:

- Pick the first non used row (u, o, r) from $\mathcal{D}$ (u is the visitor, o the displayed option, r the click or no click information).
- Input u to the recommendation algorithm and let it choose an option o_a,
- If o_a and o are identical, then give the reward r to the algorithm, otherwise discard this visitor.
- loop until no more rows in $\mathcal{D}$.
- The score of the algorithm is the average click rate for non discarded visitors.

This method based on rejection would only use $1/K$ records to evaluate the algorithm, and would act as a "time accelerator"; it is assumed that(options are uniformly distributed in

$\mathcal{D}$. Similar ideas have been used by Li et al. (2010). Here it is acceptable because there is only 6 possible options and a large number of records (so dropping 5/6 of them is not a big deal). But sometimes we can not afford such a method.

7.1 A new method for offline evaluation

In this section, we propose a new way to evaluate recommendation algorithm on a task such as the one considered in this challenge. In the next section, we show that this approach is sound and actually evaluates what it is meant for.

With the data set $\mathcal{D}$, we can create another data set $\mathcal{D}_1$ by only taking the lines of $\mathcal{D}$ associated with a reward of 1. Then we can design an online classification problem in which the classes are the options. So the goal is to map each visitor of $\mathcal{D}_1$ to the right option. The only way to perform well in that case is to find connections between visitor preferences and the options: simple visitor selection is no longer relevant. By trying to solve this problem online, we might also be able to identify some kind of dynamics in the preferences or to benefit from exploration. We call that problem P_1.

In problem P_1, a given visitor may have more than one class if he/she has been shown different options and that he/she has clicked on at least two of them. In fact one can consider that each visitor u belongs to K classes. Each time we encounter visitor u during the online process and try to classify him/her as class (or option) o the probability to be correct is given by:

$$p_{ou} = p(o|u \in \mathcal{D}_1)$$

Note that $p_{ou} = 0$ if u has never clicked on o in the data set $\mathcal{D}$ and that:

$$\forall u \quad \sum_{o \in Options} p_{ou} = 1$$

We call that kind of classification problem a *stochastic* classification problem.

We can also design a harder stochastic classification problem where we have to map each visitor of $\mathcal{D}$ to the right option. When we consider a visitor which is in $\mathcal{D}_1$, the reward is given as in P_1. Mapping an option to a visitor which is not in $\mathcal{D}_1$ always lead to a reward of 0. So in this problem that we call P_2 we have visitors with no known class. One can think of them as noise added to problem P_1. In the evaluation process, no difference is made between a misclassified visitor, and a visitor without class: in both cases, the reward is 0 and that is all. This is what makes the problem harder.

7.2 Theoretical result

Theorem 1 *With the notation introduced in section 2. We denote:*

- f^* *the optimal function that maps a visitor to an option for the recommendation problem.*
- g^* *the optimal function that maps a visitor to an option for* P_1

- h^* *the optimal function that maps a visitor to an option for* P_2

Under the assumption that the options were allocated uniformly while creating $\mathcal{D}$*, we have the following result:*

$$\forall u \in \mathcal{D}_1 \quad f^*(u) = g^*(u) = h^*(u)$$

This means that if we manage to design a good mapping function for P_1 or P_2 (two problems whose proposed solutions are easy to evaluate), we have theoretical guarantees that it will also perform well on the real recommendation problem.

The proof in provided in Appendix A. The uniformity assumption is true in the case of the Adobe data set. It should be possible to extend this result (up to a renormalisation) to the case were the probability of distribution is not uniform but known and with a non null probability of having any option attributed to visitor u.

8. Conclusion and future work

We presented a Bayesian approach to the recommendation problem. This approach led us to win the 2011 ICML Exploration & Exploitation challenge. Based on a data set provided by Adobe, we have shown that on this particular dataset, the click probability is much more related to the visitor than to the option. Moreover this behaviour is very stable over time. This means that not all visitors are equal, and that there are much more "valuable" visitors than others. In the advertising context, this means that some part of the optimisation problem of advertising display is to allocate some of this "premium" visitors to some well chosen ads. To perform this optimisation, a mixture of bandit algorithm and linear programming was proposed by Girgin et al. (2010).

We also proposed to use an other evaluation protocol to use collected data to compare the performance of new online algorithms. Further work will investigate the link between the Thompson sampling step of adPredictor and the global performance. As we outperform the Thompson sampling using likelihood maximisation, and do even better doing something in between, we are interested in the non-asymptotic behaviour of Thompson sampling and will try to exhibit better strategies (especially in the small probability case).

Acknowledgments

This work was supported by Ministry of Higher Education and Research, Nord-Pas-de-Calais Regional Council and FEDER through the "Contrat de Projets Etat Region (CPER) 2007-2013", the ANR project Lampada (ANR-09-EMER-007) and the CRE-INRIA-Orange Labs. During this work, O. Nicol was supported by a PhD grant of the University of Lille, and J. Mary acknowledges a partial secondment from INRIA.

References

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. volume 47, pages 235–256, Hingham, MA, USA, May 2002. Kluwer Academic Publishers. URL http://dx.doi.org/10.1023/A:1013689704352.

S. Girgin, J. Mary, Ph. Preux, and O. Nicol. Advertising campaigns management: Should we be greedy? In *The 10th IEEE International Conference on Data Mining (ICDM-2010)*, pages 821–826, 2010. URL http://www.grappa.univ-lille3.fr/~mary/paper/icdm-2010.pdf.

Thore Graepel, Joaquin Quiñonero Candela, Thomas Borchert, and Ralf Herbrich. Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-2010)*, pages 13–20, Haifa, Israel, June 2010. Omnipress. URL http://www.icml2010.org/papers/901.pdf.

Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill$^{\text{tm}}$: A bayesian skill rating system. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS-2006)*, pages 569–576. MIT Press, 2007.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, WWW'2010, pages 661–670, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: http://doi.acm.org/10.1145/1772690.1772758. URL http://doi.acm.org/10.1145/1772690.1772758.

Geoffrey J. Mc Lachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. Wiley-Interscience, 1st edition, November 1996. ISBN 0471123587. URL http://www.worldcat.org/isbn/0471123587.

Richard S. Sutton. Learning to predict by the methods of temporal differences. In *Machine Learning*, pages 9–44. Kluwer Academic Publishers, 1988. URL http://webdocs.cs.ualberta.ca/ sutton/papers/sutton-88-with-erratum.pdf.

Appendix A. Proof of the theorem

Let us remind the notations:

- $\mathcal{D}$ is the whole data set.
- $\mathcal{D}_1$ is the data set containing only the lines of $\mathcal{D}$ associated with a reward of 1.
- f^* is the optimal function that maps a visitor to an option for the recommendation problem.
- P_1 is the stochastic classification problem where we have to map each visitor of $\mathcal{D}_1$ to the correct option. g^* is its optimal mapping function.
- P_2 is the stochastic classification problem where we have to map each visitor of $\mathcal{D}$ to the correct option. Classifying a visitor not present in $\mathcal{D}_1$ only leads to a reward of 0. h^* is its optimal mapping function.

A.1 g^* and h^*

It is straight forward to see that if $u \in \mathcal{D}_1$ we have:

$$h^*(u) = g^*(u)$$

Indeed rewards are earned the same way in P_1 and P_2 and the only lines that matter in P_2 are the ones from $\mathcal{D}_1$. To complete h^* we can map anything to the visitors who are not in $\mathcal{D}_1$ since they do not produce rewards anyway (because all visitors not in $\mathcal{D}_1$ correspond to a non click row, and in P_2 there is no way to score even providing a different option).

A.2 f^* and g^*

An optimal mapping function for the recommendation problem is as follows:

$$f^*(u) = \underset{o \in Options}{\operatorname{argmax}}\, p\left(Click = 1 |\, (u, o)\right)$$

For the stochastic classification problem in $\mathcal{D}_1$, an optimal mapping function can be written as follows:

$$g^*(u) = \underset{o \in Options}{\operatorname{argmax}}\, p(o | u \in \mathcal{D}_1)$$

Let us call $N_{(u,o)}$ the number of times option o was displayed to visitor u in $\mathcal{D}$ and $C_{(u,o)}$ the number of times u clicked on o in $\mathcal{D}$. We have the two following equalities:

$$p(o|u \in \mathcal{D}_1) = \frac{\mathbb{E}_{(u,o)\in\mathcal{D}}[C_{(u,o)}]}{\sum_{o' \in Options} \mathbb{E}_{(u,o')\in\mathcal{D}}[C_{(u,o')}]} \tag{2}$$

$$\mathbb{E}_{(u,o)\in\mathcal{D}}[C_{(u,o)}] = \mathbb{E}_{(u,o)\in\mathcal{D}}[N_{(u,o)}].p\left(Click = 1|(u, o)\right) \tag{3}$$

If the options were allocated uniformly while creating $\mathcal{D}$ - which is the case in the data set used for the challenge according to the information we have been given by Adobe - then

$$\forall u \quad \mathbb{E}_{(u,o_1)\in\mathcal{D}}[N_{(u,o_1)}] = \mathbb{E}_{(u,o_2)\in\mathcal{D}}[N_{(u,o_2)}] = ... = \mathbb{E}_{(u,o_k)\in\mathcal{D}}[N_{(u,o_k)}], \quad k = |Options| \tag{4}$$

Replacing (2) in (1) and then simplifying using (3) we obtain:

$$\begin{aligned} p(o|u \in \mathcal{D}_1) &= \frac{\mathbb{E}_{(u,o)\in\mathcal{D}}[N_{(u,o)}] \;.\; p(Click = 1|(u,o))}{\sum_{o'\in Options} \mathbb{E}_{(u,o')\in\mathcal{D}}[N_{(u,o')}] \;.\; p\left(Click = 1|(u,o')\right)} \\ &= \frac{p\left(Click = 1|(u,o)\right)}{\sum_{o'\in Options} p\left(Click = 1|(u,o')\right)} \end{aligned}$$

Finally, using this result in the expression of g*, we get:

$$\begin{aligned} g^*(u) &= \text{argmax}_{o\in Options} \frac{p\left(Click = 1|(u,o)\right)}{\sum_{o'\in Options} p\left(Click = 1|(u,o')\right)} \\ &= \text{argmax}_{o\in Options} p\left(Click = 1|(u,o)\right) \\ &= f^*(u) \end{aligned}$$

So we do have:

$$\forall u \in \mathcal{D}_1 \quad f^*(u) = g^*(u) = h^*(u)$$

□

Appendix B. Pseudo code of the algorithm

B.1 The winning algorithm

Note that before using the algorithm, a set of discrete features is to be built. For each possible value of each feature, we keep track of a Gaussian weight (its mean m and standard deviation σ). For a given display, each feature takes on one value. The weights corresponding to these values are said to be *active* for this display.

Function chooseDisplay(D: SetOfDisplays): Display

parameter: *T: the number of initial pure exploration steps*
parameter: α
foreach *Display* $d_i \in D$ **do**
  $s[i] \leftarrow 0$
  foreach $a \in active(d_i)$ **do**
    if $\#iterations \geq T$ **then**
      draw x from $\mathcal{N}(m_a, \sigma_a \cdot \alpha)$
    else
      $x \leftarrow \sigma_a^2$
    end
    $s[i] \leftarrow s[i] + x$
  end
end
$maxIndex \leftarrow \text{argmax}_i s[i]$
return $d_{maxIndex}$

Procedure updateModel(d: Display, click: Boolean)

parameter: β
if *click* **then**
  $y \leftarrow 1$
else
  $y \leftarrow -1$
end
$uncertaintySq \leftarrow \beta^2 + \sum_{a \in active(d)} \sigma_a^2$
$uncertainty \leftarrow \sqrt{uncertaintySq}$
$correctness \leftarrow \frac{y \cdot \sum_{a \in active(d)} m_a}{uncertainty}$
foreach $a \in active(d)$ **do**
  $m_a \leftarrow m_a + y \cdot \frac{\sigma_a}{uncertainty} \cdot v(correctness)$
  $sq \leftarrow \sigma_a^2 \cdot \left(1 - \frac{\sigma_a^2}{uncertaintySq} \cdot w\left(correctness\right)\right)$
  $\sigma_a \leftarrow \sqrt{sq}$
end

Algorithm 1: Evaluation of the two previous functions during the challenge

```
foreach Batch b : B do
    d ← chooseDisplay(b)
    r ← observeReward(d)
    updateModel(d, r)
end
```

B.2 More than one model

This approach is the one presented in section 4.3.2. We note $updateModel_M$ the procedure updating model M and $chooseDisplay_M$ the function choosing a display using model M.

Algorithm 2: using more than one model to handle the dynamics

```
modelList ← emptyList
i ← 0
add a new model to modelList
foreach Batch b : B do
    M ← firstElement(modelList)
    d ← chooseDisplay_M(b)
    r ← observeReward(d)
    foreach Model M ∈ modelList do
        updateModel_M(d, r)
    end
    i ← i + 1
    if i = τ then
        if size(modelList) = m_max then
            remove the last element of modelList
        end
        add a new model at the beginning of modelList
        i = 0
    end
end
```

Front. Comput. Sci.
DOI 10.1007/s11704-012-2873-5

RESEARCH ARTICLE

Managing advertising campaigns – an approximate planning approach

Sertan GIRGIN[1,2], **Jérémie MARY**[1,2], **Philippe PREUX**[1,2], **Olivier NICOL**[1,2]

1 Team-Project Sequel, INRIA Lille Nord Europe, Villeneuve d'Ascq 59650, France
2 LIFL (UMR CNRS), Université de Lille, Villeneuve d'Ascq 59650, France

Abstract We consider the problem of displaying commercial advertisements on web pages, in the "cost per click" model. The advertisement server has to learn the appeal of each type of visitors for the different advertisements in order to maximize the profit. Advertisements have constraints such as a certain number of clicks to draw, as well as a lifetime. This problem is thus inherently dynamic, and intimately combines combinatorial and statistical issues. To set the stage, it is also noteworthy that we deal with very rare events of interest, since the base probability of one click is in the order of 10^{-4}. Different approaches may be thought of, ranging from computationally demanding ones (use of Markov decision processes, or stochastic programming) to very fast ones. We introduce NOSEED, an adaptive policy learning algorithm based on a combination of linear programming and multi-arm bandits. We also propose a way to evaluate the extent to which we have to handle the constraints (which is directly related to the computation cost). We investigate performance of our system through simulations on a realistic model designed with an important commercial web actor.

Keywords advertisement selection, web sites, optimization, non-stationary setting, linear programming, multi-arm bandit, click-through rate (CTR) estimation, exploration-exploitation trade-off.

Received June 15, 2011; accepted November 6, 2011

E-mail: sertan.girgin@inria.fr

1 Introduction

The ability to efficiently select items that are likely to be clicked by a human visitor of a web site is a very important issue. Whether for the mere comfort of the user to be able to access the content he/she is looking for, or to maximize the income of the website owner, this problem is strategic. The selection is based on generic properties (date, world news events, ...), along with available personal information (ranging from mere IP related information to more dedicated information based on the login to an account). The scope of applications of this problem ranges from advertisement or news display (see for instance the Yahoo! Front Page Today Module), to web search engine result display. There are noticeable differences between these examples: in the first two cases, the set of items from which to choose is rather small, in the order of a few dozens; in the latter case, the set contains billions of items. The lifetime of items may vary considerably, from a few hours for news, to weeks for web advertisements, to years for pages returned by search engine. Finally, the objective ranges from drawing attention and clicks on news, to providing the most useful information for search engines, to earning a maximum of money in the case of advertisement display. Hence, it seems difficult to consider all these settings at once and in this paper, we consider the problem of selecting advertisements, in order to maximize the profit earned from clicks: we consider the "cost to click" economic model in which each single click on an advertisement brings a certain profit. We wish to study principled approaches to solve this problem in the most realistic setting; for that purpose, we

consider the problem with:

- finite amounts of advertising campaigns,
- finite amounts of clicks to gather on each campaign,
- finite campaign lifetimes,
- the appearance and disappearance of campaigns along days, and
- a finite flow of visitors and page requests.

With these assumptions, we would like to emphasize that our goal is not to optimize any asymptotic behavior and exhibit algorithms that are able to achieve optimal asymptotic behavior (but perform badly for much too long). To the opposite, we concentrate on the practical problem faced here and now by the web server owner: he/she wants to make money now, and do not really care about ultimately becoming a billionaire when the universe will have collapsed (which is likely to happen in a not so remote future with regards to asymptotic times either). In the same order of ideas, we also want to keep the solution computable in "real"-time, real meaning here within a fraction of a second, and able to support the high rate of requests observed on the web server of an important web portal. Of course, such requirements impede the quality of the solution, but these requirements are necessary from the practical point of view; furthermore, since we have to deal with a lot of uncertainty originating from various sources, the very notion of optimality is quite relative here.

In Section 2, we formalize the problem under study, and introduce the vocabulary and the notation used throughout the paper. Our notations are summarized in an appendix to the paper. The problem we tackle is actually changing over time; for pedagogical reasons, in Section 3 we first study the problem under a static setting where the set of advertising campaigns are known in advance and the time horizon is fixed, before moving to the more general dynamic setting without these constraints in Section 4. We define a series of problems of increasing complexity, ranging from the case in which all information is available, to the case where key information is missing. Assessing algorithms in the latter one is difficult, in particular from a methodological point of view, and spanning this range of problems let us assess our ideas in settings in which there is a computable optimal solution against which the performance of algorithms may be judged. Section 5 presents related works. Section 6 presents some experimental results of our algorithm near optimal sequential estimation and exploration for decision (NOSEED) in both static and dynamic settings. Finally, Section 7 concludes and we briefly discuss the lines of foreseen future works.

2 Formalization of the problem

At a given time t, there is a pool of advertising campaigns. Each advertising campaign in the pool has a starting time, a lifetime and a click budget that is expected to be fulfilled during its lifetime. At each click on an advertisement of the campaign, a certain profit is made. The status of an advertising campaign can be either one of the following (Fig. 1):

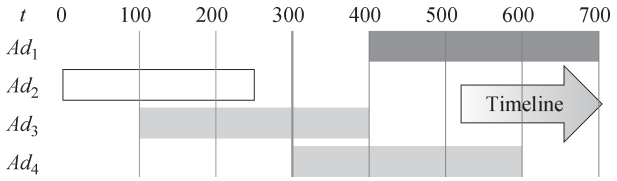


Fig. 1 At time $t = 300$, Ad_1 is in scheduled state (in dark grey), Ad_2 has expired (in white), Ad_3 and Ad_4 are running with remaining lifetimes of 100 and 300, respectively (in light grey)

scheduled when the campaign will begin at some time in the future,

running when the campaign has started but not expired yet,

expired when either the lifetime of a campaign has ended or the click budget has been reached.

The advertisements of a campaign can only be displayed when it is in the *running* state.

Each advertising campaign is assumed to have a unique identifier, and we will represent an advertising campaign by a tuple (S, L, B, cp, rb) where S, L, B and cp denote its starting time, lifetime, click budget and profit per click, respectively; $rb \leqslant B$ denotes the remaining click budget of the advertising campaign. Note that, for a given advertising campaign, all parameters except the remaining click budget are constant; the remaining click budget is initially equal to the click budget of the advertising campaign and decreases with time as it receives clicks from the visitors. Throughout the paper, the advertising campaign with identifier k will be denoted by Ad_k and its parameters will be identified by subscript k, e.g., S_k will denote the starting time of the advertising campaign with identifier k.

Now, the problem that we are interested in is as follows (Fig. 2):

- The web site receives a continuous stream of page requests. Each request originates from a "visitor", that is,

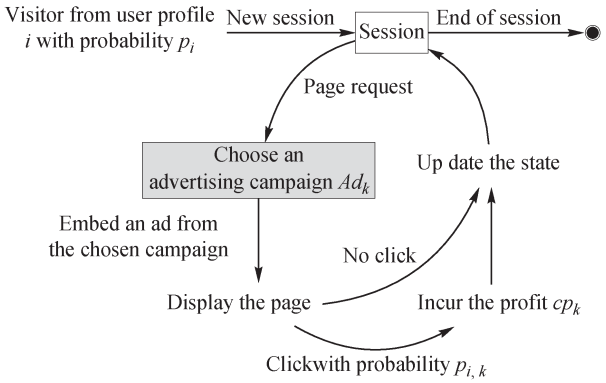


Fig. 2 The interaction between a visitor and the system

a human being browsing in some way the website. We assume that non human visitors (robots) may be identified as such, and are filtered out[1]. Each visitor is assumed to belong to one among N possible user profiles; the user profiles are numbered from 1 to N. We will use U_i to denote the i^{th} user profile and v_i to denote the probability that a visitor belongs to that user profile[2].

- When a visitor visits the web site, a new "session" begins and we observe one or several iterations of the following sequence of events:
  - The visitor requests a certain page of the web site.
  - The requested page is displayed to this visitor with an advertisement from advertising campaign with identifier k embedded in it.
  - The visitor clicks on the advertisement with a certain probability $p_{i,k}$ where i denotes the user profile of the visitor; this probability is usually called the click-through rate (CTR) and the event itself is a Bernoulli trial with success probability $p_{i,k}$.
  - If there is a click, then the profit associated with the advertising campaign is incurred.
- After a certain number of page requests, the visitor leaves the web site and the session terminates.

Returning visitors do not change the nature of the problem given that the session information persists, and for the sake of simplicity we will be assuming that there are no returning visitors.

The objective is to maximize the total profit by choosing the advertisements to be displayed "carefully". Since page requests are atomic actions, in the rest of the paper we will take a page request as the *unit of time* to simplify the discussion, i.e., a time step will denote a page request and vice versa. Note that in the real-world, some of the parameters mentioned above may not be known with certainty in advance. For example, we do not know the visit probabilities of the user profiles, their probability of click for each advertising campaign, the actual profiles of the visitors, or the number of requests that they will make; the number of visitors may change with time and new advertising campaigns may begin. These and other issues that we will address in the following sections of the paper make this problem a non-trivial one to solve.

3 Static setting

In order to better understand the problem and derive our solution, we will first investigate it under a static setting. In this setting, we assume that

- there is a pool of K advertising campaigns, and
- the properties of the advertising campaigns (i.e., their starting times, lifetimes, click budgets and click profits) are known in advance.

Note that, this leads to a fixed time horizon T which is equal to the latest ending time of the advertising campaigns. At time T, the task is finished. Other parameters of the problem, such as the click and visit probabilities, may or may not be known with certainty. We will start with the case in which all the information is available, and subsequently move to the setting in which uncertainty comes into play, and then only a part of the information will be available.

3.1 Static setting with full information

In the static setting with full information, we assume that all parameters are known. To be more precise:

(a) the visit probabilities of user profiles, v_i, and their click probabilities for each advertising campaign, $p_{i,k}$ are known, and,

(b) there is no uncertainty in the actual profiles of the visitors, i.e., we know for sure the profile of each visitor.

Note that, even if we have full information, the visitor at time t and whether the visitor will click on the displayed advertisement or not are still unknown.

We first define the problem we wish to solve as a Markov

[1] The identification of robots is not necessarily easy to perform, but even if some of them are not filtered out, that would not bring serious problems to our study.
[2] The sum of v_i over all user profiles is equal to 1, which forms a categorical distribution with probability mass function $f_{\mathbf{P}}(U_i) = v_i$
[3] To be precise, one optimal solution, or a set of equally performing optimal solutions.

decision process (MDP) [1]. This implies that the problem under consideration has an optimal solution[3)]. For the sake of clarity, we only detail the MDP formulation in the static setting, but subsequent, more complex settings, may easily be cast into the MDP (or partially-observable MDP) framework. Then, we consider the problem of determining an optimal policy for this MDP. As it will be shown in the following section, the state space of the MDP grows linearly with time and exponentially with the number of advertising campaigns. This huge state space makes it difficult to determine an optimal policy by straightforward dynamic programming approaches in a reasonable time for any practical application; this raises the question of whether there can be other approaches to solve the problem and obtain a policy that performs "well" (also this explains why we do not solve the problem using traditional MDP algorithms).

Being interested in real settings, thus looking for non asymptotic performance, and wishing to have an algorithm that performs as best as possible in an efficient way, we examine various issues and subsequently propose the NOSEED algorithm which aims to handle them both in simple and more complex settings as will be detailed in Sections 3.2, 3.3, and 4.

3.1.1 The underlying Markov decision problem for the advertising selection problem

At any time t, the state of this version of the problem can be fully represented by a tuple that consists of time t, the time horizon T, the visit and click probabilities, and a set of tuples denoting the advertising campaigns:

$$\langle t, T, \{v_i\}, \{p_{i,k}\}, Ad_1 = \langle S_1, L_1, B_1, cp_1, rb_1\rangle, \ldots, Ad_K\rangle.$$

By omitting the fixed parameters, this tuple can be more compactly represented as $\langle t, rb_1, \ldots, rb_K\rangle$.

Given a state $s = \langle t, rb_1, \ldots, rb_K\rangle$, if there is no click at that time step or there is no running advertising campaign then the next state, which we will denote by $s'_{noclick}$, has the same representation as s except the t component since click budgets of campaigns do not change, i.e., $s'_{noclick} = \langle t+1, rb_1, \ldots, rb_K\rangle$. In case an advertisement from a running advertising campaign Ad_k is clicked, the remaining click budget of Ad_k will be reduced by 1 and the next state becomes $\langle t+1, rb_1, \ldots, rb_k - 1, \ldots, rb_K\rangle$; we will denote this state by $s'_{click,k}$.

A policy is defined as a mapping from states to a distribution over the set of advertising campaigns; given a particular state, the policy determines which advertising campaign to display at that state. A policy is called optimal if it maximizes the expected total profit. Let $V(s)$ denote the expected total profit that can be obtained by following an optimal policy starting from state s until the end of time horizon; $V(s)$ is usually called the value of state s. Now, suppose that there is a visitor from the i^{th} user profile at state s; the expected total profit that can be obtained by displaying an advertisement from a running advertising campaign Ad_k can be defined as:

$$V_{i,k}(s) = p_{i,k}[cp_k + V(s'_{click,k})] + (1 - p_{i,k})V(s'_{noclick}), \quad (1)$$

and the optimal policy, i.e., the best advertising campaign to display, would be to choose advertising campaign with the maximum expected total profit, i.e., $\arg\max_{Ad_k} V_{i,k}(s)$. Note that, the value of state s can be calculated by taking the expectation of maximum $V_{i,k}(s)$ values over all user profiles and we have:

$$V(s) = \sum_{U_i} \max_{Ad_k} V_{i,k}(s). \quad (2)$$

Regarding expired campaigns, we define their value to be 0. Using Eqs. (1) and (2), the value of any state can be determined, for example, by dynamic programming; henceforth, the optimal policy can be determined too. However, the size of the state space is equal to $(T - t) \times rb_1 \ldots \times rb_K$ and grows exponentially with the number of advertising campaigns (with order equal to their budgets). From a practical point of view, this huge state space makes such solutions very computationally demanding, and unable to meet our requirements in this regard.

3.1.2 A greedy approach

When we look at Eq. (1) more carefully, it is easy to see that the value of the next state without a click, $V(s'_{noclick})$, is an upper bound for the value of the next state with a click, $V(s'_{click,k})$. Replacing the second term by $V(s'_{noclick}]) - \xi_{i,k}$ where $\xi_{i,k}$ is a constant that depends on s, Ad_k and the user profile U_i, we obtain:

$$V_{i,k}(s) = p_{i,k}cp_k - p_{i,k}\xi_{i,k} + V(s'_{noclick}). \quad (3)$$

If $\xi_{i,k}$ values are small compared to the corresponding click profits, i.e., their effect is negligible, or they are ignored, then the optimal policy becomes choosing the advertising campaign with the highest expected profit per click among the set of running campaigns at that state denoted by C:

$$\begin{aligned}\arg\max_{Ad_k \in C} V_{i,k}(s) &= \arg\max_{Ad_k \in C}[p_{i,k}cp_k + V(s'_{noclick})] \\ &= \arg\max_{Ad_k \in C} p_{i,k}cp_k.\end{aligned}$$

We will call this particularly simple method the highest expected value (HEV) policy. Alternatively, we can employ a stochastic selection method where the selection probability of a running advertising campaign is proportional to its expected profit per click. This variant will be called the stochastic expected value (SEV) policy.

As both policies exploit advertising campaigns with possibly high return and assign lower priority to those with lower return, one expects them to perform well if the lifetimes of the advertising campaigns are "long enough" to ensure their click budgets. However, they may show inferior performances even in some trivial situations, that is $\xi_{i,k}$ terms are significant. For example, assume that there is a single user profile and two advertising campaigns, Ad_1 and Ad_2, starting at time $t = 0$ with click probabilities of 0.005 and 0.01, lifetimes of $L_1 = 2\,000$ and $L_2 = 4\,000$ time steps, budgets of $B_1 = 10$ and $B_2 = 20$ clicks, and unit profits per click i.e., $cp_1 = cp_2 = 1$ (Fig. 3). In this particular case, starting from $t = 0$, HEV policy always chooses Ad_2 until this campaign expires (on expectation at $t = 2\,000$, at which point the other campaign Ad_1 also expires) and this results in an expected total profit of 20 units; SEV policy displays on average twice as many advertisements from Ad_2 compared to Ad_1 during the first 2000 time steps, and performs slightly better with an expected total profit of $23\frac{1}{3}$. However, both figures are less than the value of 25 that can be achieved by choosing one of the campaigns randomly with equal probability. Note that, by displaying advertisements from only Ad_1 in the first 2000 time steps until it expires and then Ad_2 thereafter, it is possible to obtain an expected total profit of 30 that satisfies the budget demands of both advertising campaigns; the lifetime of Ad_2, which is long enough to receive a sufficient number of clicks with the associated click probability, allows this to happen. In order to derive this solution, instead of being short-sighted, it is compulsory to take into consideration the interactions between the advertising campaigns over the entire timeline and determine which advertising campaign to display accordingly, in other words, consider a planning problem, as in the dynamic programming solution mentioned before.

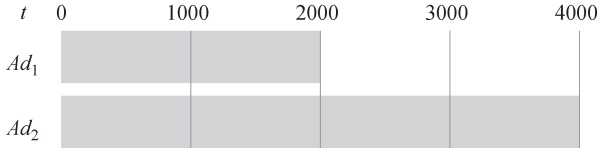


Fig. 3 A toy example in which HEV and SEV policies have suboptimal performance. Ad_1 and Ad_2 have the same unit profit per click, click probabilities of 0.005 and 0.01, and total budgets of $B_1 = 10$ and $B_2 = 20$ clicks, respectively. The expected total profits of HEV and SEV are 20 and $23\frac{1}{3}$ compared to a maximum achievable expected total profit of 30

Observing Fig. 3, it is easy to see that the interactions between the advertising campaigns materialize as *overlapping*

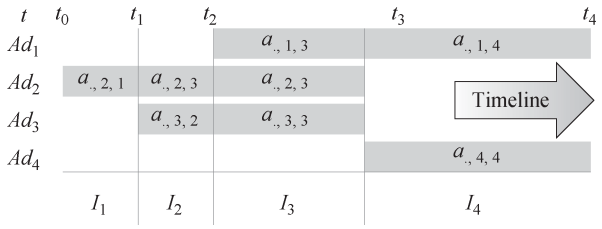


Fig. 4 The timeline divided into intervals and parts. I_j denotes the j^{th} interval $[t_{j-1}, t_j]$ and $a_{k,j}$ denotes the allocation for advertising campaign Ad_k in interval I_j. The first index of a (user profile) is left unmentioned for the sake of clarity. In this particular example, the set of running advertising campaigns in the second interval is $AI_2 = \{Ad_2, Ad_3\}$, and the set of intervals that cover Ad_1 is $IA_1 = \{I_3, I_4\}$

time intervals over the timeline[4]; in this toy example the intervals are $I_1 = [0, 2\,000]$ and $I_2 = [2\,000, 4\,000]$, and what we are trying to find is the *optimal allocation* of the number of advertising campaign displays in each interval. This can be posed as the following optimization problem where $a_{k,j}$ denotes the number of displays allocated to Ad_k in the interval I_j:

$$\begin{aligned} \text{maximize} \quad & 0.005 \times a_{1,1} + 0.01 \times (a_{2,1} + a_{2,2}), \\ \text{s.t.} \quad & a_{1,1} + a_{2,1} \leqslant 2\,000, a_{2,2} \leqslant 2\,000, \\ & 0.005 \times a_{1,1} \leqslant 10, 0.01 \times (a_{2,1} + a_{2,2}) \leqslant 20, \end{aligned}$$

which has an optimal solution of $a_{1,1} = a_{2,2} = 2000$ and $a_{2,1} = 0$. One can then use this optimal allocation to calculate the display probabilities for both advertising campaigns proportional to the number of displays allocated to them in the corresponding time intervals.

3.1.3 Optimal allocation approach

Let E_k be the ending time of advertising campaign Ad_k, which is simply equal to the sum of its starting time and lifetime. Given a pool of K advertising campaigns C, the time intervals during which the advertising campaigns overlap with each other can be found from the set of their starting and ending times. Let $t_0, t_1, \ldots, t_M$, $M \leqslant 2 \times K$, be the sorted list of elements of the set of starting and ending times of the advertising campaigns; without loss of generality, we will assume that $t_0 = 0$ as otherwise there will not be any advertising campaigns to display until t_0. By definition, the M intervals defined by $I_j = [t_{j-1}, t_j], 1 \leqslant j \leqslant M$ cover the entire timeline of the pool of the advertising campaigns. Let $AI_j = \{Ad_k | S_k < t_j \leqslant E_k\}$ be the set of running advertising

[4] See Fig. 4 for a more detailed example.

campaigns in interval I_j. Note that for some of the intervals, this set may be empty; these intervals are not of our interest as there will be no advertising campaigns to display during such intervals (which is certainly not good for the web site) and we can ignore them. Let $\mathcal{A} = \{I_j | AI_j \neq \emptyset\}$ be the set of remaining intervals, $l_j = t_j - t_{j-1}$ denote the length of interval I_j, and $IA_k = \{I_j | Ad_k \in AI_j\}$ be the set of intervals that cover Ad_k (Figure 4). Generalizing the formulation given above for the simple example and denoting the number of displays allocated to Ad_k in the interval I_j for the user profile U_i by $a_{i,k,j}$, we can define the optimization problem that we want to solve as follows:

$$\text{maximize} \quad \sum_{I_j \in \mathcal{A}} \sum_{Ad_k \in AI_j} cp_k p_{i,k} a_{i,k,j}, \tag{4}$$

$$\text{s.t.} \quad \sum_{Ad_k \in AI_j} a_{i,k,j} \leqslant v_i l_j, \quad \forall\, U_i, I_j \in \mathcal{A}, \tag{5}$$

$$\sum_{U_i} \sum_{I_j \in IA_k} p_{i,k} a_{i,k,j} \leqslant r b_k, \quad \forall\, Ad_k \in C. \tag{6}$$

The objective function (Eq. 4) aims to maximize the total expected profit, the first set of constraints (Eq. (5)) ensures that for each interval we do not make an allocation for a particular user profile that is over the capacity of the interval (i.e., the portion of the interval proportional to the visit probability of the user profile), and the second set of constraints (Eq. (6)) ensures that we do not exceed the remaining click budgets. This corresponds to the maximization of a *linear objective function* ($a_{i,k,j}$ being the variables), subject to *linear inequality constraints*, which is a *linear programming* problem. This problem can be solved efficiently using the simplex algorithm, or an interior-point method, or an other existing large scale approach if necessary. The number of constraints in the linear program is of order $O(NK)$ where N is the number of user profiles and K is the number of advertising campaigns, and the number of variables is of order $O(NK^2)$.

The solution of the linear program, i.e., the assignment of values to $a_{i,k,j}$, indicates the number of displays that should be allocated to each advertising campaign for each user profile and in each interval, but it does not provide a specific way to choose the advertising campaign to display to a particular visitor from user profile U_i at time t. For this, we need a method to calculate the display probability of each running advertising campaign from their corresponding allocated number of displays.

Let $\hat{a}_{i,k,j} = a_{i,k,j} / \sum_{Ad_k \in AI_j} a_{i,k,j}$ be the ratio of the allocation for user profile U_i and advertising campaign Ad_k in interval I_j to the total number of allocations for that user profile in the same interval. One can either pick the advertising campaign having the highest ratio in the first interval, i.e., $\arg\max_k \hat{a}_{i,k,0}$, which we will call the highest LP policy (HLP), or employ a stochastic selection method similar to SEV in which the selection probability of a campaign Ad_k is proportional to its ratio $\hat{a}_{i,k,0}$, which will be called the stochastic LP policy (SLP); SLP introduces certain degree of exploration which will be useful in more complex settings. Note that, as we are planning for the entire timeline, the solution of the linear program at time t may not allocate any advertising campaigns to a particular user profile i, i.e., it may be the case that $a_{i,k,j} = 0$ for all k, simply suggesting not to display any advertisement to a visitor from that user profile. In practice, when the current user is from such a user profile, choosing an advertising campaign with a low (or high) expected profit per click would be a better option and likely to increase the total profit at the end.

3.1.4 NOSEED: a two-phases, alternating algorithm

By defining and solving the linear program at each time step $0 \leqslant t < T$ for the current pool of non-expired advertising campaigns (which depends on the visitors that have visited the web site up until that time step, the advertising campaigns displayed to them and visitors' reactions to those displays), and employing one of the policies mentioned above, advertising campaigns can be displayed in such a way that the total expected profit is maximized, ignoring the uncertainty in the predictions of the future events (we will subsequently discuss the issues related to uncertainty).

When the number of advertising campaigns, and consequently the number of variables and constraints, is large, or when there is a need for fast response time, solving the optimization problem at each time step may not be feasible. An alternative approach would be to solve it regularly, for example, at the beginning for each interval or when an advertising campaign fulfills its click budget, and use the resulting allocation to determine the advertising campaigns to be displayed until the next resolution. In short, the algorithm alternates planning, with exploitation of this planning during multiple steps. This can be accomplished by updating the allocated number of advertising campaign displays as we move along the timeline, reducing the allocation of the chosen advertising campaigns in the corresponding intervals, and calculating the ratios that determine the advertising campaign to be displayed accordingly[5)]. Note that in practice, the planning step and the exploitation step can be asynchronous as long

[5)] The complete algorithm can be found in the appendix.

as the events that have occurred from the time that planning has started until its end are reflected properly to the resulting allocation. Such an algorithm belongs to the approximate dynamic programming family.

3.2 Dealing with uncertainty in the static setting with full information

The static setting with full information has two sources of uncertainty:

(a) the user profiles of visitors are drawn from a categorical distribution, and

(b) each advertising campaign display is a Bernoulli trial with a certain probability, which is known, and the result is either a success (i.e., click) or a failure (i.e., no click).

The aforementioned linear program solution of the optimization problem focuses on what happens in the expectation. Following the resulting policy in different instances of the same problem[6] may lead to different realizations of the total profit that vary from its expected value (due to the fact that the number of visitors from each user profile and the number of clicks on the displayed advertising campaigns will not exactly match their expected values).

As a simple example, consider the case in which there is a single user profile and two advertising campaigns Ad_1 and Ad_2 both having the same unit profit per click and a lifetime of 10^5 time steps, click probabilities of 0.001 and 0.002, and total budgets of 50 and 100, respectively. The solution of the linear program would allocate 50 000 displays to each advertising campaign with an expected total profit of 150, thus satisfying the budget demands. Figure 5 shows the cumulative distribution of the total profit over 1 000 independent runs for this problem using the stochastic LP policy and solving the optimization problem once at the beginning. Although values that are equal to or near the expected total profit are attained in more than half of the runs, one can observe a substantial amount of variability. In reality, reducing this variability may also be important and could be considered as a secondary objective to obtaining a high total profit. For the given example, slightly increasing the display probability of Ad_2 and decreasing that of Ad_1 would enable the accomplishment of this objective by preventing the risk of receiving fewer clicks than expected for Ad_2 without considerably compromising the outcome as the same risk also exists for Ad_1. This leads to the question of how to incorporate risk-awareness to our formulation of the optimization problem.

When we look closely at the objective function and the constraints of the linear program (Eqs. (4)–(6)), we can identify two sets of expressions of the form $v_i l_j$ and $p_{i,k} a_{i,k,j}$; the first one denotes the expected number of visitors from user profile U_i during the time-span of interval I_j, and the second one denotes the expected number of clicks that would be received if the advertising campaign Ad_k is displayed $a_{i,k,j}$ times to the visitors from user profile U_i. Note that visits from a particular user profile U_i occur with a known average rate v_i, and each visit occurs independently of the time since the previous visit. Therefore, the number of such visits in a fixed period of time t can be considered a random variable having a Poisson distribution with parameter $\lambda = v_i t$ which is equal to the expected number of visits that occur during that time period. Similarly, the number of clicks that would be received in a fixed period of time if advertising campaign Ad_k is displayed to the visitors from user profile U_i can also be considered a random variable having a Poisson distribution with parameter $\lambda = p_{i,k} t$. Let $Po(\lambda)$ denote a Poisson-distributed random variable with parameter λ. Replacing $v_i l_j$ and $p_{i,k} a_{i,k,j}$ terms with the corresponding random variables, we can convert the linear program into the following stochastic optimization problem:

$$\max \quad \sum_{I_j \in \mathcal{A}} \sum_{Ad_k \in AI_j} cp_k \mathbb{E}[Po(p_{i,k} a_{i,k,j})], \tag{7}$$

$$\text{s.t.} \quad \sum_{Ad_k \in AI_j} a_{i,k,j} \leqslant Po(v_i l_j), \quad \forall\, U_i, I_j \in \mathcal{A}, \tag{8}$$

$$\sum_{U_i} \sum_{I_j \in IA_k} Po(p_{i,k} a_{i,k,j}) \leqslant rb_k, \quad \forall\, Ad_k \in C. \tag{9}$$

The summation of independent Poisson-distributed random variables also follows a Poisson distribution whose parameter is the sum of the parameters of the random variables. Assuming that $Po(p_{i,k} a_{i,k,j})$ are independent, the budget constraints in Eq. (9) can be written as:

$$Po\left(\sum_{U_i} \sum_{I_j \in IA_k} p_{i,k} a_{i,k,j} \right) \leqslant rb_k, \quad \forall\, Ad_k \in C, \tag{10}$$

which is equivalent to its linear program counterpart in expectation. The rationale behind this set of constraints is to bound the total expected number of clicks for each advertising campaign (while at the same time trying to stay as close as possible to the bounds due to maximization in the objective function). Without loss of generality, assume that in the optimal allocation the budget constraint of advertising campaign Ad_k is met. This means that the expected total number of clicks for Ad_k will be a Poisson-distributed random variable with parameter rb_k and in any particular instance of the problem the probability of realizing this expectation (our target) would be 0.5. In order to increase the likelihood of reaching

[6] An "instance" refers here to a certain realization of the random problem.

the target expected total number of clicks, a possible option would be to use a higher budget limit in the constraint. Let Λ_k be our risk factor[7] and $Po(\lambda_k)$ be the Poisson-distributed random variable having the smallest parameter λ_k such that $Pr(Po(\lambda_k) > rb_k - 1) \geqslant \Lambda_k$ which is equivalent to

$$1 - \Lambda_k \geqslant \mathbf{F}_{Po(\lambda_k)}(rb_k - 1),$$

where $F_{Po(\lambda_k)}$ is the cumulative distribution function of $Po(\lambda_k)$. Note that rb_k and Λ_k are known, and λ_k can be found using numerical methods. If we replace rb_k with λ_k in the budget constraint and solve the linear optimization problem again, the expected total number of clicks for Ad_k based on the new allocation would be greater than or equal to rb_k and will have an upper bound of λ_k. Following the same strategy, one can derive new bounds for the user profile constraints and replace $v_i l_j$ terms in Eq. (8) with the smallest value of $\lambda_{i,j}$ such that the Poisson-distributed random variable $Po(\lambda_{i,j})$ satisfies $1 - \Lambda_{i,j} \geqslant \mathbf{F}_{Po(\lambda_{i,j})}(v_i l_j)$ and $\Lambda_{i,j}$ is the risk factor. In this case, an additional set of constraints defined below is necessary to ensure that for each interval the sum of advertising campaign allocations for all user profiles do not exceed the length of the interval:

$$\sum_{U_i} \sum_{Ad_k \in AI_j} a_{i,k,j} \leqslant l_j, \quad \forall I_j \in \mathcal{A}\,. \tag{11}$$

As presented in Fig. 5, in our simple example using a common risk factor of 0.95 results in a cumulative distribution of total profit which is more concentrated toward the optimal value compared to the regular linear program approach.

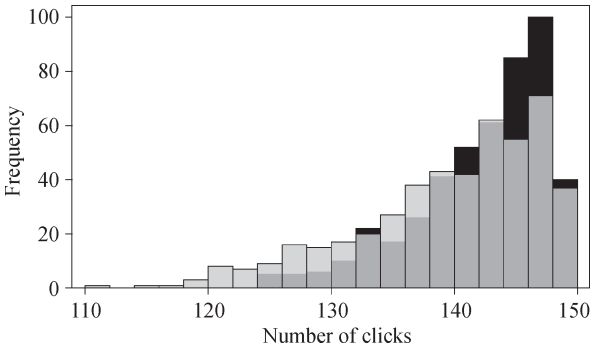


Fig. 5 The distribution of the total profit less than its expected value over 1000 independent runs on the toy example with two advertising campaigns; the dark shaded bars depict SLP with a risk factor of 0.95. In reality, the realization will be only one of the runs and therefore more concentration near the maximum value is better (see text for more explanation)

3.3 Static setting with partial information

In the settings discussed so far, we have assumed that two important sets of parameters, the visit probabilities of user profiles $\{U_i\}$ and their click probabilities for each advertising campaign $\{p_{i,k}\}$ are known. However, this is a rather strong assumption and in reality these probabilities are hardly known in advance; instead, they have to be estimated based on observations, such as the profiles of the existing visitors, the advertising campaigns that have been displayed to them and their responsive actions (i.e., whether they have clicked on a displayed advertisement or not). An accurate prediction of these probabilities results in the display of more attractive advertisements to the web site visitors.

Once this estimation problem is solved, one has to deal with probabilities to decide on which advertisement to display. This problem of decision making in face of uncertainty raises the exploration/exploitation dilemma, one having to balance the exploitation of what is already known, with the exploration of new, potentially better, decisions.

We discuss these two issues in the next two sections.

3.3.1 Estimating the probabilities

The simplest way to estimate unknown probabilities would be to use maximum likelihood estimation. In our problem, the profile of a visitor can be considered a categorical random variable $\mathbf{U}$ with profile U_i having an estimated visit probability of $\hat{v}_i$, and the click of a visitor from user profile U_i on an advertisement from advertising campaign Ad_k can be considered a Bernoulli random variable $\mathbf{p}_{i,k}$ with success probability $\hat{p}_{i,k}$.

Let $visit_i$ denote the total number of visitors from user profile U_i that have visited the web site at time $0 \leqslant t$, then the maximum likelihood estimate of $\hat{v}_i$ will be $visit_i/(t+1)$, and similarly the maximum likelihood estimate of $\hat{p}_{i,k}$ at time t will be $click_{i,k}/display_{i,k}$ where $click_{i,k}$ is the number of times that visitors from user profile U_i clicked on advertisement Ad_k and $display_{i,k}$ is the number of times Ad_k had been displayed to them[8]. Since $visit_i$ values are initially 0, the estimates will also be 0 until we observe a visit from the corresponding user profiles. In order remedy this situation, it is customary to assign a prior ϑ_i, e.g., 1, for each user profile and define $\hat{v}_i$ as

$$\hat{v}_i = \frac{visit_i + \vartheta_i}{t + 1 + \sum_{i=1}^{N} \vartheta_i}.$$

The priors of click probabilities can also be assigned in a similar manner. In practice, as the number of visits is high and the number of user profiles is low, the maximum likelihood estimates of visit probabilities will be quite accurate.

[7] Typical values include 0.90, 0.95, and 0.99.

[8] For brevity, the time indices have been dropped from $visit_i$, $display_{i,k}$ and $click_{i,k}$.

Alternatively, we can employ Bayesian maximum a posteriori estimates using the conjugate priors. The conjugate priors of the categorical and Bernoulli distributions are Beta and Dirichlet distributions, respectively. If $Beta(\alpha_{i,k}, \beta_{i,k})$ is the Beta prior with hyper-parameters $\alpha_{i,k}$ and $\beta_{i,k}$ for click probability $p_{i,k}$, then the posterior at time t is the Beta distribution with hyper-parameters $\alpha_{i,j} + click_{i,k}$ and $\beta_{i,j} + display_{i,k} - click_{i,k}$. Setting both hyper-parameters to 1 corresponds to having a uniform prior. At time t, the posterior of the prior Dirichlet distribution with hyper-parameters v_i for $\mathbf{U}$ will have hyper-parameters $v_i + visit_i$. The initial hyper-parameters can be guessed or determined empirically based on historical data. As we will see later in the experiment section, choosing good priors may have a significant effect on the outcome.

By estimating probabilities at each time step (or periodically) and replacing the actual values with the corresponding estimates, we can use the approach presented in the previous section to determine allocations (optimal up to the accuracy of the estimations) and choose advertising campaigns to display. For maximum a posteriori estimates, the mode of the posterior distribution can be used as a point estimate and a single instance of the problem can be solved, or several instances of the problem can be generated by sampling probabilities from the posterior distributions, solved separately and then the resulting allocations can be merged (for example taking their mean; note that, in this case the final allocations will likely be not bound to the initial constraints).

3.3.2 Exploration-exploitation trade-off

As in many online learning problems, one important issue is the need for balancing the exploitation of the current estimates and exploration, i.e., estimation of the unknown or less-known (e.g., with higher variance) parameters. Using the solution of the optimization problem without introducing any additional exploration may introduce substantial bias to the results. This exploration/exploitation trade-off problem can be formulated as a multi-arm bandit problem (with the advertising campaigns in the role of arms). Based on the multi-arm bandit framework, exploration can be introduced to the allocation policy in various ways, among which we mention the following two:

- **Policy-modification**

The existing non-exploratory policies can be augmented with an additional mechanism in order to have exploration. This may be achieved by an ε-greedy in which the underlying policy is followed with a high probability $1 - \varepsilon$, and a running advertising campaign is chosen at random with a small probability ε. One can derive other possible solutions from the bandit literature, such as the UCB rule [2]. Standing for Upper-Confidence Bound, UCB is a very simple way to achieve asymptotically optimal policy to choose the best action among a set of available actions. Each action is associated to a certain average return; the principle consists in sampling each action, gathering for each its average observed return $\bar{r_i}$, and the number of times each action has been selected n_i. After n actions have been performed, the next action is selected as being the one that maximize the UCB bound: $r_i + \sqrt{\frac{C \ln n}{n_i}}$, where C is an appropriately tuned constant.

- **Estimation-modification**

In this approach, the probability of click estimates are systematically modified (before solving the optimization problem) in order to favor the advertising campaign and user profile couples according to the uncertainty on their estimation based on the following principle: *the more uncertain the estimate, the more exploration may be rewarding*. By giving them artificially a higher probability of click tends to favor their use, and consequently the exploration. For this purpose, [3] use Gittins indices. Similarly one can also use UCB indices associated with the estimates, or with a value sampled from the posterior Beta distribution over the expected reward (see [4]). Empirically, this second way of increasing exploration does not seem to work as well as the first one (for example, ε-greedy with fine-tuned ε) especially if we do not re-plan at each time step. We believe that the reason for this situation is that such methods lead to solutions that only explore the most uncertain areas of the search space.

4 Dynamic setting

Under the static setting of the problem, there are two main constraints: the set of advertising campaigns is known in advance and, consequently, the time horizon is fixed. In the more general and realistic dynamic setting, we remove these constraints. The time horizon is no longer fixed, i.e., does not have a limited length T but instead it is assumed that T is infinite; furthermore, new advertisement campaigns may appear with time. Thus, to the 3 aforementioned categories of advertising campaigns (scheduled, running, and expired), we add a new category made of the yet-unknown campaigns, that is, the advertising campaigns that will come into play in the future, but which future existence is not yet known. In contrast to these unknown advertising campaigns, scheduled, running, and expired advertising campaigns will be qualified as “known”.

In the next two subsections, we will consider two main cases in which either a generative model of advertising campaigns is available, or not. Given a set of parameters and the current state, a generative model generates a stream of advertisement campaigns during a specified time period, together with all related-information, such as, the click probabilities of user profiles for each generated advertising campaign.

4.1 Model based resolution

When a generative model of advertising campaigns is available, it can be utilized to compensate for the uncertainty in future events. In this case, in addition to the set of known advertising campaigns, the model allows us to generate a set of hypothetical unknown advertising campaigns, for example, up to H_{max}[9], simulating what may happen in future, and include them in the planning phase of NOSEED. By omitting allocations made for these hypothetical advertising campaigns from the (optimal) allocation scheme found by solving the optimization problem, display probabilities that inherently take into consideration the effects of future events can be calculated. Note that this would introduce a bias in the resulting policies which can be reduced by running multiple simulations and combining their results as discussed before.

4.2 Model free resolution

When a generative model is not available, we have an incomplete and uncertain image of the timeline; we only have information about known advertising campaigns, and new advertising campaigns appear periodically or randomly according to a model which is unknown. In this setting, at any time step t, the set of known advertising campaigns (running or scheduled) implies a maximum time horizon H_{max}. Although, it is possible to apply the aforementioned methods and calculate the allocations for the known advertising campaigns, doing so would ignore the possibility of the arrival of new advertising campaigns that may overlap and interact with the existing ones; the resulting long-term policies may perform well if the degree of dynamism in the environment is not high. On the contrary, one can focus only on short or medium-term conditions omitting the scheduled advertising campaigns that start after a not-too-distant time H in the future, i.e., do planning for the advertising campaigns within the chosen planning horizon. The resulting policies will be greedier as H is smaller and disregard the long-time interactions between the existing advertising campaigns; however, they will also be less likely to be affected by the arrival of new campaigns. An example that demonstrates the effect of the planning horizon on the resulting policies is presented in Fig. 6. For such policies, choosing the optimal value of the planning horizon is not trivial due to the fact that it strongly depends on the unknown underlying model. One possible way to overcome this problem would be to solve the problem for a set of different planning horizons $H_1, \ldots, H_u = H_{max}$, and then combine the resulting probability distributions of advertising campaign displays (such as by majority voting).

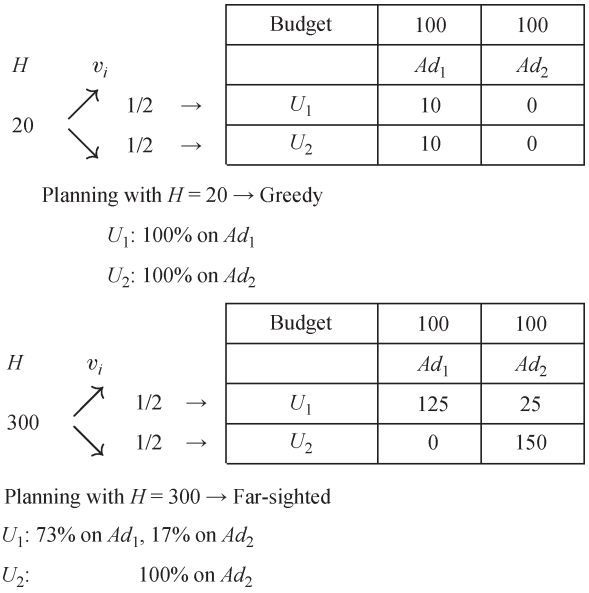


Fig. 6 The effect of the planning horizon H. Ad_1 and Ad_2 start at time 0 and have the same unit profit per click. The click probabilities are $p_{1,1} = 0.8, p_{1,2} = 0.1$ for the user profile U_1 and $p_{2,1} = 0.8, p_{2,2} = 0.5$ for the user profile U_2. Both profiles have the same visit probability

The sketch of NOSEED algorithm is presented in Fig. 7 and the complete algorithm can be found in the Appendix.

5 Related work

We review the existing work on the problem of advertisement selection for display on web pages, and related problems. We also discuss our own work in respect to these works.

The oldest reference we were able to spot is [5] who mixed a linear program with a simple estimation of CTR to select advertisements to display. In this work, no attention is paid to the exploration/exploitation trade-off and more generally, the problem of the estimation of the CTR is very crudely addressed. Then, [3] introduce a multi-arm bandit approach to balance exploration with exploitation. Their work is based on display proportions, that is unlimited resources; they also

[9] We will use H to denote the planning horizon and differentiate it from the time horizon of the problem T.

```
Initialize the visit probability estimates for each user profile.
while there is a request do
    Let U_i be the user profiole of the current visitor
    if there are new advertising campaigns then
        Initialize the click probability estimates of the new campaigns for each user profile.
    end if
    if planning is required then
        Solve the optimization problem and determine the display allocations of advertising campaigns.
    end if
    Choose an advertising campaign Ad_k basedon U_i and the display allocations of advertising campaigns for
the current time interval (with exploration).
    Display an advertisement from Ad_k to the current visitor and note the outcome (i.e. click or no click).
    Update visit and click probability estimates according to the outcome.
end while
```

Fig. 7 Sketch of the NOSEED algorithm: NOSEED selects advertising campaigns by solving the optimization problem from times to times, and then exploiting its solution to display a certain amount of advertisements

deal with a static set of advertisements. This was later improved by [6] who deal with the important problem of multi-impression of advertisements on a single page; they also deal with the exploration/exploitation trade-off by way of Gittins indices. Ideas drawn from their work on multi-impression may be introduced in ours to deal with that issue.

Aiming at directly optimizing the advertisement selection, side information (information about the type of advertisement, page, date of the request, ...) is used to improve the accuracy of prediction in several recent papers [7–11]. Interestingly, [12] also deals with the multi-impression problem. However, all these works do not consider finite budget constraints, and finite lifetime constraints, as well as the continuous creation of new advertising campaigns; they also do not consider the CTR estimation problem. Recently, [11] focuses on the exploration/exploitation trade-off and proposes interesting ideas that may be combined to ours (varying ε in the ε-greedy strategy, and taking into account the history of the displays of an advertisement). Though not dealing with advertisement selection but news selection, which implies that there is no profit maximization, and no click budget constraint, but merely maximization of the amount of clicks, [13, 14] investigate a multi-arm bandit approach.

Some works have specifically dealt with the accurate prediction of the CTRs, either in a static setting [15], or dealing with a dynamic setting, and non stationary CTRs [16]. [17, 18] also use a hierarchically organized side information on advertisements and pages. Recently, the extent of the content relevance between the pages and the personal interests of users based on intention and sentiment analysis are also considered for improving the predictions [19].

A rather different approach is that of [20] who treated this problem as an on-line bipartite matching problem with daily budget constraints. However, it assumed that we have no knowledge of the sequence of appearance of the profile, whereas in practice we often have a good estimate of it. [21] tried then to take advantage of such estimates while still maintaining a reasonable competitive ratio, in case of inaccurate estimates. Extensions to click budget were discussed in the case of extra estimates about the click probabilities. Nevertheless, the daily maximization of the income is not equivalent to a global maximization.

6 Experiments

We do not see any way to provide a relevant theoretical assessment of this work regarding the performance of the algorithm. Indeed, the algorithm we propose is aimed at dealing with large problems in an efficient way, efficient meaning with the constraint of rather short answering time (“quasi real-time”, that is in the order of the micro-second to decide which advertisement to display). Clearly this constraint on time requires an approximate solution to the problem we consider; however, even if we remove this constraint on time, we are unable to solve exactly the problem we wish to solve within a reasonable amount of time, for a significant size of the problem, so that we can not compare our results with the optimal results. All this makes the experimental assessment a necessity.

Assessing live the approach we propose is impossible; this is a well-known issue of the community. Even if we plugged NOSEED in a real advertisement server, we would have absolutely no way to assess its performance in comparison with an other algorithm. Some workarounds have been proposed

(see e.g., [22][10]), but the issue is clearly not settled today. So, we set up a set of experiments to study its performance, and study how different tunings of the parameters, and how different display policies affect the performance of the algorithm. We report on these experiments in the next sections.

6.1 The generative model

To fit the real-world problem, our approach was tested on a toy-model designed with experts from the research division of Orange Labs. Orange Labs is an important commercial web actor with tens of millions of page views per day over multiple web sites. We took care that each advertising campaign has its own characteristics that more or less appeal to the different visitor profiles.

The model assumes that each advertising campaign Ad_k has a *base click probability* p_k that is sampled from a known distribution (e.g., uniform in an interval, or normally distributed with a certain mean and variance). As clicking on an advertisement is in general a rare event, the base click probabilities are typically low (around 10^{-4}). The click probability of a visitor from a particular user profile is then set to $p_{i,k} = p_k\gamma^{\mathbf{d}-1}$ where $\gamma > 1$ is a predefined multiplicative coefficient, and the random variable $\mathbf{d}$ is sampled from the discrete probability distribution with parameter n that has the following probability mass function $Pr[\mathbf{d} = x] = 2^{n-x}/(2^n - 1), 1 \leqslant x \leqslant n$. When n is small, all advertising campaigns will have similar click probabilities that are close to the base click probability; as n increases, some advertising campaigns will have significantly higher click probabilities for some but not all of the user profiles. Note that, the number of such assignments will be exponentially low; if γ is taken as fixed, then there will be twice as many advertising campaigns with click probability p compared to those with click probability γp. This allows us to effectively model situations in which a small number of advertising campaigns end up being popular in certain user profiles.

In the experiments we used two values for the γ parameter, 2 and 4; experts recommended the use of the latter value, but as we will see shortly, having a higher value for γ may be advantageous for the greedy policy. The value of n is varied between 2 and 6.

6.2 The experiments

Similar to the way that we introduce the proposed method in the previous sections, in the experiments we will also proceed from simpler settings to more complex ones. We opted to focus on core measures and therefore omit some of the extensions that have been discussed in the text.

We begin with the static setting with full information, and uncertainty (Section 2.1.2). In this setting, we consider a fixed time horizon of one day which is assumed to be equivalent to $T = 4 \times 10^6$ page visits. The distribution of user profiles is uniform and the budget and lifetime of advertising campaigns are also sampled uniformly from fixed intervals. In order to determine the starting times of advertising campaigns, we partitioned the time horizon into M equally spaced intervals (in our case 80) and set the starting time of each advertisement campaign to the starting time of an interval chosen randomly, such that the ending times do not exceed the fixed time horizon. The base click probability is set to 10^{-4}. We solved the optimization problem every 10^4 steps.

First, we consider a setting in which there is a single user profile ($N = 1$), and there are $K = 40$ advertising campaigns with an average $L_k = \frac{1}{10}T$, i.e., 1 tenth of the time horizon. All advertising campaigns have the same budget $B_k = B$. Figure 8 shows the relative performance of HLP policy with respect to the HEV policy for different values of the click probability generation parameter n and budgets. We can make two observations: all other parameters being fixed, HLP is more effective with increasing budgets, and the performance gain depends mainly on the value of γ. For $\gamma = 4$, which is

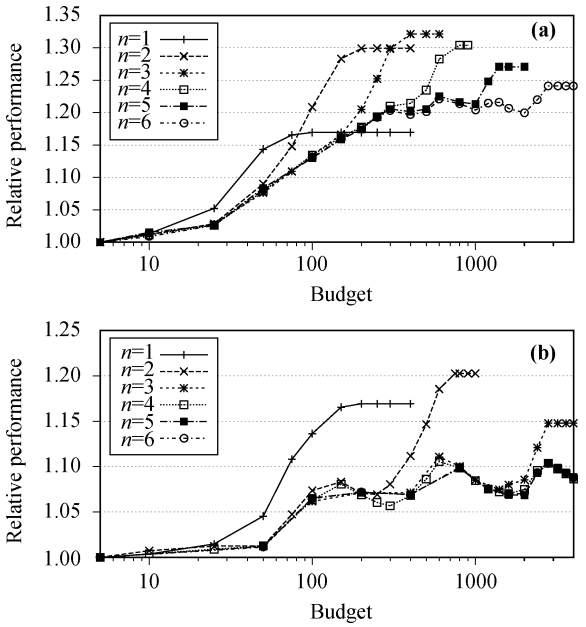


Fig. 8 The relative performance of the HLP policy with respect to the HEV policy for different values of the click probability generation parameter n and budget under the static setting with one user profile and 40 advertising campaigns. The value of γ is either 2 (a) or 4 (b) and the x-axis, i.e., budget B, is in logarithmic scale

[10] Also, Nicol O, Mary J, Preux P. ICML exploration & exploitation challenge: Keep it simple!, 2011, submitted.

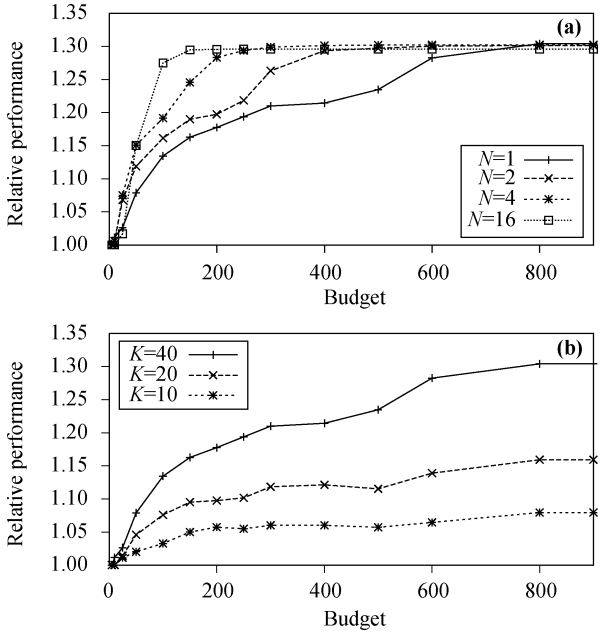


Fig. 9 The effect of the number of (a) user profiles N and (b) advertising campaigns K when other parameters are kept constant and n and γ are set to 2 and 4, respectively

considered to be a realistic value by experts, and reasonable budgets, the greedy policy would perform well. A similar situation also arises when the number of advertising campaigns (K) is low, whereas when the number of user profiles increases, non greedy policies taking longer terms consequences are better (Fig. 9).

Then, in order to isolate and figure out the effect of the overlapping of advertising campaigns on the performance of the algorithms, we conducted another set of experiments in which all advertising campaigns have high click probabilities, and their budget vary depending on their lifetimes. We set:

- n to 1 and sampled the base click probability p_k from a truncated Gaussian distribution with mean 1 and standard deviation 0.02,
- the lifetimes of advertising campaigns L_k are sampled uniformly from 0.5% to 5% of the time horizon,
- the budget B_k of each advertising campaign is set to λ times its lifetime,
- λ is sampled uniformly from the interval $[a, b]$,
- a and b are the parameters of the experiment.

As in the previous case, the time horizon is assumed to be $T = 4 \times 10^6$ page visits. Figure 10 shows the relative performances of HEV, SEV, HLP approaches, as well as the random policy for $K = 100$ advertising campaigns and different values of a and b. We can observe that when the budget to lifetime ratios ($\frac{B_k}{L_k}$) of all campaigns are either low, or high, the difference between the different approaches diminishes. This is due to the fact that in both cases, there is no particular need for taking into account long term consequences: when click probabilities are close to 1, the budget constraints can be satisfied easily when ratios are small (in short, the expected clicks will be grabbed whatever the policy is: no need to be smart), and when they are high choosing any running campaign is likely to end up with a click (in short, whatever the display policy is, however smart it is, budgets can not be fulfilled). However, when the advertising campaigns have diverse budget to lifetime ratios, the interactions between advertising campaigns do matter, and can be exploited by the planning-based approach, especially for low ratios (Fig. 10(a)). In this setting, similar to the toy example presented in Section 3.1.2, the performance of the greedy policy turns out to be inferior to that of random policy which chooses at each time step one of the running advertising campaigns with uniform probability (Fig.10(c)); hence, the stochastic version of the greedy policy, SEV, performs better than HEV (Fig. 10(b)).

Next, we tried longer static settings of over one week period with full, or partial information in which the advertising campaign lifetimes and their budget are more realistic (lifetimes ranging 2 – 5 days, budgets ranging from 500 to 4 000 clicks). The campaigns are generated on a daily basis at the beginning of a simulation, i.e., a set of seven to nine new advertisement are introduced every four million time steps. We tested different values for the click probability generation parameters. There were $N = 8$ user profiles with equal visit probabilities ($v_i = 1/8$). As presented in Fig. 11, in this setting although HLP policy performs better than the greedy policy, the performance gain remains limited. While the greedy policy quickly exploits and consumes new advertisements as they arrive, HLP tends to keep a consistent and uniform click rate at the beginning, and progressively becomes more greedy towards the end of the period (Fig. 12). Figure 13 shows the effect of the planning horizon H: H tunes whether we focus on the campaigns running in the near future (small value of H), or also take into account campaigns that will run in a more remote future (larger value of H). For this experiment, we increased the time horizon from one week to two weeks, and the planning horizon is varied from one day up to the entire time horizon. Note that, the intensity of the interactions between advertising campaigns, in terms of overlapping intervals, and their propagation through time are the main factors that determine the influence of the upcoming campaigns

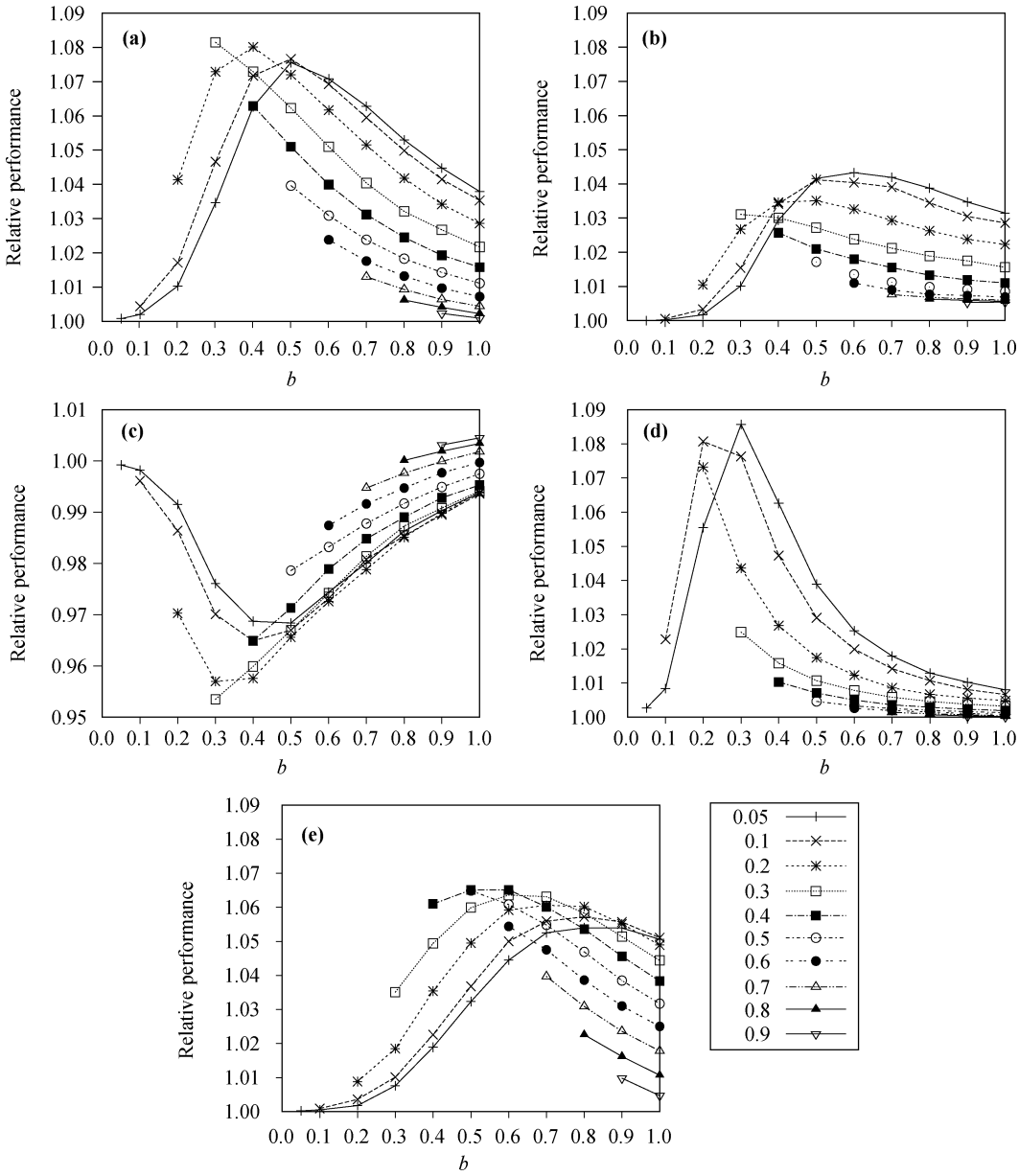


Fig. 10 The relative performances of different approaches for different budget ratios. Each curve presents the case in which the budget of each advertising campaign is set to λ times its lifetime such that λ is sampled uniformly from the interval $[a, b]$, where a is the value that corresponds to the curve and b is the value at the x-axis. (a) HLP vs. HEV, (b) HLP vs. SEV, and (c) HEV vs. random policy with 100 advertising campaigns; HLP vs. HEV with (d) 200 and (e) 50 advertising campaigns

over the display allocations for the currently running campaigns; in this and other experiments we observed that being very far-sighted may not be necessary.

As discussed in Section 3.3.2, when we move to the more realistic setting of partial information, the visits and click probabilities are not known in advance but instead are estimated online. In this setting, without sufficient exploration there is a risk of getting stuck in a local optima; we define local optima as a situation in which the values of some of the options are underestimated and these estimates cannot be improved because the corresponding options are not considered in the search process due to their seemingly low values.

To deal with the exploration-exploitation trade-off, we implemented two approaches: ε-greedy policy, and a UCB based approach. We studied their behavior under various settings in which:

- to increase the variance of the click probabilities, instead of using a fixed value, the base click probabilities p_k of the advertising campaigns are sampled from a Gaussian distribution with mean 0.001 and standard deviation 0.0002,

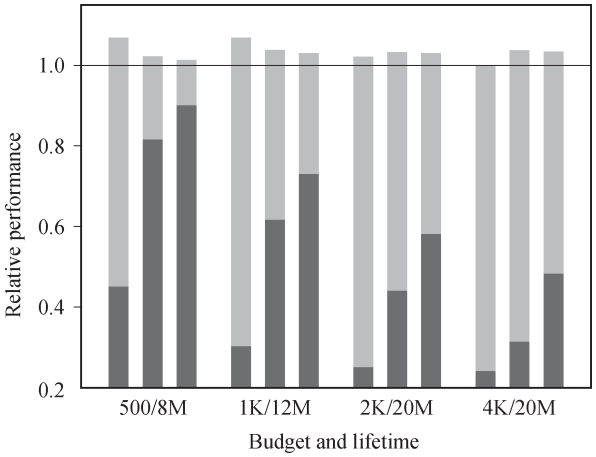


Fig. 11 The performance of the random (dark gray and lowest) and the HLP (light gray and highest) policies with respect to the HES policy under the seven days static setting for different budget (500 to 4 000), lifetime (2 — 5 days) and generation parameter n values. The three sets of bars in each group corresponds to the case where n is taken as to 2, 4, and 6 in that order

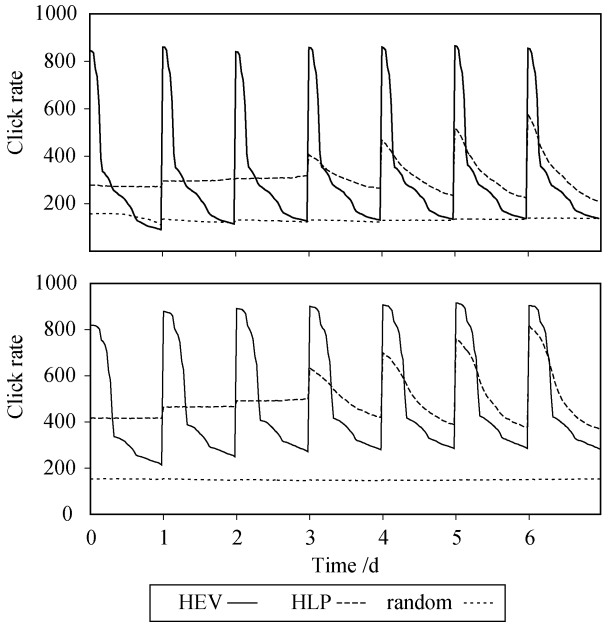


Fig. 12 The moving average of click rate for different policies under the seven day static setting; the lifetime of advertising campaigns is five days and their budgets are either 2 000 (top) or 4 000 (bottom)

- the time horizon is set to $T = 4 \times 10^6$ page visits,
- there are $K = 100$ advertising campaigns,
- their lifetimes L_k falls in the range 0.5% and 5% of the time horizon,
- budget to lifetime ratios $\frac{B_k}{L_k}$ falls in the interval $[0.1, 0.5]$,
- the multiplicative coefficient γ is set to 2.

We employed simple maximum likelihood estimates to estimate click probabilities. Figure 14 shows the results

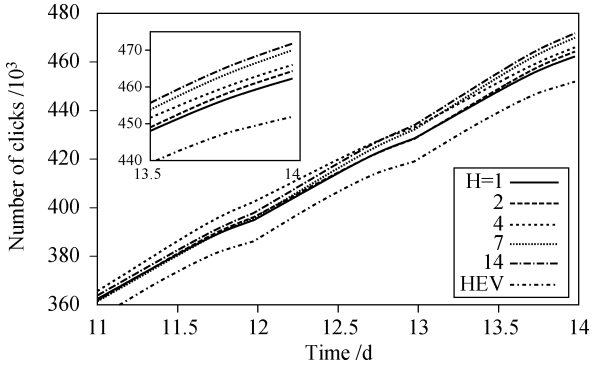


Fig. 13 The effect of horizon H (1, 2, 4, 7 and 14 days) in the 14 days static setting with full information; using less information than available hinders the performance

obtained using both approaches with HEV and HLP policies as a function of ε and the UCB tuning coefficient C. Each subfigure depicts the relative performances of these policies under partial information settings compared to the performance of the HLP policy assuming that the true click probabilities are known (i.e., full information case) for a certain value of click probability generation parameter n, ranging from 1 to 6. The figures highlight that although the performance of ε-greedy varies as a function of ε, for a wide range of values, it performs better than the UCB approach. The performance of the UCB approach is observed to be less sensitive to its tuning coefficient, especially with the HLP policy. It may still be possible that the UCB approach performs better than the ε-greedy approach for a particular value of the tuning coefficient (or a small interval), but fine-tuning the coefficient seems to be more challenging. Furthermore, although ε-greedy has a generally consistent pattern of performance across the full range of n for both HEV and HLP policies, in the UCB approach the performance of the HEV policy deteriorates relative to the HLP policy as n increases, that is, under conditions where the advertising campaigns end up having a wider range of non-homogeneous click probabilities. These results indicate that policy-modification may be a more viable option for balancing the exploitation of the current estimates and exploration.

Finally, we conducted experiments in the dynamic setting with partial information where the probabilities are not known in advance but estimated online. We employed an ε-greedy exploration mechanism with different values of ε and maximum a posteriori estimation with Beta priors. We used the same set of parameters as in two weeks static setting with full information, except that rather than generating all advertising campaigns at the beginning of the simulation, they are generated on a daily basis at the beginning of each day, i.e., a set of seven to nine new advertisement are

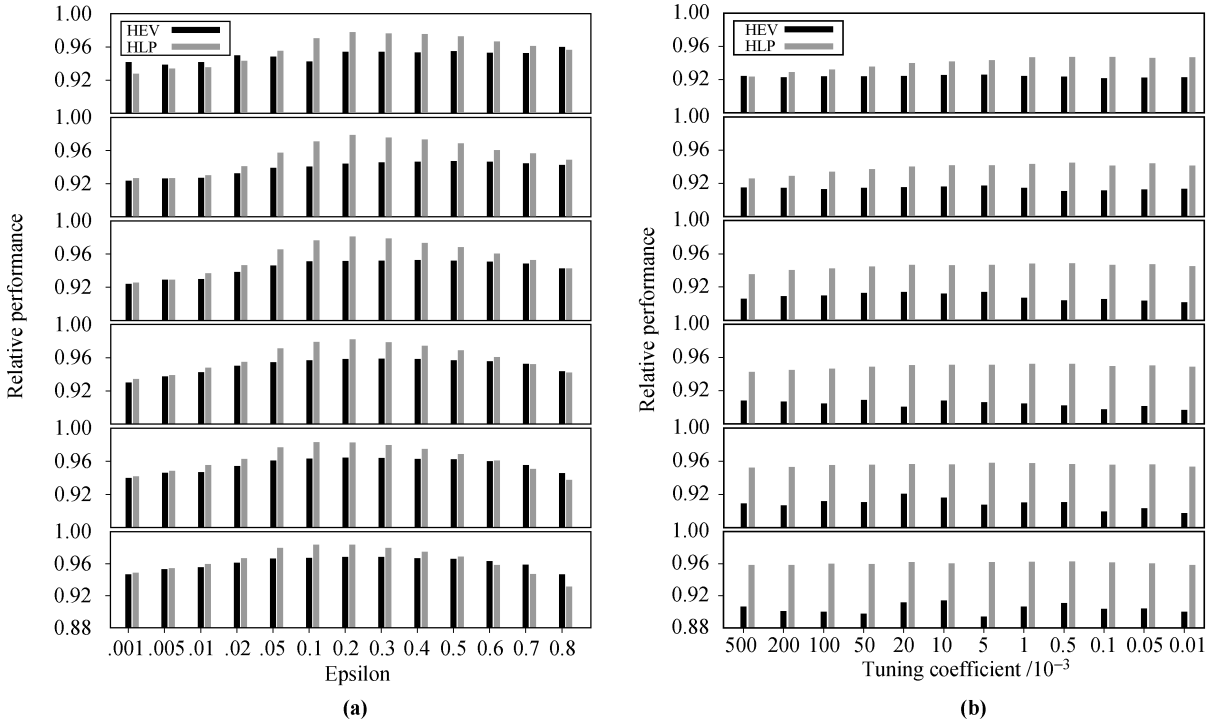


Fig. 14 The performances of HEV and HLP policies with ε-greedy (a) and UCB (b) selection in partial information case relative to the performance of the HLP policy with full information for different values of ε and UCB tuning coefficient; the click probability generation parameter n varies from 1 (top) to 6 (bottom)

introduced every 4 million time steps; the planning is done over all known advertising campaigns. The results presented in Fig. 15 show that HLP can perform better than HEV; however for both policies, the chosen set of hyper-parameters influences the outcome.

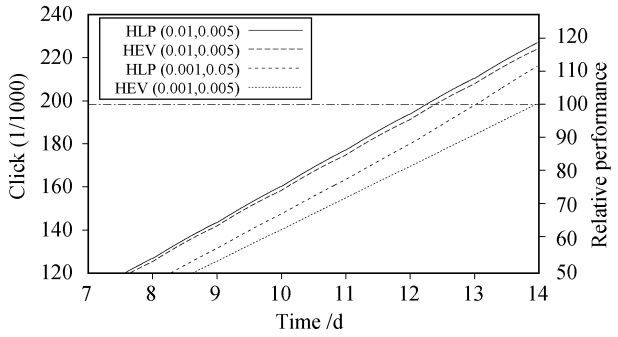


Fig. 15 The performance of HEV and HLP algorithms in the dynamic setting with partial information using ε-greedy exploration. The numbers in parentheses denote the values of the hyper-parameters of the Beta prior (α and β parameters are set to be equal to each other) and ε

7 Conclusion and future work

In this paper, we considered the advertisement selection problem for display on web pages. Aiming at considering the problem in the most realistic setting, and providing effective and efficient algorithms to perform this selection on a production system, we have formalized the problem by providing a series of increasing complexity settings. This let us discuss various algorithmic approaches, and clearly identify the issues. While defining this set of problems, we provided a way to effectively tackle this problem, and provided an experimental study of some of their key features. The experimental study is based on a realistic model, carefully designed with a major commercial Internet portal.

We have shown that optimizing advertisement display handling finite budgets and finite lifetimes in a dynamic and non stationary setting, is feasible within realistic computational time constraints. We have also given some insights in what can be gained by handling this constraint, depending on the properties of the advertisements to display. We have also exhibited that lifetime of the advertisements impact the overall performance, and so should be taken into account into the pricing policy. Moreover our work may be seen as a part of a decision aid tool. For instance, it can help to price the advertisements in the case in which a fraction of the advertising campaigns are in the "cost per display" model, while the rest is in the cost per click model. This is rather easy because the LP solution provides an estimation of the profit for each visitor profile.

Our work shows that depending on the parameters and characteristics of the existing or prospective advertising cam-

paigns, a simple greedy approach may perform well or one can benefit from using a more advanced solution, such as NOSEED, that takes into consideration the long term gains. Figure 8 illustrates how these parameters interact. To summarize, we may say that if there are few overlapping advertisements, or many advertisements with long lifetimes and good click rates, then we should be greedy. Between these two extreme solutions, one should consider the constraints associated to each advertising campaign.

This work calls for many further developments. A possibility is to solve the problem from the perspective of the advertiser, i.e., help the advertiser to set the value of a click, and adjust it optimally with respect to his/her expected number of visitors. It would be equivalent to a local sensitivity analysis of the LP problem. A more difficult issue is that of handling multiple advertisement displays on the same page. It may be possible to handle them by estimating the correlation between the advertisements, and trying to update multiple click probabilities at the same time. Some recent developments in the bandit setting [23] are interesting in this regard.

We are willing to draw some theoretical results on how far from the optimal strategy we are. Dealing with finite resources, under finite time constraints, in a dynamic setting makes that kind of study very difficult. An other work originates from the analysis of some real web server logs. We have already been very slightly using such source of information, but much more has to be done. We also think that it is important to go towards learning on-line the profiles of the visitors depending on their click behavior instead of having pre-existing ones.

Acknowledgements This research was supported, and partially funded by Orange Labs, under externalized research contract number CRE number 46 146 063 – 8360, and by Ministry of Higher Education and Research, Nord-Pas de Calais Regional Council and FEDER through the "Contrat de Plan État-Région (CPER) 2007-2013", and the contract "Vendeur Virtuel Ubiquitaire" of the "Pôle de compétitivité Industries du Commerce".

The model used in this paper, as well as its parameters have been designed with, and validated by, Orange Labs, to keep up with the essential characteristics of the real problem.

The realization of simulations were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research, INRIA, CNRS and RENATER and other contributing partners.

Appendixes

Notations

	Indices
t	Current time
i	Index of a user profile
k	Index of an advertising campaign
j	Index of a time interval
	User profile
N	# user profiles (Sec. 2)
U_i	User profile i (Sec. 2)
v_i	Probability that a certain visitor belongs to U_i (Sec. 2)
	Advertising campaign
C	The set of known (running) advertising campaigns at a given time (Sec. 3.1.2)
K	# of advertising campaigns (Sec. 3.1)
Ad_k	Advertising campaign k (Sec. 2)
S_k	Starting time of Ad_k (Sec. @2)
L_k	Lifetime of Ad_k (Sec. 2)
B_k	Budget of Ad_k (Sec. 2)
cp_k	Click profit of Ad_k (Sec. 2)
rb_k	$\leqslant B_k$: remaining budget of Ad_k (Sec. 2)
E_k	Ending time of Ad_k ($E_k = S_k + L_k$, Sec. 3.1.3)
$p_{i,k}$	Probability that a visitor $\in U_i$ clicks on an advertisement $\in Ad_k$ (Sec. 2)

	Time
T	Time horizon of the problem (Sec. 2.1.1.1)
H	Time horizon of the resolution (planning, Sec. 2.2.1)
M	# of time intervals (Sec. 3.1.3)
I_j	Time interval j (Sec. 3.1.3)
AI_j	The set of running advertising campaigns in I_j (Sec. 3.1.3)
IA_k	The set of time interval is which Ad_k is running (Sec. 3.1.3)
$visit_i$	# of visit $\in U_i$ for time $\leqslant t$ (Sec. 3.3.1)
$click_{i,k}$	# of times a user $\in U_i$ clicked on Ad_k, for time $\leqslant t$ (Sec. 3.3.1)
$display_{i,k}$	# of times an Ad_k has been displayed to a visitor $\in U_i$ for time $\leqslant t$ (Sec. 3.3.1)
	Display allocation (Sec. 3.1.3)
$a_{i,k,j}$	# of advertisement displays allocated to Ad_k in interval I_j for user profile U_i
$\hat{a}_{i,k,j}$	$= a_{i,k,j} / \sum_{Ad_k \in AI_j} a_{i,k,j}$ is the ratio of allocation of displays for U_i and Ad_k during I_j to the total allocations for that user profile in the same interval (forming a categorical distribution)
	Exploration policy parameters (Sec. 3.3.2)
ε	Parameter of the ε greedy-policy
C	Parameter of the UCB policy
	Miscellaneous
s	A state of the MDP (Sec. 3.1.1)
ϑ_i	Prior for the maximum likelihood estimation of v_i
λ (λ_k, $\lambda_{i,j}$)	Parameters of the Poisson distribution (its interpretation is a risk ratio, Sec. 3.2)
Λ_k	Risk ratio threshold (Sec. 3.2)
$\alpha_{i,k}, \beta_{i,k}$	Parameters of the Beta distribution (Sec. 3.3.1)
p_k	Base click probability for advertisements $\in Ad_k$ (Sec. 6.1)
$\gamma, n, [a, b]$	Parameters of the generative model (Sec. 6.1 and 6.2)

The NOSEED algorithm

Input: N: number of user profiles; T: time horizon; H: planning horizon; C: set of known advertising campaigns; hyper-parameters of click and visit probability estimators, (e.g. $\alpha_{i,k}, \beta_{i,k}$ for the Beta distributions).

Additional variables: C_{last}: set of known advertising campaigns at last planning; $\hat{p}_{i,k}$: probability distribution for the estimate of $p_{i,k}$ (a Beta distribution).

procedure C F N C
  for all $Ad_k \in C$ and $Ad_k \notin C_{last}$ /* *New campaigns* */ **do**
    for $i = 1$ **to** N **do**
      $\hat{p}_{i,k} = Beta(\alpha_{i,k}, \beta_{i,k})$ /* *Initial click probability estimates* */
      $click_{i,k} = display_{i,k} = 0$
    end for
  end for
end procedure

/*C: *set of advertising campaigns; rb: remaining budgets of advertising campaigns in* C, rb_k *denotes the remaining budget of* Ad_k */

function F A (t, C, rb)
  $boundaries = \{min(t, S_k)\} \cup \{E_k\}, \forall Ad_k \in C$ such that $S_k \leqslant t + H$.
  Let $t_0, \ldots, t_M$ is the sorted list of the elements of *boundaries* and define intervals $I_j = [t_{j-1}, t_j], 1 \leqslant j \leqslant M$.
  for $j = 1$ **to** M **do**

15: $l_j = t_j - t_{j-1}$ /*length of the interval */
16: $AI_j = \{Ad_k | S_k < t_j \leqslant E_k\}$ /*the set of campaigns in each interval */
17: **end for**
18: **for all** $Ad_k \in C$ **do**
19: $IA_k = \{I_j | Ad_k \in AI_j\}$ /*the set of intervals that cover Ad_k */
20: **end for**
21: $\mathcal{A} = \{I_j | AI_j \neq \emptyset\}$ /*non-empty intervals */
22: Let $a_{i,k,j}$ denote the number of displays allocated to the campaign Ad_k in interval I_j for the user profile U_i.
23: Solve the linear program maximize $\sum_{I_j \in \mathcal{A}} \sum_{Ad_k \in AI_j} cp_k p_{i,k} a_{i,k,j}$ with the set of constraints
24: (a) $\sum_{Ad_k \in AI_j} a_{i,k,j} \leqslant v_i l_j, \forall\, U_i, I_j \in \mathcal{A}$
25: (b) $\sum_{U_i} \sum_{I_j \in IA_k} p_{i,k} a_{i,k,j} \leqslant rb_k, \forall\, Ad_k \in C$
26: (c) $\sum_{U_i} \sum_{Ad_k \in AI_j} a_{i,k,j} \leqslant l_j, \forall I_j \in \mathcal{A}$
27: Let *intervals* be the list of intervals I_j and *allocations* be the list of display allocations $a_{i,k,j}$.
28: **return** [*intervals*, *allocations*]
29: **end function**
30:
31: **function** D P (t)
32: $C_{last} = C$ /*Save the current set of advertising campaigns */
33: **if** a generative model is available **then**
34: Generate a set of hypothetical campaigns C'. /* up to time $t + H$ */
35: $C_{current} = C \cup C'$
36: **else**
37: $C_{current} = C$
38: **end if**
39: Update click probability estimates, i.e., $\hat{p}_{i,k} = Beta(\alpha_{i,k} + click_{i,k}, \beta_{i,k} + display_{i,k})$
40: **if** using estimation-modification approach
41: Modify $\hat{p}_{i,k}$, e.g. using Gittins or UCB indices /* see Section 3.3.2 */
42: **end if**
43: **for all** $Ad_k \in C_{current}$ **do**
44: **if** risk factor $\Lambda_k < 1$ **them** /*Modify budget limits for dealing with uncertainty, see Section 3.2 */
45: $rb'_k = \arg\min_\lambda Pr(Po(\lambda) > rb_k - 1) \geqslant \Lambda_k$
46: **else**
47: $rb'_k = rb_k$
48: **end if**
49: **end for**
50: **return** F A $(t, C_{current}, rb')$
51: **end function**
52:
53: /* U_i is the profile of a visitor. intervals and allocations are the list of intervals and display allocations in each interval as determined in the planning phase, i.e. DoPlanning function, respectively; I_j denotes the j^{th} interval, and $a_{i,j,k}$ denotes the number of advertisement displays allocated to $Ad_k \in I_j$ for U_i. */
54: **function** C C $t, U_i, intervals, allocations$
55: Determine $I_j = [t_{j-1}, t_j] \in intervals$ such that $t_{j-1} \leqslant t < t_j$ /* current interval */
56: Let $AI_j \subseteq C$ be the set of running campaigns that span I_j.
57: **if** $AI_j = \emptyset$ **then**
58: **return** $\emptyset$ /* There is no running advertising campaign */
59: **end if**
60: $\bar{a}_{i,j} = \sum_{Ad_k \in AI_j} a_{i,k,j}$ /* Total allocations in this interval */

```
    if ā_{i,j} > 0 then
      for all Ad_k ∈ AI_j do
        â_{i,k,j} = a_{i,k,j}/ā_{i,j} /* Calculate display probabilities */
      end for
      Choose an advertising campaign Ad_k based on â_{i,k,j} /* e.g. using HLP or SLP with exploration, if any */
      a_{i,k,j} = a_{i,k,j} − 1 /* Update the allocation for Ad_k */
    else
      Choose a campaign Ad_k from AI_j (e.g. randomly). /* No allocations to display for this user profile */
    end if
    return k
end function

/* The main loop */
C_last = ∅
for i = 1 to N do /* Initialize visit probability estimates */
    visit_i = 1, v_i = 1/N /* ϑ_i = 1 */
end for
t = 0 /* Set time to 0 */
while there is a request do
    Let U_i be the user profile of the current visitor.
    C      F   N   C
    if t = 0 or planning is required /* e.g. when an advertising campaign expires or periodically */ then
        [intervals, allocations] =D  P          (t)
    end if
    k =C       C          (t, U_i, intervals, allocations)
    if Ad_k ≠ ∅ then
      display_{i,k} = display_{i,k} + 1
      if visitor clicks on Ad_k then
        click_{i,k} = click_{i,k} + 1 /* Update the click count of the user profile */
        rb_k = rb_k − 1 /*Update the remaining budget of the advertising campaign */
      end if
    end if
    t = t + 1
   /* Update the visit probability estimates */
    visit_i = visit_i + 1
    for i = 1 to N do
      v_i = visit_i/(t + N)
    end for
end while
```

References

1. Puterman M L. Markov Decision Processes: Discrete Stochastic Dynamic Programming. New York: John Wiley & Sons, 1994
2. Auer P, Cesa-Bianchi N, Fischer P. Finite-time analysis of the multiarmed bandit problem. Machine Learning, 2002, 47(2-3): 235–256
3. Abe N, Nakamura A. Learning to optimally schedule Internet banner advertisements. In: Proceedings of the 16th International Conference on Machine Learning. 1999, 12–21
4. Granmo O C. A Bayesian learning automaton for solving two-armed Bernoulli bandit problems. In: Proceedings of the 7th International Conference on Machine Learning and Applications. 2008, 23–30
5. Langheinrich M, Nakamura A, Abe N, Kamba T, Koseki Y. Unintrusive customization techniques for web advertising. Computer Networks, 1999, 31(11-16): 1259–1272
6. Nakamura A, Abe N. Improvements to the linear programming based scheduling of web advertisements. Electronic Commerce Research,

2005, 5(1): 75–98

7. Pandey S, Agarwal D, Chakrabarti D, Josifovski V. Bandits for taxonomies: a model-based approach. In: Proceedings of the 7th SIAM International Conference on Data Mining. 2007
8. Langford J, Zhang T. The epoch-greedy algorithm for multi-armed bandits with side information. In: Proceedings of 20th Advances in Neural Information Processing Systems. 2008, 817–824
9. Wang C C, S.R. Kulkarni S R, Poor H V. Bandit problems with side observations. IEEE Transactions on Automatic Control, 2005, 50(3): 338–355
10. Kakade S M, Shalev-Shwartz S, Tewari A. Efficient bandit algorithms for online multiclass prediction. In: Proceedings of the 25th International Conference on Machine Learning. 2008, 440–447
11. Li W, Wang X, Zhang R, Cui Y, Mao J, Jin R. Exploitation and exploration in a performance based contextual advertising system. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2010, 27–36
12. Pandey S, Olston C. Handling advertisements of unknown quality in search advertising. In: Proceedings of 18th Advances in Neural Information Processing Systems. 2006, 1065–1072
13. Agarwal D, Chen B, Elango P. Explore/exploit schemes for web content optimization. In: Proceedings of the 9th IEEE International Conference on Data Mining. 2009, 1–10
14. Li L, Chu W, Langford J, Schapire R E. A contextual-bandit approach to personalized article recommendation. In: Proceedings of the 19th International Conference on World Wide Web. 2010, 661–670
15. Richardson M, Dominowska E, Ragno R. Predicting clicks: estimating the click-through rate for new ads. In: Proceedings of the 16th International Conference on World Wide Web. 2007, 521–530
16. Agarwal D, Chen B C, Elango P. Spatio-temporal models for estimating click-through rate. In: Proceedings of the 18th International Conference on World Wide Web. 2009, 21–30
17. Agarwal D, Broder A, Chakrabarti D, Diklic D, Josifovski V, Sayyadian M. Estimating rates of rare events at multiple resolutions. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2007, 16–25
18. Wang X, Li W, Cui Y, Zhang B, Mao J. Clickthrough rate estimation for rare events in online advertising. In: Hua X S, Mei T, Hanjalic A, eds. Online Multimedia Advertising: Techniques and Technologies. Hershey: IGI Global, 2010
19. Fan T K, Chang C H. Sentiment-oriented contextual advertising. Knowledge and Information Systems, 2010, 23(3): 321–344
20. Mehta A, Saberi A, Vazirani U, Vazirani V. Adwords and generalized on-line matching. In: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science. 2005, 264–273
21. Mahdian M, Nazerzadeh H. Allocating online advertisement space with unreliable estimates. In: Proceedings of the 8th ACM Conference on Electronic Commerce. 2007, 288–294
22. Langford J, Strehl A, Wortman J. Exploration scavenging. In: Proceedings of the 25th International Conference on Machine Learning. 2008, 528–535
23. Koolen W M, Warmuth M K, Kivinen J. Hedging structured concepts. In: Proceedings of the 23rd Annual Conference on Learning Theory. 2010, 93–105

Sertan Girgin has two BSc degrees, one in Computer Engineering and the other in Mathematics, and a PhD degree in Computer Engineering from Middle East Technical University (METU), Turkey, 2007. He was a visiting researcher at the Department of Computer Science, University of Calgary, Canada, in 2006. For three years, Dr. Girgin worked as a postdoc researcher in team-project Sequel, INRIA Lille Nord Europe, France. Currently, he is with Google, Inc. His research interests include sequential learning, evolutionary computation, distributed AI and multi-agent systems.

Jérémie Mary is Assistant professor at University of Lille and member of the SequeL team at INRIA. He is also member of the european network of excellence PASCAL 2. He obtained his PhD on online machine learning, at Université Paris XI advised by Michèle Sebag and Antoine Cornuéjols. His mains interests in research are related to Machine Learning and more specifically sequential data. With Olivier Nicol (PhD student), he won the ICML'2011 challenge Exploration&Exploitation on data provided by Adobe.

Philippe Preux defended his PhD in computer science in 1991, at the Université de Lille, France. He is currently professor in computer science at the Université de Lille. He is the head of the SequeL research group, affiliated to both INRIA, CNRS, and the university. Since 1991, his research focuses on adaptive systems. He has worked on genetic algorithms and metaheuristics for combinatorial optimization; he then moved to reinforcement learning. These days, his main research interests are statistical learning on sequential data, data mining and sequential decision making in face of very large amounts of data, in non stationary environments.

Olivier Nicol holds a Master's degree in computer science with specialization in software engineering from the University of Lille, France. He is now studying for a PhD under Philippe Preux and Jérémie Mary in the SequeL (Sequential Learning) team at INRIA Lille. His main research interests lie in

Machine learning and especially using sequential data such as web logs. For instance he is currently working on how to use data to evaluate recommendation policies (and more generally contextual bndits policies) without having to actually test them on the real world. Together with Jérémy Mary he won the ICML 2011 Exploration and Exploitation challenge which was about balancing exploration and exploitation in order to efficiently recommend items to visitors on an Abode web site.

Chapter 7

Ergodic Processes

Consistent algorithms for clustering time series

Azadeh Khaleghi azadeh.khaleghi@inria.fr
Daniil Ryabko daniil@ryabko.net
Jérémie Mary Jeremie.Mary@inria.fr
Philippe Preux Philippe.Preux@inria.fr
SequeL-INRIA/LIFL-CNRS,
Université de Lille, France

Editor: Gábor Lugosi

Abstract

[1] The problem of clustering is considered for the case where every point is a time series. The time series are either given in batch (offline setting), or they are allowed to grow with time and new time series can be added along the way (online setting). We propose a natural notion of consistency for this problem, and show that there are simple, computationally efficient algorithms that are asymptotically consistent under extremely weak assumptions on the distributions that generate the data. The notion of consistency is as follows. A clustering algorithm is called consistent if it places two time series into the same cluster if and only if the distribution that generates them is the same. In the considered framework the time series are allowed to be highly dependent, and the dependence can have arbitrary form. For the case of a known number of clusters, the only assumption we make is that the (marginal) distribution of each time series is stationary ergodic. No parametric, memory or mixing assumptions are made. For the case of unknown number of clusters, stronger assumptions are provably necessary, but it is still possible to devise non-parametric algorithms that are consistent under very general conditions. The theoretical findings of this work are illustrated with experiments on both synthetic and real data.

Keywords: clustering, time series, ergodicity, unsupervised learning

1. Introduction

Clustering is a widely studied problem in machine learning, and is key to many applications in almost all fields of science. The goal is to partition a given dataset into a set of non-overlapping *clusters* in some natural way, thus hopefully revealing an underlying structure in the data. In particular, this problem for time series data is motivated by many research problems from a variety of disciplines, such as marketing and finance, biological and medical research, video/audio analysis, etc., with the common feature that the data are abundant while little is known about the nature of the processes that generate them.

The intuitively appealing problem of clustering is notoriously difficult to formalize. An intrinsic part of the problem is a similarity measure: the points in each cluster are supposed to be close to each other in some sense. While it is clear that the problem of finding the appropriate similarity measure is inseparable from the problem of achieving the clustering

1. This is an extended version of two conference papers: Ryabko (2010b) and Khaleghi et al. (2012).

objectives, currently there are no formulations of the clustering problem that would encompass this aspect. Instead, the available formulations simply assume the similarity measure to be given: it is either some fixed metric, or just a matrix of similarities between the points. Even with this simplification, it is still unclear how to define good clustering. Thus, Kleinberg (2002) presents a set of fairly natural properties that a good clustering function should have, and proceeds to demonstrate that there is no clustering function with these properties. A common approach is therefore to fix not only the similarity measure, but also some specific objective function — typically, along with the number of clusters — and to construct algorithms to maximize this objective. However, even this approach has some fundamental difficulties, albeit of a different, this time computational, nature: already in the case where the number of clusters is known, and the distance between the points is set to be the Euclidean distance, optimizing some fairly natural objectives (such as k-means) turns out to be NP hard (Mahajan et al., 2009).

In this paper we consider a subset of the clustering problem, namely, clustering time series data. That is, we consider the case where each data point is a sample drawn from some (unknown) time-series distribution. At first glance this does not appear to be a simplification (indeed, any data point can be considered as a time series of length 1). However, note that time series present a different dimension of asymptotic: with respect to their length, rather than with respect to the total number of points to be clustered. "Learning" along this dimension turns out to be easy to define and allows for the construction of consistent algorithms under most general assumptions. Specifically, the assumption that each time series is generated by a stationary ergodic distribution is already sufficient to estimate any finite-dimensional characteristic of the distribution with arbitrary precision, provided the series is long enough. Thus, in contrast to the general clustering setup, in the time-series case it is possible to "learn" the distribution that generates each given data point. No assumptions on the dependence between time series are necessary for this. The assumption that a given time series is stationary ergodic is one of the most general assumptions used in statistics; in particular, it allows for arbitrary long-range serial dependence, and subsumes most of the non-parametric as well as modelling assumptions used in the literature on clustering time series, such as i.i.d., (Hidden) Markov, or mixing time series.

This allows us to define the following clustering objective: *group a pair of time series into the same cluster if and only if the distribution that generates them is the same.*

Note that this intuitive objective is impossible to achieve outside the time-series framework. Even in the simplest case of a known number of Gaussian distributions, there is always a non-trivial likelihood for each point to be generated by any of the distributions. Thus, the best one can do is to estimate the parameters of the distributions, rather than to actually cluster the data points. The situation becomes hopeless in the non-parametric setting. In contrast, the fully nonparametric case is tractable in the time-series case, both in offline and online scenarios, as explained in the next section.

1.1 Problem setup

We consider two variants of the clustering problem in this setting: offline (batch) and online, defined as follows.

In the *offline (batch)* setting a finite number N of sequences $\mathbf{x}_1 = (X_1^1, \dots, X_{n_1}^1)$, $\dots, \mathbf{x}_N = (X_1^N, \dots, X_{n_N}^N)$ is given. Each sequence is generated by one of k different *unknown* time-series distributions. The target clustering is the partitioning of $\mathbf{x}_1, \dots, \mathbf{x}_N$ into k clusters, putting together those and only those sequences that were generated by the same time-series distribution. We call a batch clustering algorithm *asymptotically consistent* if, with probability 1, it stabilizes on the target clustering in the limit as the lengths $n_1, \dots, n_N$ of each of the N samples tend to infinity. The number k of clusters may be either known or unknown. Note that the asymptotic is not with respect to the number of sequences, but with respect to the individual sequence lengths.

In the *online* setting we have a growing body of sequences of data. Both the number of sequences as well as the sequences themselves grow with time. The manner of this evolution can be arbitrary; we only require that the length of each individual sequence tends to infinity. Similarly to the batch setting, the joint distribution generating the data is unknown. At time-step 1 initial segments of some of the first sequences are available to the learner. At each subsequent time-step, new data are revealed, either as an extension of a previously observed sequence, or as a new sequence. Thus, at each time-step t a total of $N(t)$ sequences $\mathbf{x}_1, \dots, \mathbf{x}_{N(t)}$ are to be clustered, where each sequence $\mathbf{x}_i$ is of length $n_i(t) \in \mathbb{N}$ for $i = 1..N(t)$. The total number of observed sequences $N(t)$ as well as the individual sequence lengths $n_i(t)$ grow with time. In the online setting, a clustering algorithm is called asymptotically consistent, if almost surely for each fixed batch of sequences $\mathbf{x}_1, \dots, \mathbf{x}_N$, the clustering restricted to this sequences coincides with the target clustering from some time on.

At first glance it may seem that one can use the offline algorithm in the online setting by simply applying it to the entire data observed at every time-step. However, this naive approach does not result in a consistent algorithm. The main challenge in the online setting can be identified with what we regard as "bad" sequences: sequences for which sufficient information has not yet been collected, and thus cannot be distinguished based on the process distributions that generate them. In this setting, using a batch algorithm at every time step results in not only mis-clustering such "bad" sequences, but also in clustering incorrectly those for which sufficient data are already available. That is, such "bad" sequences can render the entire batch clustering useless, leading the algorithm to incorrectly cluster even the "good" sequences. Since new sequences may arrive in an uncontrollable (even data-dependent, adversarial) fashion, any batch algorithm will fail in this scenario.

1.2 Results

The first result of this work is a formal definition of the problem of time series clustering which is rather intuitive and at the same time allows for the construction of efficient algorithms that are provably consistent under the most general assumptions. The second result is the construction of such algorithms.

More specifically, we propose clustering algorithms that are shown to be strongly asymptotically consistent provided that the number k of clusters is known, under the only assumption that the distribution that generates each sequence is stationary ergodic. No restrictions are placed on the dependence between the sequences: this dependence may be arbitrary

(and can be thought of as adversarial). This consistency result is established in each of the two settings (offline and online) introduced above.

As follows from the theoretical impossibility results of (Ryabko, 2010c) that are discussed further in Section 1.4, under the only assumption that the distributions generating the sequences are stationary ergodic, it is impossible to find the correct number of clusters k, that is, the total number of different time series distributions that generate the data. Moreover, non-asymptotic results (finite-time bounds on the probability of error) are also provably impossible to obtain in this setting, since this is the case already for the problem of estimating the probability of any finite-time event (Shields, 1996).

Finding the number of clusters k, as well as obtaining non-asymptotic performance guarantees, is possible under additional conditions on the distributions. In particular, we show that if k is unknown, it is possible to construct consistent algorithms provided that the distributions are mixing and bounds on the mixing rates are available.

However, the main focus of this paper remains on the general framework where no additional assumptions on the unknown process distributions are made (other than that they are stationary ergodic). As such, the main theoretical and experimental results concern the case of known k.

Finally, we show that our methods can be implemented efficiently: they are at most quadratic in each of their arguments, and are linear (up to log terms) in some formulations. To test the empirical performance of our algorithms we evaluated them on both synthetic and real data. To reflect the generality of the suggested framework in the experimental setup, we had our synthetic data generated by stationary ergodic process distributions that do not belong to any "simpler" class of distributions, and in particular cannot be modelled as hidden Markov processes with countable sets of states. In the batch setting, the error rates of both methods go to zero with sequence length. In the online setting with new samples arriving at every time step, the error rate of the offline algorithm remains consistently high, whereas that of the online algorithm converges to zero. This demonstrates that unlike the offline algorithm, the online algorithm is robust to "bad" sequences. To demonstrate the applicability of our work to real-world scenarios, we chose the problem of clustering motion-capture sequences of human locomotion. This application area has also been studied by (Li and Prakash, 2011) and (Jebara et al., 2007) that (to the best of our knowledge) constitute the state-of-the-art performance on the datasets they consider, and against which we compare the performance of our methods. We obtained consistently better performance on the datasets involving motion that can be considered ergodic (walking, running), and competitive performance on those involving non-ergodic motions (single jumps).

1.3 Methodology and algorithms

A crucial point of any clustering method is the similarity measure. Since in our formulation the objective is to cluster the time series based on the distributions that generate them, the similarity measure must reflect the difference between the underlying distributions. Since we aim to make as little assumptions as possible on the distributions that generate the data, and since we make no assumptions on the nature of differences between the distributions, the distance should take into account all possible differences between time series distributions.

Moreover, we want to be able to construct estimates of this distance that are consistent for arbitrary stationary ergodic distributions.

It turns out that a suitable distance for this purpose is the so-called distributional distance. The distributional distance $d(\rho_1, \rho_2)$ between a pair of process distributions ρ_1, ρ_2 is defined (Gray, 1988) as $\sum_{j\in\mathbb{N}} w_j|\rho_1(B_j) - \rho_2(B_j)|$ where, w_i are positive summable real weights, e.g. $w_j = 1/j(j+1)$, and B_k range over a countable field that generates the σ-algebra of the underlying probability space. For example, consider finite-alphabet processes with the binary alphabet $\mathcal{X} = \{0, 1\}$. In this case $B_i,\ i \in \mathbb{N}$ would range over the set $\mathcal{X}^* = \cup_{m\in\mathbb{N}}\mathcal{X}^m$; that is, over all tuples $0, 1, 00, 01, 10, 11, 000, 001, \ldots$; therefore, the distributional distance in this case is the weighted sum of the differences of the probability values (calculated with respect to ρ_1 and ρ_2) of all possible tuples. In this case, the distributional distance metrizes the topology of weak convergence. In this work we consider real-valued processes so that the sets B_k range over a suitable sequence of intervals, all pairs of such intervals, triples, and so on (see the formal definitions in Section 2). Although this distance involves infinite summations, we show that its empirical approximations can be easily calculated. Asymptotically consistent estimates of this distance can be obtained by replacing unknown probabilities with the corresponding frequencies (Ryabko and Ryabko, 2010); these estimators have proved useful in various statistical problems concerning ergodic processes (Ryabko and Ryabko, 2010; Ryabko, 2012; Khaleghi and Ryabko, 2012).

Armed with an estimator of the distributional distance, it is relatively easy to construct a consistent clustering algorithm for the batch setting. In particular, we show that the following algorithm is asymptotically consistent. First a set of k cluster centres are identified using k farthest point initialization (using an empirical estimate of the distributional distance). This means that the first sequence is assigned as the first cluster centre. Iterating over $2..k$, at every iteration a sequence is sought which has the largest minimum distance from the already chosen cluster centres. Next, the remaining sequences are assigned to the closest clusters.

The online algorithm is based on a *weighted combination* of several clusterings, each obtained by running the offline procedure on different portions of data. The partitions are combined with weights that depend on the batch size and on an appropriate performance measure for each individual partition. The performance measure of each clustering is the minimum inter-cluster distance.

1.4 Related Work

Some probabilistic formulations of the time series clustering problem can be considered related to ours. Perhaps the closest one is mixture models (Bach and Jordan, 2004; Biernacki et al., 2000; Kumar et al., 2002; Smyth, 1997; Zhong and Ghosh, 2003): it is assumed that the data are generated by a mixture of k different distributions that have a particular known form (such as Gaussian, Hidden Markov models, or graphical models). Thus, each of the N samples is independently generated according to one of these k distributions (with some fixed probability). Since the model of the data is specified quite well, one can use likelihood-based distances (and then, for example, the k-means algorithm), or Bayesian inference, to cluster the data. Another typical objective is to estimate the parameters of the distributions in the mixture (e.g., Gaussians), rather than actually clustering the data points. Clearly,

the main difference between this setting and ours is in that we do not assume any known model of the data; not even between-sample independence is assumed.

Taking a different perspective, the problem of clustering in our formulations generalizes two classical problems of mathematical statistics, namely, homogeneity testing (or the two-sample problem) and process classification (or the three-sample problem). In the *two-sample problem,* given two sequences $\mathbf{x}_1 = (x_1^1, \dots, x_{n_1}^1)$ and $\mathbf{x}_2 = (x_1^2, \dots, x_{n_2}^2)$ it is required to test whether they are generated by the same or by different process distributions. This corresponds to the special case of clustering only $N = 2$ data points where the number k of clusters unknown: it can be either 1 or 2. In the *three-sample problem,* three sequences $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ are given, and it is known that $\mathbf{x}_1$ and $\mathbf{x}_2$ are generated by different distributions, while $\mathbf{x}_3$ is generated either by the same distribution as $\mathbf{x}_1$ or by the same distribution as $\mathbf{x}_2$. It is required to find which one is the case. This can be seen as clustering $N = 3$ data points, with the number of clusters known: $k = 2$. The classical approach is of course to consider Gaussian i.i.d. samples, but general non-parametric solutions exist not only for i.i.d. data (Lehmann, 1986), but also for Markov chains (Gutman, 1989), as well as under certain mixing-rates conditions. Observe that the two-sample problem is more difficult to solve than the three-sample problem, as the number k of clusters is unknown in the former while it is given in the latter. Indeed, as shown by (Ryabko, 2010c) in general for stationary ergodic (binary-valued) processes there is no solution for the two-sample problem, even in the weakest asymptotic sense. A solution to the three-sample problem, for (real-valued) stationary ergodic processes was given in (Ryabko and Ryabko, 2010); it is based on estimating the distributional distance.

More generally, the area of non-parametric statistical analysis of stationary ergodic time series to which the main results of this paper belong, is full of both positive and negative results, some of which are related to the problem of clustering in our formulation. Among these we can mention change point problems (Carlstein and Lele, 1993; Khaleghi and Ryabko, 2012, 2014) hypothesis testing (Ryabko, 2012, 2014; Morvai and Weiss, 2005) and prediction (Ryabko, 1988; Morvai and Weiss, 2012).

1.5 Organization

The remainder of the paper is organized as follows. We start with introducing notation and definitions in Section 2. In Section 3 we define the considered clustering protocol. Our main theoretical results are given in Section 4, where we present our methods and prove their consistency, as well as discuss some extensions (Section 4.4). Section 5 is devoted to computational considerations. In Section 6 we provide some experimental evaluations on both synthetic and real data. Finally, in Section 7 we provide some concluding remarks and open questions.

2. Preliminaries

Let $\mathcal{X}$ be a measurable space (the domain); in this work we let $\mathcal{X} = \mathbb{R}$ but extensions to more general spaces are straightforward. For a sequence $X_1, \dots, X_n$ we use the abbreviation $X_{1..n}$. Consider the Borel σ-algebra $\mathcal{B}$ on $\mathcal{X}^\infty$ generated by the cylinders $\{B \times \mathcal{X}^\infty : B \in B^{m,l}, m, l \in \mathbb{N}\}$ where, the sets $B^{m,l}, m, l \in \mathbb{N}$ are obtained via the partitioning of $\mathcal{X}^m$ into cubes of dimension m and volume 2^{-ml} (starting at the origin). Let also $B^m := \cup_{l \in \mathbb{N}} B^{m,l}$.

We may choose any other means to generate the Borel σ-algebra $\mathcal{B}$ on $\mathcal{X}$, but we need to fix a specific choice for computational reasons. Process distributions are probability measures on the space $(\mathcal{X}^\infty, \mathcal{B})$. Similarly, one can define distributions over the space $((\mathcal{X}^\infty)^\infty, \mathcal{B}_2)$ of infinite matrices with the Borel σ-algebra $\mathcal{B}_2$ generated by the cylinders $\{(\mathcal{X}^\infty)^k \times (B \times \mathcal{X}^\infty) \times (\mathcal{X}^\infty)^\infty : \ B \in B^{m,l}, k, m, l \in \mathbb{N}\}$. For $\mathbf{x} = X_{1..n} \in \mathcal{X}^n$ and $B \in B^m$ let $\nu(\mathbf{x}, B)$ denote the *frequency* with which $\mathbf{x}$ falls in B, i.e.

$$\nu(\mathbf{x}, B) := \frac{\mathbb{I}\{n \geq m\}}{n - m + 1} \sum_{i=1}^{n-m+1} \mathbb{I}\{X_{i..i+m-1} \in B\} \tag{1}$$

A process ρ is *stationary* if for any $i, j \in 1..n$ and $B \in \mathcal{B}$, we have

$$\rho(X_{1..j} \in B) = \rho(X_{i..i+j-1} \in B).$$

A stationary process ρ is called *ergodic* if for all $B \in \mathcal{B}$ with probability 1 we have

$$\lim_{n\to\infty} \nu(X_{1..n}, B) = \rho(B).$$

By virtue of the ergodic theorem (e.g., Billingsley, 1961), this definition can be shown to be equivalent to the standard definition of stationary ergodic processes (every shift-invariant set has measure 0 or 1; see, e.g., Gray, 1988).

Definition 1 (Distributional Distance) *The distributional distance between a pair of process distributions ρ_1, ρ_2 is defined as follows (Gray, 1988):*

$$d(\rho_1, \rho_2) = \sum_{m,l=1}^{\infty} w_m w_l \sum_{B \in B^{m,l}} |\rho_1(B) - \rho_2(B)|,$$

where we set $w_j := 1/j(j+1)$, but any summable sequence of positive weights may be used.

In words, this involves partitioning the sets $\mathcal{X}^m$, $m \in \mathbb{N}$ into cubes of decreasing volume (indexed by l) and then taking a sum over the absolute differences in probabilities of all the cubes in these partitions. The differences in probabilities are weighted: smaller weights are given to larger m and finer partitions. We use *empirical estimates* of this distance defined as follows.

Remark 2 The distributional distance is more generally defined as

$$d'(\rho_1, \rho_2) := \sum_{i \in \mathbb{N}} w_i |\rho_1(A_i) - \rho_2(A_i)|$$

where A_i ranges over a countable field that generates the σ-algebra of the underlying probability space (Gray, 1988). Definition 1 above is more suitable for implementing and testing algorithms, while the consistency results of this paper hold for either. Note also that the choice of sets B_i may result in a different topology on the space of time-series distributions. The particular choice we made results in the usual topology of weak convergence (Billingsley, 1999)

Definition 3 (Empirical estimates of $d(\cdot,\cdot)$) *Define the empirical estimate of the distributional distance between a sequence $\mathbf{x} = X_{1..n} \in \mathcal{X}^n, n \in \mathbb{N}$ and a process distribution ρ by*

$$\hat{d}(\mathbf{x}, \rho) := \sum_{m=1}^{m_n} \sum_{l=1}^{l_n} w_m w_l \sum_{B \in B^{m,l}} |\nu(\mathbf{x}, B) - \rho(B)|, \tag{2}$$

and that between a pair of sequences $\mathbf{x}_i \in \mathcal{X}^{n_i}$ $n_i \in \mathbb{N},\ i = 1, 2$ by

$$\hat{d}(\mathbf{x}_1, \mathbf{x}_2) := \sum_{m=1}^{m_n} \sum_{l=1}^{l_n} w_m w_l \sum_{B \in B^{m,l}} |\nu(\mathbf{x}_1, B) - \nu(\mathbf{x}_2, B)| \tag{3}$$

where m_n and l_n are any sequences that go to infinity with n, and $(w_j)_{j \in \mathbb{N}}$ are as in Definition 1.

Remark 4 (constants m_n, l_n) The sequences of constants $m_n, l_n (n \in \mathbb{N})$ in the definition of $\hat{d}$ are intended to introduce some computational flexibility in the estimate: while already the choice $m_n, l_n \equiv \infty$ is computationally feasible, other choices give better computational complexity without sacrificing the quality of the estimate; see Section 5. For the asymptotic consistency results this choice is irrelevant. Thus, in the proofs we tacitly assume $m_n, l_n \equiv \infty$; the extension to the general case is straightforward.

Lemma 5 ($\hat{d}$ is asymptotically consistent; Ryabko and Ryabko (2010)) *For every pair of sequences $\mathbf{x}_1 \in \mathcal{X}^{n_1}$ and $\mathbf{x}_2 \in \mathcal{X}^{n_2}$ with joint distribution ρ whose marginals $\rho_i,\ i = 1, 2$ are stationary ergodic we have*

$$\lim_{n_1, n_2 \to \infty} \hat{d}(\mathbf{x}_1, \mathbf{x}_2) = d(\rho_1, \rho_2),\ \rho - a.s.,\ and \tag{4}$$

$$\lim_{n_i \to \infty} \hat{d}(\mathbf{x}_i, \rho_j) = d(\rho_i, \rho_j),\ i, j \in 1, 2,\ \rho - a.s. \tag{5}$$

The proof of this lemma can be found in (Ryabko and Ryabko, 2010); since it is important for further development but rather short and simple we reproduce it here.

Proof The idea of the proof is simple: for each set $B \in \mathcal{B}$, the frequency with which the sample $\mathbf{x}_i,\ i = 1, 2$ falls into B converges to the probability $\rho_i(B),\ i = 1, 2$. When the sample sizes grow, there will be more and more sets $B \in \mathcal{B}$ whose frequencies have already converged to the probabilities, so that the cumulative weight of those sets whose frequencies have not converged yet will tend to 0.

Fix $\varepsilon > 0$. We can find an index J such that

$$\sum_{m,l=J}^{\infty} w_m w_l \leq \varepsilon/3.$$

Moreover, for each $m, l \in 1..J$ we can find a finite subset $S^{m,l}$ of $B^{m,l}$ such that

$$\rho_i(S^{m,l}) \geq 1 - \frac{\varepsilon}{6 J w_m w_l}.$$

There exists some N (which depends on the realization $X_1^i, \ldots, X_{n_i}^i,\ i = 1, 2$) such that for all $n_i \geq N,\ i = 1, 2$ we have,

$$\sup_{B \in S^{m,l}} |\nu(\mathbf{x}_i, B) - \rho_i(B)| \leq \frac{\varepsilon \rho_i(B)}{6 J w_m w_l},\ i = 1, 2. \tag{6}$$

For all $n_i \geq N,\ i = 1, 2$ we have

$$\begin{aligned}
|\hat{d}(\mathbf{x}_1, \mathbf{x}_2) - d(\rho_1, \rho_2)| &= \left| \sum_{m,l=1}^{\infty} w_m w_l \sum_{B \in B^{m,l}} \big(|\nu(\mathbf{x}_1, B) - \nu(\mathbf{x}_2, B)| - |\rho_1(B) - \rho_2(B)| \big) \right| \\
&\leq \sum_{m,l=1}^{\infty} w_m w_l \sum_{B \in B^{m,l}} |\nu(\mathbf{x}_1, B) - \rho_1(B)| + |\nu(\mathbf{x}_2, B) - \rho_2(B)| \\
&\leq \sum_{m,l=1}^{J} w_m w_l \sum_{B \in S^{m,l}} \big(|\nu(\mathbf{x}_1, B) - \rho_1(B)| + |\nu(\mathbf{x}_2, B) - \rho_2(B)| \big) + 2\varepsilon/3 \\
&\leq \sum_{m,l=1}^{J} w_m w_l \sum_{B \in S^{m,l}} \frac{\rho_1(B)\varepsilon}{6 J w_m w_l} + \frac{\rho_2(B)\varepsilon}{6 J w_m w_l} + 2\varepsilon/3 \leq \varepsilon,
\end{aligned}$$

which proves (4). The statement (5) can be proven analogously. ■

Remark 6 The triangle inequality holds for the distributional distance $d(\cdot, \cdot)$ and its empirical estimates $\hat{d}(\cdot, \cdot)$ so that for all distributions $\rho_i,\ i = 1..3$ and all sequences $\mathbf{x}_i \in \mathcal{X}^{n_i}\ n_i \in \mathbb{N},\ i = 1..3$ we have

$$\begin{aligned}
d(\rho_1, \rho_2) &\leq d(\rho_1, \rho_3) + d(\rho_2, \rho_3), \\
\hat{d}(\mathbf{x}_1, \mathbf{x}_2) &\leq \hat{d}(\mathbf{x}_1, \mathbf{x}_3) + \hat{d}(\mathbf{x}_2, \mathbf{x}_3), \\
\hat{d}(\mathbf{x}_1, \rho_1) &\leq \hat{d}(\mathbf{x}_1, \rho_2) + d(\rho_1, \rho_2).
\end{aligned}$$

3. Clustering settings: offline and online

We start with a common general probabilistic setup for both settings (offline and online), which we use to formulate each of the two settings separately.

Consider the matrix $\mathbf{X} \in (\mathcal{X}^\infty)^\infty$ of random variables

$$\mathbf{X} := \begin{bmatrix} X_1^1 & X_2^1 & X_3^1 & \ldots \\ X_1^2 & X_2^2 & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix} \in (\mathcal{X}^\infty)^\infty \tag{7}$$

generated by some probability distribution P on $((\mathcal{X}^\infty)^\infty, \mathcal{B}_2)$. Assume that the marginal distribution of P on each row of $\mathbf{X}$ is one of κ *unknown stationary ergodic* process distributions $\rho_1, \rho_2, \ldots, \rho_\kappa$. Thus, the matrix $\mathbf{X}$ corresponds to infinitely many one-way infinite sequences, each of which is generated by a stationary ergodic distribution. Aside from this

assumption, we do not make any further assumptions on the distribution P that generates $\mathbf{X}$. This means that the rows of $\mathbf{X}$ (corresponding to different time-series samples) are allowed to be dependent, and the dependence can be arbitrary; one can even think of the dependence between samples as "adversarial." For notational convenience we assume that the distributions $\rho_k, k = 1..\kappa$ are ordered in the order of appearance of their first rows (samples) in $\mathbf{X}$.

Among the various ways a set of κ disjoint subsets of the rows of $\mathbf{X}$ may be produced, the most natural partitioning in this formulation is to group together those and only those rows of $\mathbf{X}$ for which the marginal distribution is the same. More precisely, we define the ground-truth partitioning of $\mathbf{X}$ as follows.

Definition 7 (Ground-truth $\mathcal{G}$) *Let*

$$\mathcal{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_\kappa\}$$

be a partitioning of $\mathbb{N}$ *into* κ *disjoint subsets* $\mathcal{G}_k,\ k = 1..\kappa$*, such that the marginal distribution of* $\mathbf{x}_i,\ i \in \mathbb{N}$ *is* ρ_k *for some* $k \in 1..\kappa$ *if and only if* $i \in \mathcal{G}_k$*. Call* $\mathcal{G}$ *the ground-truth clustering. We also introduce the notation* $\mathcal{G}|_N$ *for the restriction of* $\mathcal{G}$ *to the first* N *sequences:*

$$\mathcal{G}|_N := \{\mathcal{G}_k \cap \{1..N\} : k = 1..\kappa\}.$$

Offline Setting. The problem is formulated as follows. We are given a finite set $S := \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ of N samples, for some fixed $N \in \mathbb{N}$. Each sample is generated by one of κ unknown stationary ergodic process distributions $\rho_1, \dots, \rho_\kappa$. More specifically, the set S is obtained from $\mathbf{X}$ as follows. Some (arbitrary) lengths $n_i \in \mathbb{N},\ i \in 1..N$ are fixed, and $\mathbf{x}_i$ for each $i = 1..N$ is defined as $\mathbf{x}_i := X^i_{1..n_i}$.

A clustering function f takes a finite set $S := \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ of samples and an optional parameter κ (the number of target clusters) to produce a partition $f(S, \kappa) := \{C_1, \dots, C_\kappa\}$ of the index-set $\{1..N\}$. The goal is to partition $\{1..N\}$ so as to recover the ground-truth clustering $\mathcal{G}|_N$. For consistency of notation, in the offline setting we identify $\mathcal{G}$ with $\mathcal{G}|_N$. We call a clustering algorithm asymptotically consistent if it achieves this goal for long enough sequences $\mathbf{x}_i,\ i = 1..N$ in S:

Definition 8 (Consistency: offline setting) *A clustering function* f *is consistent for a set of sequences* S *if* $f(S, \kappa) = \mathcal{G}$*. Moreover, denoting* $n := \min\{n_1, \dots, n_N\}$*,* f *is called strongly asymptotically consistent in the offline sense if with probability* 1 *from some* n *on it is consistent on the set* S*:* $P(\exists n' \forall n > n' f(S, \kappa) = \mathcal{G}) = 1$*. It is weakly asymptotically consistent if* $\lim_{n\to\infty} P(f(S, \kappa) = \mathcal{G}) = 1$*.*

Note that the notion of consistency above is asymptotic with respect to the minimum sample length n, and not with respect to the number N of samples.

When considering the offline setting with a known number of clusters κ we assume that N is such that $\{1..N\} \cap \mathcal{G}_k \neq \varnothing$ for every $k = 1..\kappa$ (that is, all κ clusters are represented); otherwise we could just take a smaller κ.

Online Setting. The online problem is formulated as follows. Consider the infinite matrix $\mathbf{X}$ given by (7). At every time-step $t \in \mathbb{N}$, a part $S(t)$ of $\mathbf{X}$ is observed corresponding to the first $N(t) \in \mathbb{N}$ rows of $\mathbf{X}$, each of length $n_i(t), i \in 1..N(t)$, i.e.

$$S(t) = \{\mathbf{x}^t_1, \cdots \mathbf{x}^t_{N(t)}\} \text{ where } \mathbf{x}^t_i := X^i_{1..n_i(t)}.$$

This setting is depicted in Figure 1. We assume that the number of samples, as well as the individual sample-lengths grow with time. That is, the length $n_i(t)$ of each sequence $\mathbf{x}_i$ is non-decreasing and grows to infinity (as a function of time t). The number of sequences $N(t)$ also grows to infinity. Aside from these assumptions, the functions $N(t)$ and $n_i(t)$ are completely arbitrary.

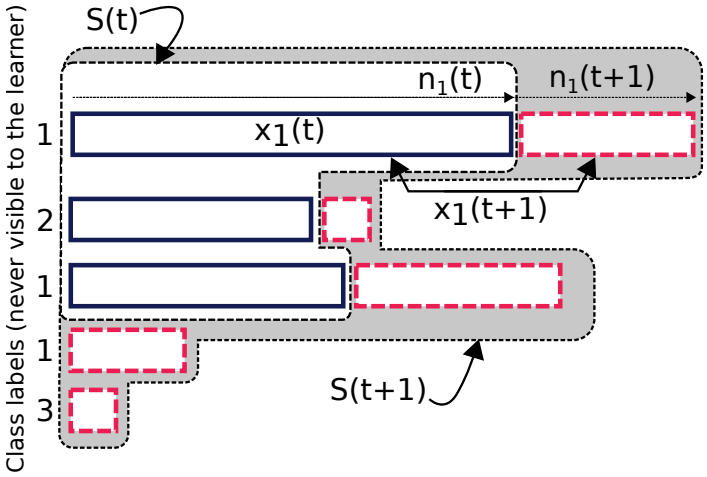


Figure 1: Online Protocol: solid rectangles correspond to sequences available at time t, dashed rectangles correspond to segments arrived at time $t+1$.

An online clustering function is strongly asymptotically consistent if, with probability 1, for each $N \in \mathbb{N}$ from some time on the first N sequences are clustered correctly (with respect to the ground-truth given by Definition 7).

Definition 9 (Asymptotic consistency: online setting) *A clustering function is strongly (weakly) asymptotically consistent in the online sense, if for every $N \in \mathbb{N}$ the clustering $f(S(t),\kappa)|_N$ is strongly (weakly) asymptotically consistent in the offline sense, where $f(S(t),\kappa)|_N$ is the clustering $f(S(t),\kappa)$ restricted to the first N sequences:*

$$f(S(t),\kappa)|_N := \{f(S(t),\kappa) \cap \{1..N\}, k = 1..\kappa\}.$$

Remark 10 Note that even if the *eventual* number κ of different time series distributions producing the sequences (that is, the number of clusters in the ground-truth clustering $\mathcal{G}$) is known, the number of observed distributions at each individual time-step is unknown. That is, it is possible that at a given time-step t we have $\left|\mathcal{G}|_{N(t)}\right| < \kappa$.

Known and unknown κ. As mentioned in the introduction, in the general framework (for both the offline and the online problems) described above consistent clustering with unknown number of clusters is impossible. This follows from the impossibility result of (Ryabko, 2010c) which states that when we have only two (binary-valued) samples generated (independently) by two stationary ergodic distributions, it is impossible to decide whether they have been generated by the same or by different distributions, even in the sense of

weak asymptotic consistency. (This holds even if the distributions come from a smaller class: the set of all B-processes.) Therefore, if the number of clusters is unknown, we are bound to make stronger assumptions. Since our main interest in this paper is to develop consistent clustering algorithms under the general framework described above, for the most part of this paper we assume that the correct number κ of clusters is known. However, in Section 4.3 we also show that under certain mixing conditions on the process distributions that generate the data it is possible to have consistent algorithms in the case of unknown κ as well.

4. Clustering algorithms and their consistency

In this section we present our clustering methods for both the offline and the online settings. The main results, presented in Sections 4.1 (offline) and 4.2 (online), concern the case where the number of clusters κ is known. In Section 4.3 we show that, given the mixing rates of the process distributions that generate the data, it is possible to find the correct number of clusters κ and thus obtain consistent algorithms for the case of unknown κ as well. Finally, section 4.4 considers some extensions of the proposed settings.

4.1 Offline Setting

Given that we have asymptotically consistent estimates $\hat{d}$ of the distributional distance d, it is relatively simple to construct asymptotically consistent clustering algorithms for the offline setting. This follows from the fact that, since $\hat{d}(\cdot,\cdot)$ converges to $d(\cdot,\cdot)$, for large enough sequence lengths n, the points $\mathbf{x}_1,\dots,\mathbf{x}_N$ have the so-called strict separation property: the points within each target cluster are closer to each other than to points in any other cluster (on strict separation in clustering see, for example, Balcan et al., 2008). This makes many simple algorithms, such as single or average linkage, or the k-means algorithm with certain initializations, provably consistent.

We present below one specific algorithm that we show to be asymptotically consistent in the general framework introduced. What makes this simple algorithm interesting is that it requires only κN distance calculations (that is, much less than is needed to calculate the distance between each two sequences). In short, Algorithm 1 initializes the clusters using farthest-point initialization, and then assigns each remaining point to the nearest cluster. More precisely, the sample $\mathbf{x}_1$ is assigned as the first cluster centre. Then a sample is found that is farthest away from $\mathbf{x}_1$ in the empirical distributional distance $\hat{d}$ and is assigned as the second cluster centre. For each $k = 2..\kappa$ the k^{th} cluster centre is sought as the sequence with the largest minimum distance from the already assigned cluster centres for $1..k-1$. (This initialization was proposed for use with k-means clustering by Katsavounidis et al., 1994.) By the last iteration we have κ cluster centres. Next the remaining samples are each assigned to the closest cluster.

Theorem 11 *Algorithm 1 is strongly asymptotically consistent (in the offline sense of Definition 8), provided that the correct number κ of clusters is known, and the marginal distribution of each sequence $\mathbf{x}_i, i = 1..N$ is stationary ergodic.*

Proof To prove the consistency statement we use Lemma 5 to show that if the samples in S are long enough, the samples that are generated by the same process distribution

Algorithm 1 Offline clustering method of (Ryabko, 2010b)

INPUT: sequences $S := \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$**, Number** κ **of clusters**
Initialize κ***-farthest points as cluster-centres:***
$c_1 \leftarrow 1$
$C_1 \leftarrow \{c_1\}$
for $k = 2..\kappa$ **do**
    $c_k \leftarrow \operatorname{argmax}_{i=1..N} \min_{j=1..k-1} \hat{d}(\mathbf{x}_i, \mathbf{x}_{c_j})$***, where ties are broken arbitrarily***
    $C_k \leftarrow \{c_k\}$
end for
Assign the remaining points to closest centres:
for $i = 1..N$ **do**
    $k \leftarrow \operatorname{argmin}_{j \in \bigcup_{k=1}^{\kappa} C_k} \hat{d}(\mathbf{x}_i, \mathbf{x}_j)$
    $C_k \leftarrow C_k \cup \{i\}$
end for
OUTPUT: clusters $C_1, C_2, \cdots, C_\kappa$

are closer to each other than to the rest of the samples. Therefore, the samples chosen as cluster centres are each generated by a different process distribution, and since the algorithm assigns the rest of the samples to the closest clusters, the statement follows. More formally, let n denote the shortest sample length in S:

$$n_{\min} := \min_{i \in 1..N} n_i.$$

Denote by δ the minimum non-zero distance between the process distributions:

$$\delta := \min_{k \neq k' \in 1..\kappa} \hat{d}(\rho_k, \rho_{k'}).$$

Fix $\varepsilon \in (0, \delta/4)$. Since there are a finite number N of samples, by Lemma 5 for all large enough $n_{\min}$ we have

$$\sup_{\substack{k \in 1..\kappa \\ i \in \mathcal{G}_k \cap \{1..N\}}} \hat{d}(\mathbf{x}_i, \rho_k) \leq \varepsilon. \tag{8}$$

where $\mathcal{G}_k,\ k = 1..\kappa$ denote the ground-truth partitions given by Definition 7. By (8) and applying the triangle inequality we obtain

$$\sup_{\substack{k \in 1..\kappa \\ i,j \in \mathcal{G}_k \cap \{1..N\}}} \hat{d}(\mathbf{x}_i, \mathbf{x}_j) \leq 2\varepsilon. \tag{9}$$

Thus, for all large enough $n_{\min}$ we have

$$\begin{aligned} \inf_{\substack{i \in \mathcal{G}_k \cap \{1..N\} \\ j \in \mathcal{G}_{k'} \cap \{1..N\} \\ k \neq k' \in 1..\kappa}} \hat{d}(\mathbf{x}_i, \mathbf{x}_j) &\geq \inf_{\substack{i \in \mathcal{G}_k \cap \{1..N\} \\ j \in \mathcal{G}_{k'} \cap \{1..N\} \\ k \neq k' \in 1..\kappa}} d(\rho_k, \rho_{k'}) - \hat{d}(\mathbf{x}_i, \rho_k) - \hat{d}(\mathbf{x}_j, \rho_{k'}) \\ &\geq \delta - 2\varepsilon \end{aligned} \tag{10}$$

where the first inequality follows from the triangle inequality, and the second inequality follows from (8) and the definition of δ. In words, (9) and (10) mean that the samples in $\mathcal{S}$ that are generated by the same process distribution are closer to each other than to the rest of the samples. Finally, for all $n_{\min}$ large enough to have (9) and (10) we obtain

$$\max_{i=1..N} \min_{k=1..\kappa-1} \hat{d}(\mathbf{x}_i, \mathbf{x}_{c_k}) \geq \delta - 2\varepsilon > \delta/2$$

where as specified by Algorithm 1, $c_1 := 1$ and $c_k := \operatorname{argmax}_{i=1..N} \min_{j=1..k-1} \hat{d}(\mathbf{x}_i, \mathbf{x}_{c_j})$, $k = 2..\kappa$. Hence, the indices $c_1, \ldots, c_\kappa$ will be chosen to index sequences generated by different process distributions. To derive the consistency statement, it remains to note that, by (9) and (10), each remaining sequence will be assigned to the cluster centre corresponding to the sequence generated by the same distribution. ■

4.2 Online setting

The online version of the problem turns out to be more complicated than the offline one. The challenge is that, since new sequences arrive (potentially) at every time-step, we can never rely on the distance estimates corresponding to all of the observed samples to be correct. Thus, as mentioned in the introduction, the main challenge can be identified with what we regard as "bad" sequences: recently-observed sequences, for which sufficient information has not yet been collected, and for which the estimates of the distance (with respect to any other sequence) are bound to be misleading. Thus, in particular, farthest-point initialization would not work in this case. More generally, using any batch algorithm on all available data at every time-step results in not only mis-clustering "bad" sequences, but also in clustering incorrectly those for which sufficient data are already available.

The solution, realized in Algorithm 2, is based on a *weighted combination* of several clusterings, each obtained by running the offline algorithm (Algorithm 1) on different portions of data. The clusterings are combined with weights that depend on the batch size and on the minimum inter-cluster distance. This last step of combining multiple clusterings with weights may be reminiscent of prediction with expert advice (see Cesa-Bianchi and Lugosi, 2006 for an overview), where experts are combined based on their past performance. However, the difference here is that the performance of each clustering cannot be measured directly.

More precisely, Algorithm 2 works as follows. Given a set $S(t)$ of $N(t)$ samples, the algorithm iterates over $j := \kappa, \ldots, N(t)$ where at each iteration Algorithm 1 is used to cluster the first j sequences $\{\mathbf{x}_1^t, \ldots, \mathbf{x}_j^t\}$ into κ clusters. In each cluster the sequence with the smallest index is assigned as the candidate cluster centre. A performance score γ_j is calculated as the minimum distance $\hat{d}$ between the κ candidate cluster centres obtained at iteration j. Thus, γ_j is an estimate of the minimum inter-cluster distance. At this point we have $N(t)-\kappa+1$ sets of κ cluster centres $c_1^j, \ldots, c_\kappa^j$, $j = 1..N(t)-\kappa+1$. Next, every sample $\mathbf{x}_i^t$, $i = 1..N(t)$ is assigned according to the weighted combination of the distances between $\mathbf{x}_i^t$ and the candidate cluster centres obtained at each iteration on j. More precisely, for

Algorithm 2 Online Clustering

INPUT: Number κ of target clusters
for $t = 1..\infty$ **do**
  Obtain new sequences $S(t) = \{\mathbf{x}_1^t, \cdots, \mathbf{x}_{N(t)}^t\}$
  Initialize the normalization factor: $\eta \leftarrow 0$
  Initialize the final clusters: $C_k(t) \leftarrow \varnothing,\ k = 1..\kappa$
  Generate $N(t) - \kappa + 1$ **candidate cluster centres:**
  for $j = \kappa..N(t)$ **do**
    $\{C_1^j, \ldots, C_\kappa^j\} \leftarrow \text{Alg1}(\{\mathbf{x}_1^t, \cdots, \mathbf{x}_j^t\}, \kappa)$
    $\mu_k \leftarrow \min\{i \in C_k^j\}, k = 1..\kappa$ ▷ Set the smallest index as cluster centre.
    $(c_1^j, \ldots, c_\kappa^j) \leftarrow \text{sort}(\mu_1, \ldots, \mu_\kappa)$ ▷ Sort the cluster centres increasingly.
    $\gamma_j \leftarrow \min_{k \neq k' \in 1..\kappa} \hat{d}(\mathbf{x}^t_{c_k^j}, \mathbf{x}^t_{c_{k'}^j})$ ▷ Calculate performance score.
    $w_j \leftarrow 1/j(j+1)$
    $\eta \leftarrow \eta + w_j\gamma_j$
  end for
  Assign points to clusters:
  for $i = 1..N(t)$ **do**
    $k \leftarrow \text{argmin}_{k' \in 1..\kappa} \frac{1}{\eta} \sum_{j=1}^{N(t)} w_j \gamma_j \hat{d}(\mathbf{x}_i^t, \mathbf{x}^t_{c_{k'}^j})$
    $C_k(t) \leftarrow C_k(t) \cup \{i\}$
  end for
  OUTPUT: $\{C_1(t), \cdots, C_k(t)\}$
end for

each $i = 1..N(t)$ the sequence $\mathbf{x}_i^t$ is assigned to the cluster k, where k is defined as

$$k := \text{argmin}_{k=1..\kappa} \sum_{j=\kappa}^{N(t)} w_j \gamma_j \hat{d}(\mathbf{x}_i^t, \mathbf{x}^t_{c_k^j}).$$

Theorem 12 *Algorithm 2 is strongly asymptotically consistent (in the online sense of Definition 9), provided the correct number of clusters κ is known, and the marginal distribution of each sequence $\mathbf{x}_i, i \in \mathbb{N}$ is stationary ergodic.*

Before giving the proof of Theorem 12, we provide an intuitive explanation as to how Algorithm 2 works. First, consider the following simple candidate solution. Take some fixed (reference) portion of the samples, run the batch algorithm on it, and then simply assign every remaining sequence to the nearest cluster. Since the offline algorithm is asymptotically consistent, this procedure would be asymptotically consistent as well, but only if we knew that the selected reference of the sequences contains at least one sequence sampled from each and every one of the κ distributions. However, there is no way to find a fixed (not growing with time) portion of data that would be guaranteed to contain a representative of each cluster (that is, of each time series distribution). Allowing such a reference set of sequences to grow with time would guarantee that eventually it contains representatives of all clusters, but it would break the consistency guarantee for the reference set; since the set grows, this formulation effectively returns us back to the original online clustering problem.

A key observation we make to circumvent this problem is the following. If, for some $j \in \{\kappa, \dots, N(t)\}$, each sample in the batch $\{\mathbf{x}_1^t, \dots, \mathbf{x}_j^t\}$ is generated by *at most* $\kappa - 1$ process distributions, any partitioning of this batch into κ sets results in a minimum inter-cluster distance γ_j that, as follows from the asymptotic consistency of $\hat{d}$, converges to 0. On the other hand, if the set of samples $\{\mathbf{x}_1^t, \dots, \mathbf{x}_j^t\}$ contains sequences generated by all κ process distributions, γ_j converges to a non-zero constant, namely, the minimum distance between the distinct process distributions $\rho_1, \dots, \rho_\kappa$. In the latter case from some time on the batch $\{\mathbf{x}_1^t, \dots, \mathbf{x}_j^t\}$ will be clustered correctly. Thus, instead of selecting one reference batch of sequences and constructing a set of clusters based on those, we consider all batches of sequences for $j = \kappa..N(t)$, and combine them with weights. Two sets of weights are involved in this step: γ_j and w_j, where

1. γ_j is used to penalise for small inter-cluster distance, canceling the clustering results produced based on sets of sequences generated by less than κ distributions;

2. w_j is used to give precedence to chronologically earlier clusterings, protecting the clustering decisions from the presence of the (potentially "bad") newly formed sequences, whose corresponding distance estimates may still be far from accurate.

As time goes on, the batches where not all clusters are represented will have their weight γ_j converge to 0, while the number of batches that have all clusters represented and are clustered correctly by the offline algorithm will go to infinity, and their total weight will approach 1. Note that, since we are combining different clusterings, it is important to use a consistent ordering of clusters, for otherwise we might sum up clusters generated by different distributions. Therefore, we always order the clusters with respect to the index of the first sequence in each cluster.

Proof [of Theorem 12] First, we show that for every $k \in 1..\kappa$ we have

$$\frac{1}{\eta} \sum_{j=1}^{N(t)} w_j \gamma_j^t \hat{d}(\mathbf{x}_{c_k^j}^t, \rho_k) \to 0 \text{ a.s.} \tag{11}$$

Denote by δ the minimum non-zero distance between the process distributions:

$$\delta := \min_{k \neq k' \in 1..\kappa} \hat{d}(\rho_k, \rho_{k'}). \tag{12}$$

Fix $\varepsilon \in (0, \delta/4)$. We can find an index J such that $\sum_{j=J}^{\infty} w_j \leq \varepsilon$. Let $S(t)|_j = \{\mathbf{x}_1^t, \cdots, \mathbf{x}_j^t\}$ denote the subset of $S(t)$ consisting of the first j sequences for $j \in 1..N(t)$. For $k = 1..\kappa$ let

$$s_k := \min\{i \in \mathcal{G}_k \cap 1..N(t)\} \tag{13}$$

index the first sequence in $S(t)$ that is generated by ρ_k where $\mathcal{G}_k,\ k = 1..\kappa$ are the ground-truth partitions given by Definition 7. Define

$$m := \max_{k \in 1..\kappa} s_k. \tag{14}$$

Recall that the sequence lengths $n_i(t)$ grow with time. Therefore, by Lemma 5 (consistency of $\hat{d}$) for every $j \in 1..J$ there exists some $T_1(j)$ such that for all $t \geq T_1(j)$ we have

$$\sup_{\substack{k \in 1..\kappa \\ i \in \mathcal{G}_k \cap \{1..j\}}} \hat{d}(\mathbf{x}_i^t, \rho_k) \leq \varepsilon. \tag{15}$$

Moreover, by Theorem 11 for every $j \in m..J$ there exists some $T_2(j)$ such that $\mathrm{Alg1}(S(t)|_j, \kappa)$ is consistent for all $t \geq T_2(j)$. Let

$$T := \max_{\substack{i=1,2 \\ j \in 1..J}} T_i(j).$$

Recall that, by definition (14) of m, $S(t)|_m$ contains samples from all κ distributions. Therefore, for all $t \geq T$ we have

$$\begin{aligned} \inf_{k \neq k' \in 1..\kappa} \hat{d}(\mathbf{x}^t_{c^m_k}, \mathbf{x}^t_{c^m_{k'}}) &\geq \inf_{k \neq k' \in 1..\kappa} d(\rho_k, \rho_{k'}) - \sup_{k \neq k' \in 1..\kappa} (\hat{d}(\mathbf{x}^t_{c^m_k}, \rho_k) + \hat{d}(\mathbf{x}^t_{c^m_{k'}}, \rho_{k'})) \\ &\geq \delta - 2\varepsilon \geq \delta/2, \end{aligned} \tag{16}$$

where the first inequality follows from the triangle inequality and the second inequality follows from the consistency of $\mathrm{Alg1}(S(t)|_m, \kappa)$ for $t \geq T$, the definition of δ given by (12) and the assumption that $\varepsilon \in (0, \delta/4)$. Recall that (as specified in Algorithm 2) we have $\eta := \sum_{j=1}^{N(t)} w_j \gamma_j^t$. Hence, by (16) for all $t \geq T$ we have

$$\eta \geq w_m \delta/2. \tag{17}$$

By (17) and noting that by definition $\hat{d}(\cdot, \cdot) \leq 1$, for all $t \geq T$ for every $k \in 1..\kappa$ we obtain

$$\frac{1}{\eta} \sum_{j=1}^{N(t)} w_j \gamma_j^t \hat{d}(\mathbf{x}^t_{c^j_k}, \rho_k) \leq \frac{1}{\eta} \sum_{j=1}^{J} w_j \gamma_j^t \hat{d}(\mathbf{x}^t_{c^j_k}, \rho_k) + \frac{2\varepsilon}{w_m \delta}. \tag{18}$$

On the other hand, by the definition (14) of m, the sequences in $S(t)|_j$ for $j = 1..m-1$ are generated by *at most* $\kappa - 1$ out of the κ process distributions. Therefore, at every iteration on $j \in 1..m-1$ there exists at least one pair of distinct cluster centres that are generated by *the same* process distribution. Therefore, by (15) and (17), for all $t \geq T$ and every $k \in 1..\kappa$ we have,

$$\frac{1}{\eta} \sum_{j=1}^{m-1} w_j \gamma_j^t \hat{d}(\mathbf{x}^t_{c^j_k}, \rho_i) \leq \frac{1}{\eta} \sum_{j=1}^{m-1} w_j \gamma_j^t \leq \frac{2\varepsilon}{w_m \delta}. \tag{19}$$

Noting that the clusters are ordered in the order of appearance of the distributions, we have $\mathbf{x}^t_{c^j_k} = \mathbf{x}^t_{s_k}$ for all $j = m..J$ and $k = 1..\kappa$, where the index s_k is defined by (13). Therefore, by (15) for all $t \geq T$ and every $k = 1..\kappa$ we have

$$\frac{1}{\eta} \sum_{j=m}^{J} w_j \gamma_j^t \hat{d}(\mathbf{x}^t_{c^j_k}, \rho_k) = \frac{1}{\eta} \hat{d}(\mathbf{x}^t_{s_k}, \rho_k) \sum_{j=m}^{J} w_j \gamma_j^t \leq \varepsilon. \tag{20}$$

Combining (18), (19), and (20) we obtain

$$\frac{1}{\eta} \sum_{j=1}^{N(t)} w_j \gamma_j^t \hat{d}(\mathbf{x}^t_{c^j_k}, \rho_k) \leq \varepsilon(1 + \frac{4}{w_m \delta}). \tag{21}$$

for all $k = 1..\kappa$ and all $t \geq T$, establishing (11).

To finish the proof of the consistency consider an index $i \in \mathcal{G}_r$ for some $r \in 1..\kappa$. By Lemma 5, increasing T if necessary, for all $t \geq T$ we have

$$\sup_{\substack{k\in 1..\kappa \\ j\in\mathcal{G}_k\cap 1..N}} \hat{d}(\mathbf{x}_j^t, \rho_k) \leq \varepsilon. \tag{22}$$

For all $t \geq T$ and all $k \neq r \in 1..\kappa$ we have,

$$\begin{aligned}
\frac{1}{\eta}\sum_{j=1}^{N(t)} w_j\gamma_j^t\hat{d}(\mathbf{x}_i^t, \mathbf{x}_{c_k^j}^t) &\geq \frac{1}{\eta}\sum_{j=1}^{N(t)} w_j\gamma_j^t\hat{d}(\mathbf{x}_i^t, \rho_k) - \frac{1}{\eta}\sum_{j=1}^{N(t)} w_j\gamma_j^t\hat{d}(\mathbf{x}_{c_k^j}^t, \rho_k) \\
&\geq \frac{1}{\eta}\sum_{j=1}^{N(t)} w_j\gamma_j^t(\hat{d}(\rho_k, \rho_r) - \hat{d}(\mathbf{x}_i^t, \rho_r)) - \frac{1}{\eta}\sum_{j=1}^{N(t)} w_j\gamma_j^t\hat{d}(\mathbf{x}_{c_k^j}^t, \rho_k) \\
&\geq \delta - 2\varepsilon(1 + \frac{2}{w_m\delta}),
\end{aligned} \tag{23}$$

where the first and second inequalities follow from the triangle inequality, and the last inequality follows from (22), (21) and the definition of δ. Since the choice of ε is arbitrary, from (22) and (23) we obtain

$$\operatorname*{argmin}_{k\in 1..\kappa}\frac{1}{\eta}\sum_{j=1}^{N(t)} w_j\gamma_j^t\hat{d}(\mathbf{x}_i^t, \mathbf{x}_{c_k^j}^t) = r. \tag{24}$$

It remains to note that for any fixed $N \in \mathbb{N}$ from some t on (24) holds for all $i = 1..N$, and the consistency statement follows. ■

4.3 Unknown number κ of clusters

So far we have shown that when κ is known in advance, consistent clustering is possible under the only assumption that the samples are generated by unknown stationary ergodic process distributions. However, as follows from the theoretical impossibility results of (Ryabko, 2010c) discussed in Section 3, the correct number κ of clusters is not possible to be estimated with no further assumptions or additional constraints. One way to overcome this obstacle is to assume known rates of convergence of frequencies to the corresponding probabilities. Such rates are provided by assumptions on the mixing rates of the process distributions that generate the data.

Here we will show that under some assumptions on the mixing rates (and still without making any modelling or independence assumptions), consistent clustering is possible when the number of clusters is unknown.

The purpose of this section, however, is not to find the weakest assumptions under which consistent clustering (with κ unknown) is possible, nor is it to provide sharp bounds under assumptions considered; our only purpose here is to demonstrate that asymptotic consistency is achievable in principle when the number of clusters is unknown, under some mild non-parametric assumptions on the time series distributions. More refined analysis

could yield sharper bounds under weaker assumptions, such as those in, for example, (Bosq, 1996; Rio, 1999; Doukhan, 1994; Doukhan et al., 2010).

We introduce *mixing coefficients*, mainly following (Rio, 1999) in formulations. Informally, mixing coefficients of a stochastic process measure how fast the process forgets about its past. Any one-way infinite stationary process $X_1, X_2, \dots$ can be extended backwards to make a two-way infinite process $\dots, X_{-1}, X_0, X_1, \dots$ with the same distribution. In the definition below we assume such an extension. Define the φ-mixing coefficients of a process μ as

$$\varphi_n(\mu) = \sup_{A\in\sigma(X_{-\infty..k}),B\in\sigma(X_{k+n..\infty}),\mu(A)\neq 0} |\mu(B|A) - \mu(B)|, \tag{25}$$

where $\sigma(..)$ stands for the sigma-algebra generated by random variables in brackets. These coefficients are non-increasing. Define also

$$\theta_n(\mu) := 2 + 8(\varphi_1(\mu) + \dots + \varphi_n(\mu)).$$

A process μ is called uniformly φ-*mixing* if $\varphi_n(\mu) \to 0$. Many important classes of processes satisfy mixing conditions. For example, a stationary irreducible aperiodic Hidden Markov process with finitely many states is uniformly φ-mixing with coefficients decreasing exponentially fast. Other probabilistic assumptions can be used to obtain bounds on the mixing coefficients, see, e.g., (Bradley, 2005) and references therein.

The method that we propose for clustering mixing time series in the offline setting, namely *Algorithm 3*, is very simple. Its inputs are: samples $S := \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, the threshold level $\delta \in (0,1)$ and the parameters $m, l \in \mathbb{N}$, $B^{m,l,n}$. The algorithm assigns to the same cluster all samples which are at most δ-far from each other, as measured by $\hat{d}$ with $m_n = m, l_n = l$ and the summation over $B^{m,l}$ restricted to $B^{m,l,n}$. The sets $B^{m,l,n}$ have to be chosen so that in asymptotic they cover the whole space, $\cup_{n\in\mathbb{N}} B^{m,l,n} = B^{m,l}$. For example, $B^{m,l,n}$ may consist of the first b_n cubes around the origin, where $b_n \to \infty$ is a parameter sequence. We do not give a pseudo code implementation of this algorithm, since it is rather clear.

The idea is that the threshold level $\delta = \delta_n$ is selected according to the minimum length n of a sample and the (known bounds on) mixing rates of the process ρ generating the samples (see Theorem 13). As we show in Theorem 13, if the distribution of the samples satisfies $\varphi_n(\rho) \leq \varphi_n \to 0$, where φ_n are known, then one can select (based on φ_n only) the parameters of Algorithm 3 in such a way that it is weakly asymptotically consistent. Moreover, a bound on the probability of error before asymptotic is provided.

Theorem 13 (Algorithm 3 is consistent for unknown κ) *Fix sequences $m_n, l_n, b_n \in \mathbb{N}$, and let, for each $m, l \in \mathbb{N}$, $B^{m,l,n} \subset B^{m,l}$ be an increasing sequence of finite sets such that $\cup_{n\in\mathbb{N}} B^{m,l,n} = B^{m,l}$. Set $b_n := \max_{l\leq l_n, m\leq m_n} |B^{m,l,n}|$ and $n := \min_{i=1..N} n_i$. Let also $\delta_n \in (0,1)$. Let $N \in \mathbb{N}$, and suppose that the samples $\mathbf{x}_1, \dots, \mathbf{x}_N$ are generated in such a way that the (unknown marginal) distributions ρ_i, $i = 1..\kappa$ are stationary ergodic and satisfy $\varphi_n(\rho) \leq \varphi_n$, for all $n \in \mathbb{N}$. Then there exist constants ε_ρ, δ_ρ and n_ρ that depend only on ρ such that for all $\delta_n < \delta_\rho$, $n > n_\rho$ Algorithm 3 satisfies*

$$P(T \neq \mathcal{G}|_N) \leq 2N(N+1)(m_n l_n b_n \gamma_{n/2}(\delta_n) + \gamma_{n/2}(\varepsilon_\rho)), \tag{26}$$

where

$$\gamma_n(\varepsilon) := \sqrt{e}\exp(-n\varepsilon^2/\theta_n),$$

T is the partition output by the algorithm and $\mathcal{G}|_N$ is the ground-truth clustering. In particular, if $\varphi_n = o(1)$, then, selecting the parameters in such a way that $\delta_n = o(1)$, $m_n, l_n, b_n = o(n)$, $m_n, l_n \to \infty$, $\cup_{k\in\mathbb{N}} B^{m,l,k} = B^{m,l}$, $b_n^{m,l} \to \infty$, for all $m, l \in \mathbb{N}$, $\gamma_n(const) = o(1)$, and, finally, $m_n l_n b_n \gamma_n(\delta_n) = o(1)$, as is always possible, Algorithm 3 is weakly asymptotically consistent *(with the number of clusters κ unknown).*

Proof We use the following bound from (Rio, 1999, Corollary 2.1): for any zero-mean $[-1,1]$-valued random process $Y_1, Y_2, \dots$ and every $n \in \mathbb{N}$ we have

$$P\left(|\sum_{i=1}^{n} Y_i| > n\varepsilon\right) \le \gamma_n(\varepsilon). \tag{27}$$

For every $j = 1..N$, every $m < n$, $l \in \mathbb{N}$, and $B \in B^{m,l}$, define the $[-1,1]$-valued processes $\mathbf{Y}^j := Y_1^j, Y_2^j, \dots$ as

$$Y_t^j := \mathbb{I}\{(X_t^j, \dots, X_{t+m-1}^j) \in B\} - \rho_k(X_{1..m}^j \in B),$$

where ρ_k is the marginal distribution of X^j (that is, k is such that $j \in \mathcal{G}_k$). It is easy to see that φ-mixing coefficients for this process satisfy $\varphi_n(\mathbf{Y}^j) \le \varphi_{n-2m}$. Thus, from (27) we have

$$P(|\nu(X_{1..n_j}^j, B) - \rho_k(X_{1..m}^j \in B)| > \varepsilon/2) \le \gamma_{n-2m_n}(\varepsilon/2). \tag{28}$$

Then for every $i, j \in \mathcal{G}_k \cap 1..N$ for some $k \in 1..\kappa$ (that is, $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same ground-truth cluster) we have

$$P(|\nu(X_{1..n_i}^i, B) - \nu(X_{1..n_j}^j, B)| > \varepsilon) \le 2\gamma_{n-2m_n}(\varepsilon/2).$$

Using the union bound, summing over m, l, and B, we obtain

$$P(\hat{d}(\mathbf{x}_i, \mathbf{x}_j) > \varepsilon) \le 2m_n l_n b_n \gamma_{n-2m_n}(\varepsilon/2). \tag{29}$$

Next, let $i \in \mathcal{G}_k \cap 1..N$ and $j \in \mathcal{G}_{k'} \cap 1..N$ for $k \ne k' \in 1..\kappa$ (i.e., $\mathbf{x}_i, \mathbf{x}_j$ are in two different target clusters). Then, for some $m_{i,j}, l_{i,j} \in \mathbb{N}$ there is $B_{i,j} \in B^{m_{i,j}, l_{i,j}}$ such that for some $\tau_{i,j} > 0$ we have

$$|\rho_k(X_{1..|B_{i,j}|}^i \in B_{i,j}) - \rho_{k'}(X_{1..|B_{i,j}|}^j \in B_{i,j})| > 2\tau_{i,j}.$$

Then for every $\varepsilon < \tau_{i,j}/2$ we have

$$P(|\nu(X_{1..n_i}^i, B_{i,j}) - \nu(X_{1..n_j}^j, B_{i,j})| < \varepsilon) \le 2\gamma_{n-2m_{i,j}}(\tau_{i,j}). \tag{30}$$

Moreover, for $\varepsilon < w_{m_{i,j}} w_{l_{i,j}} \tau_{i,j}/2$ we have

$$\rho(\hat{d}(\mathbf{x}_i, \mathbf{x}_j) < \varepsilon) \le 2\gamma_{n-2m_{i,j}}(\tau_{i,j}). \tag{31}$$

Define

$$\delta_\rho := \min_{\substack{k\neq k'\in 1..\kappa \\ i\in\mathcal{G}_k\cap 1..N \\ j\in\mathcal{G}_{k'}\cap 1..N}} w_{m_{i,j}} w_{l_{i,j}} \tau_{i,j}/2, \quad \varepsilon_\rho := \min_{\substack{k\neq k'\in 1..\kappa \\ i\in\mathcal{G}_k\cap 1..N \\ j\in\mathcal{G}_{k'}\cap 1..N}} \tau_{i,j}/2$$

$$n_\rho := 2 \max_{\substack{k\neq k'\in 1..\kappa \\ i\in\mathcal{G}_k\cap 1..N \\ j\in\mathcal{G}_{k'}\cap 1..N}} m_{i,j}.$$

Clearly, from this and (30), for every $\delta < 2\delta_\rho$ and $n > n_\rho$ we obtain

$$\rho(\hat{d}(\mathbf{x}_i, \mathbf{x}_j) < \delta) \leq 2\gamma_{n/2}(\varepsilon_\rho). \tag{32}$$

Algorithm 3 produces correct results if for every pair i, j we have $\hat{d}(\mathbf{x}_i, \mathbf{x}_j) < \delta_n$ if and only if $i, j \in \mathcal{G}_k \cap 1..N$ for some $k \in 1..\kappa$. Therefore, taking the bounds (29) and (32) together for each of the $N(N+1)/2$ pairs of samples, we obtain (26). ■

For the online setting, consider the following simple extension of Algorithm 3, that we call Algorithm 3'. It applies Algorithm 3 to the first N_t sequences, where the parameters of Algorithm 3 and N_t are chosen in such a way that the bound (26) with $N = N_t$ is $o(1)$ and $N_t \to \infty$ as time t goes to infinity. It then assigns each of the remaining sequences ($\mathbf{x_i}$, $i > N_t$) to the nearest cluster. Note that in this case the bound (26) with $N = N_t$ bounds the error of Algorithm 3' on the first N_t sequences, as long as all of the κ clusters are already represented among the first N_t sequences. Since $N_t \to \infty$, we can formulate the following result.

Theorem 14 *Algorithm 3' is weakly asymptotically consistent in the online setting when the number κ of clusters is unknown, provided that the assumptions of Theorem 13 apply to the first N sequences $\mathbf{x_1}, \ldots, \mathbf{x_N}$ for every $N \in \mathbb{N}$.*

4.4 Extensions

In this section we show that some simple yet rather general extensions of the main results of this paper are possible, namely allowing for non-stationarity and for slight differences in distributions in the same cluster.

4.4.1 AMS processes and gradual changes

Here we argue that our results can be strengthened to a more general case where the process distributions that generate the data are Asymptotically Mean Stationary (AMS) ergodic. Throughout the paper we have been concerned with stationary ergodic process distributions. Recall from Section 2 that a process ρ is *stationary* if for any $i, j \in 1..n$ and $B \in B^m$, $m \in \mathbb{N}$, we have $\rho(X_{1..j} \in B) = \rho(X_{i..i+j-1} \in B)$. A stationary process is called *ergodic* if the limiting frequencies converge to their corresponding probabilities, so that for all $B \in \mathcal{B}$ with probability 1 we have $\lim_{n\to\infty} \nu(X_{1..n}, B) = \rho(B)$. This latter convergence of all frequencies is the only property of the process distributions that is used in the proofs (via Lemma 5) which give rise to our consistency results. We observe that this property also holds for a more general class of processes, namely those that are AMS

ergodic. Specifically, a process ρ is called *AMS* if for every $j \in 1..n$ and $B \in B^m, m \mathbb{N}$ the series $\lim_{n\to\infty} \sum_{i=1}^{n-j+1} \frac{1}{n}\rho(X_{i..i+j-1} \in B)$ converges to a limit $\bar{\rho}(B)$, which forms a measure, i.e. $\bar{\rho}(X_{1..j} \in B) := \bar{\rho}(B)$, $B \in B^m,\ m \in \mathbb{N}$, called *asymptotic mean* of ρ. Moreover, if ρ is an AMS process, then for every $B \in B^m,\ m \in \mathbb{N}$, the frequency $\nu(X_{1..n}, B)$ converges ρ-a.s. to a random variable with mean $\bar{\rho}(B)$. Similarly to stationary processes, if the random variable to which $\nu(X_{1..n}, B)$ converges is a.s. constant, then ρ is called AMS ergodic. More information on AMS processes can be found in (Gray, 1988). However, the main characteristic pertaining to our work is that the class of all processes with AMS properties is composed precisely of those processes for which the almost sure convergence of frequencies to the corresponding probabilities holds. It is thus easy to check that all the asymptotic results of Sections 4.1, 4.2 carry over to the more general setting where the unknown process distributions that generate the data are AMS ergodic.

4.4.2 Strictly separated clusters of distributions

So far we have defined a cluster as a set of sequences generated by the same distribution. This seems to capture rather well the notion that in the same cluster the objects can be very different (as is the case for stochastically generated sequences), yet are intrinsically of the same nature (they have the same law).

However, one may wish to generalize this further, and allow each sequence to be generated by a different distribution, yet requiring that in the same clusters distributions must be close. Unlike the original formulation, such an extension would require fixing some similarity measure between distributions. The results of the preceding sections suggest using the distributional distance for this purpose.

Specifically, as discussed in Section 4.1, in our formulation, from some time on, the sequences possess the so-called strict separation property in the $\hat{d}$ distance: sequences in the same target cluster are closer to each other than to those in other clusters. One possible way to relax the considered setting is to impose the strict separation property on the distributions that generate the data. Here the separation would be with respect to the distributional distance d. That is, each sequence $\mathbf{x}_i, i = 1..N$, may be generated by its own distribution ρ_i, but the distributions $\{\rho_i : i = 1..N\}$ can be clustered in such a way that the resulting clustering has the strict separation property with respect to d. The goal would then be to recover this clustering based on the given samples. In fact, it can be shown that the offline Algorithm 1 of Section 4.1 is consistent in this setting as well. How this transfers to the online setting remains open. For the offline case, we can formulate the following result, whose proof is analogous to that of Theorem 11.

Theorem 15 *Assume that each sequence $\mathbf{x}_i$, $i = 1..N$ is generated by a stationary ergodic distribution ρ_i. Assume further that the set of distributions $\{1, \dots, N\}$ admits a partitioning $\mathcal{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_k\}$ that has the strict separation property with respect to d: for all $i, j = 1..k, i \neq j$, for all $\rho_1, \rho_2 \in \mathcal{G}_i$ and all $\rho_3 \in \mathcal{G}_j$ we have $d(\rho_1, \rho_2) < d(\rho_1, \rho_3)$. Then Algorithm 1 is strongly asymptotically consistent, in the sense that almost surely from some $n = \min\{n_1, \dots, n_N\}$ on it outputs the set $\mathcal{G}$.*

5. Computational considerations

In this section we show that all the proposed methods are efficiently computable. This claim is further illustrated by the experimental results in the next section. Note, however, that since the results presented are asymptotic, the question of what is the best achievable computational complexity of an algorithm that still has the same asymptotic performance guarantees is meaningless: for example, an algorithm could throw away most of the data and still be asymptotically consistent. This is why we do not attempt to find a resource-optimal way of computing the methods presented.

First, we show that calculating $\hat{d}$ is at most quadratic (up to log terms), and quasilinear if we use $m_n = \log n$. Let us begin by showing that calculating $\hat{d}$ is fully tractable with $m_n, l_n \equiv \infty$. First, observe that for fixed m and l, the sum

$$T^{m,l} := \sum_{B \in B^{m,l}} |\nu(X^1_{1..n_1}, B) - \nu(X^2_{1..n_2}, B)| \tag{33}$$

has not more than $n_1 + n_2 - 2m + 2$ non-zero terms (assuming $m \leq n_1, n_2$; the other case is obvious). Indeed, there are $n_i - m + 1$ tuples of size m in each sequence $\mathbf{x}_i,\ i = 1, 2$ namely, $X^i_{1..m}, X^i_{2..m+1}, \ldots, X^i_{n_1-m+1..n_1}$. Therefore, $T^{m,l}$ can be obtained by a finite number of calculations.

Furthermore, let

$$s = \min_{\substack{X^1_i \neq X^2_j \\ i=1..n_1, j=1..n_2}} |X^1_i - X^2_j|. \tag{34}$$

and observe that $T^{m,l} = 0$ for all $m > n$ and for each m, for all $l > \log s^{-1}$ the term $T^{m,l}$ is constant. That is, for each fixed m we have

$$\sum_{l=1}^{\infty} w_m w_l T^{m,l} = w_m w_{\log s^{-1}} T^{m,\log s^{-1}} + \sum_{l=1}^{\log s^{-1}} w_m w_l T^{m,l}.$$

so that we simply double the weight of the last non-zero term. (Note also that s is bounded above by the length of the binary precision in representing the random variables X^i_j.) Thus, even with $m_n, l_n \equiv \infty$ one can calculate $\hat{d}$ precisely. Moreover, for a fixed $m \in 1..\log n$ and $l \in 1..\log s^{-1}$ for every sequence $\mathbf{x}_i,\ i = 1, 2$ the frequencies $\nu(\mathbf{x}_i, B),\ B \in B^{m,l}$ may be calculated using suffix trees or suffix arrays, with $\mathcal{O}(n)$ worst case construction and search complexity (see, e.g., Ukkonen, 1995). Searching all $z := n - m + 1$ occurrences of subsequences of length m results in $\mathcal{O}(m + z) = \mathcal{O}(n)$ complexity. This brings the overall computational complexity of (3) to $\mathcal{O}(nm_n \log s^{-1})$; this can potentially be improved using specialized structures, e.g., (Grossi and Vitter, 2005).

The following consideration can be used to set m_n. For a fixed l the frequencies $\nu(\mathbf{x}_i, B),\ i = 1, 2$ of cells in $B \in B^{m,l}$ corresponding to values of $m > \log_n$ (in the asymptotic sense, that is, if $\log_n = o(m)$) are, in general, not consistent estimates of their probabilities (and thus only add to the estimation error). More specifically, for a subsequence $X_{j..j+m}$ with $j = 1..n - m$ of length m the probability $\rho_i(X_{j..j+m} \in B),\ i = 1, 2$ is of order $2^{-mh_i},\ i = 1, 2$ where h_i denotes the entropy rate of $\rho_i,\ i = 1, 2$. Moreover, under some (general) conditions one can show that a string of length $\log n / h_i$ on average repeats only

once in a string of length n (Kontoyiannis and Suhov, 1994). Therefore, subsequences of length m larger than $\log n$ are typically met 0 or 1 times, and thus are not consistent estimates of probabilities. By the above argument, one can set $m_n := \log n$. To choose $l_n < \infty$ one can either fix some constant based on the bound on the precision in real computations, or choose it in such a way that each cell B^{m,l_n} contains no more than $\log n$ points for all $m = 1..\log n$ largest values of l_n.

Complexity of the algorithms. The computational complexity of the presented algorithms is dominated by the complexity of calculations of $\hat{d}$ between different pairs of sequences. Thus, it is enough to bound the number of pairwise $\hat{d}$ computations.

It is easy to see that the offline algorithm for the case of known κ (Algorithm 1) requires at most κN distance calculations, while for the case of unknown κ all N^2 distance calculations are necessary.

The computational complexity of the updates in the online algorithm can be computed as follows. Assume that the pairwise distance values are stored in a database D, and that for every sequence $\mathbf{x}_i^{t-1},\ i \in \mathbb{N}$ we have already constructed a suffix tree, using for example, the online algorithm of (Ukkonen, 1995). At time-step t, a new symbol X is received. Let us first calculate the required computations to update D. We have two cases, either X forms a new sequence, so that $N(t) = N(t-1) + 1$, or it is the subsequent element of a previously received segment, say, $\mathbf{x}_j^t$ for some $j \in 1..N(t)$, so that $n_j(t) = n_j(t-1) + 1$. In either case, let $\mathbf{x}_j^t$ denote the updated sequence. Note that for all $i \neq j \in 1..N(t)$ we have $n_i(t) = n_i(t-1)$. Recall the notation $\mathbf{x}_i^t := X_1^{(i)}, \ldots X_{n_i(t)}^{(i)}$ for $i \in 1..N(t)$. In order to update D we need to update the distance between $\mathbf{x}_j^t$ and $\mathbf{x}_i^t$ for all $i \neq j \in N(t)$. Thus, we need to search for all m_n new patterns induced by the received symbol X, resulting in complexity at most $\mathcal{O}(N(t)m_n^2 l_n)$. Let $n(t) := \max\{n_1(t), \ldots n_{N(t)}(t)\},\ t \in \mathbb{N}$. As discussed previously, we let $m_n := \log n(t)$; we also define $l_n := \log s(t)^{-1}$ where

$$s(t) := \min_{\substack{i,j \in 1..N(t) \\ u=1..n_i(t), v=1..n_j(t), X_u^{(i)} \neq X_v^{(j)}}} |X_u^{(i)} - X_v^{(j)}|,\ t \in \mathbb{N}.$$

Thus, the per symbol complexity of updating D is at most $\mathcal{O}(N(t)\log^3 n(t))$. However, note that if $s(t)$ decreases from one time-step to the next, updating D will have a complexity of order equivalent to its complete construction, resulting in a computational complexity of order $\mathcal{O}(N(t)n(t)\log^2 n(t))$. Therefore, we avoid calculating $s(t)$ at every time-step; instead, we update $s(t)$ at pre-specified time-steps so that for every $n(t)$ symbols received, D is reconstructed at most $\log n(t)$ times. (This can be done, for example, by recalculating $s(t)$ at time-steps where $n(t)$ is a power of 2.) It is easy to see that with the database D of distance values at hand, the rest of the computations are of order at most $\mathcal{O}(N(t)^2)$. Thus, the computational complexity of updates in Algorithm 2 is at most $\mathcal{O}(N(t)^2 + N(t)\log^3 n(t))$.

6. Experimental Results

In this section we present empirical evaluations of Algorithms 1 and 2 on both synthetically generated and real data.

6.1 Synthetic Data

We start with synthetic experiments. In order for the experiments to reflect the generality of our approach we have selected time series distributions that, while being stationary ergodic, cannot be put into any of usual smaller classes of time series, and are difficult to approximate by finite-state models. Namely, we consider rotation processes, used, for example, by Shields (1996) as as an example of stationary ergodic processes that are not B-processes. Such time series cannot be modelled by a hidden Markov model with a finite or countably infinite set of states. Moreover, while k-order Markov (or hidden Markov) approximations of this process converge to it in distributional distance, they do not converge to it in the $\bar{d}$ distance, a stronger distance than d whose empirical approximations are often used to study general (non-Markovian) processes (e.g., Ornstein and Weiss, 1990).

6.1.1 Time series generation

To generate a sequence $\mathbf{x} = X_{1..n}$ we proceed as follows: Fix some parameter $\alpha \in (0, 1)$. Select $r_0 \in [0, 1]$; then, for each $i = 1..n$ obtain r_i by shifting r_{i-1} by α to the right, and removing the integer part, i.e. $r_i := r_{i-1} + \alpha - \lfloor r_{i-1} + \alpha \rfloor$. The sequence $\mathbf{x} = (X_1, X_2, \cdots)$ is then obtained from r_i by thresholding at 0.5, that is $X_i := \mathbb{I}\{r_i > 0.5\}$. If α is irrational then $\mathbf{x}$ forms a stationary ergodic time series. (We simulate α by a `longdouble` with a long mantissa.)

For the purpose of our experiments, first we fix $\kappa := 5$ difference process distributions specified by $\alpha_1 = 0.31...,\ \alpha_2 = 0.33...,\ \alpha_3 = 0.35...,\ \alpha_4 = 0.37...,\ \alpha_5 = 0.39....$ The parameters α_i are intentionally selected to be close, in order to make the process distributions harder to distinguish. Next we generate an $N \times M$ data matrix $\mathbf{X}$, each row of which is a sequence generated by one of the process distributions. Our task in both the online and the batch setting is to cluster the rows of $\mathbf{X}$ into $\kappa = 5$ clusters.

6.1.2 Batch Setting

In this experiment we demonstrate that in the batch setting, the clustering errors corresponding to both the online and the offline algorithms converge to 0 as the sequence-lengths grow. To this end, at every time-step t we take an $N \times n(t)$ sub-matrix $\mathbf{X}|_{\mathbf{n(t)}}$ of $\mathbf{X}$ composed of the rows of $\mathbf{X}$ terminated at length $n(t)$, where $n(t) = 5t$. Then at each iteration we let each of the algorithms, (online and offline) cluster the rows of $\mathbf{X}|_{\mathbf{n(t)}}$ into five clusters, and calculate the clustering error-rate of each algorithm. As shown in Figure 2 (top) the error rate of each algorithm decreases with sequence length.

6.1.3 Online Setting

In this experiment we demonstrate that, unlike the online algorithm, the offline algorithm is consistently confused by the new sequences arriving at each time-step in an online setting. To simulate an online setting, we proceed as follows: At every time-step t, a triangular window is used to reveal the first $1..n_i(t),\ i = 1..t$ elements of the first t rows of the data-matrix $\mathbf{X}$, with $n_i(t) := 5(t - i) + 1,\ i = 1..t$. This gives a total of t sequences, each of length $n_i(t)$, for $i = 1..t$, where the i^{th} sequence for $i = 1..t$ corresponds to the i^{th} row of $\mathbf{X}$ terminated at length $n_i(t)$. At every time-step t the online and offline algorithms

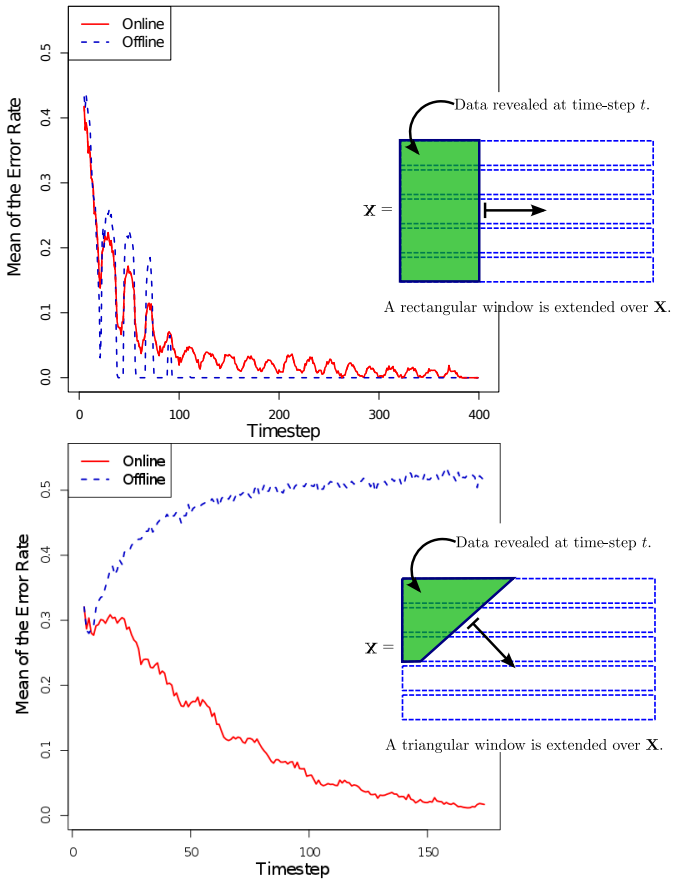


Figure 2: Top: error-rate vs. sequence length in batch setting. Bottom: error-rate vs. Number of observed samples in online setting. (error-rates averaged over 100 runs.)

are each used in turn to cluster the observed t sequences into five clusters. Note that the performance of both algorithms is measured on all sequences available at a given time, not on a fixed batch of sequences. As shown in Figure 2 (bottom), in this setting the clustering error-rate of the offline algorithm remains consistently high, whereas that of the online algorithm converges to zero.

6.2 Real Data

As an application we consider the problem of clustering motion capture sequences, where groups of sequences with similar dynamics are to be identified. Data is taken from the Motion Capture database (MOCAP) (cmu) which consists of time series data representing human locomotion. The sequences are composed of marker positions on human body which are tracked spatially through time for various activities.

We compare our results to those obtained with two other methods, namely those of (Li and Prakash, 2011) and (Jebara et al., 2007), which (to the best of our knowledge) constitute the state-of-the-art performance on these datasets. Note that we have not implemented these reference methods, rather we have taken the numerical results directly from the corresponding articles. In order to have common grounds for each comparison we use the same sets of sequences,[2] and the same means of evaluation as those used by Li and Prakash (2011); Jebara et al. (2007).

In the paper by (Li and Prakash, 2011) two MOCAP datasets[3] are used, where the sequences in each dataset are labelled with either running or walking as annotated in the database. Performance is evaluated via the conditional entropy S of the true labelling with respect to the prediction, i.e., $S := -\sum_{i,j} \frac{\mathcal{M}_{ij}}{\sum_{i',j'} \mathcal{M}_{i'j'}} \log \frac{\mathcal{M}_{ij}}{\sum_{j'} \mathcal{M}_{ij'}}$ where $\mathcal{M}$ denotes the clustering confusion matrix. The motion sequences used by Li and Prakash (2011) are reportedly trimmed to equal duration. However, we use the original sequences as our method is not limited by variation in sequence lengths. Table 1 lists performance of Algorithm 1 as well as that reported for the method of Li and Prakash (2011); Algorithm 1 performs consistently better.

In the paper (Jebara et al., 2007) four MOCAP datasets[4] are used, corresponding to four motions: run, walk, jump and forward jump. Table 2 lists performance in terms of accuracy. The datasets in Table 2 constitute two types of motions:

1. motions that can be considered ergodic: walk, run, run/jog (displayed above the double line), and

2. non-ergodic motions: single jumps (displayed below the double line).

As shown in Table 2, Algorithm 1 achieves consistently better performance on the first group of datasets, while being competitive (better on one and worse on the other) on the non-ergodic motions. The time taken to complete each task is in the order of few minutes on a standard laptop computer.

7. Discussion

We have proposed a natural notion of consistency for clustering time series in both the online and the offline scenarios. While in this work we have taken some first steps in investigating the theoretical and algorithmic questions arising in the proposed framework, there are many open problems and exciting directions for future research remaining to be explored. Some of these are discussed in this section.

Rates, optimality. The main focus of this work is on the most general case of highly dependent time series. On the one hand, this captures best the spirit of the unsupervised learning problem in question: the nature of the data is completely unknown, and one tries to find some structure in it. On the other hand, as discussed above, in this generality rates of convergence and finite-sample performance guarantees are provably impossible to obtain, and thus one cannot argue about optimality. While we have provided some results on a

2. marker position: the subject's right foot.
3. subjects #16 and #35.
4. subjects #7, #9, #13, #16 and #35.

	Dataset	(Li and Prakash, 2011)	Algorithm 1
1.	Walk vs. Run (#35)	0.1015	**0**
2.	Walk vs.Run (#16)	0.3786	**0.2109**

Table 1: Comparison with (Li and Prakash, 2011): Performance in terms of entropy; datasets concern ergodic motion captures.

	Dataset	(Jebara et al., 2007)	Algorithm 1
1.	Run(#9) vs. Run/Jog(#35)	**100%**	**100%**
2.	Walk(#7) vs. Run/Jog(#35)	95%	**100%**
3.	Jump vs. Jump fwd.(#13)	87%	**100%**
4.	Jump vs. Jump fwd.(#13, 16)	**66%**	60%

Table 2: Comparison with (Jebara et al., 2007): Performance in terms of accuracy; Rows 1 & 2 concern ergodic, Rows 3 & 4 concern non-ergodic motion captures.

more restrictive setting (time series with mixing, Section 4.3), the question of what are the optimal performance guarantees for different classes of time series remains open. In fact, the first interesting question in this direction is not about time series with mixing, but about i.i.d. series What is the minimal achievable probability of clustering error in this setting, for finite sample sizes, and what algorithms attain it?

Online setting: bad points. In the online setting of Section 4.2 we have assumed that the length of each sequence grows to infinity with time. While this is a good first approximation, this assumption may not be practical. It is interesting to consider the situation in which some sequences stop growing at some point; moreover, it can be assumed that such sequences are not representative of the corresponding distribution. While this clearly makes the problem much more difficult, already in the setting considered in this work we are dealing with "bad" sequences at each time step: these are those sequences which are as yet too short to be informative. This hints at the possibility of obtaining consistent algorithm in the extended setting outlined.

Other metrics and non-metric-based methods. All of the methods presented in this paper are based on the distributional distance d. The main property of this distance that we exploit is that it can be estimated consistently. In principle, one can use other distances with this property, in order to obtain consistent clustering algorithms. While there are not many known distances that can be consistently estimated for arbitrary stationary ergodic distributions, there is at least one, namely the telescope distance recently introduced in (Ryabko and Mary, 2012). Moreover, the definition of consistency proposed does not entail the need to use a distance between time-series distributions. As an alternative, the use of compression-based methods can be considered. Such methods have been used to solve various statistical problems concerning stationary ergodic time series (Ryabko and Astola, 2006; Ryabko, 2010a). Compression-based methods have also been used for clustering time series data before, albeit without asymptotic consistency analysis, by Cilibrasi and Vitanyi (2005). Combining our consistency framework with these compression-based methods is a promising direction for further research.

Acknowledgments

This work is supported by the French Ministry of Higher Education and Research, by FP7/2007-2013 under grant agreements 270327 (CompLACS), by the Nord-Pas-de-Calais Regional Council and FEDER. Most of the work has been done while Azadeh Khaleghi was a PhD student at INRIA Lille - Nord Europe.

References

CMU graphics lab motion capture database. URL `http://mocap.cs.cmu.edu/`.

F.R. Bach and M.I. Jordan. Learning graphical models for stationary time series. *IEEE Transactions on Signal Processing*, 52(8):2189 – 2199, aug. 2004.

M.F. Balcan, A. Blum, and S. Vempala. A discriminative framework for clustering via similarity functions. In *STOC*, 2008.

C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(7):719–725, 2000.

P. Billingsley. Statistical inference about Markov chains. *Ann. Math. Stat.*, 32(1):12–40, 1961.

P. Billingsley. *Convergence of probability measures*. John Wiley & Sons, 1999.

D. Bosq. *Nonparametric Statistics for Stochastic Processes*. Estimation and Prediction. Springer, 1996. ISBN 0-387-94713-2.

R.C. Bradley. Basic properties of strong mixing conditions. a survey and some open questions. *Probability Surveys*, 2:107–144, 2005.

E. Carlstein and S. Lele. Nonparametric change-point estimation for data from an ergodic sequence. *Teor. Veroyatnost. i Primenen.*, 38:910–917, 1993.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006. ISBN 0521841089.

R. Cilibrasi and P.M.B. Vitanyi. Clustering by compression. *IEEE Transactions on Information Theory*, 51:1523–1545, 2005.

Paul Doukhan. *Mixing*. Springer, 1994.

Paul Doukhan, Gabriel Lang, Donatas Surgailis, and Gilles Teyssière. *Dependence in Probability and Statistics*. Springer, 2010.

R. Gray. *Probability, Random Processes, and Ergodic Properties*. Springer Verlag, 1988.

Roberto Grossi and Jeffrey Scott Vitter. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. *SIAM Journal on Computing*, 35(2): 378–407, 2005.

M. Gutman. Asymptotically optimal classification for multiple tests with empirically observed statistics. *IEEE Transactions on Information Theory*, 35(2):402–408, 1989.

T. Jebara, Y. Song, and K. Thadani. Spectral clustering and embedding with hidden markov models. *Machine Learning: ECML 2007*, pages 164–175, 2007.

I. Katsavounidis, C.-C. Jay Kuo, and Zhen Zhang. A new initialization technique for generalized lloyd iteration. *IEEE Signal Processing Letters*, 1:144–146, 1994.

A. Khaleghi and D. Ryabko. Locating changes in highly-dependent data with unknown number of change points. In *Neural Information Processing Systems (NIPS)*, Lake Tahoe, Nevada, United States, 2012.

A. Khaleghi, D. Ryabko, J. Mary, and P. Preux. Online clustering of processes. In *AISTATS*, JMLR W&CP 22, pages 601–609, 2012.

Azadeh Khaleghi and Daniil Ryabko. Asymptotically consistent estimation of the number of change points in highly dependent time series. In *ICML*, JMLR W&CP, pages 539–547, Beijing, China, 2014.

J. Kleinberg. An impossibility theorem for clustering. In *15th Conf. Neiral Information Processing Systems (NIPS'02)*, pages 446–453, Montreal, Canada, 2002. MIT Press.

I. Kontoyiannis and Y.M. Suhov. Prefixes and the entropy rate for long-range sources. In *IEEE International Symposium On Information Theory*, pages 194–194, 1994.

M. Kumar, N.R. Patel, and J. Woo. Clustering seasonality patterns in the presence of errors. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 557–563. ACM, 2002.

E. Lehmann. *Testing Statistical Hypotheses, 2nd edition.* Wiley, New York, 1986.

L. Li and B.A. Prakash. Time series clustering: Complex is simpler! 2011.

M. Mahajan, P. Nimbhorkar, and K. Varadarajan. The planar k-means problem is np-hard. In *WALCOM '09: Proceedings of the 3rd International Workshop on Algorithms and Computation*, pages 274–285, Berlin, Heidelberg, 2009. Springer-Verlag.

G. Morvai and B. Weiss. On classifying processes. *Bernoulli*, 11(3):523–532, 2005.

Gusztáv Morvai and Benjamin Weiss. A note on prediction for discrete time series. *Kybernetika*, 48(4):809–823, 2012.

D.S. Ornstein and B. Weiss. How sampling reveals a process. *Annals of Probability*, 18(3): 905–930, 1990.

Emmanuel Rio. *Théorie asymptotique des processus aléatoires faiblement dépendants*, volume 31. Springer, 1999.

B. Ryabko. Prediction of random sequences and universal coding. *Problems of Information Transmission*, 24:87–96, 1988.

B. Ryabko and J. Astola. Universal codes as a basis for time series testing. *Statistical Methodology*, 3:375–397, 2006.

Boris Ryabko. Applications of universal source coding to statistical analysis of time series. *Selected Topics in Information and Coding Theory, World Scientific Publishing*, pages 289–338, 2010a.

D. Ryabko. Clustering processes. In *Proc. the 27th International Conference on Machine Learning (ICML 2010)*, pages 919–926, Haifa, Israel, 2010b.

D. Ryabko. Discrimination between B-processes is impossible. *Journal of Theoretical Probability*, 23(2):565–575, 2010c.

D. Ryabko. Testing composite hypotheses about discrete ergodic processes. *Test*, 21(2): 317–329, 2012.

D. Ryabko. Uniform hypothesis testing for finite-valued stationary processes. *Statistics*, 48 (1):121–128, 2014.

D. Ryabko and B. Ryabko. Nonparametric statistical inference for ergodic processes. *IEEE Transactions on Information Theory*, 56(3):1430–1435, 2010.

Daniil Ryabko and Jeremie Mary. Reducing statistical time-series problems to binary classification. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2069–2077. 2012.

P. Shields. *The Ergodic Theory of Discrete Sample Paths*. AMS Bookstore, 1996.

P. Smyth. Clustering sequences with hidden Markov models. In *Advances in Neural Information Processing Systems*, pages 648–654. MIT Press, 1997.

Esko Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

Shi Zhong and Joydeep Ghosh. A unified framework for model-based clustering. *Journal of Machine Learning Research*, 4:1001–1037, 2003.

Journal of Machine Learning Research 14 (2013) 2837-2856 Submitted 3/13; Revised 6/13; Published 9/13

A Binary-Classification-Based Metric between Time-Series Distributions and Its Use in Statistical and Learning Problems

Daniil Ryabko DANIIL.RYABKO@INRIA.FR
Jérémie Mary JEREMIE.MARY@INRIA.FR
SequeL-INRIA/LIFL-CNRS
Université de Lille, France
40, avenue de Halley
59650 Villeneuve d'Ascq
France

Editor: Léon Bottou

Abstract

A metric between time-series distributions is proposed that can be evaluated using binary classification methods, which were originally developed to work on i.i.d. data. It is shown how this metric can be used for solving statistical problems that are seemingly unrelated to classification and concern highly dependent time series. Specifically, the problems of time-series clustering, homogeneity testing and the three-sample problem are addressed. Universal consistency of the resulting algorithms is proven under most general assumptions. The theoretical results are illustrated with experiments on synthetic and real-world data.

Keywords: time series, reductions, stationary ergodic, clustering, metrics between probability distributions

1. Introduction

Binary classification is one of the most well-understood problems of machine learning and statistics: a wealth of efficient classification algorithms has been developed and applied to a wide range of applications. Perhaps one of the reasons for this is that binary classification is conceptually one of the simplest statistical learning problems. It is thus natural to try and use it as a building block for solving other, more complex, newer or just different problems; in other words, one can try to obtain efficient algorithms for different learning problems by reducing them to binary classification. This approach has been applied to many different problems, starting with multi-class classification, and including regression and ranking (Balcan et al., 2007; Langford et al., 2006), to give just a few examples. However, all of these problems are formulated in terms of independent and identically distributed (i.i.d.) samples. This is also the assumption underlying the theoretical analysis of most of the classification algorithms.

In this work we consider learning problems that concern time-series data for which independence assumptions do not hold. The series can exhibit arbitrary long-range dependence, and different time-series samples may be interdependent as well. Moreover, the learning problems that we consider—the three-sample problem, time-series clustering, and homogeneity testing—at first glance seem completely unrelated to classification.

We show how the considered problems can be reduced to binary classification methods, via a new metric between time-series distributions. The results include asymptotically consistent algorithms, as well as finite-sample analysis. To establish the consistency of the suggested methods, for clustering and the three-sample problem the only assumption that we make on the data is that the distributions generating the samples are stationary ergodic; this is one of the weakest assumptions used in statistics. For homogeneity testing we have to make some mixing assumptions in order to obtain consistency results (this is indeed unavoidable, as shown by Ryabko, 2010b). Mixing conditions are also used to obtain finite-sample performance guarantees for the first two problems.

The proposed approach is based on a new distance between time-series distributions (that is, between probability distributions on the space of infinite sequences), which we call *telescope distance*. This distance can be evaluated using binary classification methods, and its finite-sample estimates are shown to be asymptotically consistent. Three main building blocks are used to construct the telescope distance. The first one is a distance on finite-dimensional marginal distributions. The distance we use for this is the following well-known metric: $d_{\mathcal{H}}(P,Q) := \sup_{h\in\mathcal{H}} |\mathbf{E}_P h - \mathbf{E}_Q h|$ where P, Q are distributions and $\mathcal{H}$ is a set of functions. This distance can be estimated using binary classification methods, and thus can be used to reduce various statistical problems to the classification problem. This distance was previously applied to such statistical problems as homogeneity testing and change-point estimation (Kifer et al., 2004). However, these applications so far have only concerned i.i.d. data, whereas we want to work with highly-dependent time series. Thus, the second building block are the recent results of Adams and Nobel (2012), that show that empirical estimates of $d_{\mathcal{H}}$ are consistent (under certain conditions on $\mathcal{H}$) for arbitrary stationary ergodic distributions. This, however, is not enough: evaluating $d_{\mathcal{H}}$ for (stationary ergodic) time-series distributions means measuring the distance between their finite-dimensional marginals, and not the distributions themselves. Finally, the third step to construct the distance is what we call *telescoping*. It consists in summing the distances for all the (infinitely many) finite-dimensional marginals with decreasing weights. The resulting distance can "automatically" select the marginal distribution of the right order: marginals which cannot distinguish between the distributions give distance estimates that converge to zero, while marginals whose orders are too high to have converged have very small weights. Thus, the estimate is dominated by the marginals which can distinguish between the time-series distributions, or converges to zero if the distributions are the same. It is worth noting that a similar telescoping trick is used in different problems, most notably, in sequence prediction (Solomonoff, 1978; B. Ryabko, 1988; Ryabko, 2011); it is also used in the distributional distance (Gray, 1988), see Section 8 below.

We show that the resulting distance (telescope distance) indeed can be consistently estimated based on sampling, for arbitrary stationary ergodic distributions. Further, we show how this fact can be used to construct consistent algorithms for the considered problems on time series. Thus we can harness binary classification methods to solve statistical learning problems concerning time series. A remarkable feature of the resulting methods is that the performance guarantees obtained do not depend on the approximation error of the binary classification methods used, they only depend on their estimation error.

Moreover, we analyse some other distances between time-series distributions, the possibility of their use for solving the statistical problems considered, and the relation of these distances to the telescope distance introduced in this work.

To illustrate the theoretical results in an experimental setting, we chose the problem of time-series clustering, since it is a difficult unsupervised problem which seems most different from the

problem of binary classification. Experiments on both synthetic and real-world data are provided. The real-world setting concerns brain-computer interface (BCI) data, which is a notoriously challenging application, and on which the presented algorithm demonstrates competitive performance.

A related approach to address the problems considered here, as well as some related problems about stationary ergodic time series, is based on (consistent) empirical estimates of the distributional distance, see Ryabko and Ryabko (2010), Ryabko (2010a), Khaleghi et al. (2012), as well as Gray (1988) about the distributional distance. The empirical distance is based on counting frequencies of bins of decreasing sizes and "telescoping." This distance is described in some detail in Section 8 below, where we compare it to the telescope distance. Another related approach to time-series analysis involves a different reduction, namely, that to data compression (B. Ryabko, 2009).

1.1 Organisation

Section 2 is preliminary. In Section 3 we introduce and discuss the telescope distance. Section 4 explains how this distance can be calculated using binary classification methods. Sections 5 and 6 are devoted to the three-sample problem and clustering, respectively. In Section 7, under some mixing conditions, we address the problems of homogeneity testing, clustering with unknown k, and finite-sample performance guarantees. In Section 8 we take a look at other distances between time-series distributions and their relations to the telescope distance. Section 9 presents experimental evaluation.

2. Notation and Definitions

Let $(\mathcal{X}, \mathcal{F}_1)$ be a measurable space (the domain), and denote $(\mathcal{X}^k, \mathcal{F}_k)$ and $(\mathcal{X}^{\mathbb{N}}, \mathcal{F})$ the product probability space over $\mathcal{X}^k$ and the induced probability space over the one-way infinite sequences taking values in $\mathcal{X}$. Time-series (or process) distributions are probability measures on the space $(X^{\mathbb{N}}, \mathcal{F})$. We use the abbreviation $X_{1..k}$ for $X_1, \dots, X_k$. A set $\mathcal{H}$ of functions is called *separable* if there is a countable set $\mathcal{H}'$ of functions such that any function in $\mathcal{H}$ is a pointwise limit of a sequence of elements of $\mathcal{H}'$.

A distribution ρ is called stationary if $\rho(X_{1..k} \in A) = \rho(X_{n+1..n+k} \in A)$ for all $A \in \mathcal{F}_k$, $k, n \in \mathbb{N}$. A stationary distribution is called (stationary) ergodic if

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1..n-k+1} \mathbb{I}_{X_{i..i+k} \in A} = \rho(A) \;\; \rho - \text{a.s.}$$

for every $A \in \mathcal{F}_k$, $k \in \mathbb{N}$. (This definition, which is more suited for the purposes of this work, is equivalent to the usual one expressed in terms of invariant sets, see, e.g., Gray, 1988.)

3. A Distance between Time-Series Distributions

We start with a distance between distributions on $\mathcal{X}$, and then we extend it to distributions on $\mathcal{X}^{\mathbb{N}}$. For two probability distributions P and Q on $(\mathcal{X}, \mathcal{F}_1)$ and a set $\mathcal{H}$ of measurable functions on $\mathcal{X}$, one can define the distance

$$d_{\mathcal{H}}(P, Q) := \sup_{h \in \mathcal{H}} |\mathbf{E}_P h - \mathbf{E}_Q h|. \tag{1}$$

This metric in its general form has been studied at least since the 80's (Zolotarev, 1983); its special cases include Kolmogorov-Smirnov (Kolmogorov, 1933), Kantorovich-Rubinstein (Kantorovich

and Rubinstein, 1957) and Fortet-Mourier (Fortet and Mourier, 1953) metrics. Note that the distance function so defined may not be measurable; however, it is measurable under mild conditions which we assume whenever necessary. In particular, separability of $\mathcal{H}$ is a sufficient condition (separability is required in most of the results below).

We are interested in the cases where $d_{\mathcal{H}}(P,Q)=0$ implies $P=Q$. Note that in this case $d_{\mathcal{H}}$ is a metric (the rest of the properties are easy to see). For reasons that will become apparent shortly (see Remark below), we are mainly interested in the sets $\mathcal{H}$ that consist of indicator functions. In this case we can identify each $f\in\mathcal{H}$ with the indicator set $\{x : f(x)=1\}\subset\mathcal{X}$ and (by a slight abuse of notation) write $d_{\mathcal{H}}(P,Q) := \sup_{h\in\mathcal{H}}|P(h)-Q(h)|$. In this case it is easy to check that the following statement holds true.

Lemma 1 *$d_{\mathcal{H}}$ is a metric on the space of probability distributions over $\mathcal{X}$ if and only if $\mathcal{H}$ generates $\mathcal{F}_1$.*

The property that $\mathcal{H}$ generates $\mathcal{F}_1$ is often easy to verify directly. First of all, it trivially holds for the case where $\mathcal{H}$ is the set of halfspaces in a Euclidean $\mathcal{X}$. It is also easy to check that it holds if $\mathcal{H}$ is the set of halfspaces in the feature space of most commonly used kernels (provided the feature space is of the same or higher dimension than the input space), such as polynomial and Gaussian kernels.

Based on $d_{\mathcal{H}}$ we can construct a distance between time-series probability distributions. For two time-series distributions ρ_1,ρ_2 we take the $d_{\mathcal{H}}$ between k-dimensional marginal distributions of ρ_1 and ρ_2 for each $k\in\mathbb{N}$, and sum them all up with decreasing weights.

Definition 2 (telescope distance $D_{\mathbf{H}}$) *For two time series distributions ρ_1 and ρ_2 on the space $(\mathcal{X}^{\mathbb{N}},\mathcal{F})$ and a sequence of sets of functions $\mathbf{H}=(\mathcal{H}_1,\mathcal{H}_2,\dots)$ define the* telescope distance

$$D_{\mathbf{H}}(\rho_1,\rho_2) := \sum_{k=1}^{\infty} w_k \sup_{h\in\mathcal{H}_k} |\mathbf{E}_{\rho_1} h(X_1,\dots,X_k) - \mathbf{E}_{\rho_2} h(Y_1,\dots,Y_k)|, \tag{2}$$

where w_k, $k\in\mathbb{N}$ is a sequence of positive summable real weights (e.g., $w_k=1/k^2$ or $w_k=2^{-k}$).

Lemma 3 *$D_{\mathbf{H}}$ is a metric if and only if $d_{\mathcal{H}_k}$ is a metric for every $k\in\mathbb{N}$.*

Proof The statement follows from the fact that two process distributions are the same if and only if all their finite-dimensional marginals coincide. ■

Definition 4 (empirical telescope distance $\hat{D}$) *For a pair of samples $X_{1..n}$ and $Y_{1..m}$ define the* empirical telescope distance *as*

$$\hat{D}_{\mathbf{H}}(X_{1..n},Y_{1..m}) := \sum_{k=1}^{\min\{m,n\}} w_k \sup_{h\in\mathcal{H}_k} \left| \frac{1}{n-k+1}\sum_{i=1}^{n-k+1} h(X_{i..i+k-1}) - \frac{1}{m-k+1}\sum_{i=1}^{m-k+1} h(Y_{i..i+k-1}) \right|. \tag{3}$$

All the methods presented in this work are based on the empirical telescope distance. The key fact is that it is an asymptotically consistent estimate of the telescope distance, that is, the latter can be consistently estimated based on sampling.

Theorem 5 *Let $\mathbf{H} = (\mathcal{H}_k)_{k\in\mathbb{N}}$ be a sequence of separable sets $\mathcal{H}_k$ of indicator functions (over $\mathcal{X}^k$) of finite VC dimension such that $\mathcal{H}_k$ generates $\mathcal{F}_k$. Then for every stationary ergodic time series distributions ρ_X and ρ_Y generating samples $X_{1..n}$ and $Y_{1..m}$ we have*

$$\lim_{n,m\to\infty} \hat{D}_{\mathbf{H}}(X_{1..n}, Y_{1..m}) = D_{\mathbf{H}}(\rho_X, \rho_Y) a.s.$$

Note that $\hat{D}_{\mathbf{H}}$ is a biased estimate of $D_{\mathbf{H}}$, and, unlike in the i.i.d. case, the bias may depend on the distributions; however, the bias is $o(n)$.

Remark. The condition that the sets $\mathcal{H}_k$ are sets of indicator function of finite VC dimension comes from the results of Adams and Nobel (2012), who show that for any stationary ergodic distribution ρ, under these conditions, $\sup_{h\in\mathcal{H}_k} \frac{1}{n-k+1}\sum_{i=1}^{n-k+1} h(X_{i..i+k-1})$ is an asymptotically consistent estimate of $\sup_{h\in\mathcal{H}_k} \mathbf{E}_\rho h(X_1,\dots,X_k)$. This fact implies that $d_{\mathcal{H}_k}$ can be consistently estimated, from which the theorem is derived.

Proof [of Theorem 5] As established by Adams and Nobel (2012), under the conditions of the theorem we have

$$\lim_{n\to\infty} \sup_{h\in\mathcal{H}_k} \frac{1}{n-k+1} \sum_{i=1}^{n-k+1} h(X_{i..i+k-1}) = \sup_{h\in\mathcal{H}_k} \mathbf{E}_{\rho_X} h(X_1,\dots,X_k) \ \rho_X\text{-a.s.} \tag{4}$$

for all $k\in\mathbb{N}$, and likewise for ρ_Y. Fix an $\varepsilon > 0$. We can find a $T\in\mathbb{N}$ such that

$$\sum_{k>T} w_k \le \varepsilon. \tag{5}$$

Note that T depends only on ε. Moreover, as follows from (4), for each $k = 1..T$ we can find an N_k such that

$$\left| \sup_{h\in\mathcal{H}_k} \frac{1}{n-k+1} \sum_{i=1}^{n-k+1} h(X_{i..i+k-1}) - \sup_{h\in\mathcal{H}_k} \mathbf{E}_{\rho_X} h(X_{1..k}) \right| \le \varepsilon/T. \tag{6}$$

Let $N_k := \max_{i=1..T} N_i$ and define analogously M for ρ_Y. Thus, for $n\ge N$, $m\ge M$ we have

$$\begin{aligned}
&\hat{D}_{\mathbf{H}}(X_{1..n}, Y_{1..m}) \\
&\quad\le \sum_{k=1}^{T} w_k \sup_{h\in\mathcal{H}_k} \left| \frac{1}{n-k+1} \sum_{i=1}^{n-k+1} h(X_{i..i+k-1}) - \frac{1}{m-k+1} \sum_{i=1}^{m-k+1} h(Y_{i..i+k-1}) \right| + \varepsilon \\
&\qquad\le \sum_{k=1}^{T} w_k \sup_{h\in\mathcal{H}_k} \left\{ \left| \frac{1}{n-k+1} \sum_{i=1}^{n-k+1} h(X_{i..i+k-1}) - \mathbf{E}_{\rho_1} h(X_{1..k}) \right| \right. \\
&\qquad\qquad + |\mathbf{E}_{\rho_1} h(X_{1..k}) - \mathbf{E}_{\rho_2} h(Y_{1..k})| \\
&\qquad\qquad \left. + \left| \mathbf{E}_{\rho_2} h(Y_{1..k}) - \frac{1}{m-k+1} \sum_{i=1}^{m-k+1} h(Y_{i..i+k-1}) \right| \right\} + \varepsilon \\
&\qquad\qquad\qquad\qquad \le 3\varepsilon + D_{\mathbf{H}}(\rho_X, \rho_Y),
\end{aligned}$$

where the first inequality follows from the definition (3) of $\hat{D}_{\mathbf{H}}$ and from (5), and the last inequality follows from (6). Since ε was chosen arbitrary the statement follows. ■

4. Calculating $\hat{D}_{\mathbf{H}}$ Using Binary-Classification Methods

The methods for solving various statistical problems that we suggest are all based on $\hat{D}_{\mathbf{H}}$. The main appeal of this approach is that $\hat{D}_{\mathbf{H}}$ can be calculated using binary classification methods. Here we explain how to do it.

The definition (3) of $D_{\mathbf{H}}$ involves calculating l summands (where $l := \min\{n,m\}$), that is

$$\sup_{h\in\mathcal{H}_k}\left|\frac{1}{n-k+1}\sum_{i=1}^{n-k+1}h(X_{i..i+k-1})-\frac{1}{m-k+1}\sum_{i=1}^{m-k+1}h(Y_{i..i+k-1})\right| \tag{7}$$

for each $k = 1..l$. Assuming that $h \in \mathcal{H}_k$ are indicator functions, calculating each of the summands amounts to solving the following k-dimensional binary classification problem. Consider $X_{i..i+k-1}$, $i = 1..n-k+1$ as class-1 examples and $Y_{i..i+k-1}$, $i = 1..m-k+1$ as class-0 examples. The supremum (7) is attained on $h \in \mathcal{H}_k$ that minimizes the empirical risk, with examples weighted with respect to the sample size. Indeed, we can define the weighted empirical risk of any $h \in \mathcal{H}_k$ as

$$\frac{1}{n-k+1}\sum_{i=1}^{n-k+1}(1-h(X_{i..i+k-1}))+\frac{1}{m-k+1}\sum_{i=1}^{m-k+1}h(Y_{i..i+k-1}), \tag{8}$$

minimising which can be easily seen to be equivalent to (7).

Thus, as long as we have a way to find $h \in \mathcal{H}_k$ that minimizes empirical risk, we have a consistent estimate of $D_{\mathcal{H}}(\rho_X,\rho_Y)$, under the mild conditions on $\mathbf{H}$ required by Theorem 5. Since the dimension of the resulting classification problems grows with the length of the sequences, one should prefer methods that work in high dimensions, such as soft-margin SVMs (Cortes and Vapnik, 1995).

A particularly remarkable feature is that *the choice of $\mathcal{H}_k$ is much easier* for the problems that we consider *than in the binary classification* problem. Specifically, if (for some fixed k) the classifier that achieves the minimal (Bayes) error for the classification problem is not in $\mathcal{H}_k$, then obviously the error of an empirical risk minimizer will not tend to zero, no matter how much data we have. In contrast, all we need to achieve asymptotically 0 error in estimating $\hat{D}$ (and therefore, in the learning problems considered below) is that the sets $\mathcal{H}_k$ generate $\mathcal{F}_k$ and have a finite VC dimension (for each k). This is the case already for the set of half-spaces in $\mathbb{R}_k$. In other words, the *approximation* error of the binary classification method (the classification error of the best f in $\mathcal{H}_k$) is not important. What is important is the estimation error; for asymptotic consistency results it has to go to 0 (hence the requirement on the VC dimension); for non-asymptotic results, it will appear in the error bounds, see Section 7. Thus, we have the following statement.

Claim 1 *The error $|D_{\mathbf{H}}(\rho_X,\rho_Y)-\hat{D}_{\mathbf{H}}(X,Y)|$, and thus the error of the algorithms below, can be much smaller than the error of classification algorithms used to calculate $D_{\mathbf{H}}(X,Y)$.*

We can conclude that, beyond the requirement that $\mathcal{H}_k$ generate $\mathcal{F}_k$ for each $k \in \mathbb{N}$, the choice of H_k (or, say, of the kernel to use in SVM) is entirely up to the needs and constraints of specific applications.

Remark (number of summands in $\hat{D}_{\mathbf{H}}$) Finally, we note that while in the definition of the empirical distributional distance (3) the number of summands is l (the length of the shorter of the two

samples), it can be replaced with any γ_l such that $\gamma_l \to \infty$, without affecting any asymptotic consistency results. In other words, Theorem 5, as well as all the consistency statements below, holds true for l replaced with any non-decreasing function γ_l that tends to infinity with l. A practically viable choice is $\gamma_l = \log l$; in fact, there is no reason to choose faster growing γ_n since the estimates for higher-order summands will not have enough data to converge. This is also the value we use in the experiments.

Remark (relation to total variation) An illustrative example[1] of the choice of the sets $\mathcal{H}_k$ is the set of indicators of all measurable subsets of $\mathcal{X}^k$. In this case each summand in (2) is the total variation distance between the k-dimensional marginal distributions of ρ_1 and ρ_2. Take, for simplicity, $k = 1$; denoting P and Q the corresponding single-dimensional marginals, the distance becomes $\sup_A |P(A) - Q(A)|$ (cf. (1)). This supremum is reached on the set $A^* := \{x \in \mathcal{X} : f(x) \geq g(x)\}$, where f and g are densities of P and Q with respect to some arbitrary measure that dominates both P and Q (e.g., $1/2(P+Q)$). A binary classifier corresponding to a set A declares P if $x \in A$ and Q otherwise. The optimal classification error is $\inf_A(1 - P(A) + Q(A)) = 1 - \sup_A(P(A) + Q(A)) = 1 - P(A^*) + Q(A^*)$ (cf. (8)). In general, estimating the total variation distance (and finding the best classifier) is not possible, so using smaller sets $\mathcal{H}_k$ can be viewed as a regularization of this problem.

5. The Three-Sample Problem

We start with a conceptually simple problem known in statistics as the three-sample problem (sometimes also called time-series classification). We are given three samples $X = (X_1, \ldots, X_n)$, $Y = (Y_1, \ldots, Y_m)$ and $Z = (Z_1, \ldots, Z_l)$. It is known that X and Y were generated by different time-series distributions, whereas Z was generated by the same distribution as either X or Y. It is required to find out which one is the case. Both distributions are assumed to be stationary ergodic, but no further assumptions are made about them (no independence, mixing or memory assumptions). The three sample-problem for dependent time series has been addressed by Gutman (1989) for Markov processes and by Ryabko and Ryabko (2010) for stationary ergodic time series. The latter work uses an approach based on the distributional distance.

Indeed, to solve this problem it suffices to have consistent estimates of some distance between time series distributions. Thus, we can use the telescope distance. The following statement is a simple corollary of Theorem 5.

Theorem 6 *Let the samples $X = (X_1, \ldots, X_n)$, $Y = (Y_1, \ldots, Y_m)$ and $Z = (Z_1, \ldots, Z_l)$ be generated by stationary ergodic distributions ρ_X, ρ_Y and ρ_Z, with $\rho_X \neq \rho_Y$ and either (i) $\rho_Z = \rho_X$ or (ii) $\rho_Z = \rho_Y$. Let the sets $\mathcal{H}_k$, $k \in \mathbb{N}$ be separable sets of indicator functions over $\mathcal{X}^k$. Assume that each set $\mathcal{H}_k$, $k \in \mathbb{N}$ has a finite VC dimension and generates $\mathcal{F}_k$. A test that declares that (i) is true if $\hat{D}_{\mathbf{H}}(Z,X) \leq \hat{D}_{\mathbf{H}}(Z,Y)$ and that (ii) is true otherwise, makes only finitely many errors with probability 1 as $n, m, l \to \infty$.*

It is straightforward to extend this theorem to more than two classes; in other words, instead of X and Y one can have an arbitrary number of samples from different stationary ergodic distributions. A further generalization of this problem is the problem of time-series clustering, considered in the next section.

1. This example was suggested by an anonymous reviewer.

6. Clustering Time Series

We are given N time-series samples $X^1 = (X_1^1, \dots, X_{n_1}^1), \dots, X^N = (X_1^N, \dots, X_{n_N}^N)$, and it is required to cluster them into K groups, where, in different settings, K may be either known or unknown. While there may be many different approaches to define what should be considered a good clustering, and, thus, what it means to have a consistent clustering algorithm, for the problem of clustering time-series samples there is a natural choice, proposed by Ryabko (2010a): Assume that each of the time-series samples $X^1 = (X_1^1, \dots, X_{n_1}^1), \dots, X^N = (X_1^N, \dots, X_{n_N}^N)$ was generated by one out of K different time-series distributions $\rho_1, \dots, \rho_K$. These distributions are unknown. The *target clustering* is defined according to whether the samples were generated by the same or different distributions: the samples belong to the same cluster if and only if they were generated by the same distribution. A clustering algorithm is called *asymptotically consistent* if with probability 1 from some n on it outputs the target clustering, where n is the length of the shortest sample $n := \min_{i=1..N} n_i \geq n'$.

Again, to solve this problem it is enough to have a metric between time-series distributions that can be consistently estimated. Our approach here is based on the telescope distance, and thus we use $\hat{D}$.

The clustering problem is relatively simple if the target clustering has what is called the *strict separation property* (Balcan et al., 2008): every two points in the same target cluster are closer to each other than to any point from a different target cluster. The following statement is an easy corollary of Theorem 5.

Theorem 7 *Let the sets $\mathcal{H}_k$, $k \in \mathbb{N}$ be separable sets of indicator functions over $\mathcal{X}^k$. Assume that each set $\mathcal{H}_k$, $k \in \mathbb{N}$ has a finite VC dimension and generates $\mathcal{F}_k$. If the distributions $\rho_1, \dots, \rho_K$ generating the samples $X^1 = (X_1^1, \dots, X_{n_1}^1), \dots, X^N = (X_1^N, \dots, X_{n_N}^N)$ are stationary ergodic, then with probability 1 from some $n := \min_{i=1..N} n_i$ on the target clustering has the strict separation property with respect to $\hat{D}_{\mathbf{H}}$.*

With the strict separation property at hand, if the number of clusters K is known, it is easy to find asymptotically consistent algorithms. Here we give some simple examples, but the theorem below can be extended to many other distance-based clustering algorithms.

The *average linkage* algorithm works as follows. The distance between clusters is defined as the average distance between points in these clusters. First, put each point into a separate cluster. Then, merge the two closest clusters; repeat the last step until the total number of clusters is K. The *farthest point* clustering works as follows. Assign $c_1 := X^1$ to the first cluster. For $i = 2..K$, find the point X^j, $j \in \{1..N\}$ that maximizes the distance $\min_{t=1..i} \hat{D}_{\mathbf{H}}(X^j, c_t)$ (to the points already assigned to clusters) and assign $c_i := X^j$ to the cluster i. Then assign each of the remaining points to the nearest cluster. The following statement is a corollary of Theorem 7.

Theorem 8 *Under the conditions of Theorem 7, average linkage and farthest point clusterings are asymptotically consistent, provided the correct number of clusters K is given to the algorithm.*

Note that we do not require the samples to be independent; the joint distributions of the samples may be completely arbitrary, as long as the marginal distribution of each sample is stationary ergodic. These results can be extended to the online setting in the spirit of Khaleghi et al. (2012).

For the case of unknown number of clusters, the situation is different: one has to make stronger assumptions on the distributions generating the samples, since there is no algorithm that is consistent for all stationary ergodic distributions (Ryabko, 2010b); such stronger assumptions are considered in the next section.

7. Speed of Convergence

The results established so far are asymptotic out of necessity: they are established under the assumption that the distributions involved are stationary ergodic, which is too general to allow for any meaningful finite-time performance guarantees. While it is interesting to be able to establish consistency results under such general assumptions, it is also interesting to see what results can be obtained under stronger assumptions. Moreover, since it is usually not known in advance whether the data at hand satisfies given assumptions or not, it appears important to have methods that have *both* asymptotic consistency in the general setting and finite-time performance guarantees under stronger assumptions. It turns out that this is possible: for the methods based on $\hat{D}$ one can establish both the asymptotic performance guarantees for all stationary ergodic distributions and finite-sample performance guarantees under stronger assumptions, namely the uniform mixing conditions introduced below.

Another reason to consider stronger assumptions on the distributions generating the data is that some statistical problems, such as homogeneity testing or clustering when the number of clusters is unknown, are provably impossible to solve under the only assumption of stationary ergodic distributions, as shown by Ryabko (2010b).

Thus, in this section we analyse the speed of convergence of $\hat{D}$ under certain mixing conditions, and use it to construct solutions for the problems of homogeneity and clustering with an unknown number of clusters, as well as to establish finite-time performance guarantees for the methods presented in the previous sections.

A stationary distribution on the space of one-way infinite sequences $(\mathcal{X}^{\mathbb{N}}, \mathcal{F})$ can be uniquely extended to a stationary distribution on the space of two-way infinite sequences $(\mathcal{X}^{\mathbb{Z}}, \mathcal{F}_{\mathbb{Z}})$ of the form $\ldots, X_{-1}, X_0, X_1, \ldots$.

Definition 9 (β-mixing coefficients) *For a process distribution* ρ *define the mixing coefficients*

$$\beta(\rho, k) := \sup_{\substack{A \in \sigma(X_{-\infty..0}),\\ B \in \sigma(X_{k..\infty})}} |\rho(A \cap B) - \rho(A)\rho(B)|$$

where $\sigma(..)$ *denotes the sigma-algebra of the random variables in brackets.*

When $\beta(\rho, k) \to 0$ the process ρ is called uniformly β-mixing (with coefficients $\beta(\rho, k)$); this condition is much stronger than ergodicity, but is much weaker than the i.i.d. assumption. For more information on mixing see, for example, Bosq (1996).

7.1 Speed of Convergence of $\hat{D}$

Assume that a sample $X_{1..n}$ is generated by a distribution ρ that is uniformly β-mixing with coefficients $\beta(\rho, k)$. Assume further that $\mathcal{H}_k$ is a set of indicator functions with a finite VC dimension d_k, for each $k \in \mathbb{N}$.

Since in this section we are after finite-time bounds, we fix a concrete choice of the weights w_k in the definition of $\hat{D}$ (Definition 2),

$$w_k := 2^{-k}. \tag{9}$$

The general tool that we use to obtain performance guarantees in this section is the following bound that can be obtained from the results of Karandikar and Vidyasagar (2002).

$$q_n(\rho, \mathcal{H}_k, \varepsilon) := \rho\left(\sup_{h\in\mathcal{H}_k}\left|\frac{1}{n-k+1}\sum_{i=1}^{n-k+1} h(X_{i..i+k-1}) - \mathbf{E}_\rho h(X_{1..k})\right| > \varepsilon\right)$$
$$\leq n\beta(\rho, t_n - k) + 8t_n^{d_k+1} e^{-l_n\varepsilon^2/8}, \quad (10)$$

where t_n are any integers in $1..n$ and $l_n = n/t_n$. The parameters t_n should be set according to the values of β in order to optimize the bound.

One can use similar bounds for classes of finite Pollard dimension (Pollard, 1984) or more general bounds expressed in terms of covering numbers, such as those given by Karandikar and Vidyasagar (2002). Here we consider classes of finite VC dimension only for the ease of the exposition and for the sake of continuity with the previous section (where it was necessary).

Furthermore, for the rest of this section we assume geometric β-mixing distributions, that is, $\beta(\rho, t) \leq \gamma^t$ for some $\gamma < 1$. Letting $l_n = t_n = \sqrt{n}$ the bound (10) becomes

$$q_n(\rho, \mathcal{H}_k, \varepsilon) \leq n\gamma^{\sqrt{n}-k} + 8n^{(d_k+1)/2} e^{-\sqrt{n}\varepsilon^2/8}. \quad (11)$$

Lemma 10 *Let two samples $X_{1..n}$ and $Y_{1..m}$ be generated by stationary distributions ρ_X and ρ_Y whose β-mixing coefficients satisfy $\beta(\rho_\cdot, t) \leq \gamma^t$ for some $\gamma < 1$. Let $\mathcal{H}_k$, $k \in \mathbb{N}$ be some sets of indicator functions on $\mathcal{X}^k$ whose VC dimension d_k is finite and non-decreasing with k. Then*

$$P(|\hat{D}_{\mathbf{H}}(X_{1..n}, Y_{1..m}) - D_{\mathbf{H}}(\rho_X, \rho_Y)| > \varepsilon) \leq 2\Delta(\varepsilon/4, n') \quad (12)$$

where $n' := \min\{n, m\}$, the probability is with respect to $\rho_X \times \rho_Y$ and

$$\Delta(\varepsilon, n) := -\log\varepsilon(n\gamma^{\sqrt{n}+\log(\varepsilon)} + 8n^{(d_{-\log\varepsilon}+1)/2} e^{-\sqrt{n}\varepsilon^2/8}). \quad (13)$$

Proof From (9) we have $\sum_{k=-\log\varepsilon/2}^{\infty} w_k < \varepsilon/2$. Using this and the definitions 2 and 4 of $D_{\mathbf{H}}$ and $\hat{D}_{\mathbf{H}}$ we obtain

$$P(|\hat{D}_{\mathbf{H}}(X_{1..n_1}, Y_{1..n_2}) - D_{\mathbf{H}}(\rho_X, \rho_Y)| > \varepsilon) \leq \sum_{k=1}^{-\log(\varepsilon/2)} (q_n(\rho_X, \mathcal{H}_k, \varepsilon/4) + q_n(\rho_Y, \mathcal{H}_k, \varepsilon/4)),$$

which, together with (11), implies the statement. ■

7.2 Homogeneity Testing

Given two samples $X_{1..n}$ and $Y_{1..m}$ generated by distributions ρ_X and ρ_Y respectively, the problem of homogeneity testing (or the two-sample problem) consists in deciding whether $\rho_X = \rho_Y$. A test is called (asymptotically) consistent if its probability of error goes to zero as $n' := \min\{m, n\}$ goes to infinity. As mentioned above, in general, for stationary ergodic time series distributions there is no asymptotically consistent test for homogeneity (Ryabko, 2010b) (even for binary-valued time series); thus, stronger assumptions are in order.

Homogeneity testing is one of the classical problems of mathematical statistics, and one of the most studied ones. Vast literature exits on homogeneity testing for i.i.d. data, and for dependent

processes as well. We do not attempt to survey this literature here. Our contribution to this line of research is to show that this problem can be reduced (via the telescope distance) to binary classification, in the case of strongly dependent processes satisfying some mixing conditions.

It is easy to see that under the mixing conditions of Lemma 10 a consistent test for homogeneity exists, and finite-sample performance guarantees can be obtained. It is enough to find a sequence $\varepsilon_n \to 0$ such that $\Delta(\varepsilon_n, n) \to 0$ (see (13)). Then the test can be constructed as follows: say that the two sequences $X_{1..n}$ and $Y_{1..m}$ were generated by the same distribution if $\hat{D}_{\mathbf{H}}(X_{1..n}, Y_{1..m}) < \varepsilon_{\min\{n,m\}}$; otherwise say that they were generated by different distributions.

Theorem 11 *Under the conditions of Lemma 10 the probability of Type I error (the distributions are the same but the test says they are different) of the described test is upper-bounded by* $2\Delta(\varepsilon/4, n')$. *The probability of Type II error (the distributions are different but the test says they are the same) is upper-bounded by* $2\Delta((\delta-\varepsilon)/4, n')$ *where* $\delta := D_{\mathbf{H}}(\rho_X, \rho_Y)$.

Proof The statement is an immediate consequence of Lemma 10. Indeed, for the Type I error, the two sequences are generated by the same distribution, so the probability of error of the test is given by (12) with $D_{\mathbf{H}}(\rho_X, \rho_Y) = 0$. The probability of Type II error is given by $P(D_{\mathbf{H}}(\rho_X, \rho_Y) - \hat{D}_{\mathbf{H}}(X_{1..n_1}, Y_{1..n_2}) > \delta - \varepsilon)$, which is upper-bounded by $2\Delta((\delta-\varepsilon))/4, n')$ as follows from (12). ■

The optimal choice of ε_n may depend on the speed at which d_k (the VC dimension of $\mathcal{H}_k$) increases; however, for most natural cases (recall that $\mathcal{H}_k$ are also parameters of the algorithm) this growth is polynomial, so the main term to control is $e^{-\sqrt{n}\varepsilon^2/8}$.

For example, if $\mathcal{H}_k$ is the set of halfspaces in $\mathcal{X}^k = \mathbb{R}^k$ then $d_k = k+1$ and one can choose $\varepsilon_n := n^{-1/8}$. The resulting probability of Type I error decreases as $\exp(-n^{1/4})$.

7.3 Clustering with a Known or Unknown Number of Clusters

If the distributions generating the samples satisfy certain mixing conditions, then we can augment Theorems 7 and 8 with finite-sample performance guarantees.

Theorem 12 *Let the distributions* $\rho_1, \dots, \rho_k$ *generating the samples* $X^1 = (X_1^1, \dots, X_{n_1}^1), \dots, X^N = (X_1^N, \dots, X_{n_N}^N)$ *satisfy the conditions of Lemma 10. Let* $n := \min_{i=1..N} n_i$ *and* $\delta := \min_{i,j=1..N, i\neq j} D_{\mathbf{H}}(\rho_i, \rho_j)$. *Then with probability at least* $1 - N(N-1)\Delta(\delta/12, n')$ *the target clustering of the samples has the strict separation property. In this case single linkage and farthest point algorithms output the target clustering.*

Proof Note that a sufficient condition for the strict separation property to hold is that for every pair i, j of samples generated by the same distribution we have $\hat{D}_{\mathbf{H}}(X^i, X^j) \leq \delta/3$, and for every pair i, j of samples generated by different distributions we have $\hat{D}_{\mathbf{H}}(X^i, X^j) \geq 2\delta/3$. Using Lemma 10, the probability of such an even (for each pair) is upper-bounded by $2\Delta(\delta/12, n')$, which, multiplied by the total number $N(N-1)/2$ of pairs gives the statement. The second statement is obvious. ■

As with homogeneity testing, while in the general case of stationary ergodic distributions it is impossible to have a consistent clustering algorithm when the number of clusters k is unknown, the situation changes if the distributions satisfy certain mixing conditions. In this case a consistent clustering algorithm can be obtained as follows. Assign to the same cluster all samples that are at

most ε_n-far from each other, where the threshold ε_n is selected the same way as for homogeneity testing: $\varepsilon_n \to 0$ and $\Delta(\varepsilon_n, n) \to 0$. The optimal choice of this parameter depends on the choice of $\mathcal{H}_k$ through the speed of growth of the VC dimension d_k of these sets.

Theorem 13 *Given N samples generated by k different stationary distributions ρ_i, $i = 1..k$ (unknown k) all satisfying the conditions of Lemma 10, the probability of error (misclustering at least one sample) of the described algorithm is upper-bounded by*

$$N(N-1)\max\{\Delta(\varepsilon/4, n'), \Delta((\delta-\varepsilon)/4, n')\}$$

where $\delta := \min_{i,j=1..k, i\neq j} D_{\mathbf{H}}(\rho_i, \rho_j)$ and $n = \min_{i=1..N} n_i$, with n_i, $i = 1..N$ being lengths of the samples.

Proof The statement follows from Theorem 11. ■

8. Other Metrics for Time-Series Distributions

The previous sections introduce a new metric on the space of time-series distributions, and use its empirical estimates to solve several learning problems. In this section we attempt to put the telescope distance into a more general context, and take a broader look at metrics between time-series distributions.

Introduce the notation μ_k for the k-dimensional marginal distribution of a time-series distribution μ.

8.1 sum Distances

Observe that the telescope distance $D_{\mathbf{H}}$ has the form

$$D(\mu, \nu) = \sum_{k\in\mathbb{N}} w_k d_k(\mu_k, \nu_k), \tag{14}$$

where w_k are summable positive real weights.

It is easy to see that distances of this form can be consistently estimated, as long as d_k can be consistently estimated for each $k \in \mathbb{N}$; this is formalized in the following statement.

Proposition 14 (estimating sum-based distances) *Let $\mathcal{C}$ be a set of distributions over $\mathcal{X}^{\mathbb{N}}$. Let $d_k, k \in \mathbb{N}$ be a series of distances on the spaces of distributions over $\mathcal{X}^k$, such that $d_k(\mu_k, \nu_k) \leq a \in \mathbb{R}$ for all $\mu, \nu \in \mathcal{C}$ and such that there exists a series $\hat{d}_k(X_{1..n}, Y_{1..n}), k \in \mathbb{N}$ of their consistent estimates: for each $\mu, \nu \in \mathcal{C}$ we have $\lim_{n\to\infty} \hat{d}_k(X_{1..n}, Y_{1..n}) = d_k(\mu_k, \nu_k)$ a.s., whenever $\mu, \nu \in \mathcal{C}$ are chosen to generate the sequences. Then the distance D given by (14) can be consistently estimated using the estimate $\sum_{k\in\mathbb{N}} w_k \hat{d}_k(X_{1..n}, Y_{1..n})$.*

Proof The proof is an easy generalization of the proof of Theorem 5, with the condition on $\hat{d}_k$ used instead of (4). ■

Clearly, $D_{\mathbf{H}}$ is an example of a distance in the form (14), and it satisfies the conditions of the proposition with $\mathcal{C}$ being the set of all stationary ergodic processes.

Another example of a distance in the form (14) is given by the so-called distributional distance (Gray, 1988; Shields, 1996), whose definition is given below. Empirical estimates of this distance are asymptotically consistent for stationary ergodic time series, and thus can be used (Ryabko and Ryabko, 2010; Ryabko, 2010a; Khaleghi et al., 2012; Khaleghi and Ryabko, 2012; Ryabko, 2012) to solve various statistical problems, including those considered above.

To define the distributional distance, let, for each $k,l \in \mathbb{N}$, the set $B^{k,l}$ be some partition of the set $\mathcal{X}^k$, such that the set $B^k = \cup_{l\in\mathbb{N}} B^{k,l}$ generates $\mathcal{F}_k$. Let also $\mathcal{B} = \cup_{k=1}^{\infty} B^k$. Note that the set $\{B \times \mathcal{X}^{\mathbb{N}} : B \in B^{k,l}, k,l \in \mathbb{N}\}$ generates $\mathcal{F}$.

Definition 15 (distributional distance) *The distributional distance is defined for a pair of processes ρ_1, ρ_2 as follows*

$$D_{dd}(\rho_1,\rho_2) := \sum_{m,l=1}^{\infty} w_m w_l \sum_{B\in B^{m,l}} |\rho_1(B) - \rho_2(B)|, \tag{15}$$

where $w_k, k \in \mathbb{N}$ is a summable sequence of positive real weights (e.g., $w_j = 2^{-j}$).

Remark. A more general definition, which is not specific to time-series distributions, is to take any sequence $B_j \in \mathcal{F}_1$, $j \in \mathbb{N}$ of events that generate the sigma-algebra $\mathcal{F}$ of a probability space $(\mathcal{X}, \mathcal{F})$, and then define

$$D'_{dd}(\rho_1,\rho_2) := \sum_{j=1}^{\infty} w_j |\rho_1(B_j) - \rho_2(B_j)|; \tag{16}$$

see Gray (1988) for a general treatment. The latter definition is sometimes more convenient for theoretical analysis (Ryabko, 2012), while the distance (15), which makes explicit the marginal distributions on $\mathcal{X}^m$, $m \in \mathbb{N}$ and the level l of discretisation $B^{m,l}$ of each set $\mathcal{X}^m$, is more suited for time-series, and, specifically, for implementing algorithms, see Ryabko and Ryabko (2010), Khaleghi et al. (2012) and Khaleghi and Ryabko (2012).

In general, it is perhaps impossible to tell which distance, specifically, $D_{\mathbf{H}}$ or D_{dd}, should be preferred for which problem. Conceptually, one of the advantages of the telescope distance $D_{\mathbf{H}}$ is that one can use different sets $\mathbf{H}$—the choice that makes it adaptable to applications. Another is that one can reuse readily available classification methods for calculating its empirical estimates. One formal way to compare different metrics is to compare the resulting topologies. This is done in the end of this section.

8.2 sup Distances

A different way to construct a distance between time-series distributions based on their finite-dimensional marginals is to use the supremum instead of summation in (14):

$$d(\mu,\nu) = \sup_{k\in\mathbb{N}} d_k(\mu_k,\nu_k). \tag{17}$$

Some commonly used metrics are defined in the form (17) or have natural interpretations in this form, as the following two examples show.

Definition 16 (total variation) *For time-series distributions ν, μ the total variation distance between them is defined as $D_{tv}(\mu,\nu) := \sup_{A\in\mathcal{F}} |\mu(A) - \nu(A)|$.*

It is easy to see that $D_{tv}(\mu,\nu) = \sup_{k\in\mathbb{N}} \sup_{A\in\mathcal{F}_k} |\mu(A)-\nu(A)|$, so that the total variation distance has the form (17).

However, the total variation distance is not very useful for time-series distributions for the following two reasons. First of all, for stationary ergodic distributions it is degenerate: $D_{tv}(\mu,\nu) = 1$ if and only if $\mu \neq \nu$. This follows from the fact that any two different stationary ergodic distributions are singular. Such a distance could still be useful as a formalization of the problem of homogeneity testing. However, the problem of homogeneity testing is impossible to solve based on sampling for stationary ergodic distributions (and even for a smaller family of B processes, see below) (Ryabko, 2010b), so the use of this distance remains limited to more restrictive classes of distributions.

This hints at an intrinsic problem with distances defined in the form (17). The problem is in the difficulties to estimate such metrics based on sampling. At each time step t we observe only a sample of finite length, say n_t, and based on this we want to estimate a quantity that involves k-dimensional marginals for all k, including those with $k > n_t$. Considering a growing (with t) number of marginals for the estimate may be a route to take, but this turns out to be difficult to analyse, especially if no rates of convergence can be established for the set of time-series distributions at hand. This problem is highlighted by the example of the so-called $\bar{d}$ distance, whose definition follows.

Definition 17 ($\bar{d}$ distance) *Assume some distance δ over $\mathcal{X}$ is given. For two time-series distributions μ and ν define*

$$\bar{d}(\mu,\nu) := \sup_{k\in\mathbb{N}} \frac{1}{k} \inf_{p\in P} \sum_{i=1}^{k} \mathbf{E}_p \delta(x_i, y_i),$$

where P is the set of all distributions over $\mathcal{X}^k \times \mathcal{X}^k$ generating a pair of sequences $x_{1..k}, y_{1..k}$ whose marginal distributions are μ_k and ν_k correspondingly.

A process is called a *B-process* (or a Bernoulli process) if it is in the $\bar{d}$-closure of the set of all aperiodic stationary ergodic k-step Markov processes, where $k \in \mathbb{N}$. For more information on $\bar{d}$-distance and B-processes see Gray (1988) and Shields (1996). The set of B-processes is a strict subset of the set of all stationary ergodic time-series distributions. It turns out that $\bar{d}$ distance is impossible to estimate for the latter, while it can be estimated for the former (Ornstein and Weiss, 1990).

Theorem 18 (Ornstein and Weiss, 1990) *There exists an estimator $\hat{d}(X_{1..n}, Y_{1..n})$ such that, if $X_{1..n}, Y_{1..n}$ are generated by B-processes μ and ν then $\hat{d}(X_{1..n}, Y_{1..n}) \to \bar{d}(\mu,\nu)$ a.s. However, for any estimator $\hat{d}(X_{1..n}, Y_{1..n})$ there is a pair of stationary ergodic processes μ and ν such that $\limsup_{n\to\infty} |\hat{d}(X_{1..n}, Y_{1..n}) - \bar{d}(\mu,\nu)| > 1/2$.*

8.3 Comparison with the Distributional Distance

In this section we show that the telescope distance is stronger than the distributional distance in the topological sense. Since in fact both the telescope distance and the distributional distance are families of distances (the telescope distance depends on the sequence **H**), we will fix a simple natural choice of each of these metrics. In general, different choices of parameters produce topologically non-equivalent metrics; it is easy to check that the analysis in this section extends to many other natural choices.

Thus, for the purpose of this section, let us fix $\mathcal{X} = \mathbb{R}$ and let H_k^0 be the set of halfspaces in $\mathcal{X}^k$. Denote $\mathbf{H}^0 := (\mathcal{H}_k^0 : k \in \mathbb{N})$. Clearly, these $\mathcal{H}_k$ satisfy all the conditions of the theorems of Sections 5 and 6.

For the distributional distance (Definition 15), set $B^{k,l}$ to be the partition of the set $\mathcal{X}^k$ into k-dimensional cubes with volume $h_l^k = (1/l)^k$. Denote D_{dd}^0 the distributional distance D_{dd} with this set of parameters.

Definition 19 *A metric d_1 is said to be* stronger *than a metric d_2 if any sequence that converges in d_1 also converges in d_2. If, in addition, d_2 is not stronger than d_1, then d_1 is called* strictly stronger.

Note that for the distributional distance, if we use the same sets B_k to generate the sigma algebras $\mathcal{X}^k$ then the distance defined by (15) is stronger than the distance defined by (16).

Theorem 20 *$D_{\mathbf{H}^0}$ is strictly stronger than D_{dd}^0.*

Proof Fix any $\varepsilon > 0$ and find a $T \in \mathbb{N}$ such that $\sum_{m,l>T} w_m w_l < \varepsilon$. Let ρ_i, $i \in \mathbb{N}$ be a sequence of process measures that converges in $D_{\mathbf{H}^0}$. Let A^k be the set of all complements to $\mathcal{X}^k$ of cubes with sides of length s, for all $s \in \mathbb{N}$. Note that any cube B in B_k, as well as any set A in A^k, can be obtained by intersecting $2k$ halfspaces. Therefore, we have

$$\sup_{B \in B^k \cup A^k} |\rho_i(B) - \rho_j(B)| \le 2kd_{\mathcal{H}_k}(\rho_i, \rho_j) \le 2kw_k^{-1} D_{\mathbf{H}^0}(\rho_i, \rho_j), \tag{18}$$

where the second inequality follows from the definition of $D_{\mathbf{H}^0}$. Observe that for each $i \in \mathbb{N}$ one can find a set $A_i \in A^k$ such that $\rho_i(A_i) < \varepsilon/2$. From this, (18) and the fact that the sequence ρ_i converges in $D_{\mathbf{H}^0}$, we conclude that there is a set $A \in A^k$ such that

$$\rho_i(A) < \varepsilon$$

for all $i \ge j_k$. For all $k, l \in \mathbb{N}$ one can find $M_{k,l} \in \mathbb{N}$ such that the complement of A (which is a cube in $\mathcal{X}^k$) is contained in the union of $M_{k,l}$ cubes from $B_{k,l}$. Let $M := \max_{k,l \le T} M_{k,l}$ and $J := \max_{i \le T} j_i$. Using (18) and the definition of the partitions $B^{k,l}$ we can derive

$$\sum_{B \in B^{k,l}, B \not\subseteq A_k} |\rho_i(B) - \rho_j(B)| \le 2MTw_T^{-1} D_{\mathbf{H}^0}(\rho_i, \rho_j)$$

for any $i, j \ge J$ and all $k, l \le T$. Increasing J if necessary to have $2MTw_T^{-1} D_{\mathbf{H}^0}(\rho_i, \rho_j) < \varepsilon$ for all $i, j \ge J$, we obtain

$$D_{dd}^0(\rho_i, \rho_j) \le \sum_{m,l=1}^{T} w_m w_l \sum_{B \in B^{m,l}, B \not\subseteq A_m} |\rho_i(B) - \rho_j(B)| + 2\varepsilon \le 3\varepsilon$$

for all $i, j > J$, which means that the sequence $\rho_i, i \in \mathbb{N}$ converges in D_{dd}^0. Thus, $D_{\mathbf{H}^0}$ is stronger than D_{dd}^0.

It remains to show that D_{dd}^0 is not stronger than $D_{\mathbf{H}^0}$. To see this, consider the following sequence of subsets of $\mathcal{X} = \mathbb{R}$. f is the dot $\{0\}$, and f_k is the interval $[0, 1/k]$, for each $k \in \mathbb{N}$. Define the distributions ν_j for $j \in \mathbb{N}$ as uniform on f_j, and let ν be concentrated on f; since we need time-series distributions, extend this i.i.d. for all $n \in \mathbb{N}$. It is easy to check that $\lim_{i \in \mathbb{N}} D_{dd}^0(\nu_i, \nu_0) = 0$ while $D_{\mathbf{H}^0}(\nu_i, \nu_0) = 1$ for all $i > 0$. ■

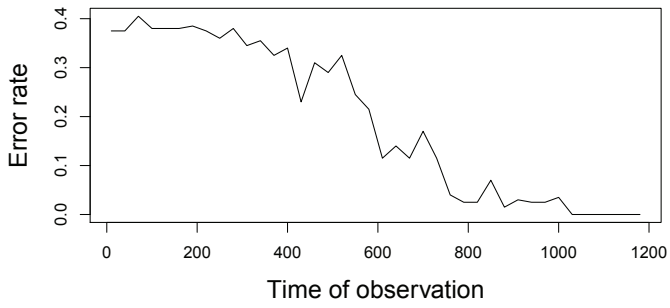


Figure 1: Error of two-class clustering using $\mathrm{TS}_{\mathrm{SVM}}$; 10 time series in each target cluster, averaged over 20 runs.

9. Experimental Evaluation

For experimental evaluation we chose the problem of time-series clustering. The average-linkage clustering is used, with the telescope distance between samples calculated using an SVM, as described in Section 4. In all experiments, SVM is used with radial basis kernel, with default parameters of libsvm (Chang and Lin, 2011). The parameters w_k in the definition of the telescope distance (Definition 2) are set to $w_k := k^{-2}$.

9.1 Synthetic Data

For the artificial setting we chose highly-dependent time-series distributions which have the same single-dimensional marginals and which cannot be well approximated by finite- or countable-state models. Variants of this family of distributions are standard examples in ergodic theory and dynamical systems (see, for example, Billingsley, 1965; Gray, 1988; Shields, 1996). The distributions $\rho(\alpha)$, $\alpha \in (0,1)$, are constructed as follows. Select $r_0 \in [0,1]$ uniformly at random; then, for each $i = 1..n$ obtain r_i by shifting r_{i-1} by α to the right, and removing the integer part. The time series $(X_1, X_2, \dots)$ is then obtained from r_i by drawing a point from a distribution law $\mathcal{N}_1$ if $r_i < 0.5$ and from $\mathcal{N}_2$ otherwise. $\mathcal{N}_1$ is a 3-dimensional Gaussian with mean of 0 and covariance matrix $\mathrm{Id} \times 1/4$. $\mathcal{N}_2$ is the same but with mean 1. If α is irrational[2] then the distribution $\rho(\alpha)$ is stationary ergodic, but does not belong to any simpler natural distribution family; in particular, it is not a *B*-processes (Shields, 1996). The single-dimensional marginal is the same for all values of α. The latter two properties make all parametric and most non-parametric methods inapplicable to this problem.

In our experiments, we use two process distributions $\rho(\alpha_i), i \in \{1,2\}$, with $\alpha_1 = 0.31...$, $\alpha_2 = 0.35...$,. The dependence of error rate on the length of time series is shown on Figure 1. One clustering experiment on sequences of length 1000 takes about 5 min. on a standard laptop.

9.2 Real Data

To demonstrate the applicability of the proposed methods to realistic scenarios, we chose the brain-computer interface data from BCI competition III (Millán, 2004). The data set consists of (pre-processed) BCI recordings of mental imagery: a person is thinking about one of three subjects

2. In the experiments we used a `longdouble` with a long mantissa

	s_1	s_2	s_3
TS_{SVM}	**84%**	**81%**	**61%**
DTW	46%	41%	36%
KCpA	79%	74%	61%
SVM	76%	69%	60%

Table 1: Clustering accuracy in the BCI data set. 3 subjects (columns), 4 methods (rows). Our method is TS_{SVM}.

(left foot, right foot, a random letter). Originally, each time series consisted of several consecutive sequences of different classes, and the problem was supervised: three time series for training and one for testing. We split each of the original time series into classes, and then used our clustering algorithm in a completely unsupervised setting. The original problem is 96-dimensional, but we used only the first 3 dimensions (using all 96 gives worse performance). The typical sequence length is 300. The performance is reported in Table 1, labelled TS_{SVM}. All the computation for this experiment takes approximately 6 minutes on a standard laptop.

The following methods were used for comparison. First, we used dynamic time wrapping (DTW) (Sakoe and Chiba, 1978) which is a popular base-line approach for time-series clustering. The other two methods in Table 1 are from the paper of Harchaoui et al. (2008). The comparison is not fully relevant, since the results of Harchaoui et al. (2008) are for different settings; the method KCpA was used in change-point estimation method (a different but also unsupervised setting), and SVM was used in a supervised setting. The latter is of particular interest since the classification method we used in the telescope distance is also SVM, but our setting is unsupervised (clustering). On this data set the telescope distance demonstrates better performance than the comparison methods, which indicates that it can be useful in real-world scenarios.

10. Outlook

We have proposed a binary-classifier-based metric and shown how it can be used to solve several problems concerning highly dependent time series. The consistency results obtained concern the use of the empirical risk minimizer as a binary classifier. For applications this suggests using classifiers that approximate empirical risk minimizers over target sets of (indicator) functions. It is easy to extend the definition of the metric so that any classifier can be used, including such classifiers as nearest-neighbours rules. However, in order to extend the obtained results to such classifiers, one would need to establish the consistency of the empirical estimates of the resulting metric between time-series distributions, which means extending the results concerning the corresponding classifiers from the i.i.d. samples to stationary ergodic time series. Note that, while consistency of the empirical estimates of the time-series metric used is sufficient for the analysis of the learning problems considered in this work, it is not sufficient for some other learning problems concerning dependent time series that rely on a metric between time-series distributions. For example, some change-point problems for stationary ergodic time series can be solved using the distributional distance (Ryabko and Ryabko, 2010; Khaleghi and Ryabko, 2012, 2013). It remains to see whether the same results can be obtained with the telescope distance and its generalizations.

Acknowledgments

This work is an extended version of the NIPS'12 paper (Ryabko and Mary, 2012). The authors are grateful to the anonymous reviewers for the numerous constructive comments that helped us to improve the paper. This research was partially supported by the French Ministry of Higher Education and Research, Nord-Pas-de-Calais Regional Council and FEDER through CPER 2007-2013, ANR project Lampada (ANR-09-EMER-007) and by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement 231495 (project CompLACS).

References

T. M. Adams and A. B. Nobel. Uniform approximation of Vapnik-Chervonenkis classes. *Bernoulli*, 18(4):1310–1319, 2012.

M.-F. Balcan, N. Bansal, A. Beygelzimer, D. Coppersmith, J. Langford, and G. Sorkin. Robust reductions from ranking to classification. In Nader Bshouty and Claudio Gentile, editors, *Learning Theory*, volume 4539 of *Lecture Notes in Computer Science*, pages 604–619. 2007.

M.-F. Balcan, A. Blum, and S. Vempala. A discriminative framework for clustering via similarity functions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 671–680. ACM, 2008.

P. Billingsley. *Ergodic Theory and Information*. Wiley, New York, 1965.

D. Bosq. *Nonparametric Statistics for Stochastic Processes*. Estimation and Prediction. Springer, 1996.

Ch.-Ch. Chang and Ch.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

R. Fortet and E. Mourier. Convergence de la répartition empirique vers la répartition théoretique. *Ann. Sci. Ec. Norm. Super., III. Ser*, 70(3):267–285, 1953.

R. Gray. *Probability, Random Processes, and Ergodic Properties*. Springer Verlag, 1988.

M. Gutman. Asymptotically optimal classification for multiple tests with empirically observed statistics. *IEEE Transactions on Information Theory*, 35(2):402–408, 1989.

Z. Harchaoui, F. Bach, and E. Moulines. Kernel change-point analysis. In *Advances in Neural Information Processing Systems 21*, pages 609–616, 2008.

L. V. Kantorovich and G. S. Rubinstein. On a function space in certain extremal problems. *Dokl. Akad. Nauk USSR*, 115(6):1058–1061, 1957.

R.L. Karandikar and M. Vidyasagar. Rates of uniform convergence of empirical means with mixing processes. *Statistics and Probability Letters*, 58:297–307, 2002.

A. Khaleghi, D. Ryabko, J. Mary, and P. Preux. Online clustering of processes. In *AISTATS*, JMLR W&CP 22, pages 601–609, 2012.

A. Khaleghi and D. Ryabko. Locating changes in highly dependent data with unknown number of change points. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 3095–3103. 2012.

A. Khaleghi and D. Ryabko. Nonparametric multiple change point estimation in highly dependent time series. In *Proc. 24th International Conf. on Algorithmic Learning Theory (ALT'13)*, Singapre, 2013. Springer.

D. Kifer, Sh. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proc. the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB'04, pages 180–191, 2004.

A.N. Kolmogorov. Sulla determinazione empirica di una legge di distribuzione. *G. Inst. Ital. Attuari*, pages 83–91, 1933.

J. Langford, R. Oliveira, and B. Zadrozny. Predicting conditional quantiles via reduction to classification. In *Proc. of the 22th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2006.

J. del R. Millán. On the need for on-line learning in brain-computer interfaces. In *Proc. of the Int. Joint Conf. on Neural Networks*, 2004.

D.S. Ornstein and B. Weiss. How sampling reveals a process. *Annals of Probability*, 18(3):905–930, 1990.

D. Pollard. *Convergence of Stochastic Processes*. Springer, 1984.

B. Ryabko. Prediction of random sequences and universal coding. *Problems of Information Transmission*, 24:87–96, 1988.

B. Ryabko. Compression-based methods for nonparametric prediction and estimation of some characteristics of time series. *IEEE Transactions on Information Theory*, 55:4309–4315, 2009.

D. Ryabko. Clustering processes. In *Proc. the 27th International Conference on Machine Learning (ICML 2010)*, pages 919–926, Haifa, Israel, 2010a.

D. Ryabko. Discrimination between B-processes is impossible. *Journal of Theoretical Probability*, 23(2):565–575, 2010b.

D. Ryabko. On the relation between realizable and non-realizable cases of the sequence prediction problem. *Journal of Machine Learning Research*, 12:2161–2180, 2011.

D. Ryabko. Testing composite hypotheses about discrete ergodic processes. *Test*, 21(2):317–329, 2012.

D. Ryabko and B. Ryabko. Nonparametric statistical inference for ergodic processes. *IEEE Transactions on Information Theory*, 56(3):1430–1435, 2010.

D. Ryabko and J. Mary. Reducing statistical time-series problems to binary classification. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2069–2077. 2012.

H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.

P. Shields. *The Ergodic Theory of Discrete Sample Paths*. AMS Bookstore, 1996.

R. J. Solomonoff. Complexity-based induction systems: comparisons and convergence theorems. *IEEE Trans. Information Theory*, IT-24:422–432, 1978.

V. M. Zolotarev. Probability metrics. *Theory of Probability and Its Applications.*, 28(2):264–287, 1983.