Online stochastic algorithms

Le Li

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

université angers

THÈSE DE DOCTORAT DE

L'UNIVERSITE D'ANGERS
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *(Mathématiques et leurs interactions)*

Le Ll

Online stochastic algorithms

Thèse présentée et soutenue à Angers, le 27/11/2018
Unité de recherche : LAREMA, UMR CNRS 6093
Thèse N° : 139335

Rapporteurs avant soutenance :

Gilles Stoltz	Directeur de recherche, CNRS - Université Paris Sud
Claire Monteleoni	Associate Professor, University of Colorado Boulder

Composition du Jury :

Président :	Gérard Biau	Professeur, Sorbonne Université
Rapporteur :	Gilles Stoltz	Directeur de recherche, CNRS - Université Paris Sud
Examinateurs :	Bertrand Michel	Professeur, Ecole Centrale de Nantes
	Sébastien Loustau	PDG, LumenAI
Dir. de thèse :	Loïc Chaumont	Professeur, Université d'Angers
Co-dir. de thèse :	Benjamin Guedj	Chargé de recherche, Inria
Invité :	Florent Gosselin	VP, iAdvize

Acknowledgements

Almost four years ago, I moved to Angers and started a journey in the joint area of statistics, online learning and artificial intelligence in LAREMA, University of Angers. As this journey is approaching its end, I feel that a few lines are not enough to thank everyone who has helped me adapting the work and life in the past days.

First of all, I would like to thank sincerely my two supervisors, Sébastien Loustau and Benjamin Guedj who have given me a lot of valuable advises in the past years. The discussions with Sébastien have enlightened me in the progress of my thesis; The suggestions of Benjamin have made me more rigorous towards scientific research. Without their earnest instruction, patience and encouragement, I would never have been able to complete this crucial personal project. In addition, I truly appreciate them for their support and friendship during these years.

In addition, I sincerely thank Gilles Stoltz and Claire Monteleoni for accepting to review this thesis with great patience and kindness. Their precious and insightful comments have helped me a lot in improving this thesis. I also thank Gérard Biau and Bertrand Michel for agreeing to be in my jury, this is a honor.

My thanks also go to Loïc Chaumont, Fabien Panloup and Frédéric Proïa who have not only organized many enlightening seminars but also had interesting talk with me about researches in statistics. In addition, I would thank Olivier Wintenberger and Pierre Alquier for the time they spent in explaining my questions about PAC-Bayesian theory and online learning. I also want to show my thankfulness to my office mates Alexis Roquefeuil, Marine Marolleau, Théo Jamin, Sa Lem Lamine, Clément Fromenteau, Johan Leray, Viet Anh Nguyen, Jérôme Spielmann and Ann du Crest de Villeneuve who spent a great deal of time on discussing work-related topics and on solving my daily-life problems as a foreigner. Additionally, I would like to thank our nice secretaries Alexandra Le Petitcorps and Caroline Chalumeau who have rendered all possible assistance to me to solve my difficulties on administrative issues.

In addition, I want to thank Julien Hervouet and Florent Gosselin who have given me a valuable opportunity to work in iAdvize from which I have learned many working skills. I am especially grateful to Florent Gosselin who is very kind and helped me a lot in adapting to the different rhythm of working in both laboratory and company. I also appreciate the lovely colleagues of my team: Marion Guillard, Emmanuel Verron, Jocelyn Drean, Jean-Baptiste Even, Thomas Pocreau, Yannis Fletberliac, François Klein and Antony Hamon and all other colleagues for their great help and patience in my daily work.

I am particularly grateful to my dear parents Gang LI and Ruigen Wang. Thanks for all their love and selfless support! Without them, this work would not be possible. I would also thank my friends Xun Jiang, Yi Zhou, Jinsong Zeng, Ji Shi, Shuxian LI, Thomas Michel, Yang Qiu, Zequn Wei, Runsheng Gu, Zhi Lu, Jintong Ren, Aiwen Lai, Ming Wang, Lili Zang, Sun Wen, Xuan Zhou, Yang Wu, Rui Zhang, Wei Wang, Shuang

Liu, Dan Yang, Hongyu Ding, Maoyue Tang and all my neighbors in France for their accompanies. I also greatly appreciate my best friends Huang Shen, Wei Lu, Jianwei Pu, Xiongtao Zhang and Kaile Xiao in China who have brought me a lot of happiness and spiritual support.

Contents

Foreword

Ce travail doctoral a été réalisé dans le cadre d'une convention CIFRE de l'ANRT (2014-00757) du 02/12/2014 au 01/06/2018 et a été hébergé par iAdvize, le Laboratoire Angevin de Recherche en Mathématiques et l'équipe-projet Modal du centre de recherche Lille - Nord Europe d'Inria.

Plan du manuscrit

Ce manuscript est composé de quatre chapitres dont chacun peut être lu indépendamment.

Le Chapitre 1 de ce manuscrit vise deux objectifs : introduire le cadre de l'apprentissage en ligne et synthétiser les principaux résultats de cette thèse. Nous introduisons la prédiction avec avis d'experts et présentons le concept de borne de regret, qui est une quantité essentielle dans l'apprentissage en ligne. Nous nous intéressons ensuite à la régression séquentielle ou en ligne pour aboutir à la présentation du clustering en ligne et de l'apprentissage de courbes principales en ligne, qui sont les deux domaines des principales contributions de ce manuscrit. La plupart des résultats obtenus sont basés sur des algorithmes stochastiques en ligne construits à l'aide d'arguments quasi-bayésiens: un panorama des principales méthodes est détaillé. Le Chapitre 1 se referme avec un résumé des principales contributions de la thèse pour les problèmes de clustering en ligne et l'apprentissage séquentiel de courbes principales.

Le Chapitre 2 se concentre sur le problème de clustering en ligne. Nous présentons un nouvel algorithme adaptatif de clustering en ligne s'appuyant sur l'approche quasi-bayésienne avec une estimation dynamique (i.e., dépendant du temps) du nombre de clusters. En reposant sur deux types de lois a priori; nous prouvons que cet algorithme a une borne de regret de l'ordre $\sqrt{T \ln T}$. De plus, nous montrons que cet algorithm est optimal au sens qu'il atteint des bornes de regret minimax. Nous proposons aussi une implémentation par RJMCMC : le paquet R correspondant est appelé PACBO et est disponible en ligne sur le CRAN [1] Enfin, nous comparons notre méthode avec d'autre méthodes classiques de clustering et présentons des expériences numériques qui illustrent le potentiel de notre algorithme. Les résultats contenus dans ce chapitre sont publiés dans Electronic Journal of Statistics, 2018, Vol. 12, No. 2, 3071–3113.

Le Chapitre 3 propose un nouvel algorithme d'apprentissage séquentiel de courbes principales, reposant sur une approche maximum a posteriori (MAP) pour le quasi-posterior de Gibbs. Cet algorithme permet ainsi de résumer l'information contenue dans un nuage de points, en proposant des courbes qui passent au milieu des points. Nous prouvons que le regret de cet algorithme et celui de sa version adaptative est sous-linéaire en l'horizon temporel T, et nous proposons une implémentation par algorithme glouton local qui ne prend qu'une partie des données pour mettre à jour la courbe. Cette implémentation

1. https://cran.rproject.org/web/packages/PACBO/index.html.

est donc adaptée à l'apprentissage en ligne dans lequel la vitesse à exécution peut être prioritaire sur la précision.

Le Chapitre 4 a une visée plus pratique, et présente les travaux effectués chez iAdvize. Il contient deux parties : la première partie se concentre sur l'application de certaines méthodes classiques de traitement automatique des langues (tf-idf) et d'apprentissage, parmi lesquelles la classification naïve bayésienne et les support vector machines pour améliorer les produits déployés par iAdvize. La deuxième partie est liée à la création d'un agent conversationnel (chatbot) en mettant en pratique certaines méthodes de pointe dans l'apprentissage profond, comme le traitement du langage naturel (NLP) et le modèle Seq2Seq.

1

State-of-the-art and contributions

Contents

1.1 Online learning

In the most basic version of online learning, the predictor (or forecaster) gets access one after another to a sequence $z_1, z_2, \ldots$ of elements. At each time $t = 1, 2, \ldots$, before the tth symbol of the sequence is revealed, the forecaster gives his guess of value z_t on the basis of the previous $t-1$ observations and of possibly other available side information.

The first significant difference between online learning and classical batch learning in statistics relies on the assumptions of the sequence of elements. In the latter one, these elements, which are often called observations, are assumed to be independent and identically distributed realizations of a stationary stochastic process. Under this assumption, statistical properties can be estimated from all available observations and prediction can be derived from these estimates. The performance of the prediction is assessed by the risk that is defined as the expected value of a certain loss function measuring the discrepancy between true observation and predicted value. However, in online setting, we do not impose such assumption on the sequence of elements and they can be deterministic, stochastic or even play an adversary role against forecaster's prediction.

Since no probabilistic model is assumed on the sequence, the notion of risk cannot be defined. Instead, we consider a cumulative loss during many rounds of predictions to measure the performance of a forecaster, where the loss at each round is measured by the discrepancy between forecaster's prediction and true value. To get a baseline or an oracle with which the cumulative loss can be compared, we introduce a class of reference forecasters, often called experts in online setting. These experts also make their predictions at each round before the revealing of next outcome. With an additional access to experts' predictions (or advice), the goal of a forecaster is to make predictions such that his cumulative loss is as close as possible to that of the best reference forecaster. The difference between a forecaster's cumulative loss and that of best reference forecaster is called regret, as it measures the extent of forecaster's regret, in hindsight, of not having followed the advice of this expert. Intuitively, given a set of reference forecasters and a loss function, we seek for a forecaster whose regret is bounded by a term that is as small as possible or even negative in the best cases. This term is called regret bound.

The abstract notion of reference forecasters as well as the available side information can have different interpretation, depending on the specific settings that one considers. In what follows, we shall detail the notion of experts and side information in three settings: prediction with expert advice, online regression and online learning in an unsupervised setting. Investigations and methods in the first two settings have shed light on the contribution of this manuscript which is focused on the unsupervised case.

1.1.1 Prediction with expert advice

In the framework of prediction with expert advice, the sequence $z_1, z_2, \ldots$ are supposed to be from an outcome space $\mathcal{Z}$ and a forecaster makes sequentially his prediction $\hat{z}_1, \hat{z}_2, \ldots$ from a decision space $\mathcal{D}$ that is assumed to be a convex subset vector space. This decision space $\mathcal{D}$ may vary from $\mathcal{Z}$. In addition, the notion of experts is a set $\{\mathbf{f}_{i,t} \in \mathcal{D} : i \in \mathcal{E}\}, t = 1, 2, \ldots$ of experts' predictions, where $\mathbf{f}_{i,t}$ is the prediction of expert e at time t and $\mathcal{E}$ is a set of experts. These experts could be some statistical models with different values of parameters, different statistical estimators trained on a different set or some stochastic decisions given by several black boxes. At each time $t = 1, 2, \ldots$, before the next outcome z_t, the forecaster yields prediction $\hat{z}_t$ on the basis of experts' current and past predictions $\{\mathbf{f}_{i,s} \in \mathcal{D} : i \in \mathcal{E}, s \leq t\}$ at t. In other words, $\hat{z}_t$ is a function of $\{\mathbf{f}_{i,s} \in \mathcal{D} : i \in \mathcal{E}, s \leq t\}$. Prediction made by the forecaster (*resp.* experts) is assessed by $\ell(\hat{z}_t, z_t)$ (*resp.* $\ell(\mathbf{f}_{i,t}, z_t)$) where ℓ is a loss function $\ell : \mathcal{D} \times \mathcal{Z} \longrightarrow \mathbb{R}_+$. We now list some examples of ℓ with different $\mathcal{D}$ and $\mathcal{Z}$:

— square loss: $\ell(y, z) = (y - z)^2$ where $\mathcal{D} = \mathcal{Z} = [0, 1]$.
— logarithm loss: $\ell(y, z) = -\mathbb{1}_{\{z=1\}} \ln y - \mathbb{1}_{\{z=0\}} \ln(1 - y)$ where $\mathcal{D} = [0, 1], \mathcal{Z} = \{0, 1\}$.

— linear loss: $\ell(y,z) = \sum_{j=1}^{N} y_j z_j$ where $N \in \mathbb{N}^*$, $\mathscr{D} = \mathscr{X}_N$ is the simplex $\{y \in \mathbb{R}^N, \sum_{j=1}^{N} y_j = 1, y_j \geq 0, j = 1,\dots,N\}$ and $\mathscr{Z} = [0,1]^N$.

The forecaster aims to have a cumulative loss as small as possible with respect to the best expert. In other term, it corresponds to minimizing the difference (*i.e.*, the regret) of a forecaster's cumulative loss and that of the best expert

$$\sum_{t=1}^{T} \ell(\hat{z}_t, z_t) - \min_{i \in \mathscr{E}} \left\{ \sum_{t=1}^{T} \ell\left(\mathbf{f}_{i,t}, z_t\right) \right\}.$$

This quantity can be interpreted as the regret of the forecaster of not following the advice of the best expert. In this setting, one wants to find a forecaster with regret sublinear in T (since linear in T is trivial: *e.g.,* ℓ is bounded by a constant) and that this sublinearity is uniformly over any sequence $z_t \in \mathscr{Z}, t = 1,\dots,T$ and any $\mathbf{f}_{i,t} \in \mathscr{D}, i \in \mathscr{E}, t = 1,\dots,T$, *i.e.,* a forecaster satisfying the following

$$\sup_{\substack{z_t \in \mathscr{Z} \\ \mathbf{f}_{i,t} \in \mathscr{D}, i \in \mathscr{E}, t = 1,\dots,T}} \left\{ \sum_{t=1}^{T} \ell(\hat{z}_t, z_t) - \min_{i \in \mathscr{E}} \left\{ \sum_{t=1}^{T} \ell\left(\mathbf{f}_{i,t}, z_t\right) \right\} \right\} \leq R_T(\mathscr{E}), \tag{1.1}$$

where $R_T(\mathscr{E})$ is called regret bound satisfying $\lim_{T \to \infty} \frac{R_T(\mathscr{E})}{T} = 0$.

If $\mathscr{E}$ is finite with cardinality N, and loss ℓ is bounded and convex in its first argument, a classical and fundamental example of forecaster which satisfies (1.1) is the *Exponentially Weighted Average forecaster* (EWA forecaster) whose prediction is

$$\hat{z}_t = \sum_{i=1}^{N} w_{i,t} \mathbf{f}_{i,t}. \tag{1.2}$$

EWA forecaster (1.2) is a convex combination of experts advice, with $w_{i,1} = 1/N$ for all $i \in \mathscr{E}$, and with

$$w_{i,t} = \frac{\exp\left(-\lambda \sum_{s=1}^{t-1} \ell(\mathbf{f}_{i,s}, z_s)\right)}{\sum_{j=1}^{N} \exp\left(-\lambda \sum_{s=1}^{t-1} \ell(\mathbf{f}_{j,s}, z_s)\right)}, \tag{1.3}$$

for all $t \geq 2$, and $\lambda > 0$ is known as the inverse temperature parameter. The quantity $w_{i,t}$ is the weight associated with expert $i \in \mathscr{E}$. Experts with better performance (*i.e.,* smaller cumulative loss $\sum_{s=1}^{t-1} \ell(\mathbf{f}_{i,s}, z_s)$) in the past $t-1$ rounds will have higher weight at time t.

Theorem 1.1 (Theorem 2.2 of Cesa-Bianchi and Lugosi, 2006)**.** *If ℓ is bounded in $[0,1]$ and convex in its first argument, EWA forecaster with $\lambda > 0$ satisfies uniformly over any sequence $z_t \in \mathscr{Z}$ and $\mathbf{f}_{i,t} \in \mathscr{D}$ that*

$$\sum_{t=1}^{T} \ell(\hat{z}_t, z_t) \leq \min_{i \in \mathscr{E}} \sum_{t=1}^{T} \ell(\mathbf{f}_{i,t}, z_t) + \frac{\ln N}{\lambda} + \frac{\lambda T}{8}.$$

The right hand side is minimized (with respect to λ) at $\lambda = \sqrt{8 \ln N / T}$ and yields a regret bound $R_T(\mathscr{E}) = \sqrt{(T/2) \ln N}$.

The proof is given in Chapter 2 of Cesa-Bianchi and Lugosi (2006). In addition, the order $\sqrt{T}$ of $R_T(\mathscr{E})$ with respect to the time horizon T is optimal in the sense that there does not exist any other forecaster which can achieve a faster convergence rate in T. For

similar and more detailed results, we refer the reader to Littlestone and Warmuth (1994), Cesa-Bianchi *et al.* (1997) and Cesa-Bianchi and Lugosi (2006).

The convexity assumption of both decision space $\mathscr{D}$ and loss function ℓ is important in Theorem 1.1 to achieve a regret sublinear in T since the former guarantees that the linear combination $\hat{z}_t$ belongs to $\mathscr{D}$ and the latter is required in the proof. However, this assumption cannot always be satisfied in some cases. For example, in the binary classification problem where $\mathscr{D} = \mathscr{Z} = \{0,1\}$ and $\ell(y,z) = \mathbb{1}_{\{y \neq z\}}$, if one considers only two experts who always give constant prediction $\mathbf{f}_{1,t} \equiv 0$ and $\mathbf{f}_{2,t} \equiv 1$ respectively during all rounds, then no matter what predictions given by the forecaster, there always exists a sequence z_t (say $1 - \hat{z}_t$), $t = 1,2,\ldots,T$ such that $\sum_{t=1}^{T} \ell(\hat{z}_t, z_t) = T$. However, since the best expert predicts correctly at least half of the rounds, the regret of the forecaster is at least $\frac{T}{2}$, *i.e.*,

$$\sum_{t=1}^{T} \ell(\hat{z}_t, z_t) - \min_{i=1,2} \sum_{t=1}^{T} \ell(\mathbf{f}_{i,t}, z_t) \geq \frac{T}{2}.$$

Hence, in this setting, a regret sublinear in T is impossible. Other cases violating the convexity assumption include the online clustering problem and sequential principal curves that will be introduced in Chapter 2 and Chapter 3. It is clear that the loss functions defined in (2.1) and (3.2) are not convex in their first argument.

Randomized forecasting

One way to conquer the non-convexity is via randomized forecasting: at each time t, the forecaster firstly chooses a distribution ρ_t from the decision space $\mathscr{D}$ which is now assumed to be a set of probability distributions over $\mathscr{E}$, then a randomized index I_t is generated from ρ_t and the forecaster yields his prediction as $\hat{z}_t = \mathbf{f}_{I_t,t}$. The distribution ρ_t may depend not only on the past outcome $z_1, z_2, \ldots, z_{t-1}$ but also on its past predictions $I_1, \ldots, I_{t-1}$. The blackbox is only allowed to access to ρ_t but I_t before yielding z_t. Hence, even though the blackbox could know in advance a "rough outline" ρ_t of forecaster's strategy, this advantage would be cancelled out by the randomized prediction of forecaster, leading to an achievable regret sublinear in T. In this randomized prediction setting, the loss of forecaster can be defined as the expected value $\overline{\ell}$ of loss with respect to ρ_t, *i.e.*,

$$\overline{\ell}(\rho_t, z_t) = \mathbb{E}_t\left[\ell\left(\mathbf{f}_{I_t,t}, z_t\right)\right] = \int_{\mathscr{E}} \ell\left(\mathbf{f}_{i,t}, z_t\right) \mathrm{d}\rho_t(i),$$

where $\mathbb{E}_t$ denotes the conditional expectation of $\ell\left(\mathbf{f}_{I_t,t}, z_t\right)$ given $I_1, \ldots, I_{t-1}$ since ρ_t may depend on them. Due to the linearity of expectation, one makes the loss function $\overline{\ell}$ convex in its first argument. In addition, by taking again the example of binary classification discussed above, the merit of randomized forecasting can be shown clearly: if the forecaster simply chooses a trivial uniform distribution on $\{0,1\}$ at each time t, then whatever the outcome sequence is, the expected loss of forecaster can be reduced to $\frac{T}{2}$. In other words, without any deep consideration on the choice of ρ_t, the cumulative loss of forecaster has already been cut to half via randomized forecasting with a simple uniform distribution.

The randomized forecasting in the prediction with expert advice has been extensively and thoroughly studied in the chapter 4 of Cesa-Bianchi and Lugosi (2006). It has also been analysed under different restrictions on the available information to which the forecaster can be access. They include: the label-efficient prediction where the forecaster can only inquiry the outcomes z_t within limited times; the bandit games where the forecaster can only access to the loss of his own decisions while not knowing that of any other experts;

the sleeping experts where only a subset of experts is available and this availability can varies with time.

In the next section, we shall introduce another popular setting in online learning, namely, online regression. It allows the forecaster to access to other side information, not merely expert's advice.

1.1.2 Online regression

In the framework of online regression, the sequence z_t is a pair of observations $z_t = (x_t, y_t) \in \mathscr{X} \times \mathscr{Y}, t \geq 1$, where $\mathscr{X} \subset \mathbb{R}^d, d \geq 1$ and $\mathscr{Y} \subset \mathbb{R}$. At each t, prediction $\hat{y}_t$ of forecaster is on the basis of not only past information $z_s, s < t$ but also side information x_t, *i.e.*, $\hat{y}_t : (\mathscr{X} \times \mathscr{Y})^{t-1} \times \mathscr{X} \to \mathscr{Y}$. The experts set is now a set of prediction functions $\{\mathbf{f}_\theta, \theta \in \Theta\}$ with parameter θ, where $\mathbf{f}_\theta : \mathscr{X} \to \mathscr{Y}$. Given a loss function $\ell(\cdot,\cdot)$ on $\mathscr{Y} \times \mathscr{Y}$, the goal is to build a forecaster whose performance is uniformly nearly as good as the best one in $\{\mathbf{f}_\theta, \theta \in \Theta\}$, *i.e.*, uniformly over all $(x_1, y_1), \ldots, (x_T, y_T) \in \mathscr{X} \times \mathscr{Y}$

$$\sum_{t=1}^{T} \ell(\hat{y}_t, y_t) \leq \inf_{\theta \in \Theta} \left\{ \sum_{t=1}^{T} \ell\left(\mathbf{f}_\theta(x_t), y_t\right) + R_T(d, \theta) \right\}, \tag{1.4}$$

where $R_T(d,\theta)$ is a remainder term (for simplicity, we omit the possible dependence of $R_T(d,\theta)$ on $\mathscr{X}, \mathscr{Y}$ and $\mathbf{f}_\theta$). At first glance, regret bound of form (1.4) is different from the following counterpart of (1.1) in online regression setting:

$$\sum_{t=1}^{T} \ell(\hat{y}_t, y_t) - \inf_{\theta \in \Theta} \left\{ \sum_{t=1}^{T} \ell\left(\mathbf{f}_\theta(x_t), y_t\right) \right\} \leq R_T(d, \Theta). \tag{1.5}$$

However, (1.4) and (1.5) are equivalent in the sense that (1.5) can be deduced from (1.4) by taking $R_T(d,\Theta) = \sup_{\theta \in \Theta} R_T(d,\theta)$ and that (1.4) holds by taking $R_T(d,\theta) = R_T(d,\Theta)$ for any $\theta \in \Theta$. Regret bounds of both forms will be used in the sequel. Online regression has been addressed by numerous contributions to the literature. In particular, Azoury and Warmuth (2001) and Vovk (2001) each provide an algorithm close to the ridge regression with a regret bound of order $\mathscr{O}(d \ln T)$. Other contributions have investigated the Gradient-Descent algorithm (Cesa-Bianchi *et al.*, 1996; Kivinen and Warmuth, 1997) and the Exponentiated Gradient Forecasters (Kivinen and Warmuth, 1997; Cesa-Bianchi, 1999). However, most of these results require some constraint on the loss function such as the convexity of the loss function on its first argument or the exp-concavity of the square loss function.

Audibert (2009) generalizes the regret bound in online regression by considering loss functions with less regularities, especially loosing the convexity constraint. He presents a sequential randomized algorithm (*SeqRand*, section 4.2 of Audibert, 2009) which considers the decision space $\mathscr{D}$ as set of probability distributions on the prediction function set $\{\mathbf{f}_\theta, \theta \in \Theta\}$ (also denoted by Θ for simplicity if no confusion is made). At the beginning (*i.e.*, t=1), *SeqRand* chooses a prior distribution π on Θ and the first randomized prediction function is draw according to π. Then it iteratively updates the prior to (Gibbs) posterior distributions $\hat{\rho}_t, t = 1, \ldots, T$ that concentrate simultaneously on functions having both small cumulative loss and low variance with respect to the previously drawn prediction function. In addition, Gerchinovitz (2011) considers online regression in the so-called high dimension setting ($d \gg T$) with quadratic loss function $\ell(y_1, y_2) = (y_1 - y_2)^2$, and with particular prediction functions $\mathbf{f}_\theta = \theta \cdot \varphi = \sum_{j=1}^{d} \theta_j \varphi_j$ where $\varphi = (\varphi_1, \ldots, \varphi_d)$ is a dictionary

of base predictors $\varphi_j : \mathscr{X} \to \mathscr{Y}$ and $\theta \in \Theta = \mathbb{R}^d$. Under the sparsity hypothesis that there exists a $\theta^\star$ with $s \ll d$ non-zero coordinates whose cumulative loss $\sum_{t=1}^T \ell(\mathbf{f}_{\theta^\star}, y_t)$ is small, one can obtain the following sparsity regret bound:

$$\sum_{t=1}^T \ell(\hat{y}_t, y_t)^2 \leq \inf_{\theta \in \mathbb{R}^d} \left\{ \sum_{t=1}^T \ell(\mathbf{f}_\theta, y_t) + (||\theta||_0 + 1) g_{T,d}\left(||\theta||_1, ||\varphi||_\infty\right) \right\},$$

where $||\theta||_0$ is the number of non-zero coordinates of θ, $||\theta||_1 = \sum_{j=1}^d |\theta_j|$ and $||\varphi||_\infty = \sup_{x \in \mathscr{X}} \max_{j=1,\dots,d} |\varphi_j(x)|$; $g_{T,d}$ is a function growing logarithmically with d and T. Predictions $\hat{y}_t$ achieving this regret bound are

$$\hat{y}_t = \int [\theta \cdot \varphi(x_t)]_B \mathrm{d}\rho_t(\theta),$$

where $[x]_B = \max\{-B, \min\{B, x\}\}$ truncates x at $B > \sup_{t=1,\dots,T} |y_t|$ and

$$\mathrm{d}\rho_t(\theta) \propto \exp\left(-\lambda \sum_{s=1}^{t-1} \left(y_s - [\theta \cdot \varphi(x_s)]_B \right)^2 \right) \mathrm{d}\pi(\theta), \quad t \geq 1 \tag{1.6}$$

is a posterior distribution with a prior π on $\mathbb{R}^d$ in favor of generating sparse θ (see Gerchinovitz, 2011 for more details). One can see that the posterior ρ_t concentrates on "expert" θ having smaller cumulative loss. Prediction $\hat{y}_t$ is the mean of a truncated linear combination $\theta \cdot \varphi$ with respect to ρ_t. Moreover, Gerchinovitz (2011) also investigates online regression with ℓ_1 regularization on Θ, *i.e.,* with $\Theta = \{\theta, ||\theta||_1 \leq U\}$ in (1.4) where $U > 0$ and obtains a regret bound sublinear in T.

Results in Audibert (2009) and Gerchinovitz (2011) rely on the notion of Kullback-Leibler divergence and Pac-Bayes. These two notions, which will be introduced in the next section, are also indispensable to our study in unsupervised setting.

1.1.3 A quasi-Bayesian online analysis

Given a measurable space Θ (embedded with its Borel σ-algebra), we let $\mathscr{P}(\Theta)$ denote the set of probability distributions on Θ, and for some reference measure ν, we let $\mathscr{P}_\nu(\Theta)$ be the set of probability distributions absolutely continuous with respect to ν. For any probability distributions $\rho, \pi \in \mathscr{P}(\Theta)$, the Kullback-Leibler divergence $\mathscr{K}(\rho, \pi)$ is defined as

$$\mathscr{K}(\rho, \pi) = \begin{cases} \int_\Theta \ln\left(\frac{\mathrm{d}\rho}{\mathrm{d}\pi}\right) \mathrm{d}\rho & \text{when } \rho \in \mathscr{P}_\pi(\Theta), \\ +\infty & \text{otherwise.} \end{cases}$$

Note that for any bounded measurable function $h : \Theta \to \mathbb{R}$ and any probability distribution $\rho \in \mathscr{P}(\Theta)$ such that $\mathscr{K}(\rho, \pi) < +\infty$,

$$-\ln \int_\Theta \exp(-h) \mathrm{d}\pi = \inf_{\rho \in \mathscr{P}(\Theta)} \left\{ \int_\Theta h \mathrm{d}\rho + \mathscr{K}(\rho, \pi) \right\}. \tag{1.7}$$

This result, which may be found in Csiszár (1975) and Catoni (2004, Equation 5.2.1), is critical to prove Theorem 1.2 and Theorem 1.3. Further, the infimum is achieved at the so-called Gibbs quasi-posterior $\hat{\rho}$, defined by

$$\mathrm{d}\hat{\rho} = \frac{\exp(-h)}{\int \exp(-h) \mathrm{d}\pi} \mathrm{d}\pi. \tag{1.8}$$

Gibbs distribution $\mathrm{d}\hat{\rho}$ concentrates on prediction functions θ that are close to minimizing the function h if prior π is uniform over Θ.

The use of Gibbs quasi-posterior is especially advocated by the PAC-Bayesian theory which originates in the machine learning community in the late 1990s, in the seminal works of Shawe-Taylor and Williamson (1997) and McAllester (1999b,a) (see also Seeger, 2002, 2003). In the statistical learning community, the PAC-Bayesian approach has been extensively developed by Catoni (2004, 2007), Audibert (2004b) and Alquier (2006), and later on adapted to the high dimensional setting Dalalyan and Tsybakov (2007, 2008), Alquier and Lounici (2011), Alquier and Biau (2013), Guedj and Alquier (2013), Guedj and Robbiano (2018) and Alquier and Guedj (2017).

In a parallel effort, the online learning community has contributed to the PAC-Bayesian theory in the online regression setting (Kivinen and Warmuth, 1999). Audibert (2009) and Gerchinovitz (2011) have been the first attempts to merge both lines of research. One can find that both weights $w_{i,t}$ of (1.3) in prediction with expert advice and the posterior ρ_t of (1.6) in online regression are closely related to (1.8). More precisely, setting $\Theta = \mathscr{E}$, the set of N experts, prior $\pi(i) = 1/N$ uniform on Θ, and $h := h_t(i) = \sum_{s=1}^{t-1} \ell(\mathbf{f}_{i,s}, z_s)$ at round t in (1.8) makes $\hat{\rho} = (w_{1,t}, w_{2,t}, \ldots, w_{N,t})$ a simplex distribution on $\mathscr{E}$, and it is more obvious for ρ_t in (1.6).

The above duality formula can help us to refine, for instance, the regret bound of Theorem 1.1, as indicated by the Theorem 1.2 below.

Theorem 1.2. *If the loss function $\ell : \mathscr{D} \times \mathscr{Z} \longrightarrow \mathbb{R}_+$ is bounded in $[0,1]$ and convex in its first argument, then for all $T \in \mathbb{N}^*$ and $\lambda > 0$, the EWA forecaster satisfies uniformly over all $\mathbf{f}_{i,t} \in \mathscr{D}$ and $z_t \in \mathscr{Z}$*

$$\sum_{t=1}^{T} \ell(\hat{z}_t, z_t) \leq \inf_{\rho \in \mathscr{X}_N} \left\{ \sum_{i=1}^{N} \rho_i \sum_{t=1}^{T} \ell(\mathbf{f}_{i,t}, z_t) + \frac{\mathscr{K}(\rho, \pi)}{\lambda} \right\} + \frac{\lambda T}{8},$$

where $\mathscr{X}_N$ is the set of all simplex distribution on finite set $\mathscr{E}$; $\pi = (1/N, 1/N, \ldots, 1/N) \in \mathscr{X}_N$ is the initial weight vector for EWA forecaster, and $\rho = (\rho_1, \rho_2, \ldots, \rho_N) \in \mathscr{X}_N$.

The above result is a simpler version of Theorem 4.6 of Audibert (2009) by taking convex and bounded loss function. Its proof, which can be found in proposition 2.2 of Gerchinovitz (2011), relies on the duality formula (1.7) and Hoeffding's lemma.

In comparison with Theorem 1.1, regret bound in Theorem 1.2 is refined in the sense that the former can be deduced from the latter by choosing ρ as Dirac mass $\delta_{I^\star}$ at $I^\star$, where $I^\star = \mathbf{argmin}_{i \in \mathscr{E}} \sum_{t=1}^{T} \ell(\mathbf{f}_{i,t}, z_t)$. Moreover, if the best expert is not unique, let $\mathscr{B}$ be the set containing $s > 1$ best experts. Setting ρ as an uniform on $\mathscr{B}$ leads to $\mathscr{K}(\rho, \pi) = \ln(N/s)$, a smaller value than $\ln(N)$ in Theorem 1.1. Hence, one can have a smaller regret bound by duality formula under some circumstances.

The Gibbs quasi-posterior as well as the Kullback-Leibler divergence are also important tools for online clustering and sequential principal curves (Chapter 2 and Chapter 3), the two main contributions of this manuscript. They will be summarized respectively in section 1.3.1 and 1.3.6. Before proceeding to these two sections, we first introduce Markov Chain Monte Carlo that are important tools for obtaining randomized samples from quasi-posterior since explicit form of Gibbs quasi-posterior is often impossible (Alquier and Biau, 2013, Guedj and Alquier, 2013 and Chapter 2 of this manuscript).

1.2 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) methods are a class of algorithms for generating a sequence of random samples from a probability distribution for which direct sampling

is generally difficult. These samples can play a role of randomized predictions or can be used to approximate the mean of estimators. The working principle of MCMC method is quite straightforward to describe. Given a target distribution ρ (on Θ) from which one wants to generate samples, we build a Markov kernel K with stationary distribution ρ and then generate a Markov chain $\theta^{(n)}, n = 0, 1, \ldots$, (abbreviated as $\left(\theta^{(n)}\right)$ in the sequel) using this kernel so that the limiting distribution of $\left(\theta^{(n)}\right)$ is ρ. The MCMC method is quite useful in many fields such as machine learning, Bayesian inference etc. One of the most well-known examples of MCMC method is Metropolis-Hastings algorithm which was firstly introduced by Metropolis *et al.* (1953) and later generalized by Hasting (1970) and Peskun (1973) to statistical simulation. In the sequel, we first specify the definition of Metropolis-Hastings algorithm and then introduce the Reversible Jump MCMC which will help us to generate $\hat{\mathbf{c}}_t$ from the posterior in Algorithm 1.2 and Algorithm 2.2.

1.2.1 Metropolis-Hastings algorithm

In this section, the distribution of interest is referred to as the *target* distribution ρ. The (very limited) assumption underlying the Metropolis-Hastings algorithm, besides the availability of ρ, is that one can directly simulate samples from a transition density q, often called *proposal* or *instrumental* distribution, whose functional form is known. The Metropolis-Hastings(MH) algorithm associated with *target* distribution ρ and *proposal* distribution q produces a Markov chain $\left(\theta^{(n)}\right)$ in the following way:

Algorithm 1.1 *Metropolis-Hastings algorithm*

1: **Given**: $\theta^{(0)}$
2: **For** $n = 0, 1, \ldots$
3: generate $\theta' \sim q\left(\cdot|\theta^{(n)}\right)$
4: take

$$\theta^{(n+1)} = \begin{cases} \theta' & \text{with probability} \quad \alpha = \left\{ \dfrac{\rho\left(\theta'\right) q\left(\theta^{(n)}|\theta'\right)}{\rho\left(\theta^{(n)}\right) q\left(\theta'|\theta^{(n)}\right)} \wedge 1 \right\}, \\ \theta^{(n)} & \text{otherwise.} \end{cases} \tag{1.9}$$

5: **End for**

where the initial value $\theta^{(0)}$ is chosen arbitrarily in Θ such that $\rho\left(\theta^{(0)}\right) > 0$, q is a dependent *proposal* on the current state $\theta^{(n)}$ and $a \wedge b \triangleq \min\{a, b\}$.

The MH algorithm compares the importance ratio $\frac{\rho(\theta')}{q(\theta'|\theta^{(n)})}$ of the candidate state θ' with the corresponding ratio $\frac{\rho(\theta^{(n)})}{q(\theta^{(n)}|\theta')}$ for a "reverse" move from state θ' to current state $\theta^{(n)}$. The Metropolis chain moves to candidate state if it has higher importance ratio (or importance weight) than the current state; otherwise, it moves there with the probability defined by the relative magnitudes α of importance ratios, which is called the acceptance rate of MH algorithm.

Under certain conditions (Theorem 2.1 of Mengersen and Tweedie, 1996), the Markov chain $\left(\theta^{(n)}\right)$ generated by MH algorithm is ergodic and has limiting distribution ρ. The ergodicity guarantees that the marginal distribution of $\theta^{(n)}$ converges finally to the target function ρ with regardless of the initial value for $\theta^{(0)}$.

The rate of convergence of MH algorithm depends on the choice of *proposal*. The support of the *proposal* distribution should be compatible with *target* distribution (*i.e.,* $q(\theta) > 0$ if $\rho(\theta) > 0$). In other words, the proposal should have an ability to explore the entire support of *target* distribution. If the support of *proposal* were quite small or if the *proposal* were not disperse enough(*i.e.,* having a much lighter tail) compared to the *target*, then the candidate state θ' would be very close to the current state $\theta^{(n)}$, leading to a comparatively high acceptance rate. Hence the chain $\left(\theta^{(n)}\right)$ will move frequently but "locally" around the state space. On the other hand, a too disperse *proposal* that generates large move is more likely to have a very low value of $\rho(\theta')$, hence would suffer from a low acceptance rate even though it is enable to explore freely the whole support of *target*. Therefore, the choice of proposal distribution is crucial to the performance of MH algorithm.

The dependent proposal $q\left(\cdot|\theta^{(n)}\right)$ in MH algorithm can be replaced by an independent one $q(\cdot)$, leading the acceptance rate α in (1.9) reduce to

$$\alpha = \left\{ \frac{\rho\left(\theta'\right) q\left(\theta^{(n)}\right)}{\rho\left(\theta^{(n)}\right) q\left(\theta'\right)} \wedge 1 \right\}.$$

1.2.2 Reversible Jump MCMC

In the previous section, we introduced the general setting of Metropolis-Hastings Markov Chain Monte Carlo. However, MH algorithm can only cope with generating samples from posterior that is defined on a parameter space of fixed dimensions. In this section, we present the Reversible Jump Markov Chain Monte Carlo (RJMCMC), an extension of MCMC, that applied to varying-dimension problem and creates a novel method capable of reversible jumping between subspaces of differing dimensions. This method, first proposed by Green (1995), serves as a tool for us to cope with online clustering discussed in Chapter 2.

More precisely, in Bayesian model selection or online clustering, the parameter space $\Theta = \cup_{k=1}^{p} \Theta_k$ can be a union of finite (or countable) sub-spaces Θ_k of different dimensions, where k is a model indicator and $\Theta_k \in \mathbb{R}^{d_k}$ the parameter space associated to model k. We suppose that ρ is a certain distribution defined on Θ, and that direct sampling from it is impossible. For example, ρ can be the posterior distribution in Bayesian analysis. Denote by (θ, k) the current state of Markov chain $\left(\theta^{(n)}, k^{(n)}\right), n = 1, \ldots$, where $\theta \in \Theta_k$. The proposal θ' can be constructed by a two-step transition: choosing firstly a proposal indicator k' from a model transition probability $\mathrm{p}_{kk'}$ such that $\sum_{k'=1}^{p} \mathrm{p}_{kk'} = 1$; then θ' is formed as

$$\theta' = g_{1,kk'}(\theta, v),$$

where $g_{1,kk'}$ is a deterministic mapping of θ and v, and v is a realization of a random vector V who has a density $q_{kk'}(\cdot|\theta)$ on $\mathbb{R}^{d_{kk'}}$.

The mapping $g_{1,kk'}$ is one of the two components of $g_{kk'}$ whose definition is given below:

$$(\theta', v') = g_{kk'}(\theta, v) = \left(g_{1,kk'}(\theta, v), g_{2,kk'}(\theta, v)\right),$$

where $g_{kk'}$ must be an one-to-one and differentiable function from $\mathbb{R}^{d_k} \times \mathbb{R}^{d_{kk'}}$ to $\mathbb{R}^{d_{k'}} \times \mathbb{R}^{d_{k'k}}$. The requirement of $g_{kk'}$ indicates that $d_k + d_{kk'} = d_{k'} + d_{k'k}$, an equality known as the dimension matching condition. It is a prerequisite for the detailed balance condition which requires the equilibrium probability of moving from θ to θ' and vice versa. It guarantees the convergence of RJMCMC.

The proposal θ' is then accepted with probability

$$\alpha(\theta,\theta') = \left\{ \frac{\rho(\theta',k')\,\mathrm{p}_{k'k}\,q_{k'k}(v')}{\rho(\theta,k)\,\mathrm{p}_{kk'}\,q_{kk'}(v)} \left| \det\left(\frac{\partial g_{kk'}(\theta,v)}{\partial\theta\partial v} \right) \right| \wedge 1 \right\},$$

where **det** denotes the Jacobian determinant and $|\cdot|$ the absolute value. The acceptance probability here is different from that defined in (1.9) with an additional Jacobian determinant which compensates for the variation of dimensions between θ' and θ.

1.3 Contributions

The theoretical work of this manuscript is presented in two chapters. In Chapter 2, we generalize the quasi-Bayesian methodology to the online clustering problem. We introduce a new and adaptive online clustering algorithm relying on a quasi-Bayesian approach, with a dynamic (*i.e.*, time-dependent) estimation of the (unknown and changing) number of clusters. We prove that our approach is supported by minimax regret bound. We also provide an RJMCMC-flavored implementation (called PACBO [1]) for which we give a convergence guarantee. In Chapter 3, we extend our work to sequential principal learning problem. We propose an adaptive algorithm that can sequentially summarize the data by polygonal line. The algorithm is different but inspired by the quasi-Bayesian methodology whose performance is supported by sublinear regret bounds. An implementation based on local greed search that combines both the notion of sleeping experts and bandit games is also given in this chapter.

The principal contributions of Chapter 2 are detailed in sections 1.3.1 to 1.3.4 while that of Chapter 3 are detailed in sections 1.3.6 to 1.3.8.

1.3.1 Online clustering framework

In the framework of online clustering, the sequence of observations is denoted as x_t for $t = 1,\ldots,$, where $x_t \in \mathscr{X} \subset \mathbb{R}^d$ (to keep the consistency of notation with Chapter 2, we use x_t in the sequel). Each x_t is a point in $\mathbb{R}^d$. Our goal is to predict a partition of the observed points into K_t cells, for any $t = 1,\ldots,T$. The motivation for considering online clustering can be summarized from two perspectives: firstly, to our best knowledge, clustering in online setting has not been largely considered, for example, Guha *et al.* (2003), Barbakh and Fyfe (2008) and Liberty *et al.* (2016) study the so-called data streaming clustering problem. It amounts to cluster online data to a fixed number of groups in a single pass, or a small number of passes, while using little memory. Choromanska and Monteleoni (2012) considers online clustering by aggregating finite number of batch clustering algorithms and the regret is measured by the difference between the cumulative loss of their algorithm and that of the best batch one. This is different from our setting since we shall compare the performance of our algorithm with that of the best partition in the hindsight. Secondly, in practice, if the data x_t contains information about customer's commercial behavior, the clusters of customers, combined with other information such as customer's transaction or the problems they met, can help commerce company to make suitable criterion to target customers. Since the criterion need to reflect instantaneously the change of interest of customers on site, offline clustering might not well suit this need. To this aim, the output of our algorithm at time t is a vector $\hat{\mathbf{c}}_t = (\hat{c}_{t,1}, \hat{c}_{t,2}, \ldots, \hat{c}_{t,K_t})$ of K_t centers in $\mathbb{R}^{dK_t}$,

1. https://cran.r-project.org/web/packages/PACBO/index.html

depending on the past information $(x_s)_{1:(t-1)}$ (we use notation $(x_s)_{a:b}, a \le b$ in the sequel to represent sequence $x_a, x_{a+1}, \ldots, x_b$) and $(\hat{\mathbf{c}}_s)_{1:(t-1)}$. A partition is then created by assigning any point in $\mathbb{R}^d$ to its closest center. If several centers satisfy it, we make a convention that this point is assigned to the center with smallest index. Hence, each vector of centers corresponds to a Voronoi partition of $\mathbb{R}^d$. When x_t is newly revealed, the quality of prediction $\hat{\mathbf{c}}_t$ is evaluated by the instantaneous loss computed as

$$\ell(\hat{\mathbf{c}}_t, x_t) = \min_{1 \le k \le K_t} \left| \hat{c}_{t,k} - x_t \right|_2^2,$$

where $|\cdot|_2$ is the ℓ_2-norm in $\mathbb{R}^d$. This quantity, often used in k-means algorithm, measures the squared distance of point x_t to its closest center in $\hat{\mathbf{c}}_t$. However, since $\hat{\mathbf{c}}_t$ is build based on the past observations $(x_s)_{1:t-1}$, the loss $\ell(\hat{\mathbf{c}}_t, x_t)$ can be regarded as a measurement which emphasis more on the prediction ability of our algorithm. Our goal is to predict almost as good as the best constant partition $\mathbf{c}$, in the hindsight, within $\mathscr{C} = \cup_{k=1}^{p} \mathbb{R}^{dk}$, where $\mathscr{C}$ is the partition space allowing for at most $p > 0$ cells, *i.e.*, to satisfy uniformly over all $(x_t)_{1:T} \in \mathscr{X}^T$, a regret bound of the form,

$$\mathbb{E}\left[\sum_{t=1}^{T} \ell(\hat{\mathbf{c}}_t, x_t) \right] \le \inf_{k \in \{1,\ldots,p\}} \left\{ \inf_{\mathbf{c} \in \mathbb{R}^{dk}} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) + R_{T,k}(\mathbf{c}) \right\}, \tag{1.10}$$

where $R_{T,k}(\mathbf{c})$ should be as small as possible and in particular sublinear in T (for simplicity, we omit here the dependence of $R_{T,k}(\mathbf{c})$ on d, p and $\max_{t=1,\ldots,T} |x_t|_2$). Since $\ell(\hat{\mathbf{c}}_t, x_t)$ is not convex in its first argument in $\mathbb{R}^{dK_t}$ and it is impossible to combine constant partitions with different number of cells, we resort to the randomized prediction, therefore the expectation in (1.10) is taken with respect to randomized predictors $\hat{\mathbf{c}}_t$.

1.3.2 Regret bounds for clustering

To simplify the analysis, we assume that the sequence $(x_t)_{1:T}$ is bounded in a ℓ_2-ball $B_d(R) = \{x \in \mathbb{R}^d, |x|_2 \le R\}$, centered in $0 \in \mathbb{R}^d$ with radius $R > 0$. At the beginning, we assume that both R and T is known a priori. The first version of our algorithm is given in Algorithm 1.2. It is a derivative of algorithm *SeqRand* but adapted to the unsupervised setting. More precisely,

---

Algorithm 1.2 The quasi-Bayesian online clustering algorithm (cf. Algorithm 2.1 in Chapter 2)

1: **Input parameters**: $p > 0, \pi \in \mathscr{P}(\mathscr{C})$, $\lambda > 0$ and $S_0 \equiv 0$
2: **Initialization**: Draw $\hat{\mathbf{c}}_1 \sim \pi = \hat{\rho}_1$
3: **For** $t \in [\![1, T]\!]$
4: Get the data x_t
5: Draw $\hat{\mathbf{c}}_{t+1} \sim \hat{\rho}_{t+1}(\mathbf{c})$ where $\mathrm{d}\hat{\rho}_{t+1}(\mathbf{c}) \propto \exp\big(-\lambda S_t(\mathbf{c})\big)\mathrm{d}\pi(\mathbf{c})$, and

$$S_t(\mathbf{c}) = S_{t-1}(\mathbf{c}) + \ell(\mathbf{c}, x_t) + \frac{\lambda}{2}\big(\ell(\mathbf{c}, x_t) - \ell(\hat{\mathbf{c}}_t, x_t)\big)^2.$$

6: **End for**

---

where $\mathscr{P}(\pi)$ is the set of all probability distributions on $\mathscr{C}$ and π is a prior. The quantity $S_t(\mathbf{c})$ is computed recursively by adding at each time t both the loss $\ell(\mathbf{c}, x_t)$ and a quadratic term measuring the loss difference between partition $\mathbf{c}$ and prediction $\hat{\mathbf{c}}_t$. When no data

is available, we give our first partition by sampling from a prior π and at each time $t+1, t=0,1,\ldots,T-1$, the partition $\hat{\mathbf{c}}_{t+1}$ is sampled from a posterior distribution $\hat{\rho}_{t+1}$ which concentrates on partition having both small cumulative loss and low variance. In addition, one can notice that the randomness of $\hat{\mathbf{c}}_{t+1}$ originates from not only the way it is sampled but also the dependence of $\hat{\rho}_{t+1}$ on random variables $\hat{\mathbf{c}}_t, \hat{\mathbf{c}}_{t-1}, \ldots, \hat{\mathbf{c}}_1$.

By adapting the lemma of Audibert (2009) to unsupervised setting, with a uniform prior π on $\mathscr{C}$, defined as

$$\pi(\mathbf{c}) = \sum_{k \in [\![1,p]\!]} q(k) \pi_k(\mathbf{c}) \mathbb{1}_{\{\mathbf{c} \in \mathbb{R}^{dk}\}}, \tag{1.11}$$

where q a discrete probability distribution with parameter $\eta \geq 0$ on the set $[\![1,p]\!] := \{1,\ldots,p\}$, *i.e.*,

$$q(k) = \frac{\exp(-\eta k)}{\sum_{i=1}^{p} \exp(-\eta i)}, \tag{1.12}$$

and π_k is the product of k independent uniform distributions on ℓ_2-balls $B_d(2R)$ in $\mathbb{R}^d$ (we refer to (2.6) to see its concrete form). We prove that Algorithm 1.2 satisfies the following bound.

Theorem 1.3. *For any deterministic sequence $(x_t)_{1:T} \in \mathbb{R}^{dT}$ and any $p \geq 1$, consider π defined by (1.11) with $\eta \geq 0$ and $R \geq \max_{t=1,\ldots,T} |x_t|_2$. If $\lambda \geq (d+2)/(2TR^2)$, the procedure described in Algorithm 1.2 satisfies*

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) \leq \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\mathbf{c} \in \mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) + \frac{dk}{2\lambda} \ln\left(\frac{8R^2\lambda T}{d+2}\right) + \frac{\eta}{\lambda} k \right\} + \left(\frac{\ln p}{\lambda} + \frac{d}{2\lambda} + \frac{81\lambda T R^4}{2} \right),$$

where the expectation on the left hand side is taken with respect to the joint distribution, denoted by $(\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_t)$, of $(\hat{\mathbf{c}}_1, \hat{\mathbf{c}}_2, \ldots, \hat{\mathbf{c}}_t)$; $\mathscr{C}(k,R) = \{\mathbf{c} = (c_j)_{j=1,2,\ldots,k} \in \mathbb{R}^{dk}, c_i \neq c_j, i \neq j,$ such that $|c_j|_2 \leq R, \forall j\}$ is a subset of $\mathbb{R}^{dk}$ which defines the range of k centers within $\mathbf{c} \in \mathscr{C}(k,R)$.

Notice that $\inf_{\mathbf{c} \in \mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t)$, the cumulative loss of the best partition with exactly k cells, is a non-increasing function of the number k of cells while the penalty is linearly increasing with k. Small values for λ (or equivalently, large values for R) lead to small values for k. The additional term induced by the complexity of $\mathscr{C} = \cup_{k=1,\ldots,p} \mathbb{R}^{dk}$ is $\ln p$. A suitable calibration of λ yields a sublinear remainder term in the following corollary.

Corollary 1.1. *(cf. Corollary 2.2) Under the previous notation with $\lambda = (d+2)\sqrt{\ln T}/(2\sqrt{T}R^2)$, the procedure described in Algorithm 1.2 satisfies*

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) \leq \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\mathbf{c} \in \mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) + c_{d,R,\eta} k \sqrt{T \log T} \right\} + \left(c_{p,d,R} + c_{d,R}\right) \sqrt{T \log T}, \tag{1.13}$$

where $c_{d,R,\eta}, c_{p,d,R}$ and $c_{d,R}$ are constants (see Corollary 2.2 for their explicit form).

We see that the expected cumulative loss is close to that of the best partition in batch setting up to a reminder term of the order of $\sqrt{T \ln T}$. Moreover, the right hand side of

the above inequality also indicates that our procedure seeks to partitions which does not simply have a good performance on the quantization error but also on its number of cells. In addition, If we assume that the sequence $x_1,\ldots,x_T$ is generated from a distribution with $k^\star \in [\![1,p]\!]$ clusters whose centers belong to $\mathscr{C}(k^\star,R)$, then Corollary 1.1 can yield the following

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\ldots,\hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t,x_t) - \inf_{\mathbf{c}\in\mathscr{C}(k^\star,R)} \sum_{t=1}^{T} \ell(\mathbf{c},x_t) \leq Jk^\star\sqrt{T\ln T},$$

where J is a constant depending on d, R, η and $\ln p$. The first term on the left is the expected cumulative loss of our algorithm and the second one is the oracle cumulative loss. The above inequality indicates that the regret of our randomized procedure, defined as the difference between expected cumulative loss and oracle cumulative loss, is of the order $\sqrt{T\ln T}$. However, whenever $k^\star > p$, the term $k\sqrt{T\ln T}$ in (1.13) emerges and the bound in Corollary 1.1 is deteriorated (we refer to Corollary 2.2 for more details).

1.3.3 Adaptive regret bounds for clustering

The parameter λ in Corollary 1.1 depends both on time horizon T and on bound R of ℓ_2-norm of sequence $(x_t)_{1:T}$. These two quantities are usually unknown a priori. It prompts us to make λ adaptive to both T and R. In what follows, we begin with the case when only R is known and then proceed to the case when both of them are unknown. When R is known, we propose an adaptive generalization of Algorithm 1.2 whose steps are detailed in Algorithm 2.2. The difference with respect to Algorithm 1.2 is that a sequence of temperature parameters $\lambda_0 = \lambda_1 = 1, \lambda_t = (d+2)\sqrt{\ln t}/\left(2\sqrt{t}R^2\right), t = 2,\ldots,T$ is initialized and substitute for λ in each iteration.

The price to pay for not knowing the time horizon T (which is a much more realistic assumption for online learning) is a multiplicative factor 2 in front of the term $c_{d,R}$ in the upper bound (cf. Corollary 2.3). However, this does not degrade the rate of convergence $\sqrt{T\ln T}$.

When both R and T remain unknown, we use the doubling trick (Section 2.3, Cesa-Bianchi and Lugosi, 2006 and Cesa-Bianchi *et al.*, 2007) in **Alg-R** of Chapter 2, to show how can we make our procedure to be a fully adaptive one. The key idea of doubling trick relies on dividing time steps into several epochs $(t_{r-1}+1, t_{r-1}+2,\ldots,t_r), r = 0,1,\ldots,t_{-1}=0)$ whose length increases exponentially. For time round t within epoch r, *i.e.*, $t \in [t_{r-1}+1, t_{r-1}+2,\ldots,t_r]$, we set $\lambda := \lambda_{r,t} = \frac{(d+2)\sqrt{\ln t}}{2\sqrt{t}R_{t_{r-1}}^2}$ where $R_0 = 1$ and $R_t = \max_{s=1,\ldots,t} 2^{\lceil \log_2(|x_s|_2)\rceil}, t \geq 1$ and $\lceil x\rceil$ represents the least integer greater than or equal to $x \in \mathbb{R}$.

Corollary 1.2. *(cf. Corollary 2.3) The regret of algorithm* ***Alg-R*** *satisfies,*

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\ldots,\hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t,x_t) \leq \inf_{k\in[\![1,p]\!]} \left\{ \inf_{\mathbf{c}\in\mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c},x_t) + \frac{28}{3} c_{d,R,\eta} k\sqrt{T\log T} \right\} + \frac{28}{3}\left(c_{p,d,R} + 2c_{d,R}\right)\sqrt{T\log T} + \frac{112}{3}R^2,$$

where $R = \max_{t=1,\ldots,T} |x_t|_2$.

Note that an additional multiplicator $\frac{28}{3}$ and an additional term $\frac{112}{3}R^2$ is what we need to pay for making our algorithm also adaptive to unknown R. Our algorithm is fully adaptive to both the upper bound R for the range of sequence $(x_t)_{1:T}$ and the time horizon T.

In the next section, we investigate the lower bound of minimax regret which is a new notion that will be defined. We then prove that this lower bound is asymptotically of the same order as that of regret bounds discussed above.

1.3.4 Minimax regret for online clustering

We obtain in Corollary 1.1 that the regret bound is of the order $\sqrt{T \ln T}$. In this section, we first define notion of minimax regret in online clustering setting and then prove a lower bound on it. This lower bound shares the same order in T as regret bound indicating that our algorithm is optimal in the minimax sense.

Recall that we supposed the sequence $(x_t)_{1:T}$ is bounded in a $\mathscr{X} = B_d(R) = \{x \in \mathbb{R}^d, |x|_2 \leq R\}$, and denote by $\mathscr{C} = \cup_{k=1}^{p} \mathbb{R}^{dk}$ the partition space. We consider first following definition.

Definition 1.1. *We call minimax regret on sequence $(x_t)_{1:T}$ with loss function ℓ the quantity*

$$\inf_{\hat{P}} \sup_{(x_t)_{1:T}} \left\{ \sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\ldots,\hat{\rho}_t)}[\ell(\hat{\mathbf{c}}_t, x_t)] - \inf_{\mathbf{c} \in \mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) \right\}, \tag{1.14}$$

where the infimum is taken over all forecaster's strategy $\hat{P} \stackrel{\Delta}{=} (\hat{\rho}_t)_{1:T}$ and the supremum is taken over all sequence $(x_t)_{1:T} \in \mathscr{X}^T$. More precisely, the strategy $\hat{P}$ is a sequence $\hat{\rho}_1, \hat{\rho}_2, \ldots$, of functions $\hat{\rho}_t : \mathscr{X}^{t-1} \longrightarrow \mathscr{P}(\mathscr{C})$, where we recall that $\mathscr{P}(\mathscr{C})$ is the set of all probability distribution over $\mathscr{C}$. Each $\hat{\rho}_t$ in the expectation denotes however the probability distribution which is the value of forecasting strategy applied on the available data, i.e., *$\hat{\rho}_t \stackrel{\Delta}{=} \hat{\rho}_t(x_1, x_2, \ldots, x_{t-1})$. In the following, we abuse the notations $\hat{\rho}_t$ if no confusion arises. In addition, notation $(\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_t)$ in the above expectation denotes the joint distribution of $(\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_t)$. This definition of minimax regret in online clustering is an analogue of minimax defined in prediction with expert advice (section 2.3 of Cesa-Bianchi and Lugosi, 2006) and online regression (section 2.10 of Gerchinovitz, 2011).*

However, value of (1.14) can be arbitrarily small or even negative since the infimum is taken over any probability distribution on $\mathscr{C}$ and there is no restriction on the number of cells of each partition $\hat{\mathbf{c}}_t$. The lower bound does not match the regret bound of Corollary 1.1. To handle this, we need to introduce a penalized term which accounts for the number of cells of each partition to the loss function ℓ. In the sequel, denote by $|\mathbf{c}|$ the number of cells (or centers) in any partition $\mathbf{c} \in \mathscr{C}$. We proceed to the minimax regret with penalized loss function ℓ_{pen}.

Definition 1.2. *We call minimax regret on sequence $(x_t)_{1:T}$ with penalized loss function ℓ_{pen} the quantity*

$$\mathscr{V}_T(k) \stackrel{\Delta}{=} \inf_{\hat{P}} \sup_{(x_t)_{1:T}} \left\{ \sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\ldots,\hat{\rho}_t)} \left[\ell_{pen}(\hat{\mathbf{c}}_t, x_t) \right] - \inf_{\mathbf{c} \in \mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) \right\}, \tag{1.15}$$

where $\ell_{pen}(\mathbf{c}, x) = \ell(\mathbf{c}, x) + \sqrt{\frac{\ln T}{T}} |\mathbf{c}|$ for any $\mathbf{c} \in \mathscr{C}$ and $x \in \mathscr{X}$.

Lower bound on the minimax regret

In the next theorem we show that the lower bound of (1.15) is asymptotically of the order $k\sqrt{T\ln T}$ and that the regret bound of Algorithm 1.2 (similar for its adaptive version) with the penalized loss function ℓ_{pen} is of the same order as the lower bound. It indicates that the regret bound of Algorithm 1.2 cannot be improved asymptotically with respect to its order in T. In other words, our algorithm is asymptotically optimal.

Theorem 1.4. *(cf. Theorem 2.2) Let $k \in \mathbb{N}^*$, $R > 0$ such that*

$$2 \leq k \leq \left\lfloor \left(\frac{RT^{\frac{1}{4}}}{6 \log T^{\frac{1}{4}}} \right)^{\frac{d}{d+1}} \right\rfloor, \tag{1.16}$$

where $\lfloor x \rfloor$ represents the largest integer that is smaller than x. If T is sufficient large, then the procedure described in Algorithm 1.2 with penalized loss ℓ_{pen} satisfies the following

$$k\sqrt{T\ln T}\left(1 - \frac{2}{T}\left[1 + \frac{k-1}{2k^2}\right]\right) \leq \mathcal{V}_T(k) \leq \text{const.} \times k\sqrt{T\ln T}. \tag{1.17}$$

The proof of left hand side of (1.17) depends on a probabilistic method: we lower bound the supremum of the regret over all individual sequences $(x_t)_{1:T}$ by the expected regret on a suitably chosen i.i.d random sequence $(X_t)_{1:T}$. The distribution of $(X_t)_{1:T}$ is a discrete distribution inspired from the work of Bartlett *et al.* (1998). The proof of right hand side of (1.17) is similar to that of Theorem 1.3. We see that the lower bound is asymptotically of the order of $\sqrt{T\ln T}$ indicating that our procedure is asymptotically optimal in the minimax sense.

1.3.5 Implementation for online clustering

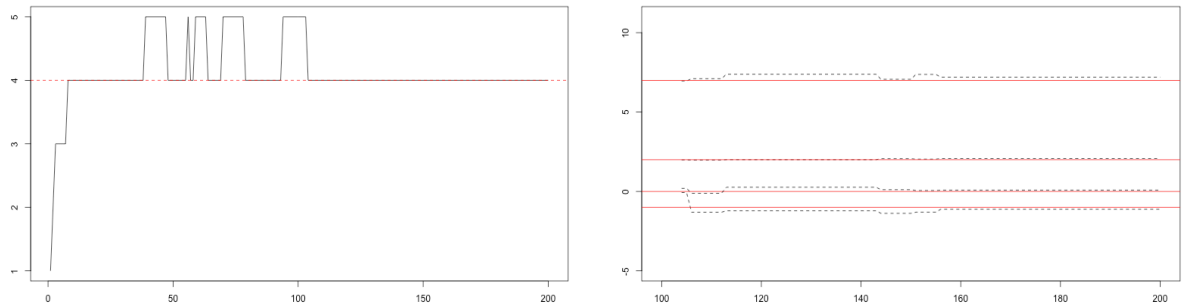
The randomized predictions $\hat{\mathbf{c}}_t, t = 1, \ldots, T$ in both Algorithm 1.2 and Algorithm 2.2 are drawn from the quasi-posterior defined on a complex space of varying dimensions $\mathscr{C} = \cup_{k=1}^{p} \mathbb{R}^{dk}$. Hence at each t, the partition $\hat{\mathbf{c}}_t$ can be of different dimension. For its ability to cope with transdimensional moves, we resort to the RJMCMC introduced in section 1.2.2, coupled with ideas from the Guedj and Alquier (2013) and Subspace Carlin and Chib algorithm proposed by Dellaportas *et al.* (2002) and Petralias and Dellaportas (2013). More precisely, we set specifically the parameter space Θ equaling to the partition space $\mathscr{C}$, and ρ to Gibbs quasi posterior $\hat{\rho}_t$ in online clustering. Let us denote by $\mathbf{c}^{(n)}$ the current state of partition at n-th iteration of RJMCMC. One should notice that $\mathbf{c}^{(n)}$ in fact depends on t since the target is $\hat{\rho}_t$ at time t. However since the schema in the RJMCMC for target $\hat{\rho}_t$ is the same for $t = 1, 2, \ldots, T$, we omit this dependence of $\mathbf{c}^{(n)}$ on t for writing convenience. Denote by $k^{(n)}$ the number of cells. We can generate a proposal $\mathbf{c}'$ by following the steps introduced in section 1.2.2 with specific transition probability $\mathrm{p}_{k^{(n)}k'}$, density $q_{k^{(n)}k'}$ and $g_{k^{(n)}k'}$. Firstly, we choose the transition probability $\mathrm{p}_{k^{(n)}k'}$ as

$$\mathrm{p}_{k^{(n)}k'} = \begin{cases} \frac{1}{3} & \text{if} \quad k^{(n)} \in \{2, \ldots, p\} \quad \text{and} \quad k' \in \left\{k^{(n)} - 1, k^{(n)}, k^{(n)} + 1\right\}, \\ \frac{1}{2} & \text{if} \quad k^{(n)} = 1 \quad \text{and} \quad k' \in \{1, 2\}, \\ \frac{1}{2} & \text{if} \quad k^{(n)} = p \quad \text{and} \quad k' \in \{p-1, p\}, \\ 0 & \text{otherwise.} \end{cases} \tag{1.18}$$

This transition $\mathbf{p}_{k^{(n)}k'}$ will only propose new number k' of cells within the neighborhood of current $k^{(n)}$. This choice of $\mathbf{p}_{k^{(n)}k'}$ that favors local transition of number of cells can lead to a comparatively higher acceptance probability in RJMCMC since otherwise, the ratio between $\hat{\rho}_t(\mathbf{c}')$ and $\hat{\rho}_t(\mathbf{c}^{(n)})$ would be small due to the large variation of number of cells between $\mathbf{c}' \in \mathbb{R}^{dk'}$ and $\mathbf{c}^{(n)} \in \mathbb{R}^{dk^{(n)}}$. Then, an auxiliary vector v is sampled from a rescaled Student distribution $\rho_{k'}(\cdot)$ on $\mathbb{R}^{dk'}$ whose density is detailed in (2.14) of Chapter 2. Finally, we set the one-to-one and differentiable function as $g_{k^{(n)}k'} : (\mathbf{c}^{(n)}, v) \mapsto (v, \mathbf{c}^{(n)})$. The proposal $\mathbf{c}'$ is then accepted with probability α defined in Algorithm 2.3 of Chapter 2 where a detailed implementation is given.

Numerical experiments

We show here a simple example illustrating the performance of our implementation in batch setting. For more numerical experiments in both batch setting and online setting, one can refer to Chapter 2. The observations in batch setting are sampled from four bivariate Gaussian distributions with identity covariance matrix, whose mean vectors are respectively $(0,0),(4,1),(0,7),(5,2)$. Each observation is uniformly drawn from one of the four groups and the number of observations is $n = 200$. Figure 1.1 illustrates respectively the convergence of RJMCMC in both the number of clusters and position of centers. The left plot includes number $k^{(n)}$ of clusters along the 200 iterations (solid black line) and true number of clusters (dashed red line); The right plot presents the values (*i.e.,* $0, -1, 7, 2$, solid red line) of second coordinate of mean vectors of four Gaussian distributions and that of second coordinate of all centers in $\mathbf{c}^{(n)}$ along iterations. It is noticed that the chain converges to the true values slightly after 100 iterations.



(a) Number of clusters.

(b) Coordinate of centers.

Figure 1.1 – Typical RJMCMC output. (a) $k^{(n)}_{1:N}$, number of clusters along the 200 iterations (a solid black line). The true number 4 of clusters in this example (a dashed red line). (b) values of second coordinate of all centers in $\mathbf{c}^{(n)}_{1:N}$ along the 200 iterations (a dashed black line). The true values of second coordinate of mean vectors of 4 Gaussian distributions.

In addition, instead of choosing arbitrarily an initial value $\mathbf{c}^{(0)} \in \mathscr{C}$, one can choose $\mathbf{c}^{(0)} = \hat{\mathbf{c}}_{t-1}$ *i.e.,* setting initial value for RJMCMC at round t (corresponding to sampling from $\hat{\rho}_t$) to the value of randomized partition $\hat{\mathbf{c}}_{t-1}$ at round $t-1$. In practice, this warm start helps to accelerate the convergence of the chain. In Chapter 2, we compare the running times of our algorithm with several competitors in the online setting. It is found that

our method, equipped with a warm start, has a better performance in running time (cf. Table 2.2 in Chapter 2).

1.3.6 Sequential principal curves framework

In Chapter 3, we proceed to another setting: sequential principal curves which can be seen as a generalization of online clustering since we seeks to "summarize" online data by a continuous curve rather than several discrete centers. This is useful especially when the sequence of data shows a certain curve pattern. We introduce first the definition of principal curve in classical statistical setting and then consider sequential principal curves (*i.e.,* principal curves in online setting). Our task is to construct an algorithm proposing sequential principal curves at each t such that its regret bound is sublinear in T.

Principal curves can be regarded as a nonlinear generalization of first principal component. Different from clustering that several centers are used to summarize the data, the goal of principal curves is to summarize the data by a curve that passes "in the middle of data", as illustrated by Figure 1.2.

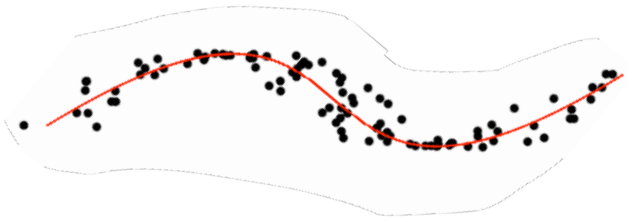


Figure 1.2 – An example of principal curve.

The original definition of principal curve in classical statistical setting dates back to Hastie and Stuetzle (1989). It is defined as a parameterized curve which does not intersect itself, has finite length inside any bounded subset of $\mathbb{R}^d$ and is self-consistent (we refer the reader to Hastie and Stuetzle, 1989 for more details). However, the existence of principal curve is guaranteed only for a very limited number of distributions. Kégl (1999) proposed a new concept of principal curves which ensures the existence for a large class of distributions. Before giving how principal curve is defined in Kégl (1999), let us first give give some notation: denote by X a random vector in $\mathbb{R}^d$ and by $\mathbf{f}(s) = (f_1(s), \ldots, f_d(s)) \in \mathbb{R}^d$ a parameterized curve in $\mathbb{R}^d$, where $s \in I$ and $I \subset \mathbb{R}$ is a closed interval. Denoted by $\mathscr{F}_L$ a class of continuous curves whose length is smaller than $L > 0$. For $\mathbf{f} \in \mathscr{F}_L$, the expected squared distance $\Delta(\mathbf{f})$ of $\mathbf{f}$ is defined by

$$\Delta(\mathbf{f}) = \mathbb{E}[\Delta(\mathbf{f}, X)] = \mathbb{E}\left[\inf_{s \in I} \|\mathbf{f}(s) - X\|_2^2\right],$$

where $\|\cdot\|_2^2$ is the ℓ_2 norm in $\mathbb{R}^d$. To maintain consistency with notations in previous investigations of principal curves, we use $\Delta(\mathbf{f}, X)$ (rather than $\ell(\mathbf{f}, X)$) to denote the loss (distance) between $\mathbf{f}$ and X.

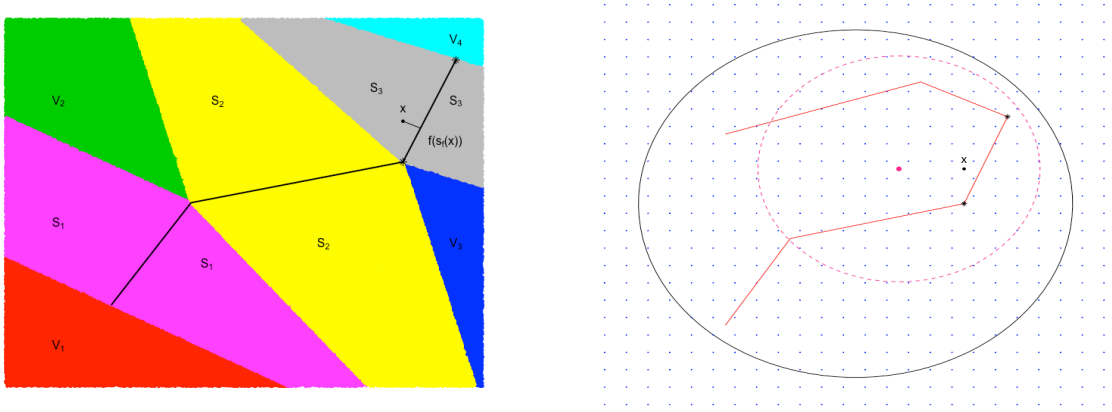
The new definition of principal curves $\mathbf{f}^\star$ by Kégl (1999) are defined as curves minimizing the expected squared distance over $\mathscr{F}_L$, namely,

$$\mathbf{f}^\star \in \operatorname*{arg\,inf}_{\mathbf{f} \in \mathscr{F}_L} \Delta(\mathbf{f}).$$

It is proved that if $\mathbb{E}\|X\|_2^2 < \infty$, $\mathbf{f}^\star$ always exists but may be not unique. It is seen that the existence of principal curve under Kégl's definition is guaranteed for class of distributions on which very loosen constraint is imposed.

To approximate unknown $\mathbf{f}^\star$ (since the distribution of X is unknown in practice), Kégl (1999) uses classes of polygonal lines with limited length. He proves that the segment of optimal polygonal line approximating $\mathbf{f}^\star$ is proportional to $n^{1/3}$ (the number of observations). Biau and Fischer (2012) also use classes of polygonal lines to estimate $\mathbf{f}^\star$ and extend the results of Kégl by choosing an optimal segments from a model selection point of view.

In Chapter 3, we adopt the notions of classes of polygonal lines defined in Biau and Fischer (2012). More precisely, let $B(\mathbf{c}, R)$ stands for the ℓ_2-ball centered in $\mathbf{c} \in \mathbb{R}^d$ with radius $R > 0$. Let $\mathcal{Q}_\delta$ be a grid over $B(\mathbf{0}, \sqrt{d}R)$, *i.e.*, $\mathcal{Q}_\delta = B(\mathbf{0}, \sqrt{d}R) \cap \Gamma_\delta$ where Γ_δ is a lattice in $\mathbb{R}^d$ with spacing $\delta > 0$. Let $L > 0$ and define for each $k \in [\![1, p]\!]$ the collection $\mathcal{F}_{k,L}$ of polygonal lines $\mathbf{f}$ with k segments whose vertices are in $\mathcal{Q}_\delta$ and whose length is smaller than L. Figure 1.3b presents an example of polygonal line with 5 segments and with vertices on $\mathcal{Q}_\delta$ (blue points, $\delta = 1$) in $\mathbb{R}^2$, where the black solid circle represents $B(\mathbf{0}, 10)$. In addition, denote by $\mathcal{F}_p = \cup_{k=1}^p \mathcal{F}_{k,L}$ all polygonal lines whose vertices are in $\mathcal{Q}_\delta$ and whose length is at most L. Finally, let $\mathcal{K}(\mathbf{f})$ denote the number of segments of $\mathbf{f} \in \mathcal{F}_p$.



(a) partition step (b) local greedy search step

Figure 1.3 – (a) a Voronoi-like partition given a polygonal line (black solid line); (b) a local greedy search region (blue points inside pink dashed circle), where the pink dashed circle is a ℓ_2 ball with center $\bar{\mathcal{N}}_t(x)$ and with radius $\mathcal{D}(\mathcal{N}_t(x))$; x is an observation in $\mathbb{R}^2$ and its $\mathcal{V}(x)$ is v_3 and v_4, represented by asterisks.

Similar to online clustering, our goal is now to learn a time-dependent curve $\hat{\mathbf{f}}_t \in \mathcal{F}_p$ which passes through the "middle" of available observations $(x_s)_{1:(t-1)}$ and whose performance is almost as good as the best "expert" curve in hindsight in $\mathcal{F}_p$, *i.e.*,

$$\sum_{t=1}^{T} \mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t, x_t)\right] \leq \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\substack{\mathbf{f} \in \mathcal{F}_p \\ \mathcal{K}(\mathbf{f}) = k}} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) + R_T(\mathbf{f}) \right\} \right\},$$

where $R_T(\mathbf{f})$ as small as possible, in particular sublinear in T.

1.3.7 Regret bounds for principal curves

Algorithms achieving the above goal can be obtained similarly as Algorithm 1.2 in online clustering by considering the Gibbs quasi-posterior $\hat{\rho}_{t+1}$ now defined on $\mathscr{F}_p$:

$$\hat{\rho}_{t+1}(\mathbf{f}) \propto \exp(-\lambda S_t(\mathbf{f}))\pi(\mathbf{f}),$$

where $\pi(\mathbf{f})$ is a prior on $\mathscr{F}_p$ and

$$S_t(\mathbf{f}) = S_{t-1}(\mathbf{f}) + \Delta(\mathbf{f}, x_t) + \frac{\lambda}{2}\big(\Delta(\mathbf{f}, x_t) - \Delta(\hat{\mathbf{f}}_t, x_t)\big)^2.$$

If an RJMCMC implementation is considered again for sampling randomized predictor from $\hat{\rho}_{t+1}$, it may take much longer time in this setting than in online clustering setting since calculation of distance of a point to a polygonal line is more complicated: not only distances of a point to all vertices of polygonal but also that to all line segments should be computed. Therefore, in what follows, we shall rather consider a different procedure linked to the mode of Gibbs quasi-posterior. At first, a simple calculation tells us that the mode of quasi-posterior $\hat{\rho}_{t+1}$ equals to

$$\arg\min_{\mathbf{f}\in\mathscr{F}_p}\left\{\sum_{s=1}^{t}\Delta(\mathbf{f}, x_s) + \frac{\lambda}{2}\sum_{s=1}^{t}\big(\Delta(\mathbf{f}, x_t) - \Delta(\hat{\mathbf{f}}_t, x_t)\big)^2 + \frac{\ln\pi(\mathbf{f})}{\lambda}\right\}, \tag{1.19}$$

which contains a cumulative loss $\sum_{s=1}^{t}\Delta(\mathbf{f}, x_s)$ of prediction $\mathbf{f}$ in the first t rounds, a term $\frac{\lambda}{2}\sum_{s=1}^{t}\big(\Delta(\mathbf{f}, x_t) - \Delta(\hat{\mathbf{f}}_t, x_t)\big)^2$ controlling the variance of prediction $\mathbf{f}$ to past predictions $\hat{\mathbf{f}}_s, s \le t$ and a term $\ln\pi(\mathbf{f})$ which can be regarded as a penalty function of the complexity of $\mathbf{f}$ if π is well chosen. If we regard each $\mathbf{f}\in\mathscr{F}_p$ as an expert giving constant advice, then (1.19) means to find a best expert. This mode hence has a flavor of following the best expert or the perturbed leader in the setting of prediction with experts (see Hutter and Poland, 2005 and Cesa-Bianchi and Lugosi, 2006, Chapters 3 and 4). Hence it makes us to consider the following different procedure: at each t, the forecaster $\hat{\mathbf{f}}_t$ is given by

$$\begin{aligned}
\hat{\mathbf{f}}_1 &= \arg\inf_{\mathbf{f}\in\mathscr{F}_p}\left\{\frac{1}{\eta}(h(\mathbf{f}) - z_{\mathbf{f}})\right\},\\
\hat{\mathbf{f}}_t &= \arg\inf_{\mathbf{f}\in\mathscr{F}_p}\left\{\sum_{s=1}^{t-1}\Delta(\mathbf{f}, x_s) + \frac{h(\mathbf{f})}{\eta} - \frac{z_{\mathbf{f}}}{\eta}\right\}, \quad t \ge 2,
\end{aligned} \tag{1.20}$$

where $h(\mathbf{f})$ is a positive penalty function of $\mathbf{f}\in\mathscr{F}_p$, $z_{\mathbf{f}}$ are i.i.d samples of distribution $\pi(z) = e^{-z}\mathbb{1}_{\{z>0\}}$ and $\eta > 0$ is a parameter controlling the variance of algorithm. In this procedure, we measure the complexity of $\mathbf{f}$ directly by a penalty function h. It is seen in (1.20) that at each time, our prediction of polygonal line is the one minimizing the sum of cumulative loss, a penalty term and a perturbed value. This penalty terms enables our algorithm to avoid overfitting since otherwise it would always choose the most complicated polygonal line. Moreover, the perturbed value $z_{\mathbf{f}}$ enables us to choose at each time a perturbed best "expert" (rather than the strict best one). It can help us to avoid a trivial regret bound of the order T, as indicated in chapter 4, Cesa-Bianchi and Lugosi (2006).

Theorem 1.5. *(cf. Theorem 3.1) The procedure satisfies, for any sequence* $(x_t)_{1:T} \in B(\mathbf{0}, \sqrt{d}R)$ *with* $R > 0$

$$\sum_{t=1}^{T}\mathbb{E}_{\pi_\eta}\left[\Delta(\hat{\mathbf{f}}_t, x_t)\right] \le (1 + c_0(\mathrm{e}-1)\eta)S_{T,h,\eta} + \frac{1 + c_0(\mathrm{e}-1)\eta}{\eta}\left(1 + \ln\sum_{\mathbf{f}\in\mathscr{F}_p}\mathrm{e}^{-h(\mathbf{f})}\right),$$

where $c_0 = d(2R+\delta)^2$ *and*

$$S_{T,h,\eta} = \inf_{k\in[\![1,p]\!]} \left\{ \inf_{\substack{\mathbf{f}\in\mathscr{F}_p \\ \mathscr{K}(\mathbf{f})=k}} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f},x_t) + \frac{h(\mathbf{f})}{\eta} \right\} \right\}.$$

The expectation of cumulative loss of polygonal lines $\hat{\mathbf{f}}_1,\dots,\hat{\mathbf{f}}_T$ is upper-bounded by the smallest penalized cumulative loss over all $k \in \{1,\dots,p\}$ up to a multiplicative factor $(1+c_0(\mathrm{e}-1)\eta)$ and an additional term $\big((1+c_0(\mathrm{e}-1)\eta)/\eta\big)\Big(1+\ln\sum_{\mathbf{f}\in\mathscr{F}_p}\mathrm{e}^{-h(\mathbf{f})}\Big)$. The multiplicative factor before $S_{T,h,\eta}$ can be made arbitrarily close to 1 by choosing an η small enough. In Chapter 3, we detail the choice of penalty function h which is advocated in Barron *et al.* (1999), Birgé and Massart (2007) and Biau and Fischer (2012), and under this choice of penalty function, $\Big(1+\ln\sum_{\mathbf{f}\in\mathscr{F}_p}\mathrm{e}^{-h(\mathbf{f})}\Big)$ can be non-positive.

Theorem 1.6. *(cf. Corollary 3.1) Under the assumptions of Theorem 1.5, let* $\eta = \min\left\{\frac{1}{d(2R+\delta)^2}, \sqrt{\frac{c_1p+c_2L+c_3}{c_0(\mathrm{e}-1)\inf_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^{T}\Delta(\mathbf{f},x_t)}}\right\}$*, then, with a penalty function* h *advocated by Biau and Fischer (2012), we have*

$$\sum_{t=1}^{T}\mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t,x_t)\right] \le \inf_{k\in[\![1,p]\!]}\left\{\inf_{\substack{\mathbf{f}\in\mathscr{F}_p \\ \mathscr{K}(\mathbf{f})=k}}\left\{\sum_{t=1}^{T}\Delta(\mathbf{f},x_t) + \sqrt{c_0(\mathrm{e}-1)r_{T,k,L}}\right\}\right\} + \sqrt{c_0(\mathrm{e}-1)r_{T,p,L}} + 2\mathrm{e}c_0(c_1p+c_2L+c_3),$$

where $c_0 = d(2R+\delta)^2$*;* $r_{T,k,L} = \inf_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^{T}\Delta(\mathbf{f},x_t)(c_1k+c_2L+c_3)$*;* c_1,c_2,c_3 *are constants depending on* R,d,δ *(we refer to Lemma 3.3 for their explicit form).*

The regret of our procedure is at most of the order of $\sqrt{T}$ since $r_{T,k,L} \le c_0(c_1k+c_2L+c_3)T$ for all $k = 1,2,\dots,p$. However we see that the optimal value for η depends on $\inf_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^{T}\Delta(\mathbf{f},x_t)$ which is obviously unknown a priori in practice. To make the above procedure more practical, we provide an adaptive refinement of it.

1.3.8 Adaptive regret bounds for principal curves

Noticing that $\Delta(\mathbf{f},x_t)$ is uniformly bounded by c_0 is $x_t \in B(\mathbf{0},\sqrt{d}R)$ and $\mathbf{f}\in\mathscr{F}_p$. Hence one has

$$\inf_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^{T}\Delta(\mathbf{f},x_t) \le c_0T.$$

Replacing $\inf_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^{T}\Delta(\mathbf{f},x_t)$ by its upper bound c_0T makes η depend on the time horizon T (*i.e.*, $\eta \overset{\Delta}{=} \eta_T$). It hence allows us to choose an adaptive η_t similarly as the adaptive choice of λ_t in online clustering. More precisely, an adaptive version of (1.20) is

$$\hat{\mathbf{f}}_1 = \operatorname*{arg\,inf}_{\mathbf{f}\in\mathscr{F}_p}\left\{\frac{1}{\eta_0}(h(\mathbf{f})-z_{\mathbf{f}})\right\},$$
$$\hat{\mathbf{f}}_t = \operatorname*{arg\,inf}_{\mathbf{f}\in\mathscr{F}_p}\left\{\sum_{s=1}^{t-1}\Delta(\mathbf{f},x_s) + \frac{h(\mathbf{f})}{\eta_{t-1}} - \frac{z_{\mathbf{f}}}{\eta_{t-1}}\right\}, \quad t\ge 2,$$

and we have a regret bound for it as follows:

Theorem 1.7. *(cf. Theorem 3.2) Under the assumptions of Theorem 1.6, if*

$$\eta_0 = \frac{1}{c_0}, \qquad \eta_t = \min\left\{\frac{1}{c_0}, \frac{\sqrt{c_1 p + c_2 L + c_3}}{c_0\sqrt{(\mathrm{e}-1)t}}\right\}, \qquad t \geq 1,$$

then the above adaptive procedure satisfies

$$\sum_{t=1}^{T} \mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t, x_t)\right] \leq \inf_{k\in[\![1,p]\!]} \left\{ \inf_{\substack{\mathbf{f}\in\mathscr{F}_p \\ \mathscr{K}(\mathbf{f})=k}} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) + c_0\sqrt{(\mathrm{e}-1)T(c_1 k + c_2 L + c_3)} \right\} \right\}$$
$$+ 2c_0\sqrt{(\mathrm{e}-1)T(c_1 p + c_2 L + c_3)} + 3c_0(c_1 p + c_2 L + c_3).$$

The expected cumulative loss of polygonal lines $\hat{\mathbf{f}}_1, \ldots, \hat{\mathbf{f}}_T$ is upper-bounded by the minimal cumulative loss over all $k \in \{1, \ldots, p\}$, up to an additive term which is sublinear in T. The actual magnitude of this remainder term is $\sqrt{kT}$. When L is fixed, the number k of segments is a measure of complexity of the retained polygonal line. This bound therefore yields the same magnitude as (3.1) in Biau and Fischer (2012).

1.3.9 Implementation for sequential principal curves

To find $\hat{\mathbf{f}}_t$ at each time t, one needs to traverse all $\mathbf{f}$ in $\mathscr{F}_p$ whose cardinality is $\mathscr{O}(R/\delta)^{dp}$. A greedy search of $\mathbf{f}$ hence would be quite time-consuming and is hardly applicable in practice especially when d and p are large. Since the speed of an algorithm may have higher priority over its precision in practice, we instead resort to a strategy which can be described as follow: given a polygonal line $\hat{\mathbf{f}}_t \in \mathscr{F}_{k_t,L}$ with k_t segments, the new polygonal line $\hat{\mathbf{f}}_{t+1}$ is chosen, with a proportion $\epsilon \in (0,1)$, inside a neighborhood $\mathscr{U}(\hat{\mathbf{f}}_t) \subseteq \bigcup_{k_t-1}^{k_t+1} \mathscr{F}_{k,L}$ of $\mathscr{F}_{k_t,L}$. In other words, when x_t is available, the number k_{t+1} of segments of $\hat{\mathbf{f}}_{t+1}$ varies with k_t within at most 1 unit. Moreover, we can reduce the combinatories of $\hat{\mathbf{f}}_{t+1}$ to polygonal lines whose vertices only differ with that of $\hat{\mathbf{f}}_t$ in a neighborhood of new observation x_t. The neighborhood consideration can therefore on the one hand simplify the computation complexity and, on the other hand, control the variance of our sequential predictions which is appeared in the second term of (1.19). In addition, the consideration of updating principal curves in a neighbourhood with a proportion $\epsilon < 1$ enables our algorithm to still have the chance to explore the complement of neighbourhood.

More precisely, our procedure starts with a **partition** step which aims to identifying the "relevant" neighborhood of an observation $x \in \mathbb{R}^d$ with respect to a given polygonal line, and the proceeds with the definition of the **neighborhood** of an action $\mathbf{f}$. We then provide the full implementation and give a regret bound for it.

Partition For any polygonal line $\mathbf{f}$ with k segments, we denote by $\vec{\mathbf{V}} = (v_1, \ldots, v_{k+1})$ its vertices and by $s_i, i = 1, \ldots, k$ the line segments connecting v_i and v_{i+1}. In the sequel, we use $\mathbf{f}(\vec{\mathbf{V}})$ to represent the polygonal line formed by connecting consecutive vertices in $\vec{\mathbf{V}}$ if no confusion arises. Let $V_i, i = 1, \ldots, k+1$ and $S_i, i = 1, \ldots, k$ be the Voronoi partitions of $\mathbb{R}^d$ with respect to $\mathbf{f}$, *i.e.,* regions consisting of all points closer to vertex v_i or segment s_i. Figure 1.3a shows an example of Voronoi partition with respect to $\mathbf{f}$ with 3 segments.

Neighbourhood For any $x \in \mathbb{R}^d$, we define the neighbourhood $\mathscr{N}(x)$ with respect to $\mathbf{f}$ as the union of all Voronoi partitions whose closure intersects with two vertices connecting the projection $\mathbf{f}(s_{\mathbf{f}}(x))$ of x to $\mathbf{f}$. For example, for the point x in Figure 1.3a,

its neighbourhood $\mathcal{N}(x)$ is the union of S_2, V_3, S_3 and V_4. In addition, let $\mathcal{N}_t(x) = \{x_s \in \mathcal{N}(x), s = 1, \ldots, t.\}$ be the set of observations $x_{1:t}$ belonging to $\mathcal{N}(x)$ and $\bar{\mathcal{N}}_t(x)$ be its average. Let $\mathcal{D}(M) = \sup_{x,y \in M} ||x - y||_2$ denote the diameter of set $M \subset \mathbb{R}^d$. We finally define the local grid $\mathcal{Q}_{\delta,t}(x)$ of $x \in \mathbb{R}^d$ at time t as

$$\mathcal{Q}_{\delta,t}(x) = B\left(\bar{\mathcal{N}}_t(x), \mathcal{D}\left(\mathcal{N}_t(x)\right)\right) \cap \mathcal{Q}_{\delta}.$$

We can finally proceed to the definition of the neighbourhood $\mathcal{U}(\hat{\mathbf{f}}_t)$ of $\hat{\mathbf{f}}_t$. Assume $\hat{\mathbf{f}}_t$ has $k_t + 1$ vertices $\vec{\mathbf{V}} = (\underbrace{v_{1:i_t-1}}_{(i)}, \underbrace{v_{i_t:j_t-1}}_{(ii)}, \underbrace{v_{j_t:k_t+1}}_{(iii)})$, where vertices of (ii) belong to $\mathcal{Q}_{\delta,t}(x_t)$ while those of (i) and (iii) do not. The neighbourhood $\mathcal{U}(\hat{\mathbf{f}}_t)$ consists of $\mathbf{f}$ sharing vertices $(i),(iii)$ with $\hat{\mathbf{f}}_t$, but can be equipped with different vertices (ii) in $\mathcal{Q}_{\delta,t}(x_t)$, *i.e.*,

$$\mathcal{U}(\hat{\mathbf{f}}_t) = \left\{\mathbf{f}(\vec{\mathbf{V}}), \quad \vec{\mathbf{V}} = \left(v_{1:i_t-1}, v_{1:m}, v_{j_t:k_t+1}\right)\right\},$$

where $v_{1:m} \in \mathcal{Q}_{\delta,t}(x_t)$ and m is given by

$$m = \begin{cases} j_t - i_t - 1 & \text{reduce segments by 1 unit,} \\ j_t - i_t & \text{same number of segments,} \\ j_t - i_t + 1 & \text{increase segments by 1 unit.} \end{cases}$$

Moreover, if we regard each $\mathbf{f}$ as an expert, the consideration of updating $\hat{\mathbf{f}}_{t+1}$ in the neighborhood $\mathcal{U}(\hat{\mathbf{f}}_t)$ is equivalent to assuming that not all experts are available all the times and that the set of available experts can vary at each time. This is a model known as "sleeping expert" (or actions) in prior work (Freund *et al.*, 1997, Auer *et al.*, 2003, Blum and Mansour, 2007, Kleinberg *et al.*, 2008). In this setting, let us denote by σ an ordering of $|\mathcal{F}_p|$ actions, and $\mathcal{A}_t$ an available subset of the actions at round t. We let $\sigma(\mathcal{A}_t)$ denote the highest ranked action in $\mathcal{A}_t$. In addition, for any action $\mathbf{f} \in \mathcal{F}_p$ we define the reward $r_{\mathbf{f},t}$ of $\mathbf{f}$ at round $t, t \geq 1$ by

$$r_{\mathbf{f},t} = c_0 - \Delta(\mathbf{f}, x_t).$$

It is clear that $r_{\mathbf{f},t} \in (0, c_0)$. The convention from losses to gains is done in order to facilitate the subsequent performance analysis. The reward of an ordering σ is the cumulative rewards of the selected action at each time, *i.e.*,

$$\sum_{t=1}^{T} r_{\sigma(\mathcal{A}_t),t},$$

and the reward of the best ordering is $\max_\sigma \sum_{t=1}^{T} r_{\sigma(\mathcal{A}_t),t}$ ($\mathbb{E}\left[\max_\sigma \sum_{t=1}^{T} r_{\sigma(\mathcal{A}_t),t}\right]$ when $\mathcal{A}_t$ is stochastic).

Our locally greedy algorithm that incorporates sleeping experts can be described as follows: when $t = 1$, we obtain $\hat{\mathbf{f}}_1$ as the first principal component based on a small dataset $(x_t)_{t=1,\ldots,t_0}$, we set in addition estimation $\hat{r}_{\mathbf{f},1} = r_{\mathbf{f},1}$ for all $\mathbf{f} \in \mathcal{F}_p$; For $t = 2, \ldots, T$, the available set from which the action $\hat{\mathbf{f}}_t$ will be chosen is $\mathcal{A}_t = \mathcal{U}(\hat{\mathbf{f}}_{t-1})\mathbb{1}_{\{I_t=0\}} + \mathcal{F}_p \mathbb{1}_{\{I_t=1\}}$, where I_t is a Bernoulli variable with parameter $0 < \epsilon < 1$. And

$$\hat{\mathbf{f}}_t = \hat{\sigma}^t(\mathcal{A}_t),$$

where $\hat{\sigma}^t$ is a descending ordering of all $\mathbf{f} \in \mathcal{F}_p$ according to their perturbed cumulative estimated reward till $t-1$: $\sum_{s=1}^{t-1} \hat{r}_{\mathbf{f},s} - \frac{1}{\eta_{t-1}} h(\mathbf{f}) + \frac{1}{\eta_{t-1}} z_{\mathbf{f}}, \mathbf{f} \in \mathcal{F}_p$ where $\hat{r}_{\mathbf{f},s}$ are estimations of reward $r_{\mathbf{f},s}$ whose exact values are given in Algorithm 3.3 in Chapter 3. Then we have

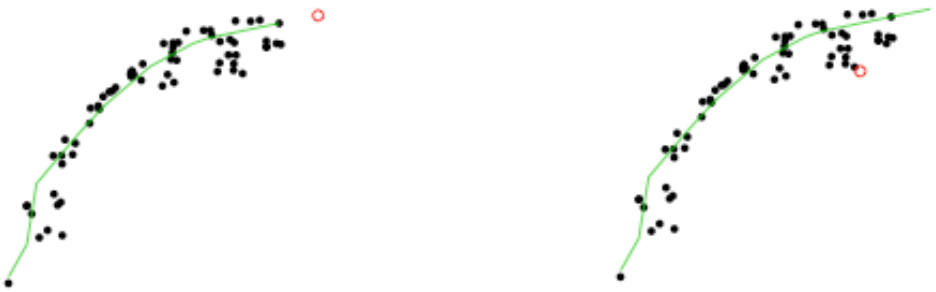
Theorem 1.8. *(cf. Theorem 3.3) Assume that $p > 6$ and T is sufficiently large with a proper choice of parameters, then the above locally greedy algorithm satisfies*

$$\sum_{t=1}^{T} \mathbb{E}\left[\Delta\left(\hat{\mathbf{f}}_t, x_{t+t_0}\right)\right] \leq \inf_{\mathbf{f}\in\mathscr{F}_p} \left[\sum_{t=1}^{T} \Delta\left(\mathbf{f}, x_{t+t_0}\right)\right] + \mathcal{O}(T^{\frac{3}{4}}).$$

We refer to Theorem 3.3 for explicit form of constants. Note that since we use a small set of data $(x_t)_{1:t_0}$ to obtain $\hat{\mathbf{f}}_1$, the loss of each polygonal line at time t is taken with respect to x_{t+t_0} rather than x_t. In addition, the algorithm achieves a regret of the order $T^{\frac{3}{4}}$ that is sublinear in T but is slower than $\sqrt{T}$. This is the price to pay for considering local update of principal curve and for the use of estimation of rewards in the algorithm.

Numerical experiments

We show in this part the potential of our algorithm on a toy example where the data set is constructed as follows: observations $\left\{x_t \in \mathbb{R}^2, t = 1, \ldots, 100\right\}$ are generated uniformly along a curve $y = 0.05 \times (x-5)^3$, $x \in [0, 10]$.



(a) t=65 (b) t = 67

Figure 1.4 – A principal curve $\hat{\mathbf{f}}_{t+1}$ (green) sequentially learned in consecutive time $t = 66, 67$ where the black dots represent data $x_{1:t}$ and the red one is the new observation x_{t+1}.

Figure 1.4 illustrates respectively two sequential principal curves $\hat{\mathbf{f}}_{t+1}$ (green line) in consecutive time $t = 66, 67$, where the black dots are observations $x_{1:t}$ and the red dot is the new observation x_{t+1}. We see that only vertex in the neighborhood of new observation is changed whereas other vertices of principal curve remain the same. For examples on real seismic data and daily commute data, we refer the reader to Section 3.5.

1.4 iAdvize context and results

1.4.1 Introduction of iAdvize

Nowadays, although the number of website visitors is huge, only a few (about 2%) of them would finally complete a transaction. In other words, the conversion rate is rather low for many commercial websites. iAdvize is a company that is committed to helping the

clients to increase their conversion rate. More precisely , it is a conversational platform that integrates different channels (*e.g.,* chat, call, social media etc) for the purpose of connecting visitors and consultants with expertise. This connection, which comes into being mostly via conversation, is quite useful for increasing the conversion rate since consultants can accompany their customers to solve difficulties that may impede the final transaction. For example, visitors may quit the website if they meet problems of payment at the final stage of purchase; visitors, although having an intention to buy a cellphone, may not complete their purchase since they hesitate between several cellphone marks and cannot make a decision due to the lack of expertise. With the help of experienced consultants, this kind of impeding is likely to be avoided.

iAdvize identifies those visitors that may need help by analyzing their behavior on the websites such as their browsing time on a particular page, the amount of money in their chart, the categories of goods that visitors are seeking etc. Once the behavior of a certain visitor is consistent with several prefixed criteria, then this visitor would be targeted and a channel enabling the visitor to connect to a consultant would be proposed. The online clustering algorithm introduced above is dedicated to ameliorate and develop the setting of criteria. It clusters firstly visitors into groups such that those with similar behavior are in the same group. Then behavior information within and across groups will be extracted and summarized to help the elaboration of criteria. In addition to online clustering algorithm which will be detailed in Chapter 2, we present in the sequel two other tasks that have been done within iAdvize.

1.4.2 Sentiment analysis for text messages

The first one is the sentiment analysis of social messages such as tweets and facebook comments. Social text messages can often reflect visitors' attitude (positive or negative) towards a certain subject. Knowing such attitude especially the negative one is important to our clients since consultants can interfere and give proper advises to visitors as soon as possible, hence help our clients to decrease the unsatisfactory rate of visitors and improve their services. In Chapter 4, we detail necessary procedures and techniques to fulfil the task of sentiment analysis. The pipeline includes pre-processing of text messages, vector representation of sentence and the use of classical supervised learning methods to predict the sentiment of social messages. Pre-processing of raw text messages is preliminary and important in sentiment analysis since raw messages, especially tweets and Facebook comments, often contain a lot of noisy such as misspelling and abbreviation of words (*e.g.,* bojour or bjr), special strings (*i.e.,* urls, email, telephone, tag characters) and conjugation of verbs or nouns depending on the tenses, subjects etc. These noisy might have a great impact on the prediction accuracy. Pre-processing aims therefore to reducing those noisy contained in the raw texts via normalization methods such as regular expression, replacement of special strings by particular tokens, lemmatisation and stemming. In addition, pre-processed text messages are needed to be transferred to numerical vectors such that classification methods can work. Typical procedures which seeks to fulfil this task include bag-of-words and tf-idf (*i.e.,* term frequency–inverse document frequency). The first one tries to represent a text (such as a sentence or a document) as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity while the second intends to reflect how important a word is to a document in a collection or corpus. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. In Chapter 4,

we use tf-idf as well as classical classification methods to compose a pipeline that can attain an accuracy of over **66%** in identifying the sentiment of French tweets and that it outperforms a sentiment classifier already existing in *pattern* library of python.

1.4.3 Neural networks and chatbot

The second work is related to the creation of conversational agent (also known as chatbot). Conversational agent is a system that can interact with customers by providing natural conversations indistinguishable from human. It is founded that, on certain scenarios such as delivery or payment, a large amount of conversations between visitors and consultants have very similar structure: it begins with similar exchanges such as greetings, then may proceed to the demanding the mode or reference of delivery, asking for additional question concerning the date or the fees of delivery and finally ends with salutation. In order to free the consultants such that they can have more time concentrating on more complicated and specific questions such as asking for details or description of certain product, a chatbot that is able to give automatically responses to simple and repeated questions is therefore preferred. In Chapter 4, we give firstly a general introduction to chatbots, ranging from their capability of handling different tasks to the underlying chatbot models. We also present neural networks from the most basic prototype to the state-of-the-art derivatives such as Long Short-Term Memory (LSTM) and Sequence to Sequence (seq2seq) model. These models are designed for treading sequential data. The very first and typical application of these models is to solve translation task between languages. It has also shown that they achieve a great potential in building a chatbot (Vinyal and Le, 2015). Therefore, we apply these model to build a chatbot in the delivery and payment scenario. However, the sequence to sequence based chatbot is often generative, in other words, given a question, the response is automatically generated word by word. It can lead to a possible problem that the final generated response may not be in correspondent with linguistic convention. To solve this, we design a score that can help us to choose the most consistent response from a prefixed set of responses, given a question. Its performance is shown at the end of Chapter 4.

2

A Quasi-Bayesian Perspective to Online Clustering

We consider the problem of online clustering on arbitrary bounded deterministic sequences. We introduce a new and adaptive online clustering algorithm relying on a quasi-Bayesian approach, with a dynamic estimation of unknown number of clusters. We prove both regret bounds for the algorithm and lower bound in the minimax sense. We also give an RJMCMC-flavored implementation called PACBO for which the convergence is guaranteed. We also illustrate the performance of the algorithm in both batch and online setting.

NOTA: This chapter is the full version (with extended proofs) of the paper Li *et al.* (2018), published in *Electronic Journal of Statistics*.

Contents

2.1 Introduction

The purpose of the present work is to generalize the framework of online learning introduced in Chapter 1 to the clustering problem, which has attracted attention from the machine learning and streaming communities. As an example, Guha *et al.* (2003), Barbakh and Fyfe (2008) and Liberty *et al.* (2016) study the so-called data streaming clustering problem. It amounts to cluster online data to a fixed number of groups in a single pass, or a small number of passes, while using little memory. From a machine learning perspective, Choromanska and Monteleoni (2012) aggregate online clustering algorithms, with a fixed number K of centers. The present chapter investigates a more general setting since we aim to perform online clustering with an unfixed and changing number K_t of centers. To the best of our knowledge, this is the first attempt of the sort in the literature. Let us stress that our approach only requires an upper bound p to K_t, which can be either a constant or an increasing function of the time horizon T.

Our main contribution is to generalize algorithms suited for supervised learning to the unsupervised setting. Our online clustering algorithm is adaptive in the sense that it does not require the knowledge of the time horizon T to be used and studied. The regret bounds that we obtain have a remainder term of magnitude $\sqrt{T \log T}$ and we prove that they are asymptotically minimax optimal.

The chapter is organised as follows. Section 2.2 introduces our notation and our online clustering procedure. Section 2.3 contains our mathematical claims, consisting in regret bounds for our online clustering algorithm. Remainder terms which are sublinear in T are obtained for a model selection-flavored prior. We also prove that these remainder terms are minimax optimal. We then discuss in Section 2.4 the practical implementation of our method, which relies on an adaptation of the RJMCMC algorithm to our setting. In particular, we prove its convergence towards the target quasi-posterior. The performance of the resulting algorithm, called PACBO, is evaluated on synthetic data. For the sake of clarity, proofs are postponed to Section 2.5. Finally, Section 2.6 contains an extension of our work to the case of a rescaled Student prior (inspired by Dalalyan and Tsybakov 2012a) along with additional numerical experiments.

2.2 A quasi-Bayesian perspective to online clustering

Let $(x_t)_{1:T}$ be a sequence of data, where $x_t \in \mathbb{R}^d$. Our goal is to learn a time-dependent parameter K_t and a partition of the observed points into K_t cells, for any $t = 1, \dots, T$. To this aim, the output of our algorithm at time t is a vector $\hat{\mathbf{c}}_t = (\hat{c}_{t,1}, \hat{c}_{t,2}, \dots, \hat{c}_{t,K_t})$ of K_t centers in $\mathbb{R}^{dK_t}$, depending on the past information $(x_s)_{1:(t-1)}$ and $(\hat{\mathbf{c}}_s)_{1:(t-1)}$. A partition is then created by assigning any point in $\mathbb{R}^d$ to its closest center. When x_t is newly revealed, the instantaneous loss is computed as

$$\ell(\hat{\mathbf{c}}_t, x_t) = \min_{1 \le k \le K_t} |\hat{c}_{t,k} - x_t|_2^2, \tag{2.1}$$

where $|\cdot|_2$ is the ℓ_2-norm in $\mathbb{R}^d$. Note that ℓ can be interpreted as follows: given a set of observations $x_1, x_2, \dots, x_{t-1}$, the partition $\hat{\mathbf{c}}_t$ aims to separating the space $\mathbb{R}^d$ into K_t cells and it allows us to classify future data into one of these cells. Moreover, it measures the error of our prediction at each time since $\hat{\mathbf{c}}_t$ only depends on the information before the arrival of x_t. In what follows, we investigate regret bounds for cumulative loss $\sum_{t=1}^{T} \ell(\hat{\mathbf{c}}_t, x_t)$ (or its expected version if $\ell(\hat{\mathbf{c}}_t, x_t)$ are random). This cumulative loss gives us a global

measurement of the ability (*i.e.,* the quality) of our algorithm in prediction. It can be seen as an analogue of the principle of studying the risk of an estimator in batch setting since an online algorithm can be converted to a batch one by considering a strategy which draws its prediction uniformly from sequential predictions or predicts as the average of sequential predictions (Littlestone, 1989, Helmbold and Warmuth, 1995, Dekel and Singer, 2006, Audibert, 2009, Gerchinovitz, 2011). Its risk then equals to the cumulative loss divided by the number of all sequential predictions. Given a measurable space Θ (embedded with its Borel σ-algebra), we let $\mathscr{P}(\Theta)$ denote the set of probability distributions on Θ, and for some reference measure ν, we let $\mathscr{P}_\nu(\Theta)$ be the set of probability distributions absolutely continuous with respect to ν. For any probability distributions $\rho, \pi \in \mathscr{P}(\Theta)$, the Kullback-Leibler divergence $\mathscr{K}(\rho,\pi)$ is defined as

$$\mathscr{K}(\rho,\pi) = \begin{cases} \int_\Theta \log\left(\frac{\mathrm{d}\rho}{\mathrm{d}\pi}\right)\mathrm{d}\rho & \text{when } \rho \in \mathscr{P}_\pi(\Theta), \\ +\infty & \text{otherwise.} \end{cases}$$

Note that for any bounded measurable function $h : \Theta \to \mathbb{R}$ and any probability distribution $\rho \in \mathscr{P}(\Theta)$ such that $\mathscr{K}(\rho,\pi) < +\infty$,

$$-\log \int_\Theta \exp(-h)\mathrm{d}\pi = \inf_{\rho\in\mathscr{P}(\Theta)} \left\{ \int_\Theta h \mathrm{d}\rho + \mathscr{K}(\rho,\pi) \right\}. \tag{2.2}$$

This result, which may be found in Csiszár (1975) and Catoni (2004, Equation 5.2.1), is critical to our scheme of proofs. Further, the infimum is achieved at the so-called Gibbs quasi-posterior $\hat{\rho}$, defined by

$$\mathrm{d}\hat{\rho} = \frac{\exp(-h)}{\int \exp(-h)\mathrm{d}\pi}\mathrm{d}\pi.$$

We now introduce the notation to our online clustering setting. Let $\mathscr{C} = \cup_{k=1}^{p} \mathbb{R}^{dk}$ for some integer $p \geq 1$. We denote by q a discrete probability distribution on the set $[\![1,p]\!] := \{1,\dots,p\}$. For any $k \in [\![1,p]\!]$, let π_k denote a probability distribution on $\mathbb{R}^{dk}$. For any vector of cluster centers $\mathbf{c} \in \mathscr{C}$, we define $\pi(\mathbf{c})$ as

$$\pi(\mathbf{c}) = \sum_{k\in[\![1,p]\!]} q(k)\mathbb{1}_{\{\mathbf{c}\in\mathbb{R}^{dk}\}}\pi_k(\mathbf{c}). \tag{2.3}$$

Note that (2.3) may be seen as a distribution over the set of Voronoi partitions of $\mathbb{R}^d$: any $\mathbf{c} \in \mathscr{C}$ corresponds to a Voronoi partition of $\mathbb{R}^d$ with at most p cells. In the sequel, we denote by $\mathbf{c} \in \mathscr{C}$ either a vector of centers or its associated Voronoi partition (or partition for writing convenience) of $\mathbb{R}^d$ if no confusion arises, and we denote by $\pi \in \mathscr{P}(\mathscr{C})$ a prior over $\mathscr{C}$. Let $\lambda > 0$ be some (inverse temperature) parameter. At each time t, we observe x_t and a random partition $\hat{\mathbf{c}}_{t+1} \in \mathscr{C}$ is sampled from the Gibbs quasi-posterior

$$\mathrm{d}\hat{\rho}_{t+1}(\mathbf{c}) \propto \exp\big(-\lambda S_t(\mathbf{c})\big)\mathrm{d}\pi(\mathbf{c}). \tag{2.4}$$

This quasi-posterior distribution will allow us to sample partitions with respect to the prior π defined in (2.3) and bent to fit past observations through the following cumulative loss

$$S_t(\mathbf{c}) = S_{t-1}(\mathbf{c}) + \ell(\mathbf{c},x_t) + \frac{\lambda}{2}\big(\ell(\mathbf{c},x_t) - \ell(\hat{\mathbf{c}}_t,x_t)\big)^2,$$

where the latter one is a variance term. It is essential to make the *online variance inequality* hold true for general loss ℓ with quasi-posterior distribution, *i.e.,* no constraint such as the convexity or boundedness is imposed on ℓ (as discussed in Audibert, 2009,

Section 4.2). $S_t(\mathbf{c})$ consists in the cumulative loss of $\mathbf{c}$ in the first t rounds and a term that controls the variance of the next prediction. Note that since $(x_t)_{1:T}$ is deterministic, no likelihood is attached to our approach, hence the terms "quasi-posterior" for $\hat{\rho}_{t+1}$ and "quasi-Bayesian" for our global method. The resulting estimate is a realization of $\hat{\rho}_{t+1}$ with a random number K_t of cells. This scheme is described in Algorithm 2.1. Note that this algorithm is an instantiation of Audibert's online SeqRand algorithm (Audibert, 2009, Section 4) to the special case of the loss defined in (2.1). However SeqRand does not account for adaptive rates $\lambda = \lambda_t$, as discussed in the next section.

Algorithm 2.1 The quasi-Bayesian online clustering algorithm

1: **Input parameters**: $p > 0, \pi \in \mathscr{P}(\mathscr{C})$, $\lambda > 0$ and $S_0 \equiv 0$
2: **Initialization**: Draw $\hat{\mathbf{c}}_1 \sim \pi = \hat{\rho}_1$
3: **For** $t \in [\![1, T]\!]$
4: Get the data x_t
5: Draw $\hat{\mathbf{c}}_{t+1} \sim \hat{\rho}_{t+1}(\mathbf{c})$ where $\mathrm{d}\hat{\rho}_{t+1}(\mathbf{c}) \propto \exp\big(-\lambda S_t(\mathbf{c})\big)\mathrm{d}\pi(\mathbf{c})$, and

$$S_t(\mathbf{c}) = S_{t-1}(\mathbf{c}) + \ell(\mathbf{c}, x_t) + \frac{\lambda}{2}\big(\ell(\mathbf{c}, x_t) - \ell(\hat{\mathbf{c}}_t, x_t)\big)^2.$$

6: **End for**

The randomness of partitions $\hat{\mathbf{c}}_{t+1}, t = 1, 2, \ldots, T$ comes from two sources: the first one is from the way $\hat{\mathbf{c}}_{t+1}$ is sampled and the second one originates from $\hat{\rho}_{t+1}$ which depends on all past randomized partitions $\hat{\mathbf{c}}_t, \hat{\mathbf{c}}_{t-1}, \ldots, \hat{\mathbf{c}}_1$, *i.e.*, $\hat{\rho}_{t+1}(\cdot) = \hat{\rho}_{t+1}(\cdot|\hat{\mathbf{c}}_t, \hat{\mathbf{c}}_{t-1}, \ldots, \hat{\mathbf{c}}_1)$. When $\hat{\mathbf{c}}_t, \hat{\mathbf{c}}_{t-1}, \ldots, \hat{\mathbf{c}}_1$ are fixed (*i.e.*, conditionally on $\hat{\mathbf{c}}_t, \hat{\mathbf{c}}_{t-1}, \ldots, \hat{\mathbf{c}}_1$), the quasi-posterior $\hat{\rho}_{t+1}(\cdot)$ is deterministic. In what follows, for any partition $h : \mathscr{C} \longleftrightarrow \mathbb{R}$, the notation $\mathbb{E}_{\hat{\rho}_{t+1}}[h(\hat{\mathbf{c}}_{t+1})]$ is used to represent the conditional expectation of $h(\hat{\mathbf{c}}_{t+1})$ (with respect to $\hat{\rho}_{t+1}$) on $\hat{\mathbf{c}}_t, \hat{\mathbf{c}}_{t-1}, \ldots, \hat{\mathbf{c}}_1$, *i.e.*, $\mathbb{E}_{\hat{\rho}_{t+1}}[h(\hat{\mathbf{c}}_{t+1})] \overset{\Delta}{=} \mathbb{E}_{\hat{\rho}_{t+1}}[h(\hat{\mathbf{c}}_{t+1}|\hat{\mathbf{c}}_t, \ldots, \hat{\mathbf{c}}_1)]$. Moreover, the notation $\mathbb{E}_{(\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_{t+1})}[h(\hat{\mathbf{c}}_{t+1})]$ denotes the expectation of $h(\hat{\mathbf{c}}_{t+1})$ with respect to the joint distribution of $\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_{t+1}$, for example, $\mathbb{E}_{(\hat{\rho}_1, \hat{\rho}_2)}[h(\hat{\mathbf{c}}_2)] = \int h(\hat{\mathbf{c}}_2)\mathrm{d}\hat{\rho}_2(\hat{\mathbf{c}}_2|\hat{\mathbf{c}}_1)\mathrm{d}\hat{\rho}_1(\hat{\mathbf{c}}_1)$. In addition, let $\mathbb{E}_{\mathbf{c}\sim\nu}$ stands for the expectation with respect to the distribution ν of $\mathbf{c}$ (abbreviated as $\mathbb{E}_\nu$ where no confusion is possible).

2.3 Minimax regret bounds

We start with the following pivotal result.

Proposition 2.1. *For any sequence $(x_t)_{1:T} \in \mathbb{R}^{dT}$, for any prior distribution $\pi \in \mathscr{P}(\mathscr{C})$ and any $\lambda > 0$, the procedure described in Algorithm 2.1 satisfies*

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) \leq \inf_{\rho \in \mathscr{P}_\pi(\mathscr{C})} \left\{ \mathbb{E}_{\mathbf{c}\sim\rho}\left[\sum_{t=1}^{T} \ell(\mathbf{c}, x_t)\right] + \frac{\mathscr{K}(\rho, \pi)}{\lambda} + \frac{\lambda}{2} \mathbb{E}_{(\hat{\rho}_1, \ldots, \hat{\rho}_T)} \mathbb{E}_{\mathbf{c}\sim\rho} \sum_{t=1}^{T} [\ell(\mathbf{c}, x_t) - \ell(\hat{\mathbf{c}}_t, x_t)]^2 \right\}.$$

Proposition 2.1 is a straightforward consequence of Audibert (2009, Theorem 4.6) who gave an upper bound for the expectation of the cumulative loss of randomized prediction functions in the regressing setting. We applied to the loss function defined in (2.1), the partitions $\mathscr{C}$, and any prior $\pi \in \mathscr{P}(\mathscr{C})$.

2.3.1 Preliminary regret bounds

In the following, we instantiate the regret bound introduced in Proposition 2.1. Distribution q in (2.3) is chosen as the following discrete distribution on the set $[\![1,p]\!]$

$$q(k) = \frac{\exp(-\eta k)}{\sum_{i=1}^{p} \exp(-\eta i)}, \quad \eta \geq 0. \tag{2.5}$$

When $\eta > 0$, the larger the number of cells k, the smaller the probability mass. Further, π_k in (2.3) is chosen as a product of k independent uniform distributions on ℓ_2-balls in $\mathbb{R}^d$:

$$\mathrm{d}\pi_k(\mathbf{c}, R) = \left(\frac{\Gamma\left(\frac{d}{2}+1\right)}{\pi^{\frac{d}{2}}} \right)^k \frac{1}{(2R)^{dk}} \left\{ \prod_{j=1}^{k} \mathbb{1}_{\{B_d(2R)\}}(c_j) \right\} \mathrm{d}\mathbf{c}, \tag{2.6}$$

where $R > 0$, Γ is the Gamma function and

$$B_d(r) = \left\{ x \in \mathbb{R}^d, |x|_2 \leq r \right\} \tag{2.7}$$

is an ℓ_2-ball in $\mathbb{R}^d$, centered in $0 \in \mathbb{R}^d$ with radius $r > 0$. Finally, for any $k \in [\![1,p]\!]$ and any $R > 0$, let

$$\mathscr{C}(k,R) = \left\{ \mathbf{c} = (c_j)_{j=1,2,\ldots,k} \in \mathbb{R}^{dk}, c_i \neq c_j, i \neq j, \text{such that } |c_j|_2 \leq R, \forall j \right\}.$$

Corollary 2.1. *For any sequence $(x_t)_{1:T} \in \mathbb{R}^{dT}$ and any $p \geq 1$, consider π defined by (2.3), (2.5) and (2.6) with $\eta \geq 0$ and $R \geq \max_{t=1,\ldots,T} |x_t|_2$. If $\lambda \geq (d+2)/(2TR^2)$, the procedure described in Algorithm 2.1 satisfies*

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) \leq \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\mathbf{c} \in \mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) + \frac{dk}{2\lambda} \log\left(\frac{8R^2\lambda T}{d+2} \right) + \frac{\eta}{\lambda} k \right\} + \left(\frac{\log p}{\lambda} + \frac{d}{2\lambda} + \frac{81\lambda T R^4}{2} \right),$$

Note that $\inf_{\mathbf{c} \in \mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t)$ is a non-increasing function of the number k of cells while the penalty is linearly increasing with k. Small values for λ (or equivalently, large values for R) lead to small values for k. The additional term induced by the complexity of $\mathscr{C} = \bigcup_{k=1,\ldots,p} \mathbb{R}^{dk}$ is $\log p$. A reasonable choice of λ would be such that $d/\lambda \log(\lambda T R^2/d + 2)$ and $\lambda T R^4$ are of the same order in T. The calibration $\lambda = (d+2)\sqrt{\log T}/(2\sqrt{T}R^2)$ yields a sublinear remainder term in the following corollary.

Corollary 2.2. *Under the previous notation with $\lambda = (d+2)\sqrt{\log T}/(2\sqrt{T}R^2)$, $R \geq \max_{t=1,\ldots,T} |x_t|_2$ and $T > 2$, the procedure described in Algorithm 2.1 satisfies*

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) \leq \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\mathbf{c} \in \mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) + \frac{2(d+\eta)R^2}{d+2} k \sqrt{T \log T} \right\} + \left(\frac{2R^2 \log p}{d+2} + \frac{dR^2}{d+2} + \frac{81(d+2)R^2}{4} \right) \sqrt{T \log T}. \tag{2.8}$$

Remark 1. *If we assume T and R are constants, the reason that λ is chosen to be of order of magnitude of d here, rather than of $\sqrt{d}$, is to guarantee that it satisfies the condition $\lambda \geq (d+2)/(2TR^2)$ in Corollary 2.1. However, if T is sufficiently large, e.g., $T \geq (d+2)^2/d$, then the choice $\lambda = \sqrt{d \log T}/(2\sqrt{T}R^2)$ will satisfy the condition and will make the right hand side of the above inequality grow linearly in $\sqrt{d}$ while keeping the order of magnitude for T and R.*

If we denote by $R_T(\mathbf{c})$ the regret of our algorithm withe respect to $\mathbf{c} \in \mathscr{C}$ in the first T rounds, *i.e.*,

$$R_T(\mathbf{c}) = \sum_{t=1}^{T} \ell(\hat{\mathbf{c}}_t, x_t) - \sum_{t=1}^{T} \ell(\mathbf{c}, x_t),$$

then the expectation $\mathbb{E}[R_T(\mathbf{c})]$ of the regret equals to $\mathbb{E}_{(\hat{\rho}_1,\dots,\hat{\rho}_T)}\left[\sum_{t=1}^{T} \ell(\hat{\mathbf{c}}_t, x_t)\right] - \sum_{t=1}^{T} \ell(\mathbf{c}, x_t)$ and Corollary 2.2 indicates that for any $\mathbf{c} \in \mathscr{C}$,

$$\mathbb{E}[R_T(\mathbf{c})] \leq c_{\eta,d,R}(|\mathbf{c}| + \log p)\sqrt{T \log T},$$

where $|\mathbf{c}|$ denotes the number of centers of $\mathbf{c}$ and $c_{\eta,d,R}$ is a constant depending on η, d, R. The above inequality shows that the expected regret of our algorithm with respect to $\mathbf{c}$ is bounded by a quantity equaling to the sum of the number $|\mathbf{c}|$ of centers and $\log p$ times $\sqrt{T \log T}$, up to a constant $c_{\eta,d,R}$. Hence the supremum $\sup_{\mathbf{c} \in \mathscr{C}} \mathbb{E}[R_T(\mathbf{c})]$ of the regret of our algorithm is at most $c_{\eta,d,R}(p + \log p)\sqrt{T \log T}$.

In addition, let us assume that the sequence $x_1, \dots, x_T$ is generated from a distribution with $k^\star \in [\![1, p]\!]$ clusters. We then define the expected cumulative loss (ECL) and oracle cumulative loss (OCL) as

$$\mathrm{ECL} = \sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1, \hat{\rho}_2, \dots, \hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t),$$

$$\mathrm{OCL} = \inf_{\mathbf{c} \in \mathscr{C}(k^\star, R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t).$$

Then Corollary 2.2 yields

$$\mathrm{ECL} - \mathrm{OCL} = \sup_{\mathbf{c} \in \mathscr{C}(k^\star, R)} \mathbb{E}[R_T(\mathbf{c})] \leq J k^\star \sqrt{T \log T}, \tag{2.9}$$

where J is a constant depending on d, R and $\log p$. In (2.9) the regret of our randomized procedure, defined as the difference between ECL and OCL is sublinear in T. However, whenever $k^\star > p$, we can deduce from Corollary 2.2 that

$$\begin{aligned} \sup_{\mathbf{c} \in \mathscr{C}(k^\star, R)} \mathbb{E}[R_T(\mathbf{c})] \leq \inf_{k \in [\![1,p]\!]} \Bigg\{ & \inf_{\mathbf{c} \in \mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) - \inf_{\mathbf{c} \in \mathscr{C}(k^\star, R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) \\ & + \frac{2(d+\eta)R^2}{d+2} k \sqrt{T \log T} \Bigg\} + \\ & \left(\frac{R^2(2 \log p + d)}{d+2} + \frac{81(d+2)R^2}{4} \right) \sqrt{T \log T}, \end{aligned}$$

where $\inf_{\mathbf{c} \in \mathscr{C}(k^\star, R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t)$ is the oracle cumulative loss (*i.e.,* OCL) with $k^\star$ clusters. If there exists a $k \in [\![1, p]\!]$ such that $\inf_{\mathbf{c} \in \mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t)$ is close to OCL, then our ECL is also close to OCL up to a term of order $k\sqrt{T \log T}$. However, if no such k exists, then the term $\frac{2(d+\eta)R^2}{d+2} k \sqrt{T \log T}$ starts to dominate, hence the quality of bound is deteriorated.

Finally, note that the dependency in k inside the braces on the right-hand side of (2.8) may be improved by choosing $\lambda = (d+2)\sqrt{p \log T}/(2\sqrt{T R^2})$ in Corollary 2.2. This allows

to achieve the optimal rate $\sqrt{k}$ instead of k, since $k/\sqrt{p} \leq \sqrt{k}$ for any $k \in [\![1,p]\!]$. However, this makes the last term in Corollary 2.2 of order of $\sqrt{pT\log T}$. Note that the effort to make the regret bound grow in $\sqrt{k}$, rather than $\sqrt{p}$ for $k \in [\![1,p]\!]$ may be achieved by using a similar strategy to the one of Wintenberger (2017), which introduces a recursive aggregation procedure with distinct learning rates for each expert in a finite set. Those learning rates are computed with a second order refinement of losses (or a linearized version when the loss is convex in its second argument) for each expert, at each time round. The regret of his strategy with respect to best aggregation of M finite experts is of the order of $\log M\sqrt{T}\log\log T$. However, the context for this procedure is not the same as ours, as we resort to the Gibbs quasi-posterior which is defined on $\mathscr{C}$, a continuous set. In addition, we focus on a single temperature parameter λ for the sake of computational complexity since the second order refinement requires the computation of the expectation of loss with respect to each expert in a finite set while, in our case, the "expert set" (*i.e.*, $\mathscr{C}$) is continuous, leading to the tedious computation of second order refinement.

2.3.2 Adaptive regret bounds

The time horizon T is usually unknown, prompting us to choose a time-dependent inverse temperature parameter $\lambda = \lambda_t$. We thus propose a generalization of Algorithm 2.1, described in Algorithm 2.2.

Algorithm 2.2 The adaptive quasi-Bayesian online clustering algorithm

1: **Input parameters**: $p > 0, \pi \in \mathscr{P}(\mathscr{C})$, $(\lambda_t)_{0:T} > 0$ and $S_0 \equiv 0$
2: **Initialization**: Draw $\hat{\mathbf{c}}_1 \sim \pi = \hat{\rho}_1$
3: **For** $t \in [\![1,T]\!]$
4: Get the data x_t
5: Draw $\hat{\mathbf{c}}_{t+1} \sim \hat{\rho}_{t+1}(\mathbf{c})$ where $\mathrm{d}\hat{\rho}_{t+1}(\mathbf{c}) \propto \exp\big(-\lambda_t S_t(\mathbf{c})\big)\mathrm{d}\pi(\mathbf{c})$, and

$$S_t(\mathbf{c}) = S_{t-1}(\mathbf{c}) + \ell(\mathbf{c},x_t) + \frac{\lambda_{t-1}}{2}\big(\ell(\mathbf{c},x_t) - \ell(\hat{\mathbf{c}}_t,x_t)\big)^2.$$

6: **End for**

This adaptive algorithm is supported by the following more involved regret bound.

Theorem 2.1. *For any sequence $(x_t)_{1:T} \in \mathbb{R}^{dT}$, any prior distribution π on $\mathscr{C}$, if $(\lambda_t)_{0:T}$ is a non-increasing sequence of positive numbers, then the procedure described in Algorithm 2.2 satisfies*

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\ldots,\hat{\rho}_t)}\ell(\hat{\mathbf{c}}_t,x_t) \leq \inf_{\rho\in\mathscr{P}_\pi(\mathscr{C})}\left\{\mathbb{E}_{\mathbf{c}\sim\rho}\left[\sum_{t=1}^{T}\ell(\mathbf{c},x_t)\right] + \frac{\mathscr{K}(\rho,\pi)}{\lambda_T} + \mathbb{E}_{(\hat{\rho}_1,\ldots,\hat{\rho}_T)}\mathbb{E}_{\mathbf{c}\sim\rho}\left[\sum_{t=1}^{T}\frac{\lambda_{t-1}}{2}[\ell(\mathbf{c},x_t) - \ell(\hat{\mathbf{c}}_t,x_t)]^2\right]\right\}.$$

The technique used in the proof of Theorem 2.1 is standard (Cesa-Bianchi and Lugosi, 2006, Györfi and Ottucsák, 2007 and Gerchinovitz, 2011), where we apply Jensen's inequality to control a telescopic formula.

If λ is chosen in Proposition 2.1 as $\lambda = \lambda_T$, the only difference between Proposition 2.1 and Theorem 2.1 lies on the last term of the regret bound. This term will be larger in the

adaptive setting than in the simpler non-adaptive setting since $(\lambda_t)_{0:T}$ is non-increasing. In other words, here is the price to pay for the adaptivity of our algorithm. However, a suitable choice of λ_t allows, again, for a refined result.

Corollary 2.3. *For any deterministic sequence $(x_t)_{1:T} \in \mathbb{R}^{dT}$, if q and π_k in (2.3) are taken respectively as in (2.5) and (2.6) with $\eta \geq 0$ and $R \geq \max_{t=1,\dots,T} |x_t|_2$, if $\lambda_t = (d+2)\sqrt{\log t}/\left(2\sqrt{t}R^2\right)$ for any $t \in [\![2,T]\!]$ and $\lambda_0 = \lambda_1 = 1$, then for $T \geq 5$ the procedure described in Algorithm 2.2 satisfies*

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\dots,\hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) \leq \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\mathbf{c} \in \mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) + \frac{2(d+\eta)R^2}{d+2} k\sqrt{T \log T} \right\} + \left(\frac{2R^2 \log p}{d+2} + \frac{dR^2}{d+2} + \frac{81(d+2)R^2}{2} \right) \sqrt{T \log T}.$$

Therefore, the price to pay for not knowing the time horizon T (which is a much more realistic assumption for online learning) is a multiplicative factor 2 in front of the term $\frac{81(d+2)R^2}{4}\sqrt{T \log T}$. This does not degrade the rate of convergence $\sqrt{T \log T}$.

In the next corollary, we use the doubling trick (Section 2.3, Cesa-Bianchi and Lugosi, 2006 and Cesa-Bianchi *et al.*, 2007) to show how can we overcome the difficulty when a priori bound R on the ℓ_2-norm of sequence $(x_t)_{1:T}$ is unknown.

Let us first denote by $R_0 = 1$, and for $t \geq 1$

$$R_t = \max_{s=1,\dots,t} 2^{\lceil \log_2(|x_s|_2) \rceil},$$

where $\lceil x \rceil$ represents the least integer greater than or equal to $x \in \mathbb{R}$. It is clear that $(R_t)_{t \geq 1}$ is non-decreasing and satisfies $\max_{s=1,\dots,t} |x_s|_2 \leq R_t \leq 2 \max_{s=1,\dots,t} |x_s|_2$, $t = 1, 2, \dots,$. We call epoch r, $r = 0, 1, \dots$, the sequence $(t_{r-1}+1, t_{r-1}+2, \dots, t_r)$ of time steps where the last step t_r is the time step $t = t_r$ when $R_t > R_{t_{r-1}}$ take places for the first time (we set conventionally $t_{-1} = 0$). Within each epoch $r \geq 0$, *i.e.,* for $t \in [t_{r-1}+1, \dots, t_r]$, let

$$\lambda_{r,t} = \frac{(d+2)\sqrt{\log t}}{2\sqrt{t}R_{t_{r-1}}^2}.$$

Let **Alg-R** be a prediction algorithm that runs Algorithm 2.2 in each epoch r with parameter $\lambda_{r,t}$, then we have the following result

Corollary 2.4. *For any deterministic sequence $(x_t)_{1:T} \in \mathbb{R}^{dT}$, if q and π_k in (2.3) are taken respectively as in (2.5) and (2.6) with $\eta \geq 0$, the regret of algorithm* **Alg-R** *satisfies*

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\dots,\hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) \leq \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\mathbf{c} \in \mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) + \frac{56(d+\eta)R^2}{3(d+2)} k\sqrt{T \log T} \right\} + \frac{28}{3} \left(\frac{2R^2 \log p}{d+2} + \frac{dR^2}{d+2} + \frac{81(d+2)R^2}{2} \right) \sqrt{T \log T} + \frac{112}{3} R^2,$$

where $R = \max_{t=1,\dots,T} |x_t|_2$.

Note that the price to pay for making our algorithm adaptive to unknown bound R is a multiplicator $\frac{28}{3}$ and an additional term $\frac{112}{3}R^2$ in the regret bound.

2.3.3 Minimax regret

This section is devoted to the study of the minimax optimality of our approach. The regret bound in Corollary 2.3 has a rate $\sqrt{T\log T}$, which is not a surprising result. Indeed, many online learning problems give rise to similar bounds depending also on the properties of the loss function. However, in the online clustering setting, it is legitimate to wonder whether the upper bound is tight, and more generally if there exists other algorithms which provide smaller regrets. The sequel answers both questions in a minimax sense.

Let us first denote by $|\mathbf{c}|$ the number of cells for a partition $\mathbf{c} \in \mathscr{C}$. We also introduce the following assumption.

Assumption $\mathscr{H}(s)$: *Let* $R > 0$ *and* $T \in \mathbb{N}^*$. *For a given* $s \in [\![1, p]\!]$, *we assume that the number of cells* $\left|\mathbf{c}^\star_{T,R}\right|$ *for partition* $\mathbf{c}^\star_{T,R}$ *defined by the following*

$$\mathbf{c}^\star_{T,R} = \operatorname*{arg\,min}_{\mathbf{c} \in \cup_{k=1}^{p} \mathscr{C}(k,R)} \left\{ \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) + |\mathbf{c}| \sqrt{T \log T} \right\}.$$

equals to s, i.e., $\left|\mathbf{c}^\star_{T,R}\right| = s$.

Since $(x_t)_{1:T}$ are uniformly bounded by R, $\mathbf{c}^\star_{T,R}$ always exists (Pollard, 1981). Note that several partitions may achieve the minimum. In that case, we adopt the convention that $\mathbf{c}^\star_{T,R}$ is any such partition with the smallest number of cells. Assumption $\mathscr{H}(s)$ means that $(x_t)_{1:T}$ could be well summarized by s cells since the minimum is reached for the partition $\mathbf{c}^\star_{T,R}$. We introduce the set

$$\omega_{s,R} = \left\{ (x_t) \text{ such that } \mathscr{H}(s) \text{ holds} \right\} \subseteq \mathbb{R}^{dT}.$$

For Algorithm 2.2, we have from Corollary 2.3 that

$$\sup_{(x_t) \in \omega_{s,R}} \left\{ \sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) - \inf_{\mathbf{c} \in \mathscr{C}(s,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) \right\} \leq c_1 \times s \sqrt{T \log T},$$

where c_1 is a constant depending on R, d, p (recall that they are respectively the bound on the ℓ_2-norm of sequence $(x_t)_{1:T}$, the dimension of data point and the maximum number of cells allowed for clustering).

Then for any $s \in \mathbb{N}^*$, $R > 0$, our goal is to obtain a lower bound of the form

$$\inf_{\hat{P}} \sup_{(x_t) \in \omega_{s,R}} \left\{ \sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) - \inf_{\mathbf{c} \in \mathscr{C}(s,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) \right\} \geq c_2 \times s \sqrt{T \log T},$$

where c_2 is some constant satisfying $c_2 \leq c_1$. The first infimum is taken over all forecaster's strategy $\hat{P} \stackrel{\Delta}{=} (\hat{\rho}_t)_{1:T}$ and the supremum is taken over all sequence $(x_t)_{1:T} \in \mathscr{X}^T$. More precisely, the strategy of forecaster is a sequence $\hat{\rho}_1, \hat{\rho}_2, \ldots$, of functions $\hat{\rho}_t : \mathscr{X}^{t-1} \longrightarrow \mathscr{D}$, where $\mathscr{D}$ is the set of all probability distribution over partition space $\mathscr{C}$. Notations $(\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_t)$ in the above expectation denote therefore probability distributions which are values of forecasting strategy applied on available data, *i.e.*, $\hat{\rho}_t = \hat{\rho}_t(x_1, x_2, \ldots, x_{t-1}), t = 1, \ldots, T$. Here, we made abuse of notations by dropping these dependencies for writing convenience. This formulation of minimax regret in online clustering can be regarded as

a counterpart of minimax defined in prediction with expert advice (section 2.3 of Cesa-Bianchi and Lugosi, 2006) and online regression (section 2.10 of Gerchinovitz, 2011). In fact, one may consider each Voronoi partition $\mathbf{c}$ as a constant expert. Next, we obtain

$$\inf_{\hat{P}} \sup_{(x_t)\in\omega_{s,R}} \left\{ \sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\ldots,\hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) - \inf_{\mathbf{c}\in\mathscr{C}(s,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) \right\}$$
$$\geq \inf_{\hat{P}} \mathbb{E}_{\mu^T} \left\{ \sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\ldots,\hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, X_t) - \inf_{\mathbf{c}\in\mathscr{C}(s,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, X_t) \right\} \mathbb{1}_{\{(X_t)\in\omega_{s,R})\}}, \quad (2.10)$$

where X_t, $t = 1,\ldots,T$ are i.i.d with distribution μ defined on $\mathbb{R}^d$ and μ^T stands for the joint distribution of $(X_1,\ldots,X_T)$. Unfortunately, in (2.10), since the infimum is taken over any distribution $(\hat{\rho}_t)$, the number of cells of each partition $\hat{\mathbf{c}}_t, t = 1,2,\ldots,T$ could be all larger than s if $s < p$. Hence, the left hand side of (2.10) could be arbitrarily small or even negative and the lower bound does not match the upper bound of Corollary 2.3. To handle this, we need to introduce a penalized term which accounts for the number of cells of each partition to the loss function ℓ. The upcoming theorem provides minimax results for an augmented value $\mathscr{V}_T(s)$ defined as

$$\mathscr{V}_T(s) = \inf_{\hat{P}} \sup_{(x_t)\in\omega_{s,R}} \left\{ \sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\ldots,\hat{\rho}_t)} \left(\ell(\hat{\mathbf{c}}_t, x_t) + \frac{\sqrt{\log T}}{\sqrt{T}} |\hat{\mathbf{c}}_t| \right) - \inf_{\mathbf{c}\in\mathscr{C}(s,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) \right\}. \quad (2.11)$$

In (2.11), we add a term which penalizes the number of cells of each partition. To capture the behavior (or asymptotic behavior) of $\mathscr{V}_T(s)$, we derive an upper bound for the penalized loss in (2.11). This is done in the following theorem which combines both upper and lower bound for the regret, hence proving that it is minimax optimal.

Theorem 2.2. *Let $s \in \mathbb{N}^*$, $R > 0$ such that*

$$2 \leq s \leq \left\lfloor \left(\frac{R T^{\frac{1}{4}}}{6 \log T^{\frac{1}{4}}} \right)^{\frac{d}{d+1}} \right\rfloor, \quad (2.12)$$

where $\lfloor x \rfloor$ represents the largest integer that is smaller than x. If T satisfies $T^{\frac{d+2}{2}} \geq 8R^{2d}\sqrt{\log T}$, then

$$s\sqrt{T\log T}\left(1 - \frac{2}{T}\left[1 + \frac{s-1}{2s^2}\right]\right) \leq \mathscr{V}_T(s) \leq \text{const.} \times s\sqrt{T\log T}. \quad (2.13)$$

The lower bound on $\mathscr{V}_T(s)/T$ is asymptotically of the order of $\sqrt{\log T}/\sqrt{T}$. Note that Bartlett *et al.* (1998) obtained a minimax lower bound in batch setting with less satisfying rate of $1/\sqrt{T}$, however holding with no restriction on the number of cells retained in the partition whereas our claim has to comply with (2.12). This is the price to pay for our additional $\sqrt{\log T}$ factor. Note however that this price is mild as s can tend to $+\infty$ whenever T or R does, casting our procedure onto the online setting where the time horizon is not assumed finite and the number of clusters evolves along time.

As a conclusion to the theoretical part of the manuscript, let us summarize our results. Regret bounds for Algorithm 2.1 are produced for our specific choice of prior π (Corollary 2.1) and with an involved choice of λ (Corollary 2.2). For the adaptive version Algorithm 2.2, the pivotal result is Theorem 2.1, which is instantiated for our prior in Corollary 2.3. Finally, the lower bound is stated in Theorem 2.2, proving that our regret bounds are minimax whenever the number of cells retained in the partition satisfies (2.12). We now move to the implementation of our approach.

2.4 The PACBO algorithm

Since direct sampling from the Gibbs quasi-posterior is usually not possible, we focus on a stochastic approximation in this section, called PACBO (available in the companion eponym R package from Li, 2016). Both implementation and convergence (towards the Gibbs quasi-posterior) of this scheme are discussed. This section also includes a short numerical experiment on synthetic data to illustrate the potential of PACBO compared to other popular clustering methods.

2.4.1 Structure and links with RJMCMC

In Algorithm 2.1 and Algorithm 2.2, it is required to sample at each t from the Gibbs quasi-posterior $\hat{\rho}_t$. Since $\hat{\rho}_t$ is defined on the massive and complex-structured space $\mathscr{C}$ (let us recall that $\mathscr{C}$ is a union of heterogeneous spaces), direct sampling from $\hat{\rho}_t$ is not an option and is much rather an algorithmic challenge. Our approach consists in approximating $\hat{\rho}_t$ through MCMC under the constraint of favouring local moves of the Markov chain. To do it, we shall use resort to Reversible Jump MCMC (Green, 1995), adapted with ideas from the Subspace Carlin and Chib algorithm proposed by Dellaportas *et al.* (2002) and Petralias and Dellaportas (2013). Since sampling from $\hat{\rho}_t$ is similar for any $t = 1, \dots, T$, the time index t is now omitted for the sake of brevity.

Let $(k^{(n)}, \mathbf{c}^{(n)})_{0 \leq n \leq N}$, $N \geq 1$ be the states of the Markov Chain of interest of length N, where $k^{(n)} \in [\![1, p]\!]$ and $\mathbf{c}^{(n)} \in \mathbb{R}^{dk^{(n)}}$. At each RJMCMC iteration, only local moves are possible from the current state $(k^{(n)}, \mathbf{c}^{(n)})$ to a proposal state $(k', \mathbf{c}')$, in the sense that the proposal state should only differ from the current state by at most one covariate. Hence, $\mathbf{c}^{(n)} \in \mathbb{R}^{dk^{(n)}}$ and $\mathbf{c}' \in \mathbb{R}^{dk'}$ may be in different spaces ($k' \neq k^{(n)}$). Two auxiliary vectors $v_1 \in \mathbb{R}^{d_1}$ and $v_2 \in \mathbb{R}^{d_2}$ with $d_1, d_2 \geq 1$ are needed to compensate for this dimensional difference, *i.e.*, satisfying the dimension matching condition introduced by Green (1995)

$$dk^{(n)} + d_1 = dk' + d_2,$$

such that the pairs $(v_1, \mathbf{c}^{(n)})$ and $(v_2, \mathbf{c}')$ are of analogous dimension. This condition is a preliminary to the detailed balance condition that ensures that the Gibbs quasi-posterior $\hat{\rho}_t$ is the invariant distribution of the Markov chain. The structure of PACBO is presented in Figure 2.1.

Let $\rho_{k'}(\cdot, \mathfrak{c}_{k'}, \tau_{k'})$ denote the product of k' rescaled Student distribution on $\mathbb{R}^{dk'}$

$$\rho_{k'}(\mathbf{c}, \mathfrak{c}_{k'}, \tau_{k'}) = \prod_{j=1}^{k'} \left\{ C_{\tau_{k'}}^{-1} \left(1 + \frac{|c_j - \mathfrak{c}_{k',j}|_2^2}{6\tau_{k'}^2} \right)^{-\frac{3+d}{2}} \right\} \mathrm{d}\mathbf{c}, \tag{2.14}$$

where $C_{\tau_{k'}}^{-1}$ denotes a normalizing constant. The rescaled Student distribution is a special version of d-multivariate Student distribution (as presented in Kotz and Nadarajah, 2004) by taking the degree of freedom $v = 3$ and $\Sigma = 2\tau_{k'}^2 I_d$, where $\tau_{k'} > 0$ is a scale parameter and I_d is the d-dimensional identity matrix, and $\mathfrak{c}_{k',j}, j = 1, \dots, k'$ are mean vectors in $\mathbb{R}^d$ (we refer to Section 2.6 for detailed definition of rescaled Student distribution). Let us now detail the proposal mechanism. First, a local move from $k^{(n)}$ to k' is proposed by choosing $k' \in [\![k^{(n)} - 1, k^{(n)} + 1]\!]$ with probability $q(k^{(n)}, \cdot)$. Next, choosing $d_1 = dk'$, $d_2 = dk^{(n)}$, we sample v_1 from $\rho_{k'}$ in (2.14). Finally, the pair $(v_2, \mathbf{c}')$ is obtained by

$$(v_2, \mathbf{c}') = g\left(v_1, \mathbf{c}^{(n)}\right),$$

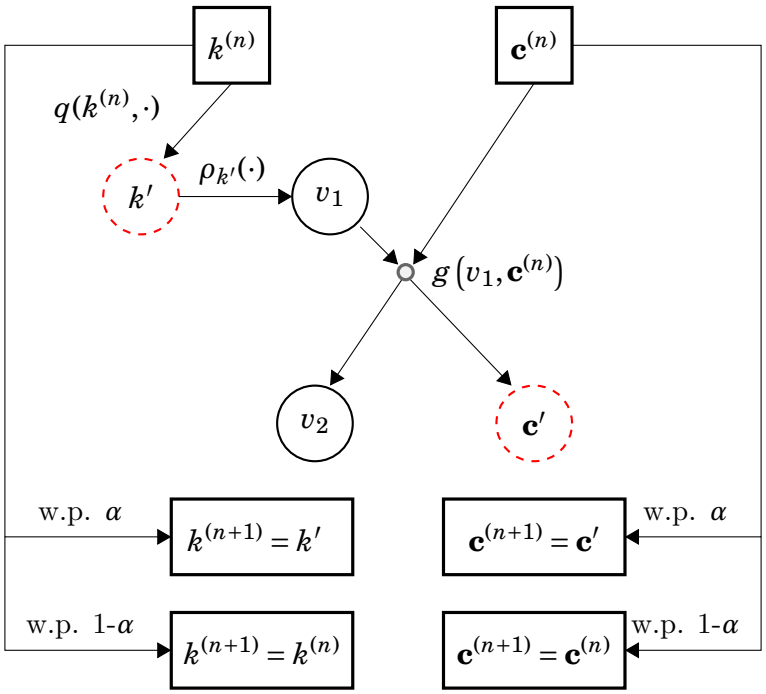


Figure 2.1 – General structure of PACBO.

where $g:(x,y)\in\mathbb{R}^{dk'}\times\mathbb{R}^{dk^{(n)}}\mapsto(y,x)\in\mathbb{R}^{dk^{(n)}}\times\mathbb{R}^{dk'}$ is a one-to-one, first order derivative mapping. The resulting RJMCMC acceptance probability is

$$\begin{aligned}\alpha\left[\left(k^{(n)},\mathbf{c}^{(n)}\right),\left(k',\mathbf{c}'\right)\right]&=\min\left\{1,\frac{\hat{\rho}_t(\mathbf{c}')q(k',k^{(n)})\rho_{k^{(n)}}(v_2)}{\hat{\rho}_t(\mathbf{c}^{(n)})q(k^{(n)},k')\rho_{k'}(v_1)}\left|\frac{\partial g\left(v_1,\mathbf{c}^{(n)}\right)}{\partial v_1\partial\mathbf{c}^{(n)}}\right|\right\},\\&=\min\left\{1,\frac{\hat{\rho}_t(\mathbf{c}')q(k',k^{(n)})\rho_{k^{(n)}}(\mathbf{c}^{(n)},\mathfrak{c}_{k^{(n)}},\tau_{k^{(n)}})}{\hat{\rho}_t(\mathbf{c}^{(n)})q(k^{(n)},k')\rho_{k'}(\mathbf{c}',\mathfrak{c}_{k'},\tau_{k'})}\right\},\end{aligned}$$

since the determinant of the Jacobian matrix of g is 1. The resulting PACBO algorithm is described in Algorithm 2.3.

2.4.2 Convergence of PACBO towards the Gibbs quasi-posterior

We prove that Algorithm 2.3 builds a Markov chain whose invariant distribution is precisely the Gibbs quasi-posterior as N goes to $+\infty$. To do so, we need to prove that the chain is $\hat{\rho}_t$-irreducible, aperiodic and Harris recurrent, see Robert and Casella (2004, Theorem 6.51) and Roberts and Rosenthal (2006, Theorem 20).

Recall that at each RJMCMC iteration in Algorithm 2.3, the chain is said to propose a "between model move" if $k'\neq k^{(n)}$ and a "within model move" if $k'=k^{(n)}$ and $\mathbf{c}'\neq\mathbf{c}^{(n)}$. The following result gives a sufficient condition for the chain to be Harris recurrent.

Lemma 2.1. *Let D be the event that no "within-model move" is ever accepted and $\mathscr{E}$ be the support of $\hat{\rho}_t$. Then the chain generated by Algorithm 2.3 satisfies*

$$\mathbb{P}\left[D|\left(k^{(0)},\mathbf{c}^{(0)}\right)=(k,\mathbf{c})\right]=0,$$

for any $k\in[\![1,p]\!]$ and $\mathbf{c}\in\mathbb{R}^{dk}\cap\mathscr{E}$.

Algorithm 2.3 PACBO

Initialization: $(\lambda_t)_{1:T}$
For $t \in [\![1,T]\!]$
Initialization: $\left(k^{(0)}, \mathbf{c}^{(0)}\right) \in [\![1,p]\!] \times \mathbb{R}^{dk^{(0)}}$. Typically $k^{(0)}$ is set to $k^{(N)}$ from iteration $t-1$ ($k^{(0)}=1$ at iteration $t=1$).
For $n \in [\![1,N-1]\!]$
    Sample $k' \in \left[\!\left[\max(1,k^{(n)}-1), \min(p,k^{(n)}+1)\right]\!\right]$ from $q(k^{(n)},\cdot) = \frac{1}{3}$.
    Let $\mathfrak{c}' \leftarrow$ standard k'-means output trained on $(x_s)_{1:(t-1)}$.
    Let $\tau' = 1/\sqrt{pt}$.
    Sample $v_1 \sim \rho_{k'}(\cdot, \mathfrak{c}_{k'}, \tau_{k'})$.
    Let $(v_2, \mathbf{c}') = g(v_1, \mathbf{c}^{(n)})$.
    Accept the move $(k^{(n)}, \mathbf{c}^{(n)}) = (k', \mathbf{c}')$ with probability

$$\alpha\left[(k^{(n)}, \mathbf{c}^{(n)}), (k', \mathbf{c}'))\right] = \min\left\{1, \frac{\hat{\rho}_t(\mathbf{c}')q(k', k^{(n)})\rho_{k^{(n)}}(v_2, \mathfrak{c}_{k^{(n)}}, \tau_{k^{(n)}})}{\hat{\rho}_t(\mathbf{c}^{(n)})q(k^{(n)}, k')\rho_{k'}(v_1, \mathfrak{c}_{k'}, \tau_{k'})} \left|\frac{\partial g(v_1, \mathbf{c}^{(n)})}{\partial v_1 \partial \mathbf{c}^{(n)}}\right|\right\}$$
$$= \min\left\{1, \frac{\hat{\rho}_t(\mathbf{c}')q(k', k^{(n)})\rho_{k^{(n)}}(\mathbf{c}^{(n)}, \mathfrak{c}_{k^{(n)}}, \tau_{k^{(n)}})}{\hat{\rho}_t(\mathbf{c}^{(n)})q(k^{(n)}, k')\rho_{k'}(\mathbf{c}', \mathfrak{c}_{k'}, \tau_{k'})}\right\}$$

    Else $(k^{(n+1)}, \mathbf{c}^{(n+1)}) = (k^{(n)}, \mathbf{c}^{(n)})$.
End for
Let $\hat{\mathbf{c}}_t = \mathbf{c}^{(N)}$.
End for

Lemma 2.1 states that the chain must eventually accept a "within-model move". It remains true for other choices of $q(k^{(n)},\cdot)$ in Algorithm 2.3, provided that the stationarity of $\hat{\rho}_t$ is preserved.

Theorem 2.3. *Let $\mathscr{E}$ denote the support of $\hat{\rho}_t$. Then for any $\mathbf{c}^{(0)} \in \mathscr{E}$, the chain $\left(\mathbf{c}^{(n)}\right)_{1:N}$ generated by Algorithm 2.3 is $\hat{\rho}_t$-irreducible, aperiodic and Harris recurrent.*

Theorem 2.3 legitimates our approximation PACBO to perform online clustering, since it asymptotically mimics the behavior of the computationally unavailable $\hat{\rho}_t$. To the best of our knowledge, this kind of guarantee is original in the PAC-Bayesian literature.

Finally, let us stress that obtaining an explicit rate of convergence is beyond the scope of the present work. Controlling the error between approximation algorithm and the target is difficult though several work (Kalai and Vempala, 2002, Pérez, 2015) have addressed it. For our RJMCMC, even though we have proved that it converges to a stationary distribution which is the target, it is an open question to know until which iteration the chain becomes stationary. However, in most cases the chain converges rather quickly in practice, as illustrated by Figure 2.2. Moreover, this algorithm has been put into production at iAdvize and it works well on real data of several clients in helping them to well-target online visitors. At time t, we advocate for setting $k^{(0)}$ as $k^{(N)}$ from round $t-1$, as a warm start.

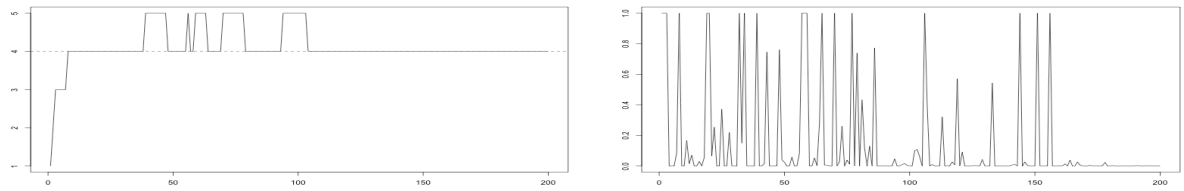
2.4.3 Numerical study

This section is devoted to the illustration of the potential of our quasi-Bayesian approach on synthetic data. Let us stress that all experiments are reproducible, thanks to the

PACBO R package (Li, 2016). We do not claim to be exhaustive here but rather show the (good) behavior of our implementation on a toy example.

Calibration of parameters and mixing properties

We set R to be the maximum ℓ_2-norm of the observations. Note that a too small value will yield acceptance ratios to be close to zero and will degrade the mixing of the chain. When calibrating λ, recall that large values will enforce the quasi-posterior to account more for past data, whereas small values make the quasi-posterior alike the prior. In order to find an optimal value for it in batch setting, we consider $\lambda = c \times (d+2)\sqrt{\log T}/(2\sqrt{T}R^2)$ with different equidistant values in the range $(0, R^2)$ for coefficient c, and we choose $\lambda = 0.6 \times (d+2)\sqrt{\log T}/(2\sqrt{T})$ for its global performance on different models below on minimizing the distances between test set and estimated centers. We keep this parameter also in online setting. We illustrate in Figure 2.2 the mixing behavior of PACBO. The convergence occurs quickly, and the default length of the RJMCMC runs is set to 500 in the PACBO package: this was a ceiling value in all our simulations.



(a) Number of clusters. (b) Acceptance probability.

Figure 2.2 – Typical RJMCMC output in PACBO. (a) plot of number of clusters $k^{(n)}$ (black solid line) at each n-th ($n = 1, \ldots, N$) iteration of RJMCMC and the red dashed line indicates the true number of clusters of a Gaussian mixture model (Model 2) (b) trace plot of acceptance ratio $\alpha^{(n)}$ along the 200 iterations, showing that the RJMCMC does not get stuck to "only reject" or "only accept" patterns.

Batch clustering setting

A large variety of methods have been proposed in the literature for selecting the number k of clusters in batch clustering (see Milligan and Cooper, 1985; Gordon, 1999, for a survey). These methods may be of local or global nature. For local methods, at each step, each cluster is either merged with another one, split in two or remains. Global methods evaluate the empirical distortion of any clustering as a function of the number k of cells over the whole dataset, and select the minimizer of this distortion. The rule of Hartigan (1975) is a well-known representative of local methods. Popular global methods include the works of Calinski and Harabasz (1974), Krzanowski and Lai (1988) and Kaufman and Rousseeuw (1990), where functions based on the empirical distortion or on the average of within-cluster dispersion of each point are constructed and the optimal number of clusters is the maximizer of these functions. In addition, the Gap Statistic (Tibshirani *et al.*, 2001) compares the change in within-cluster dispersion with the one expected under an appropriate reference null distribution. More recently, CAPUSHE (CAlibrating Penalty Using Slope Heuristics) introduced by Fischer (2011) and Baudry *et al.* (2012) addresses the problem from the penalized model selection perspective, in the form of two methods: DDSE (Data-Driven Slope Estimation) and Djump (Dimension jump).

R packages implementing those methods are used with their default parameters in our simulations.

In this section, we compare PACBO to the aforecited methods in a batch setting with $n = 200$ observations simulated from the following 4 models.

Model 1 (1 group in dimension 5)**.** *Observations are sampled from a uniform distribution on the unit hypercube in* $\mathbb{R}^5$.

Model 2 (4 Gaussian groups in dimension 2)**.** *Observations are sampled from 4 bivariate Gaussian distributions with identity covariance matrix, whose mean vectors are respectively* $(0,0),(-2,-1),(0,4),(3,1)$. *Each observation is uniformly drawn from one of the four groups.*

Model 3 (7 Gaussian groups in dimension 50)**.** *Observations are sampled from 7 multivariate Gaussian distributions in* $\mathbb{R}^{50}$ *with identity covariance matrix, whose mean vectors are chosen randomly according to an uniform distribution on* $[-10,10]^{50}$. *Each observation is uniformly drawn from one of the seven groups.*

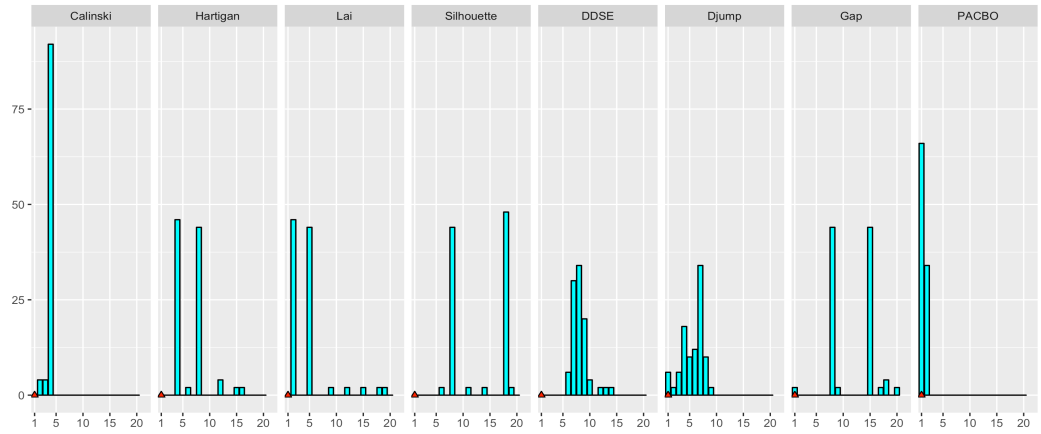
Model 4 (3 lognormal groups in dimension 3)**.** *Observations are sampled from 3 multivariate lognormal distributions in* $\mathbb{R}^3$ *with identity covariance matrix, whose mean vectors are respectively* $(1,1,1),(6,5,7),(10,9,11)$. *Each observation is uniformly drawn from one of the three groups.*

Figure 2.3 and Figure 2.4 present the percentage of the estimated number of cells k on 50 realizations of the 4 aforementioned models, for 8 methods including PACBO. In each graph, the red dot indicates the real number of groups. The methods used for selecting k are presented on the top of each panel, where DDSE (Data-Driven Slope Estimation) and Djump (Dimension jump) are the two methods introduced in CAPUSHE (Baudry *et al.*, 2012). The maximum number of cells is set to 20.

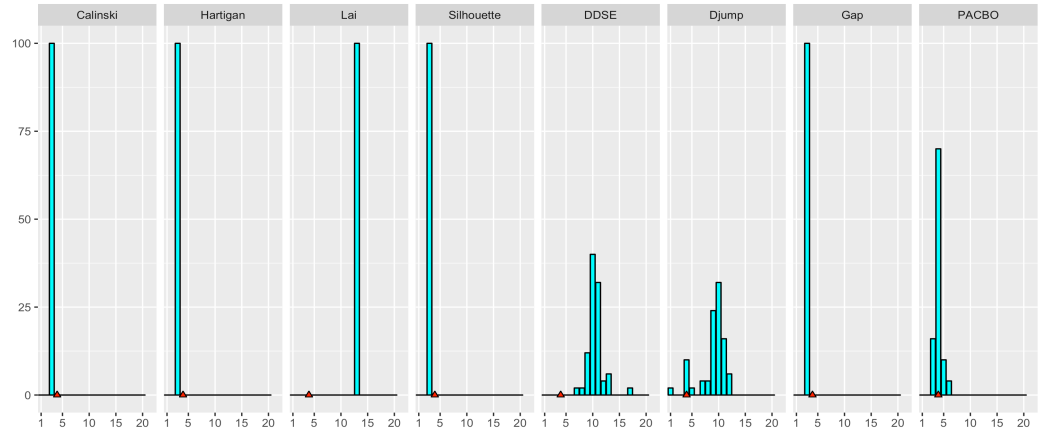
For Model 1 PACBO outperforms all competitors, since it selects the correct number of cells in almost 70% of our simulations, when all other methods barely find it (Figure 2.3a). For Model 2 Calinski, Hartigan, Silhouette and Gap underestimate the number of cells by identifying 3 groups. Djump finds the true value $k = 4$ less than 10%. PACBO identifies 4 groups in 60% of our runs (Figure 2.3b).
For Model 3 PACBO is one of the two best methods, together with Gap (Figure 2.4a).
For Model 4 where 3 groups of observations are generated from a heavy-tailed distribution, we consider a variant of PACBO with the ℓ_1-norm in $\mathbb{R}^d$, *i.e.*, we replace the loss in (2.1) by $\ell(\hat{\mathbf{c}}_t, x_t) = \min_{1\le k\le K_t} |\hat{c}_{t,k} - x_t|_1$. Figure 2.4b shows that most methods perform poorly, to the notable exception of this PACBO(ℓ_1).

Online clustering setting

In the last part, we have compared, in the batch setting, our method with 7 other methods on different datasets. However let us stress here that none of the aforementioned methods is specifically designed for *online* clustering. Indeed, to the best of our knowledge PACBO is the sole procedure that explicitly takes advantage of the *sequential nature* of data. For that reason, we present below the behavior and a comparison of running times between PACBO and the aforementioned methods, on the following synthetic online clustering toy example.

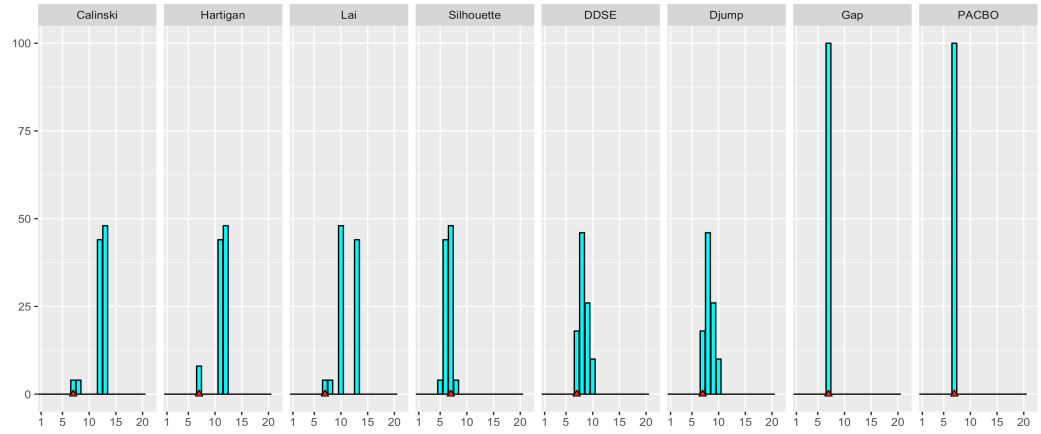


(a) Model 1.

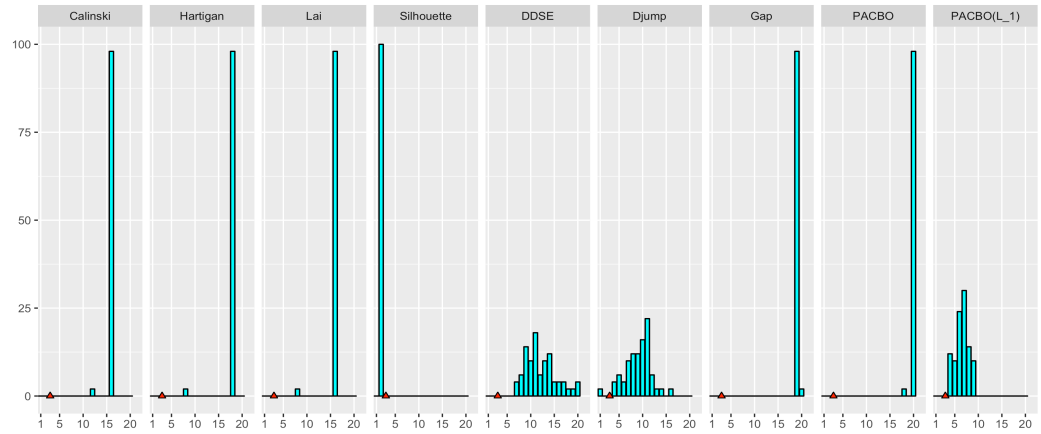


(b) Model 2.

Figure 2.3 – Histograms of the estimated number of cells on 50 realizations. The red mark indicates the true number of cells.



(a) Model 3.



(b) Model 4.

Figure 2.4 – Histograms of the estimated number of cells on 50 realizations. The red mark indicates the true number of cells.

Model 5 (10 mixed groups in dimension 2). *Observations* $(x_t)_{t=1,\dots,T=200}$ *are simulated in the following way: define firstly for each* $t \in [\![1,T]\!]$ *a pair* $(c_{1,t}, c_{2,t}) \in \mathbb{R}^2$, *where* $c_{1,t} = -\frac{5}{2}\pi + \frac{5\pi}{9}\left(\lfloor \frac{t-1}{20} \rfloor - 1\right)$ *and* $c_{2,t} = 5\sin(c_{1,t})$. *Then for* $t \in [\![1,100]\!]$, x_t *is sampled from a uniform distribution on the unit cube in* $\mathbb{R}^2$, *centered at* $(c_{x,t}, c_{y,t})$. *For* $t \in [\![101,200]\!]$, x_t *is generated by a bivariate Gaussian distribution, centered at* $(c_{x,t}, c_{y,t})$ *with identity covariance matrix.*

In this online setting, the true number $k_t^\star$ of groups will augment of 1 unit every 20 time steps to eventually reach 10 (and the maximal number of clusters is set to $\mathbf{20}$ for all methods). Figure 2.5a shows ECL for PACBO and OCL along with 95% confidence intervals computed on 100 realizations with $T = 200$ observations, with $\lambda_t = 0.6 \times (d+2)\sqrt{\log t}/(2\sqrt{t})$ and $R = 15$ (so that all observations are in the ℓ_2-ball $B_2(R)$. The experiment is replicated 100 times, yielding 100 estimates of the expectation (ECL). The confidence intervals are drawn from the empirical 95% quantile. Jumps in the ECL occur when new clusters of data are observed. Since PACBO outputs a partition based only on the past observations, the instantaneous loss is larger whenever a new cluster appears. However PACBO quickly identifies the new cluster. This is also supported by Figure 2.5b which represents the true and estimated numbers of clusters.

In addition we also count the number of correct estimations of the true number $k_t^\star$ of clusters. Table 2.1 contains its mean (and standard deviation, on $\mathbf{100}$ repetitions) for PACBO and its seven competitors. PACBO has the largest mean by a significant margin and identifies the correct number of clusters of about 120 observations out of 200.

Calinski	Hartigan	Lai	Silhouette	DDSE	Djump	Gap	PACBO
34.92 (8.24)	63.72 (4.81)	52.23 (4.64)	72.44 (4.39)	22.73 (4.17)	38.38 (6.21)	56.73 (14.38)	**119.95 (7.08)**

Table 2.1 – Mean and standard deviation of correct estimations of the true number of clusters.

Next, we compare the running times of PACBO and its competitors, in the online setting. At each time $t = 1,\dots,200$, we measure the running time of each method. Table 2.2 presents the mean (and standard deviation) on $\mathbf{100}$ repetitions of the total running times. The superiority of PACBO is a straightforward consequence of the fact that it adapts to the *sequential nature* of data, whereas all other methods conduct a batch clustering at each time step.

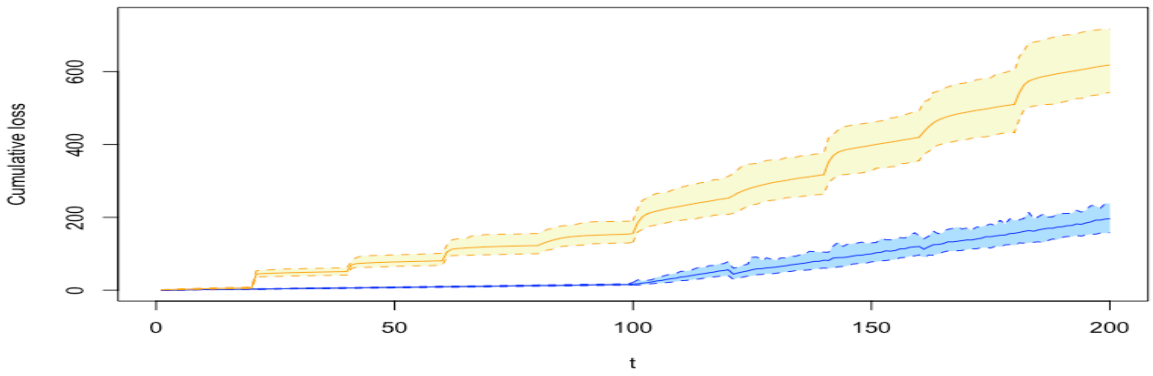
Calinski	Hartigan	Lai	Silhouette	DDSE	Djump	Gap	PACBO
46.86 (5.66)	39.27 (2.75)	52.07 (3.53)	118.44 (1.98)	33.85 (6.82)	33.85 (6.82)	207.55 (2.72)	**28.13 (4.06)**

Table 2.2 – Mean (and standard deviation) of total running time (in seconds).

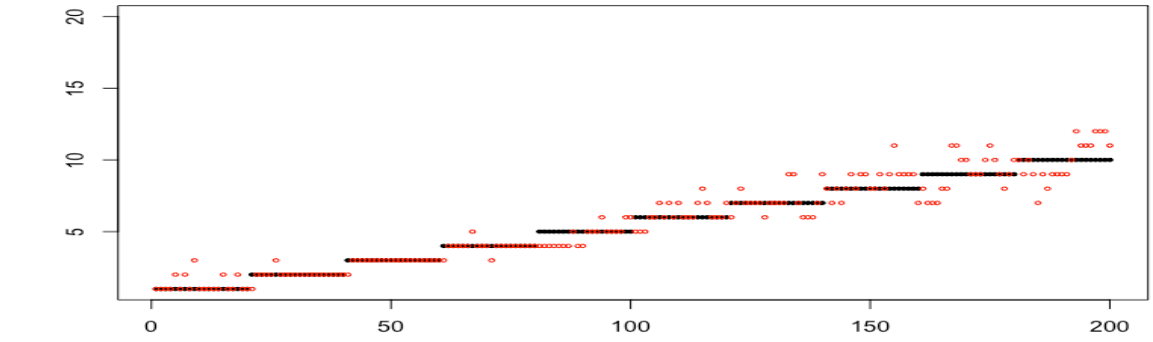
For the sake of completeness, Appendix 2.6 contains an instance of the performance of all methods to estimate the true number of clusters.

2.5 Proofs

This section contains the proofs to all original results claimed in Section 2.3 and Section 2.4.



(a) ECL (yellow line) and OCL (blue line) as function of t, with 95% confidence intervals (dashed line).



(b) Estimated number of cells (red dots) by PACBO as a function of t. Black lines represent the true number of cells.

Figure 2.5 – Performance of PACBO.

2.5.1 Proof of Corollary 2.1

Let us first introduce some notation. For any $k \in [\![1,p]\!]$ and $R > 0$, let

$$\mathscr{C}(k,R) = \left\{ \mathbf{c} = (c_j)_{j=1,2,\dots,k} \in \mathbb{R}^{dk}, c_i \neq c_j, i \neq j, \text{such that } |c_j|_2 \leq R, \forall j \right\},$$
$$\Xi(k,R) = \left\{ \xi = (\xi_j)_{j=1,\dots,k} \in \mathbb{R}^k : 0 < \xi_j \leq R, \forall j \right\}.$$

We denote by $\rho_k(\mathbf{c},\mathfrak{c},\xi)$ the density consisting in the product of k independent uniform distributions on ℓ_2-balls in $\mathbb{R}^d$, namely,

$$\mathrm{d}\rho_k(\mathbf{c},\mathfrak{c},\xi) = \prod_{j=1}^{k} \left\{ \frac{\Gamma(\frac{d}{2}+1)}{\pi^{\frac{d}{2}}} \left(\frac{1}{\xi_j}\right)^d \mathbb{1}_{\{B_d(\mathfrak{c}_j,\xi_j)\}}(c_j) \right\} \mathrm{d}\mathbf{c},$$

where $\mathfrak{c} \in \mathscr{C}(k,R)$, $\xi \in \Xi(k,R)$ and $B_d(\mathfrak{c}_j,\xi_j)$ is an ℓ_2-ball in $\mathbb{R}^d$, centered in $\mathfrak{c}_j$ with radius ξ_j. In the following, we shall shorten $\rho_k(\mathbf{c},\mathfrak{c},\xi)$ to ρ_k when no confusion can arise. The proof relies on choosing a specific ρ in Proposition 2.1. For any $k \in [\![1,p]\!]$, $\mathfrak{c} \in \mathscr{C}(k,R)$ and $\xi \in \Xi(k,R)$, let $\rho = \rho_k \mathbb{1}_{\{\mathbf{c} \in \mathbb{R}^{dk}\}}$. Then ρ is a well-defined distribution on $\mathscr{C}$ and belongs to

$\mathscr{P}_\pi(\mathscr{C})$. Proposition 2.1 yields

$$\sum_{t=1}^{T}\mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\ldots,\hat{\rho}_t)}\ell(\hat{\mathbf{c}}_t,x_t)\leq \inf_{k\in[\![1,p]\!]}\ \inf_{\substack{\rho\in\mathscr{P}_\pi(\mathscr{C})\\ \rho=\rho_k\mathbb{1}_{\{\mathbf{c}\in\mathbb{R}^{dk}\}}}}\left\{\mathbb{E}_{\mathbf{c}\sim\rho}\sum_{t=1}^{T}[\ell(\mathbf{c},x_t)]+\frac{\mathscr{K}(\rho,\pi)}{\lambda}\right.$$
$$\left.+\frac{\lambda}{2}\mathbb{E}_{(\hat{\rho}_1,\ldots,\hat{\rho}_T)}\mathbb{E}_{\mathbf{c}\sim\rho}\sum_{t=1}^{T}[\ell(\mathbf{c},x_t)-\ell(\hat{\mathbf{c}}_t,x_t)]^2\right\}. \tag{2.15}$$

For any $\rho=\rho_k\mathbb{1}_{\{\mathbf{c}\in\mathbb{R}^{dk}\}}$, the first term on the right-hand side of (2.15) satisfies

$$\begin{aligned}
\sum_{t=1}^{T}\mathbb{E}_{\mathbf{c}\sim\rho}[\ell(\mathbf{c},x_t)] &= \sum_{t=1}^{T}\mathbb{E}_{\mathbf{c}\sim\rho_k}[\ell(\mathbf{c},x_t)]\\
&= \sum_{t=1}^{T}\mathbb{E}_{\mathbf{c}\sim\rho_k}\left[\min_{j=1,2,\ldots,k}|c_j-x_t|_2^2\right]\\
&\leq \sum_{t=1}^{T}\min_{j=1,\ldots,k}\mathbb{E}_{\mathbf{c}\sim\rho_k}\left[\langle c_j-x_t,c_j-x_t\rangle\right]\\
&\leq \sum_{t=1}^{T}\min_{j=1,\ldots,k}\mathbb{E}_{\mathbf{c}\sim\rho_k}\left[|c_j-\mathfrak{c}_j|_2^2+2\langle c_j-\mathfrak{c}_j,\mathfrak{c}_j-x_t\rangle+|\mathfrak{c}_j-x_t|_2^2\right]\\
&= \sum_{t=1}^{T}\min_{j=1,\ldots,k}\left\{\mathbb{E}_{\mathbf{c}\sim\rho_k}\left[|c_j-\mathfrak{c}_j|_2^2\right]+|\mathfrak{c}_j-x_t|_2^2\right\}\\
&= \sum_{t=1}^{T}\min_{j=1,\ldots,k}\left\{\frac{d}{d+2}\xi_j^2+|\mathfrak{c}_j-x_t|_2^2\right\}\\
&\leq \frac{dT}{d+2}\max_{j=1,\ldots,k}\xi_j^2+\sum_{t=1}^{T}\ell(\mathfrak{c},x_t),
\end{aligned}\tag{2.16}$$

where the third equality is due to the fact that the expectation of c_j under ρ_k is $\mathfrak{c}_j$, $j=1,2,\ldots,k$. Let us now compute the second term on the right-hand side of (2.15).

$$\begin{aligned}
\mathscr{K}(\rho,\pi) &= \int_{\mathscr{C}}\log\frac{\rho(\mathbf{c})}{\pi(\mathbf{c})}\rho(\mathbf{c})\mathrm{d}\mathbf{c}\\
&= \int_{\mathbb{R}^{dk}}\left(\log\frac{\rho_k(\mathbf{c})}{\pi_k(\mathbf{c})}+\log\frac{\pi_k(\mathbf{c})}{\pi(\mathbf{c})}\right)\rho_k(\mathbf{c})\mathrm{d}\mathbf{c}\\
&= \mathscr{K}(\rho_k,\pi_k)+\log\frac{1}{q(k)}\\
&=: A+B,
\end{aligned}$$

where

$$A=\int_{\mathbb{R}^{dk}}\log\prod_{j=1}^{k}\frac{\left(\frac{1}{\xi_j}\right)^d}{\left(\frac{1}{2R}\right)^d}\rho_k(\mathbf{c})\mathrm{d}\mathbf{c}=d\sum_{j=1}^{k}\log\left(\frac{2R}{\xi_j}\right).$$

Since the function $x\mapsto(1-e^{-\eta x})/x$ is non-increasing for $x>0$ and $\eta>0$, we have

$$\begin{aligned}
B &= \log\left(\frac{e^{-\eta}(1-e^{-\eta p})}{1-e^{-\eta}}e^{\eta k}\right)\\
&\leq \log\left(pe^{\eta(k-1)}\right)
\end{aligned}$$

$$= \eta(k-1) + \log p. \tag{2.17}$$

When $\eta = 0$, q is a uniform distribution on $[\![1,p]\!]$, and the above inequality holds as well. Then, $\mathscr{K}(\rho,\pi)/\lambda$ in (2.15) may be upper bounded as follows:

$$\frac{\mathscr{K}(\rho,\pi)}{\lambda} \le \frac{d}{\lambda}\sum_{j=1}^{k}\log\left(\frac{2R}{\xi_j}\right) + \frac{\eta(k-1)}{\lambda} + \frac{\log p}{\lambda}. \tag{2.18}$$

Finally,

$$\begin{aligned}|\ell(\mathbf{c},x_t) - \ell(\hat{\mathbf{c}}_t,x_t)| &= \left|\min_{j=1,\dots,k}|c_j - x_t|_2^2 - \min_{j=1,\dots,K_t}|\hat{c}_{t,j} - x_t|_2^2\right| \\ &\le \left(2R + \max_{t=1,\dots,T}|x_t|_2\right)^2 =: C_1.\end{aligned}$$

Then, the third term of the right-hand side in (2.15) is controlled as

$$\frac{\lambda}{2}\mathbb{E}_{(\hat{\rho}_1,\dots,\hat{\rho}_T)}\mathbb{E}_{\mathbf{c}\sim\rho_k}\sum_{t=1}^{T}[\ell(\mathbf{c},x_t) - \ell(\hat{\mathbf{c}}_t,x_t)]^2 \le \frac{\lambda T}{2}C_1^2. \tag{2.19}$$

Combining inequalities (2.16), (2.18) and (2.19) gives, for any $\xi \in \Xi(k,R)$,

$$\begin{aligned}\sum_{t=1}^{T}\mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\dots,\hat{\rho}_t)}\ell(\hat{\mathbf{c}}_t,x_t) \le \inf_{k\in[\![1,p]\!]}\inf_{\mathfrak{c}\in\mathscr{C}(k,R)}\Bigg\{&\sum_{t=1}^{T}\ell(\mathfrak{c},x_t) + \frac{dT}{d+2}\max_{j=1,\dots,k}\xi_j^2 \\ &+ \frac{d}{\lambda}\sum_{j=1}^{k}\log\left(\frac{2R}{\xi_j}\right) + \frac{\eta}{\lambda}(k-1)\Bigg\} + \frac{\lambda T}{2}C_1^2 + \frac{\log p}{\lambda}.\end{aligned}$$

Under the assumption that $\lambda > (d+2)/(2TR^2)$, the global minimizer of the function

$$(\xi_1,\dots,\xi_k) \mapsto \frac{Td}{d+2}\max_{j=1,\dots,k}\xi_j^2 + \frac{d}{\lambda}\sum_{j=1}^{k}\log\left(\frac{2R}{\xi_j}\right) \tag{2.20}$$

does not necessarily belong to $\Xi(k,R)$. A possible choice of $(\xi_j)_{1:k} \in \Xi(k,R)$ is given by

$$\xi_1^\star = \xi_2^\star = \cdots = \xi_k^\star = \sqrt{\frac{d+2}{2\lambda T}}.$$

Then (2.20) amounts to

$$\frac{d}{2\lambda} + \frac{dk}{2\lambda}\log\left(\frac{8R^2\lambda T}{d+2}\right).$$

Hence,

$$\begin{aligned}\sum_{t=1}^{T}\mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\dots,\hat{\rho}_t)}\ell(\hat{\mathbf{c}}_t,x_t) \le \inf_{k\in[\![1,p]\!]}\inf_{\mathfrak{c}\in\mathscr{C}(k,R)}&\left\{\sum_{t=1}^{T}\ell(\mathfrak{c},x_t) + \frac{dk}{2\lambda}\log\left(\frac{8R^2\lambda T}{(d+2)k}\right) + \frac{\eta}{\lambda}k\right\} \\ &+ \left(\frac{\log p}{\lambda} + \frac{d}{2\lambda} + \frac{\lambda T}{2}C_1^2\right).\end{aligned}$$

2.5.2 Proof of Theorem 2.1

The proof builds upon the online variance inequality described in Audibert (2009), *i.e.*, for any $\lambda > 0$, any $\hat{\rho} \in \mathscr{P}_\pi(\mathscr{C})$ and any $x \in \mathbb{R}^d$,

$$\mathbb{E}_{\mathbf{c}' \sim \hat{\rho}}[\ell(\mathbf{c}', x)] \leq -\frac{1}{\lambda} \mathbb{E}_{\mathbf{c}' \sim \hat{\rho}} \log \mathbb{E}_{\mathbf{c} \sim \hat{\rho}} \left[e^{-\lambda \left[\ell(\mathbf{c}, x) + \frac{\lambda}{2} (\ell(\mathbf{c}, x) - \ell(\mathbf{c}', x))^2 \right]} \right]. \tag{2.21}$$

By (2.21), we have

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) &= \sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1, \ldots, \hat{\rho}_{t-1})} \mathbb{E}_{\hat{\rho}_t} [\ell(\hat{\mathbf{c}}_t, x_t) \mid \hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_{t-1}] \\
&\leq \sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1, \ldots, \hat{\rho}_{t-1})} \left[-\frac{1}{\lambda_{t-1}} \mathbb{E}_{\hat{\mathbf{c}}_t \sim \hat{\rho}_t} \log \mathbb{E}_{\mathbf{c} \sim \hat{\rho}_t} \left(e^{-\lambda_{t-1} [\ell(\mathbf{c}, x_t) + \frac{\lambda_{t-1}}{2} (\ell(\mathbf{c}, x_t) - \ell(\hat{\mathbf{c}}_t, x_t))^2]} \right) \middle| \, \hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_{t-1} \right] \\
&\leq \mathbb{E}_{(\hat{\rho}_1, \ldots, \hat{\rho}_T)} \left[\sum_{t=1}^{T} -\frac{1}{\lambda_{t-1}} \log \frac{\int e^{-\lambda_{t-1} S_t(\mathbf{c})} \mathrm{d}\pi(\mathbf{c})}{\int e^{-\lambda_{t-1} S_{t-1}(\mathbf{c})} \mathrm{d}\pi(\mathbf{c})} \right] \\
&= \mathbb{E}_{(\hat{\rho}_1, \ldots, \hat{\rho}_T)} \left[\sum_{t=1}^{T} -\frac{1}{\lambda_{t-1}} \log \frac{V_t}{W_{t-1}} \right] \\
&= \mathbb{E}_{(\hat{\rho}_1, \ldots, \hat{\rho}_T)} \left[\sum_{t=1}^{T} \left[\frac{1}{\lambda_{t-1}} \log W_{t-1} - \frac{1}{\lambda_{t-1}} \log V_t \right] \right],
\end{aligned} \tag{2.22}$$

where $V_t = \mathbb{E}_{\mathbf{c} \sim \pi} \left[e^{-\lambda_{t-1} S_t(\mathbf{c})} \right]$ and $W_t = \mathbb{E}_{\mathbf{c} \sim \pi} \left[e^{-\lambda_t S_t(\mathbf{c})} \right]$. Applying Jensen's inequality, for any $1 \leq t \leq T$,

$$\begin{aligned}
\frac{1}{\lambda_{t-1}} \log V_t &= \frac{1}{\lambda_{t-1}} \log \mathbb{E}_{\mathbf{c} \sim \pi} \left[\left(e^{-\lambda_t S_t(\mathbf{c})} \right)^{\frac{\lambda_{t-1}}{\lambda_t}} \right] \\
&\geq \frac{1}{\lambda_{t-1}} \log \left(\mathbb{E}_{\mathbf{c} \sim \pi} \left[e^{-\lambda_t S_t(\mathbf{c})} \right] \right)^{\frac{\lambda_{t-1}}{\lambda_t}} \\
&= \frac{1}{\lambda_t} \log W_t.
\end{aligned}$$

Therefore, since $W_0 = 1$,

$$\sum_{t=1}^{T} \left[\frac{1}{\lambda_{t-1}} \log W_{t-1} - \frac{1}{\lambda_{t-1}} \log V_t \right] \leq -\frac{1}{\lambda_T} \log W_T, \tag{2.23}$$

and by (2.22), (2.23) and the duality formula (2.2), we have

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) &\leq \mathbb{E}_{(\hat{\rho}_1, \ldots, \hat{\rho}_T)} \left[-\frac{1}{\lambda_T} \log \mathbb{E}_{\mathbf{c} \sim \pi} \left[e^{-\lambda_T S_T(\mathbf{c})} \right] \right] \\
&\leq -\frac{1}{\lambda_T} \log \mathbb{E}_{\mathbf{c} \sim \pi} \left[e^{-\lambda_T \mathbb{E}_{(\hat{\rho}_1, \ldots, \hat{\rho}_T)} S_T(\mathbf{c})} \right] \qquad \text{(by Audibert, 2009, Lemma 3.2)} \\
&= \inf_{\rho \in \mathscr{P}_\pi(\mathscr{C})} \left\{ \mathbb{E}_{\mathbf{c} \sim \rho} \left[\sum_{t=1}^{T} \ell(\mathbf{c}, x_t) \right] + \mathbb{E}_{\mathbf{c} \sim \rho} \mathbb{E}_{(\hat{\rho}_1, \ldots, \hat{\rho}_T)} \left[\sum_{t=1}^{T} \frac{\lambda_{t-1}}{2} (\ell(\mathbf{c}, x_t) - \ell(\hat{\mathbf{c}}_t, x_t))^2 \right] \right. \\
&\quad \left. + \frac{\mathscr{K}(\rho, \pi)}{\lambda_T} \right\},
\end{aligned}$$

which achieves the proof.

2.5.3 Proof of Corollary 2.3

The proof is similar to the proof of Corollary 2.1, the only difference lies in the fact that (2.19) is replaced with

$$\begin{aligned}\mathbb{E}_{(\hat\rho_1,\dots,\hat\rho_T)}\mathbb{E}_{\mathbf{c}\sim\rho_k}\sum_{t=1}^{T}\frac{\lambda_{t-1}}{2}[\ell(\mathbf{c},x_t)-\ell(\hat{\mathbf{c}}_t,x_t)]^2 &\le \frac{(d+2)C_1^2}{4R^2}\left(1+\sum_{t=2}^{T}\frac{\sqrt{\log(t-1)}}{\sqrt{t-1}}\right)\\ &\le \frac{(d+2)C_1^2}{4R^2}\left(1+\frac{\sqrt{\log 2}}{\sqrt{2}}+\frac{\sqrt{\log 3}}{\sqrt{3}}+\sum_{t=4}^{T-1}\int_{t-1}^{t}\frac{\sqrt{\log x}}{\sqrt{x}}\mathrm{d}x\right)\\ &\le \frac{(d+2)C_1^2}{2R^2}\sqrt{T\log T},\end{aligned}$$

where the second inequality above is due to the fact that $\frac{\sqrt{\log t}}{\sqrt{t}} \le \int_{t-1}^{t}\frac{\sqrt{\log x}}{\sqrt{x}}\mathrm{d}x$ when $t\ge 4$ and the last inequality is deduced from the following with change of variable $y=\sqrt{\log x}$, *i.e.*,

$$\begin{aligned}\int_3^{T-1}\frac{\sqrt{\log x}}{\sqrt{x}}\mathrm{d}x &= \int_{\sqrt{\log 3}}^{\sqrt{\log(T-1)}} 2y^2 e^{\frac{y^2}{2}}\mathrm{d}y\\ &\le \sqrt{\log(T-1)}\int_{\sqrt{\log 3}}^{\sqrt{\log(T-1)}} 2y e^{\frac{y^2}{2}}\mathrm{d}y\\ &= 2\sqrt{\log(T-1)}\left(\sqrt{T-1}-\sqrt{3}\right).\end{aligned}$$

2.5.4 Proof of Corollary 2.4

Let us denote by M the index of the last epoch and let $t_M = T$. We assume $M \ge 1$ (otherwise, the corollary follows directly from Corollary 2.3 applied with an upper bound R_0 of ℓ_2-norm of sequence $(x_t)_{1:T}$). If $R_{t_M} \le R_{t_{M-1}}$, then we have $R_T = R_{t_M} = R_{t_{M-1}}$, hence one always has $R_{t_M} \ge R_{t_{M-1}}$. In addition, since $M \ge 1$, we also have $R_{t_M} \le 2\max_{t=1,\dots,T}|x_t|_2 = 2R$.

Let us introduce for each epoch $r, r=0,1,\dots,M$ the following notation

$$E^{(r)} = \sum_{t=t_{r-1}+1}^{t_r-1}\mathbb{E}_{(\hat\rho_1,\dots,\hat\rho_t)}\ell(\hat{\mathbf{c}}_t,x_t),$$

and for $k\in[\![1,p]\!]$, $\mathbf{c}\in\mathscr{C}(k,R)$

$$L^{(r)}(k,\mathbf{c}) = \sum_{t=t_{r-1}+1}^{t_r-1}\ell(\mathbf{c},x_t).$$

Within each epoch $r=0,1,\dots,M$, since

$$\max_{t=t_{r-1}+1,t_{r-1}+2,\dots,t_r-1}|x_s|_2 \le R_{t_{r-1}}, \tag{2.24}$$

then applying Corollary 2.3 to each epoch r can give us that, for each $k\in[\![1,p]\!]$,

$$E^{(r)} - \inf_{\mathbf{c}\in\mathscr{C}(k,R_{t_{r-1}})} L^{(r)}(k,\mathbf{c}) \le \big(C(d,\eta)k + C(p,d)\big)R_{t_{r-1}}^2\sqrt{(t_r-1)\log(t_r-1)}, \tag{2.25}$$

where $C(d,\eta)=\frac{2(d+\eta)}{d+2}$ and $C(p,d)=\frac{2\log p+d}{d+2}+\frac{81(d+2)}{2}$.

In addition, since all observations $x_t, t = t_{r-1}+1,,\ldots,t_r-1$ in the epoch r are bounded in a convex ball $B_d\left(R_{t_{r-1}}\right)$, centered in $0 \in \mathbb{R}^d$ with radius $R_{t_{r-1}}$ as indicated by (2.24), we have for each $\mathbf{c}' \in \mathscr{C}(k,R) \setminus \mathscr{C}\left(k,R_{t_{r-1}}\right)$ (set of element in $\mathscr{C}(k,R)$ not in $\mathscr{C}\left(k,R_{t_{r-1}}\right)$), $k = 1,2,\ldots,p$ that

$$\inf_{\mathbf{c}\in\mathscr{C}\left(k,R_{t_{r-1}}\right)} L^{(r)}(k,\mathbf{c}) \leq L^{(r)}(k,\mathbf{c}'). \tag{2.26}$$

By (2.25) and (2.26), we can have that for any $k \in [\![1,p]\!]$ and $\mathbf{c} \in \mathscr{C}(k,R)$, the following inequality holds,

$$E^{(r)} - L^{(r)}(k,\mathbf{c}) \leq \left(C(d,\eta)k + C(p,d)\right) R_{t_{r-1}}^2 \sqrt{(t_r-1)\log(t_r-1)}.$$

Therefore, for any $\mathbf{c} \in \mathscr{C}(k,R)$, one has

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\ldots,\hat{\rho}_t)} \ell\left(\hat{\mathbf{c}}_t, x_t\right) - \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) &= \sum_{r=0}^{M} \left(E^{(r)} - L^{(r)}(k,\mathbf{c})\right) + \sum_{r=0}^{M} \left(\mathbb{E}_{(\hat{\rho}_1,\ldots,\hat{\rho}_{t_r})} \ell\left(\hat{\mathbf{c}}_{t_r}, x_{t_r}\right) - \ell(\mathbf{c}, x_{t_r})\right) \\
&\leq \sum_{r=0}^{M} \left[C(d,\eta)k + C(p,d)\right] R_{t_{r-1}}^2 \sqrt{(t_r-1)\log(t_r-1)} + 4\sum_{r=0}^{M} R_{t_r}^2 \\
&\leq \sum_{r=0}^{M} \left[C(d,\eta)k + C(p,d)\right] R_{t_{r-1}}^2 \sqrt{T\log T} + 4\sum_{r=0}^{M} R_{t_r}^2.
\end{aligned}$$

Since $R_{t_s} \geq 2^{s-r} R_{t_r}$ for $0 \leq r \leq s \leq M-1$, then for $s \leq M-1$,

$$\sum_{r=0}^{s} R_{t_r}^2 \leq \sum_{r=0}^{s} 4^{r-s} R_{t_s}^2 \leq \frac{4}{3} R_{t_s}^2.$$

Hence,

$$\begin{aligned}
\sum_{r=0}^{M} R_{t_{r-1}}^2 &\leq R_{t_{-1}}^2 + \frac{4}{3} R_{t_{M-1}}^2 \leq \frac{7}{3} R_{t_M}^2, \\
4\sum_{r=0}^{M} R_{t_r}^2 &\leq 4\left(\frac{4}{3} R_{t_{M-1}}^2 + R_{t_M}^2 +\right) \leq \frac{28}{3} R_{t_M}^2.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\ldots,\hat{\rho}_t)} \ell\left(\hat{\mathbf{c}}_t, x_t\right) - \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) &\leq \frac{7}{3} \left[C(d,\eta)k + C(p,d)\right] R_{t_M}^2 \sqrt{T\log T} + \frac{28}{3} R_{t_M}^2 \\
&\leq \frac{28}{3} \left[C(d,\eta)k + C(p,d)\right] R^2 \sqrt{T\log T} + \frac{112}{3} R^2,
\end{aligned}$$

where $R = \max_{t=1,2,\ldots,T} |x_t|_2$ and the second inequality is due to the fact that $R_{t_M} \leq 2R$. Taking the infimum of $\sum_{t=1}^{T} \ell(\mathbf{c}, x_t)$ over the set $\mathscr{C}(k,R)$, $k \in [\![1,p]\!]$ leads to

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\ldots,\hat{\rho}_t)} \ell\left(\hat{\mathbf{c}}_t, x_t\right) \leq \inf_{\mathbf{c}\in\mathscr{C}(k,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) + \frac{28}{3} \left[C(d,\eta)k + C(p,d)\right] R^2 \sqrt{T\log T} + \frac{112}{3} R^2.$$

Finally, taking the infimum of the right hand side of the above inequality with respect to k terminates the proof.

2.5.5 Proof of Theorem 2.2

The proof for the upper bound is straightforward: by replacing the loss function $\ell(\mathbf{c},x)$ by the penalized loss $\ell_\alpha(\mathbf{c},x)=\ell(\mathbf{c},x)+\alpha|\mathbf{c}|$ with $\alpha=\sqrt{\log T}/\sqrt{T}$ in the proof of Theorem 2.1, we obtain

$$\sum_{t=1}^{T}\mathbb{E}_{\hat\rho_1,\dots,\hat\rho_t}\ell_\alpha(\hat{\mathbf{c}}_t,x_t)\le\inf_{\rho\in\mathscr{P}_\pi(\mathscr{C})}\left\{\mathbb{E}_{\mathbf{c}\sim\rho}\left[\sum_{t=1}^{T}\ell_\alpha(\mathbf{c},x_t)\right]+\frac{\mathscr{K}(\rho,\pi)}{\lambda_T}\right.$$
$$\left.+\mathbb{E}_{(\hat\rho_1,\dots,\hat\rho_T)}\mathbb{E}_{\mathbf{c}\sim\rho}\left[\sum_{t=1}^{T}\frac{\lambda_{t-1}}{2}[\ell_\alpha(\mathbf{c},x_t)-\ell_\alpha(\hat{\mathbf{c}}_t,x_t)]^2\right]\right\},$$

and choosing $\lambda=\sqrt{\log T}/\sqrt{T}$ and $p=T^{\frac{1}{4}}$ yields the desired upper bound.

We now proceed to the proof of the lower bound. The trick is to replace the supremum over the (x_t) in $\mathscr{V}_T(s)$ by an expectation.

We first introduce the event $\Omega_{s,R}=\left\{(X_1,\dots,X_T)\in\mathbb{R}^{dT}:\text{ such that }\left|\mathbf{c}^\star_{T,R}\right|=s\right\}$, where $\mathbf{c}^\star_{T,R}$ is defined as in **Assumption** $\mathscr{H}(s)$. Then, we have

$$\mathscr{V}_T(s)\ge\inf_{(\hat\rho_t)}\mathbb{E}_{\mu^T}\left\{\sum_{t=1}^{T}\mathbb{E}_{(\hat\rho_1,\dots,\hat\rho_t)}\left(\ell(\hat{\mathbf{c}}_t,X_t)+\frac{\sqrt{\log T}}{\sqrt{T}}|\hat{\mathbf{c}}_t|\right)-\inf_{\mathbf{c}\in\mathscr{C}(s,R)}\sum_{t=1}^{T}\ell(\mathbf{c},X_t)\right\}\mathbb{1}\left(\Omega_{s,R}\right),$$

where $\mu^T\in\mathscr{P}(\mathbb{R}^{dT})$ is the joint distribution of i.i.d. sample $(X_1,\dots,X_T)$. Now, we have to choose μ in order to maximize the right-hand side of the above inequality. This is the purpose of the following lemmas.

Lemma 2.2. *Let $s\in\mathbb{N}^*, s\le p$. Let $\mu\in\mathscr{P}(\mathbb{R}^d)$ a distribution concentrated on $2s$ fixed points $\mathscr{S}_\mu=\{z_i,z_i+w,i=1,\dots,s\}$ such that $w=(2\Delta,0,\dots,0)\in\mathbb{R}^d$ with $\Delta>0$ and that $z_1,\dots,z_s\in B_d(R)$. Suppose that for any $i\ne j$, $d(z_i,z_j)\ge 2A\Delta$ for some $A>0$. Define μ as the uniform distribution over $\mathscr{S}_\mu$. Then, if $A>\sqrt{2}+1$, we have*

$$\arg\min_{\mathbf{c}\in\mathscr{C}(s,R)}\mathbb{E}_\mu\ell(\mathbf{c},X)=\{z_i+w/2,\quad i=1,\dots,s\}=:\mathbf{c}^\star_{\mu,s}.$$

The proof of Lemma 2.2 is similar to Bartlett *et al.* (1998, Section III.A, step 3). The next lemma controls the probability of the event $|\mathbf{c}^\star_{T,R}|\ne s$ with a proper choice of Δ^2 and A in the definition of μ.

Lemma 2.3. *Let $s\in\mathbb{N}^*$, $2\le s\le p$, and μ is defined in Lemma 2.2. Then, if we choose $A=\sqrt{2}s+1$ and*

$$\frac{2(s-1)s\sqrt{\log T}}{(A-1)^2\sqrt{T}}<\Delta^2<\frac{\sqrt{\log T}}{\sqrt{T}},$$

then for any $\epsilon>0$ and $T>8s^2\log\frac{2s^2}{\epsilon}$, we have

$$\mathbb{P}\left(\left|\mathbf{c}^\star_{T,R}\right|\ne s\right)\le\epsilon.$$

Proof. For any $k\in[\![1,p]\!]$, let $\mathbf{c}^\star_{T,k}$ firstly denote the optimal partition in $\mathscr{C}(k,R)$ that minimizes the penalized empirical loss on $(X_1,\dots,X_T)$, *i.e.*,

$$\mathbf{c}^\star_{T,k}=\arg\min_{\mathbf{c}\in\mathscr{C}(k,R)}\left\{\frac{1}{T}\sum_{t=1}^{T}\ell(\mathbf{c},X_t)+|\mathbf{c}|\frac{\sqrt{\log T}}{\sqrt{T}}\right\}.\tag{2.27}$$

Note that the optimal partition $\mathbf{c}^\star_{\mu,s}$ defined in Lemma 2.2 is also the partition minimizing the expected penalized loss, *i.e.*, $\mathbf{c}^\star_{\mu,s} = \mathbf{arg\,min}_{\mathbf{c}\in\mathscr{C}(s,R)} \left\{ \mathbb{E}_\mu \ell(\mathbf{c}, X) + |\mathbf{c}| \frac{\sqrt{\log T}}{\sqrt{T}} \right\}$ since $|\mathbf{c}| \frac{\sqrt{\log T}}{\sqrt{T}}$ equals to the same value $s\frac{\sqrt{\log T}}{\sqrt{T}}$ for all $\mathbf{c} \in \mathscr{C}(s,R)$. Next

$$\begin{aligned}
\mathbb{P}\left(\left|\mathbf{c}^\star_{T,R}\right| > s\right) &= \sum_{k=s+1}^{2s} \mathbb{P}\left(\left|\mathbf{c}^\star_{T,R}\right| = k\right) \\
&\leq \sum_{k=s+1}^{2s} \mathbb{P}\left(\frac{1}{T}\sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{T,k-1}, X_t\right) - \frac{1}{T}\sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{T,k}, X_t\right) > \sqrt{\frac{\log T}{T}} \right) \\
&\leq \sum_{k=s+1}^{2s} \mathbb{P}\left(\frac{1}{T}\sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{T,k-1}, X_t\right) > \sqrt{\frac{\log T}{T}} \right) \\
&\leq s\mathbb{P}\left(\frac{1}{T}\sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{\mu,s}, X_t\right) > \sqrt{\frac{\log T}{T}} \right) \\
&= s\mathbb{P}\left(\Delta^2 > \sqrt{\frac{\log T}{T}} \right) = 0, \qquad (2.28)
\end{aligned}$$

where the first inequality is induced by the definition of $\mathbf{c}^\star_{T,R}$; the third inequality is due to the fact that we have almost surely

$$\sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{\mu,s}, X_t\right) \geq \sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{T,s}, X_t\right) \geq \sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{T,k-1}, X_t\right), \quad \text{for } k > s. \qquad (2.29)$$

Note that the first inequality of (2.29) is due to the definition of $\mathbf{c}^\star_{T,s}$ since it minimizes the cumulative loss within all $\mathscr{C}(s,R)$, as indicated by the definition in (2.27) with $k = s$. Note that since all $\mathbf{c} \in \mathscr{C}(s,R)$, values for $|\mathbf{c}|$ would be the same. The second inequality is due to the fact that the number of $k-1$ of centers in $\mathbf{c}^\star_{T,k-1}$ is not smaller than the number s of centers in $\mathbf{c}^\star_{T,s}$.

Moreover, the first equality in the last line of (2.28) is by Lemma 2.2 where we recall that $\mathbf{c}^\star_{\mu,s} = \{z_1 + w/2, z_2 + w/2, \ldots, z_s + w/2\}$ is the best partition under the uniform distribution μ and its minimum squared distance to each X_t is Δ^2 (*i.e.*, $\ell(\mathbf{c}^\star_{\mu,s}, X_t) = \Delta^2$) since $X_t, t = 1, 2, \ldots, T$ locate either on z_i or $z_i + w$. Finally, the second equality in this line is due to the constraint (*i.e.*, $\Delta^2 < \sqrt{\log T}/\sqrt{T}$ in Lemma 2.3) imposed on the distance Δ between z_i and $z_i + w$, $i = 1, 2, \ldots, s$ when we construct them in Lemma 2.2. The equation (2.28) can be interpreted as follows: when the distance (*i.e.*, 2Δ) between z_i and its twin $z_i + w$ is sufficiently small, then the increasing of the penalty term $|\mathbf{c}|\sqrt{T \log T}$ in the assumption $\mathscr{H}(s)$ by adding the number of centers in clustering will surpass the reducing of $\sum_{t=1}^{T} \ell(\mathbf{c}, x_t)$, hence it is impossible that $\mathbf{c}^\star_{T,R}$ will have number of centers bigger than s.

In order to control the probability $\mathbb{P}(|\mathbf{c}^\star_{T,R}| < s)$, let us first consider the Voronoi partition of $\mathbb{R}^d$ induced by the set of points $\{z_i, z_i + w, i = 1, \ldots, s\}$ and for each i define V_i as the union of the Voronoi cells belonging to z_i and $z_i + w$. Let N_i denotes the number of X_t, $t = 1, \ldots, T$ falling in V_i. Hence $(N_1, \ldots, N_s)$ follows a multinomial distribution with

parameter $(T, q_1, q_2, \ldots, q_s)$, where $q_1 = q_2 = \cdots = q_s = 1/s$. Then

$$\begin{aligned}
\mathbb{P}\left(\left|\mathbf{c}^\star_{T,R}\right| < s\right) &= \sum_{k=1}^{s-1} \mathbb{P}\left(\left|\mathbf{c}^\star_{T,R}\right| = k\right) \\
&\leq \sum_{k=1}^{s-1} \mathbb{P}\left(\frac{1}{T}\sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{T,k}, X_t\right) - \frac{1}{T}\sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{T,s}, X_t\right) \leq \frac{(s-k)\sqrt{\log T}}{\sqrt{T}}\right) \\
&\leq \sum_{k=1}^{s-1} \mathbb{P}\left(\frac{1}{T}\sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{T,k}, X_t\right) - \frac{1}{T}\sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{\mu,s}, X_t\right) \leq \frac{(s-k)\sqrt{\log T}}{\sqrt{T}}\right) \\
&\leq (s-1)\mathbb{P}\left(\frac{1}{T}\min_{i=1,\ldots,s} N_i \cdot (A-1)^2\Delta^2 - \Delta^2 \leq \frac{(s-k)\sqrt{\log T}}{\sqrt{T}}\right) \\
&\leq (s-1)s\mathbb{P}\left(N_1 \leq \frac{T\Delta^2 + (s-1)\sqrt{T\log T}}{(A-1)^2\Delta^2}\right).
\end{aligned}$$

The third inequality is due to the fact that $\sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{T,k}, X_t\right) \geq \min_{i=1,\ldots,s} N_i (A-1)^2\Delta^2$ for $k < s$, and the last inequality holds since the marginal distributions of the N_is $(i = 1, \ldots, s)$ are the same binomial distribution with parameter $(T, 1/s)$. Finally, we can bound the last term by Hoeffding's inequality, *i.e.*, for any $t > 0$

$$\mathbb{P}(N_1 - \mathbb{E}(N_1) \leq -t) \leq 2\exp\left(-\frac{2t^2}{T}\right).$$

Hoeffding's inequality implies that if $s > 2, A = \sqrt{2}s+1, T > 8s^2 \log\frac{2s^2}{\epsilon}$ and $\Delta^2 > \frac{2s(s-1)\sqrt{\log T}}{(A-1)^2\sqrt{T}}$, then

$$\mathbb{P}\left(N_1 \leq \frac{T\Delta^2 + (s-1)\sqrt{T\log T}}{(A-1)^2\Delta^2}\right) < \frac{\epsilon}{s^2}.$$

□

Next, we proceed to the proof of Theorem 2.2. First of all, since $(X_1, \ldots, X_T)$ are i.i.d, following the distribution μ and by the definition of $\Omega_{s,R}$, we can write

$$\begin{aligned}
&\inf_{(\hat{\rho}_t)} \mathbb{E}_{\mu^T}\left\{\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\ldots,\hat{\rho}_t)}\left(\ell(\hat{\mathbf{c}}_t, X_t) + \sqrt{\frac{\log T}{T}}|\hat{\mathbf{c}}_t|\right)\right\} \mathbb{1}\left(\Omega_{s,R}\right) \\
=&\inf_{(\hat{\rho}_t)} \mathbb{E}_{(\hat{\rho}_1,\ldots,\hat{\rho}_T)} \sum_{t=1}^{T} \mathbb{E}_{\mu^T}\left[\left(\ell(\hat{\mathbf{c}}_t, X_t) + \sqrt{\frac{\log T}{T}}|\hat{\mathbf{c}}_t|\right)\mathbb{1}(\Omega_{s,R})\right] \\
\geq&\mathbb{E}_{\mu^T}\left\{\sum_{t=1}^{T} \ell(\hat{\mathbf{c}}, X_t) + \sqrt{T\log T}|\hat{\mathbf{c}}|\right\} \mathbb{1}\left(\Omega_{s,R}\right) \\
\geq&\mathbb{E}_{\mu^T}\left\{\sum_{t=1}^{T} \ell(\mathbf{c}^\star_{T,R}, X_t) + s\sqrt{T\log T}\right\} \mathbb{1}\left(\Omega_{s,R}\right) \\
\geq&\mathbb{E}_{\mu^T}\left\{\sum_{t=1}^{T} \ell(\mathbf{c}^\star_{T,R}, X_t)\right\}\left(1 - \mathbb{1}\left(\Omega^C_{s,R}\right)\right) + s\sqrt{T\log T}\mathbb{P}\left(\Omega_{s,R}\right) \\
\geq&\mathbb{E}_{\mu^T}\left\{\sum_{t=1}^{T} \ell(\mathbf{c}^\star_{T,R}, X_t)\right\} - T\Delta^2\mathbb{P}\left(\Omega^C_{s,R}\right) + s\sqrt{T\log T}\left(\mathbb{P}\left(\Omega_{s,R}\right) - \mathbb{P}\left(\Omega^C_{s,R}\right)\right) \\
\geq&T\inf_{\mathbf{c}\in\mathscr{C}(s,R)} \mathbb{E}_\mu \ell(\mathbf{c}, X) - T\Delta^2\mathbb{P}\left(\Omega^C_{s,R}\right) + s\sqrt{T\log T}\left(\mathbb{P}\left(\Omega_{s,R}\right) - \mathbb{P}\left(\Omega^C_{s,R}\right)\right),
\end{aligned}$$

where $\hat{\mathbf{c}}$ in the first inequality is given by

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}\in\mathscr{C}} \mathbb{E}_{\mu^T}\left[\left(\ell(\mathbf{c}, X_t) + |\mathbf{c}|\sqrt{\log T}/\sqrt{T}\right)\mathbb{1}(\Omega_{s,R})\right].$$

Note that $\hat{\mathbf{c}}$ does not depend on t since $X_t, t = 1, \ldots, T$ are i.i.d and μ is a symmetric uniform distribution (definition in Lemma 2.2). The second inequality is due to Jensen's inequality and the fourth inequality relies on the fact that with the definition of $\mathbf{c}^\star_{T,R}$ and μ, we have almost surely that

$$\sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{T,R}, X_t\right) \le \sum_{t=1}^{T} \ell\left(\mathbf{c}^\star_{\mu,s}, X_t\right) + s\sqrt{T\log T} = T\Delta^2 + s\sqrt{T\log T},$$

where $\Delta > 0$ is related with the choice of μ in Lemma 2.2 and its value is constrained according to Lemma 2.3. Then we obtain for any $\epsilon > 0$

$$\inf_{(\hat{\rho}_t)} \mathbb{E}_{\mu^T}\left\{\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\ldots,\hat{\rho}_t)}\ell(\hat{\mathbf{c}}_t, X_t) + \frac{\sqrt{\log T}}{\sqrt{T}}|\hat{\mathbf{c}}_t|\right\}\mathbb{1}\left(\Omega_{s,R}\right) \ge T \inf_{\mathbf{c}\in\mathscr{C}(s,R)} \mathbb{E}_\mu \ell(\mathbf{c}, X) - T\epsilon\Delta^2 + s\sqrt{T\log T}(1-2\epsilon). \quad (2.30)$$

Moreover, by Jensen's inequality

$$\mathbb{E}_{\mu^T}\left[\inf_{\mathbf{c}\in\mathscr{C}(s,R)} \sum_{t=1}^{T} \ell(\mathbf{c}, X_t)\mathbb{1}\left(\Omega_{s,R}\right)\right] \le T \inf_{\mathbf{c}\in\mathscr{C}(s,R)} \mathbb{E}_\mu \ell(\mathbf{c}, X). \quad (2.31)$$

Combining (2.30) and (2.31), we obtain

$$\mathcal{V}_T(s) \ge s\sqrt{T\log T}\left(1 - 2\epsilon\left[1 + \frac{\sqrt{T}\Delta^2}{2s\sqrt{\log T}}\right]\right). \quad (2.32)$$

Furthermore, by taking $\epsilon = 1/T$ and choosing the minimum value of Δ^2 allowed in Lemma 2.3, (2.32) yields

$$\mathcal{V}_T(s) \ge s\sqrt{T\log T}\left(1 - \frac{2}{T}\left[1 + \frac{s-1}{2s^2}\right]\right).$$

Finally, we need to ensure that s pairs of points $\{z_i, z_i + w\}$ can be packed in $B_d(R)$ such that the distance between any two of the z_is is at least $2A$. A sufficient condition (Kolmogorov and Tikhomirov, 1961) is

$$s \le \left(\frac{R - 2\Delta}{2A\Delta}\right)^d.$$

If $\Delta \le R/6$ (which is satisfied if T is large enough), the above inequality holds if

$$s \le \left(\frac{R}{3A\Delta}\right)^d.$$

As $A = \sqrt{2}s + 1$ and $\Delta^2 < \sqrt{\log T}/\sqrt{T}$, we get the desired result.

2.5.6 Proof of Lemma 2.1

Let D_n denote the event that no "within-model move" is ever accepted in the first n moves. Then $D_1 = D_1^{\text{within}} \cup D_1^{\text{between}}$, where D_1^{within} stands for the event that a "within-model move" is proposed but rejected in one step and D_1^{between} that a "between-model move" is proposed in one step. Then we have

$$\begin{aligned}\mathbb{P}\left[D_1|(k^{(0)},\mathbf{c}^{(0)})=(k,\mathbf{c})\right] =& \mathbb{P}\left[k'\neq k|(k,\mathbf{c})\right]+\mathbb{P}\left[k'=k,\text{but rejected}|(k,\mathbf{c})\right]\\ =&\frac{2}{3}+\frac{1}{3}\left[1-\int_{\mathbb{R}^{dk}}\alpha\left[(k,\mathbf{c}),(k,\mathbf{c}')\right]\rho_k\left(\mathbf{c}',\mathfrak{c}_k,\tau_k\right)\mathrm{d}\mathbf{c}'\right],\end{aligned}$$

where

$$\begin{aligned}\alpha\left[(k,\mathbf{c}),(k,\mathbf{c}')\right] &= \min\left\{1,\frac{\hat{\rho}_t(\mathbf{c}')\rho_k(\mathbf{c},\mathfrak{c}_k,\tau_k)}{\hat{\rho}_t(\mathbf{c})\rho_k(\mathbf{c}',\mathfrak{c}_k,\tau_k)}\right\}\\ &= \min\left\{1,h_t\left(\mathbf{c}'|(k,\mathbf{c})\right)\right\}.\end{aligned}$$

Under the assumption of $k'=k$, we have that $\mathbf{c}',\mathbf{c}\in\mathbb{R}^{dk}$, therefore the restriction of $\hat{\rho}_t$ to $\mathbb{R}^{dk}$ is well defined. Moreover, by the definition of π_k in (2.6), the support of the restriction of $\hat{\rho}_t$ to $\mathbb{R}^{dk}$ is $\mathbb{R}^{dk}\cap\mathscr{E}=(B_d(2R))^k$. Hence the function $(\mathbf{c}',\mathbf{c})\mapsto h_t\left(\mathbf{c}'|(k,\mathbf{c})\right)$ is strictly positive and continuous on the compact set $(B_d(2R))^k\times(B_d(2R))^k$. As a consequence, the minimum of $h_t\left(\mathbf{c}'|(k,\mathbf{c})\right)$ on $(B_d(2R))^k\times(B_d(2R))^k$ is achieved and we denote it by m_k, *i.e.*,

$$m_k=\inf_{\mathbf{c}',\mathbf{c}\in(B_d(2R))^k} h_t\left(\mathbf{c}'|(k,\mathbf{c})\right)>0.$$

In addition, due to the continuity and positivity of ρ_k on $\mathbb{R}^{dk}$, it is clear that for any $k\in[\![1,p]\!]$

$$z_k=\int_{(B_d(2R))^k}\rho_k\left(\mathbf{c}',\mathfrak{c}_k,\tau_k\right)\mathrm{d}\mathbf{c}'>0.$$

Therefore, for any k,

$$\begin{aligned}\int_{\mathbb{R}^{dk}}\alpha\left[(k,\mathbf{c}),(k,\mathbf{c}')\right]\rho_k\left(\mathbf{c}',\mathfrak{c}_k,\tau_k\right)\mathrm{d}\mathbf{c}' &\geq \inf_{k\in[\![1,p]\!]}(m_k z_k)\\ &=: m^\star>0.\end{aligned}$$

Hence, uniformly on $k\in[\![1,p]\!]$ and $\mathbf{c}\in\mathbb{R}^{dk}\cap\mathscr{E}$, we have,

$$\mathbb{P}[D_1|(k,\mathbf{c})]\leq\left[\frac{2}{3}+\frac{1}{3}(1-m^\star)\right]<1.$$

To conclude,

$$\mathbb{P}[D|(k,\mathbf{c})]=\lim_{n\longrightarrow\infty}\mathbb{P}[D_n|(k,\mathbf{c})]\leq\lim_{n\longrightarrow\infty}\left[\frac{2}{3}+\frac{1}{3}(1-m^\star)\right]^n=0.$$

2.5.7 Proof of Theorem 2.3

For any $\mathbf{c}\in\mathscr{E}$, there exists some $k\in[\![1,p]\!]$ such that $\mathbf{c}\in(B_d(2R))^k\subset\mathscr{E}$. For any $k'\in[\![k-1,k+1]\!]$ and for any $A\in\mathscr{B}\left(\mathbb{R}^{dk'}\right)$ such that $\hat{\rho}_t(A)>0$, the transition kernel H of the chain is given by

$$H\left(\mathbf{c},\mathbf{c}'\in A\right)=\int\mathbb{1}_{\{v_1\in A\}}\alpha\left[(k,\mathbf{c}),\left(k',v_1\right)\right]q(k,k')\rho_{k'}(v_1,\mathfrak{c}_{k'},\tau_{k'})\mathrm{d}v_1+r(\mathbf{c})\delta_{\mathbf{c}}(A),\qquad(2.33)$$

where $\rho_{k'}(\cdot, \mathfrak{c}_{k'}, \tau_{k'})$ is the rescaled Student distribution in (2.14) and

$$r(\mathbf{c}) = \sum_{k' \in [\![k-1, k+1]\!]} q(k, k') \int \left(1 - \alpha\left[(k, \mathbf{c}), (k', v_1)\right]\right) \rho_{k'}(v_1, \mathfrak{c}_{k'}, \tau_{k'}) \mathrm{d}v_1$$

is the probability of rejection when starting at state $\mathbf{c}$, and $\delta_{\mathbf{c}}(\cdot)$ is a Dirac measure in $\mathbf{c}$. One can easily note that $H(\mathbf{c}, \mathbf{c}' \in A)$ in (2.33) is strictly positive, indicating that the chain, when starting from $\mathbf{c}$, has a positive chance to move. Therefore, for any $A \in \mathscr{B}(\mathscr{C})$ such that $\hat{\rho}_t(A) > 0$, we can prove with the Chapman-Kolmogorov equation that there exists some $m \in \mathbb{N}^*$ such that

$$H^m(\mathbf{c}, A) > 0,$$

where $H^m(\mathbf{c}, A) = \int H^{m-1}(y, A) H(\mathbf{c}, \mathrm{d}y)$ is the m-step transition kernel. In other words, the chain is $\hat{\rho}_t$-irreducible. Finally, a sufficient condition for the chain to be aperiodic is that Algorithm 2.3 allows transitions such as $\left\{\left(k^{(n+1)}, \mathbf{c}^{(n+1)}\right) = \left(k^{(n)}, \mathbf{c}^{(n)}\right)\right\}$, *i.e.*,

$$\mathbb{P}\left(\alpha\left[(k^{(n)}, \mathbf{c}^{(n)}), (k', \mathbf{c}')\right] < 1\right) = \mathbb{P}\left(\frac{\hat{\rho}_t(\mathbf{c}') q(k', k^{(n)}) \rho_{k^{(n)}}(\mathbf{c}^{(n)}, \mathfrak{c}_{k^{(n)}}, \tau_{k^{(n)}})}{\hat{\rho}_t(\mathbf{c}^{(n)}) q(k^{(n)}, k') \rho_{k'}(\mathbf{c}', \mathfrak{c}_{k'}, \tau_{k'})} < 1\right) > 0. \tag{2.34}$$

Since for any $\mathbf{c}' \in A \subset \mathscr{B}\left(\mathbb{R}^{dk'}\right) \cap \mathscr{E}^c$ such that $\mathbb{P}\left(\mathbf{c}' \in A\right) = \int_A \rho_{k'}(\mathbf{c}', \mathfrak{c}_{k'}, \tau_{k'}) \mathrm{d}\mathbf{c}' > 0$, we have $\hat{\rho}_t(\mathbf{c}') = 0$, (2.34) holds. Therefore,

$$\mathbb{P}\left(\frac{\hat{\rho}_t(\mathbf{c}') q(k', k^{(n)}) \rho_{k^{(n)}}(\mathbf{c}^{(n)}, \mathfrak{c}_{k^{(n)}}, \tau_{k^{(n)}})}{\hat{\rho}_t(\mathbf{c}^{(n)}) q(k^{(n)}, k') \rho_{k'}(\mathbf{c}', \mathfrak{c}_{k'}, \tau_{k'})} < 1\right) \geq \mathbb{P}\left(\mathbf{c}' \in A\right) > 0.$$

The chain is therefore aperiodic. Finally, the Harris recurrence of the chain is a consequence of Lemma 2.1 (based on Roberts and Rosenthal, 2006, Theorem 20). As a conclusion, the chain converges to the target distribution $\hat{\rho}_t$.

2.6 Appendix

For the sake of completeness, this appendix presents additional regret bounds for a different heavy-tailed prior. Doing so, we stress that the quasi-Bayesian approach is flexible in the sense that it allows for regret bounds for a large variety of priors.

A d-multivariate Student distribution (as presented in Kotz and Nadarajah, 2004) is defined as the ratio between a Gaussian vector and the square root of an independent χ^2 with v degrees of freedom and has the density

$$\frac{\Gamma(\frac{d+v}{2})}{\Gamma(\frac{v}{2}) v^{\frac{d}{2}} \pi^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} \left[1 + \frac{(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)}{v}\right]^{-\frac{v+d}{2}},$$

where μ, Σ, v are parameters. Here, we consider a different distribution: a rescaled Student distribution with $v = 3$ and $\Sigma = 2\tau^2 I_d$, where $\tau > 0$ is a scale parameter and I_d is the d-dimensional identity matrix. This distribution has the density

$$\frac{\Gamma(\frac{d+3}{2})}{\Gamma(\frac{3}{2}) 3^{\frac{d}{2}} \pi^{\frac{d}{2}} (2\tau^2)^{\frac{d}{2}}} \left[1 + \frac{|\mathbf{x} - \mu|_2^2}{6\tau^2}\right]^{-\frac{3+d}{2}} \triangleq A_{d,\tau} \left[1 + \frac{|\mathbf{x} - \mu|_2^2}{6\tau^2}\right]^{-\frac{3+d}{2}}, \tag{2.35}$$

where $A_{d,\tau}$ stands for the renormalizing constant. In this setting, the parameters for our rescaled Student distribution are the mean vector μ and the scale parameter $\tau > 0$.

This rescaled Student distribution is a generalization of the prior introduced by Dalalyan and Tsybakov (2012a) where a product of univariate Student is considered and serves to enforce sparsity (since it is heavy-tailed).

Let us consider π_k as a product of k independent truncated rescaled Student distributions in $\mathbb{R}^d$, namely, for any $\mathbf{c} \in \mathbb{R}^{dk} \subset \mathscr{C}$,

$$\mathrm{d}\pi_k(\mathbf{c}, \tau_0, 2R) = \prod_{j=1}^{k} \left\{ C_{2R,\tau_0}^{-1} \left(1 + \frac{|c_j|_2^2}{6\tau_0^2}\right)^{-\frac{3+d}{2}} \mathbb{1}_{\{|c_j|_2 \le 2R\}} \right\} \mathrm{d}\mathbf{c}, \tag{2.36}$$

where $\tau_0 > 0$ and $R > 0$ are respectively the scale and truncation parameters, and C_{2R,τ_0} is the normalizing constant accounting for the truncation. When $R = +\infty$, $\pi_k(\mathbf{c}, \tau_0, 2R)$ amounts to a distribution without truncation. In the following, we shorten $\pi_k(\mathbf{c}, \tau_0, 2R)$ to π_k whenever no confusion is possible.

Denote by ν the rescaled Student distribution in $\mathbb{R}^d$, with mean vector $0 \in \mathbb{R}^d$ and scale parameter 1. Fix $k \in [\![1, p]\!]$, $R > 0$ and $\mathfrak{c} \in \mathscr{C}(k, R)$, and recall that $\Xi(k, R)$ denotes the hypercube in $\mathbb{R}^k$ defined by

$$\Xi(k, R) := \left\{ \xi = (\xi_j)_{j=1,\dots,k} \in \mathbb{R}^k : 0 < \xi_j \le R, \forall j \right\}.$$

For any $k \in [\![1, p]\!]$, $\mathbf{c} \in \mathbb{R}^{dk} \subset \mathscr{C}$, $\mathfrak{c} \in \mathscr{C}(k, R)$, $\xi \in \Xi(k, R)$, $0 < \tau^2 \le \sqrt{3}R^2/(6\sqrt{d})$ and $R > 0$, we define the probability distribution ρ_k on $\mathbb{R}^{dk}$ by

$$\rho_k(\mathbf{c}, \mathfrak{c}, \tau, \xi) = \prod_{j=1}^{k} \left\{ C_{\xi_j,\tau}^{-1} \left(1 + \frac{|c_j - \mathfrak{c}_j|_2^2}{6\tau^2}\right)^{-\frac{3+d}{2}} \mathbb{1}_{\{|c_j - \mathfrak{c}_j|_2 \le \xi_j\}} \right\}, \tag{2.37}$$

where $C_{\xi_j,\tau}$ are normalizing constants defined as $C_{\xi_j,\tau} = \mathbb{P}\left(|\nu|_2 \le \xi_j/\sqrt{2}\tau\right)/A_{d,\tau}$, where $A_{d,\tau}$ is the constant defined in (2.35). Moreover, when $(\xi_j)_{j=1,\dots,k} = +\infty$, we let $\rho_k(\mathbf{c}, \mathfrak{c}, \tau, \xi)$ denote the rescaled Student distribution without truncation. In the sequel, we shall shorten $\rho_k(\mathbf{c}, \mathfrak{c}, \tau, \xi)$ to ρ_k whenever no confusion is possible.

Lemma 2.4. *Assume that q and π_k in (2.3) are defined respectively as in (2.5) and (2.36), and that ρ_k is defined as (2.37) for each $k \in [\![1, p]\!]$. For the probability distribution $\rho(\mathbf{c}, \mathfrak{c}, \tau, \xi) = \mathbb{1}_{\{\mathbf{c} \in \mathbb{R}^{dk}\}} \rho_k(\mathbf{c}, \mathfrak{c}, \tau, \xi)$ defined on $\mathscr{C}$, if $R \ge \max_{t=1,\dots,T} |x_t|_2$, then*

$$\begin{aligned}\mathscr{K}(\rho, \pi) \le \sum_{j=1}^{k} \left[\frac{3+d}{2} \log\left(1 + \frac{\xi_j^2}{6\tau^2}\right) - \frac{d}{2} \log \xi_j^2 \right] - k \log c_d \\ + (3+d)k \log\left(1 + \frac{\tau}{\tau_0} + \frac{\sum_{j=1}^{k} |\mathfrak{c}_j|_2}{\sqrt{6}k\tau_0}\right) + kd \log \tau_0 + \log p + \eta(k-1).\end{aligned}$$

Proof. By the definition of the Kullback-Leibler divergence, we have

$$\mathscr{K}(\rho, \pi) = \mathscr{K}(\rho_k, \pi_k) + \log \frac{1}{q(k)} =: A + B, \tag{2.38}$$

where

$$A = \int_{\mathbb{R}^{dk}} \log \left[\prod_{j=1}^{k} \frac{C_{2R,\tau_0}}{C_{\xi_j,\tau}} \left(\frac{\tau_0^2}{\tau^2} \frac{6\tau^2 + |c_j - \mathfrak{c}_j|_2^2}{6\tau_0^2 + |c_j|_2^2} \right)^{-\frac{3+d}{2}} \right] \rho_k(\mathbf{c}) \mathrm{d}\mathbf{c}$$

$$
\begin{aligned}
&= \sum_{j=1}^{k} \log \frac{C_{2R,\tau_0}}{C_{\xi_j,\tau}} + \frac{3+d}{2} \int_{\mathbb{R}^{dk}} \sum_{j=1}^{k} \log \left(\frac{\tau^2}{\tau_0^2} \frac{6\tau_0^2 + |c_j|_2^2}{6\tau^2 + |c_j - \mathfrak{c}_j|_2^2} \right) \rho_k(\mathbf{c}) \mathrm{d}\mathbf{c} \\
&= \sum_{j=1}^{k} \log \frac{\mathbb{P}\left(|\nu|_2 \le \frac{2R}{\sqrt{2}\tau_0} \right)}{\mathbb{P}\left(|\nu)|_2 \le \frac{\xi_j}{\sqrt{2}\tau} \right)} + kd \log \frac{\tau_0}{\tau} + \frac{3+d}{2} \int_{\mathbb{R}^{dk}} \sum_{j=1}^{k} \log \left(\frac{\tau^2}{\tau_0^2} \frac{6\tau_0^2 + |c_j|_2^2}{6\tau^2 + |c_j - \mathfrak{c}_j|_2^2} \right) \rho_k(\mathbf{c}) \mathrm{d}\mathbf{c} \\
&=: A_1 + A_2 + A_3. \qquad (2.39)
\end{aligned}
$$

By the definition of the rescaled Student distribution ν,

$$
\begin{aligned}
\mathbb{P}\left(|\nu|_2 \le \frac{\xi_j}{\sqrt{2}\tau} \right) &= \int_{|\nu|_2 \le \frac{\xi_j}{\sqrt{2}\tau}} \frac{\Gamma(\frac{3+d}{2})}{\Gamma(\frac{3}{2})(3\pi)^{\frac{d}{2}}} \left(1 + \frac{|\nu|_2^2}{3} \right)^{-\frac{3+d}{2}} \mathrm{d}\nu \\
&\ge \left(1 + \frac{\xi_j^2}{6\tau^2} \right)^{-\frac{3+d}{2}} \frac{\Gamma(\frac{3+d}{2})}{\Gamma(\frac{3}{2})(3\pi)^{\frac{d}{2}}} \int_{|\nu|_2 \le \frac{\xi_j}{\sqrt{2}\tau}} \mathrm{d}\nu \\
&= c_d \tau^{-d} \left(1 + \frac{\xi_j^2}{6\tau^2} \right)^{-\frac{3+d}{2}} \xi_j^d,
\end{aligned}
$$

where $\Gamma(\cdot)$ is the Gamma function and $c_d = \frac{\Gamma\left(\frac{3+d}{2}\right)}{\Gamma(\frac{3}{2})\Gamma\left(\frac{d}{2}+1\right)6^{\frac{d}{2}}}$. Hence, the term A_1 in (2.39) verifies

$$
\begin{aligned}
A_1 &= k \log \mathbb{P}\left(|\nu|_2 \le \frac{2R}{\sqrt{2}\tau_0} \right) - \sum_{j=1}^{k} \log \mathbb{P}\left(|\nu)|_2 \le \frac{\xi_j}{\sqrt{2}\tau} \right) \\
&\le - \sum_{j=1}^{k} \log \mathbb{P}\left(|\nu|_2 \le \frac{\xi_j}{\sqrt{2}\tau} \right) \\
&\le \sum_{j=1}^{k} \left[\frac{3+d}{2} \log \left(1 + \frac{\xi_j^2}{6\tau^2} \right) - \frac{d}{2} \log \xi_j^2 \right] + kd \log \tau - k \log c_d. \qquad (2.40)
\end{aligned}
$$

In addition, we have

$$
\begin{aligned}
\frac{6\tau_0^2 + |c_j|_2^2}{6\tau^2 + |c_j - \mathfrak{c}_j|_2^2} &\le 1 + \frac{2|\mathfrak{c}_j|_2}{2\sqrt{6}\tau} \frac{2\sqrt{6}\tau |c_j - \mathfrak{c}_j|_2}{6\tau^2 + |c_j - \mathfrak{c}_j|_2^2} + \frac{|\mathfrak{c}_j|_2^2}{6\tau^2 + |c_j - \mathfrak{c}_j|_2^2} + \frac{\tau_0^2}{\tau^2} \\
&= 1 + \frac{|\mathfrak{c}_j|_2}{\sqrt{6}\tau} + \frac{|\mathfrak{c}_j|_2^2}{6\tau^2} + \frac{\tau_0^2}{\tau^2} \le \left(1 + \frac{|\mathfrak{c}_j|_2}{\sqrt{6}\tau} + \frac{\tau_0}{\tau} \right)^2,
\end{aligned}
$$

where we used the Cauchy–Schwarz inequality. Due to the above inequality, the term A_3 in (2.39) satisfies

$$
\begin{aligned}
A_3 &\le (3+d) \int \sum_{j=1}^{k} \log \left(1 + \frac{\tau}{\tau_0} + \frac{|\mathfrak{c}_j|_2}{\sqrt{6}\tau_0} \right) \rho_k(\mathbf{c}) \mathrm{d}\mathbf{c} \\
&\le (3+d)k \int \log \left(1 + \frac{\tau}{\tau_0} + \frac{\sum_{j=1}^{k} |\mathfrak{c}_j|_2}{\sqrt{6}k\tau_0} \right) \rho_k(\mathbf{c}) \mathrm{d}\mathbf{c} \\
&= (3+d)k \log \left(1 + \frac{\tau}{\tau_0} + \frac{\sum_{j=1}^{k} |\mathfrak{c}_j|_2}{\sqrt{6}k\tau_0} \right). \qquad (2.41)
\end{aligned}
$$

Combining (2.38), (2.39), (2.40), (2.41) with (2.17) completes the proof. □

Corollary 2.5. *For any sequence $(x_t)_{1:T} \in \mathbb{R}^{dT}$, for any $\lambda > 0$, if q and π_k in (2.3) are taken respectively as in (2.5) and (2.36) with parameter $\eta \geq 0$, $\tau_0 > 0$ and $R \geq \max_{t=1,\dots,T} |x_t|_2$, Algorithm 2.1 satisfies, for any $0 < \tau^2 \leq (\sqrt{3}R^2)/(6\sqrt{d})$,*

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\dots,\hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) \leq \inf_{k \in [\![1,p]\!]} \inf_{\mathbf{c} \in \mathscr{C}(k,R)} \left\{ \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) + \frac{kd}{\lambda} \log \frac{\tau_0}{c_d \tau} + \frac{\eta}{\lambda} k \right.$$
$$\left. + \frac{(3+d)k}{\lambda} \log \left(1 + \frac{\tau}{\tau_0} + \frac{\sum_{j=1}^{k} |c_j|_2}{\sqrt{6k}\tau_0} \right) + \frac{1}{\lambda} \sqrt{kd(12\tau^2 T\lambda + 3k)} \right\} + \frac{\lambda T}{2} C_1^2 + \frac{\log p}{\lambda},$$

where $C_1 = (2R + \max_{t=1,\dots,T} |x_t|_2)^2$ and $c_d = \left(\frac{\Gamma(\frac{3+d}{2})}{\Gamma(\frac{3}{2})\Gamma(\frac{d}{2}+1)} \right)^{1/d}$.

Proof. By Proposition 2.1,

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\dots,\hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) \leq \inf_{k \in [\![1,p]\!]} \inf_{\substack{\rho \in \mathscr{P}_\pi(\mathscr{C}) \\ \rho = \rho_k \mathbb{1}_{\{\mathbf{c} \in \mathbb{R}^{dk}\}}}} \left\{ \mathbb{E}_{\mathbf{c} \sim \rho} \sum_{t=1}^{T} [\ell(\mathbf{c}, x_t)] + \frac{\mathscr{K}(\rho, \pi)}{\lambda} \right.$$
$$\left. + \frac{\lambda}{2} \mathbb{E}_{(\hat{\rho}_1,\dots,\hat{\rho}_T)} \mathbb{E}_{\mathbf{c} \sim \rho} \sum_{t=1}^{T} [\ell(\mathbf{c}, x_t) - \ell(\hat{\mathbf{c}}_t, x_t)]^2 \right\} \tag{2.42}$$

As in (2.16), the first term on the right-hand side of (2.42) may be upper bounded.

$$\sum_{t=1}^{T} \mathbb{E}_{\mathbf{c} \sim \rho} [\ell(\mathbf{c}, x_t)] \leq \sum_{t=1}^{T} \ell(m, x_t) + T \max_{j=1,\dots,k} \xi_j^2. \tag{2.43}$$

For the second term in the right-hand side of (2.42), by Lemma 2.4,

$$\frac{\mathscr{K}(\rho, \pi)}{\lambda} \leq \frac{(3+d)k}{\lambda} \log \left(1 + \frac{\tau}{\tau_0} + \frac{\sum_{j=1}^{k} |\mathfrak{c}_j|_2}{\sqrt{6k}\tau_0} \right) + \frac{1}{\lambda} \sum_{j=1}^{k} \left[\frac{3+d}{2} \log \left(1 + \frac{\xi_j^2}{6\tau^2} \right) - \frac{d}{2} \log \xi_j^2 \right]$$
$$+ \frac{kd}{\lambda} \log \tau_0 - \frac{k}{\lambda} \log c_d + \frac{\eta}{\lambda}(k-1) + \frac{\log p}{\lambda}. \tag{2.44}$$

Likewise to (2.19), the third term on the right-hand side of (2.42) is upper bounded by

$$\frac{\lambda}{2} \mathbb{E}_{(\hat{\rho}_1,\dots,\hat{\rho}_T)} \mathbb{E}_{\mathbf{c} \sim \rho_k} \sum_{t=1}^{T} [\ell(\mathbf{c}, x_t) - \ell(\hat{\mathbf{c}}_t, x_t)]^2 \leq \frac{\lambda T}{2} C_1^2. \tag{2.45}$$

Combining inequalities (2.43), (2.44) and (2.45) yields for $\xi \in \Xi(k,R)$ and $0 < \tau^2 \leq \sqrt{3}R^2/(6\sqrt{d})$ that

$$\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\dots,\hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) \leq \inf_{k \in [\![1,p]\!]} \inf_{\mathfrak{c} \in \mathscr{C}(k,R)} \left\{ \sum_{t=1}^{T} \ell(\mathfrak{c}, x_t) + \xi_j^2 + \frac{(3+d)k}{\lambda} \log \left(1 + \frac{\tau}{\tau_0} + \frac{\sum_{j=1}^{k} |\mathfrak{c}_j|_2}{\sqrt{6k}\tau_0} \right) \right.$$
$$\left. + T \max_{j=1,\dots,k} \xi_j^2 + \frac{3+d}{2\lambda} \sum_{j=1}^{k} \log \left(1 + \frac{\xi_j^2}{6\tau^2} \right) - \frac{d}{2\lambda} \sum_{j=1}^{k} \log \xi_j^2 + \frac{kd}{\lambda} \log \tau_0 - \frac{k}{\lambda} \log c_d + (k-1) \right\}$$
$$+ \frac{\lambda T}{2} C_1^2 + \frac{\log p}{\lambda}.$$

Let $\hat{\xi}_j = \xi_j^2/6\tau^2$ for any $j = 1,\dots,k$, then $0 < \hat{\xi}_j \le R^2/6\tau^2$ since $\xi = (\xi_j)_{j=1,\dots,k} \in \Xi(k,R)$. This yields

$$\begin{aligned}
&T \max_{j=1,\dots,k} \xi_j^2 + \frac{3+d}{2\lambda} \sum_{j=1}^{k} \log\left(1 + \frac{\xi_j^2}{6\tau^2}\right) - \frac{d}{2\lambda} \sum_{j=1}^{k} \log \xi_j^2 \\
&= 6\tau^2 T \max_{j=1,\dots,k} \hat{\xi}_j + \frac{3}{2\lambda} \sum_{j=1}^{k} \log\left(1+\hat{\xi}_j\right) + \frac{d}{2\lambda} \sum_{j=1}^{k} \log\left(1 + \frac{1}{\hat{\xi}_j}\right) - \frac{kd}{2\lambda} \log(6\tau^2) \\
&\le 6\tau^2 T \max_{j=1,\dots,k} \hat{\xi}_j + \frac{3}{2\lambda} \sum_{j}^{k} \hat{\xi}_j + \frac{d}{2\lambda} \sum_{j=1}^{k} \frac{1}{\hat{\xi}_j} - \frac{kd}{2\lambda} \log(6\tau^2) \\
&\le \left(6\tau^2 T + \frac{3k}{2\lambda}\right) \max_{j=1,\dots,k} \hat{\xi}_j + \frac{d}{2\lambda} \sum_{j=1}^{k} \frac{1}{\hat{\xi}_j} - \frac{kd}{2\lambda} \log(6\tau^2). \qquad (2.46)
\end{aligned}$$

The minimum of the right-hand side of (2.46) is reached for

$$\hat{\xi}_1 = \dots = \hat{\xi}_k = \sqrt{\frac{kd}{12\tau^2 T\lambda + 3k}} \le \frac{R^2}{6\tau^2}, \quad \text{if } 0 < \tau^2 \le \frac{\sqrt{3}R^2}{6\sqrt{d}}.$$

Therefore for a fixed k, $\mathfrak{c} \in \mathscr{C}(k,R)$ and $0 < \tau^2 \le \frac{\sqrt{3}R^2}{6\sqrt{d}}$,

$$\begin{aligned}
\inf_{\xi \in \Xi(k,R)} &\left\{ T \max_{j=1,\dots,k} \xi_j^2 + \frac{3+d}{2\lambda} \sum_{j=1}^{k} \log\left(1 + \frac{\xi_j^2}{6\tau^2}\right) - \frac{d}{2\lambda} \sum_{j=1}^{k} \log \xi_j^2 \right\} \\
&\le \frac{1}{\lambda} \sqrt{kd(12\tau^2 T\lambda + 3k)} - \frac{kd}{2\lambda} \log 6\tau^2.
\end{aligned}$$

Hence

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\dots,\hat{\rho}_t)} \ell(\hat{\mathfrak{c}}_t, x_t) \le \inf_{k \in [\![1,p]\!]} \inf_{\mathfrak{c} \in \mathscr{C}(k,R)} &\left\{ \sum_{t=1}^{T} \ell(\mathfrak{c}, x_t) + \frac{(3+d)k}{\lambda} \log\left(1 + \frac{\tau}{\tau_0} + \frac{\sum_{j=1}^{k} |\mathfrak{c}_j|_2}{\sqrt{6k}\tau_0}\right) \right. \\
&\left. + \frac{1}{\lambda} \sqrt{kd(12\tau^2 T\lambda + 3k)} + \frac{kd}{\lambda} \log \frac{\tau_0}{\sqrt{6}\tau c_d^{1/d}} + \frac{\eta}{\lambda}(k-1) \right\} + \frac{\lambda T}{2} C_1^2 + \frac{\log p}{\lambda}.
\end{aligned}$$

which concludes the proof. □

Tuning parameters λ, τ and η can be chosen to obtain a sublinear regret bound for the cumulative loss of Algorithm 2.1.

Corollary 2.6. *For any sequence $(x_t)_{1:T} \in \mathbb{R}^{dT}$, under the assumptions of Corollary 2.5, if $T \ge \max\left\{2, \tau_0^2\sqrt{d}/\left(\sqrt{3}R^2 c_d^{2/d}\right)\right\}$, $\lambda = (d+2)\sqrt{\log T}/\left(2\sqrt{T}R^2\right)$, $\tau^2 = \tau_0^2/\left(6Tc_d^{2/d}\right)$ and $\eta \ge 0$, Algorithm 2.1 satisfies*

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\dots,\hat{\rho}_t)} \ell(\hat{\mathbf{c}}_t, x_t) \le \inf_{k \in [\![1,p]\!]} \inf_{\mathbf{c} \in \mathscr{C}(k,R)} &\left\{ \sum_{t=1}^{T} \ell(\mathbf{c}, x_t) + \frac{(6+2d)R^2}{d+2} k \frac{\sqrt{T}}{\sqrt{\log T}} \log\left(1 + \frac{1}{c_d T^{\frac{1}{2}}} + \frac{\sum_{j=1}^{k} |c_j|_2}{\sqrt{6k}\tau_0}\right) \right. \\
&\left. + \frac{k(d+\eta)R^2}{d+2} \sqrt{T \log T} + \frac{2\sqrt{T}R}{\sqrt{(d+2)\log T}} \sqrt{3k^2 d + kd\tau_0^2 (c_d)^{-1/d}} \right\} + \left(\frac{2R^2 \log p}{d+2} + \frac{81(d+2)R^2}{4}\right) \sqrt{T \log T},
\end{aligned}$$

where $c_d = \left(\frac{\Gamma(\frac{3+d}{2})}{\Gamma(\frac{3}{2})\Gamma(\frac{d}{2}+1)}\right)^{1/d}$.

If we compare the regret bound in this result with the one in Corollary 2.2, we see that both last terms outside braces have the same order of $\sqrt{T\log T}$. Inside the braces, due to the different prior (2.36) which is considered here, an additional term

$$k\frac{\sqrt{T}}{\sqrt{\log T}}\log\left(1+\frac{1}{c_d T^{\frac{1}{2}}}+\frac{\sum_{j=1}^k |c_j|_2}{\sqrt{6}k\tau_0}\right)$$

appears. If the scale parameter τ_0 in the prior (2.36) is small, say $\tau_0 < 1/T$, the logarithmic term will be of the order of $\log T$, hence this additional term is of order $\sqrt{T\log T}$. Moreover, the smaller the value of τ_0, the larger the coefficient before k, indicating that a smaller number of clusters k is preferred, which is consistent with the sparsity-inducing prior assumption since the sparsity is decided by τ_0.

In the adaptive setting (Algorithm 2.2), applying Theorem 2.1 to the specific q and π_k in (2.5) and (2.36) leads to the following result.

Corollary 2.7. *For any deterministic sequence $(x_t)_{1:T} \in \mathbb{R}^{dT}$, under the assumptions of Corollary 2.5, set $T \geq \max\left\{2, \tau_0^2\sqrt{d}/\left(\sqrt{3}R^2 c_d^{2/d}\right)\right\}$, $\lambda = (d+2)\sqrt{\log t}/\left(2\sqrt{t}R^2\right)$ and $\eta \geq 0$ and $\lambda_0 = 1$. Then Algorithm 2.2 satisfies*

$$\sum_{t=1}^T \mathbb{E}_{(\hat{\rho}_1,\hat{\rho}_2,\ldots,\hat{\rho}_t)}\ell(\hat{\mathbf{c}}_t, x_t) \leq \inf_{k\in[\![1,p]\!]}\inf_{\mathbf{c}\in\mathscr{C}(k,R)}\left\{\sum_{t=1}^T \ell(\mathbf{c}, x_t) + \frac{(6+2d)R^2}{d+2}k\frac{\sqrt{T}}{\sqrt{\log T}}\log\left(1+\frac{1}{c_d T^{\frac{1}{2}}}+\frac{\sum_{j=1}^k |c_j|_2}{\sqrt{6}k\tau_0}\right)\right.$$
$$\left.+\frac{k(d+\eta)R^2}{d+2}\sqrt{T\log T} + \frac{2\sqrt{T}R}{\sqrt{(d+2)\log T}}\sqrt{3k^2 d + kd\tau_0^2 (c_d)^{-1/d}}\right\} + \left(\frac{2R^2\log p}{d+2}+\frac{81(d+2)R^2}{2}\right)\sqrt{T\log T},$$

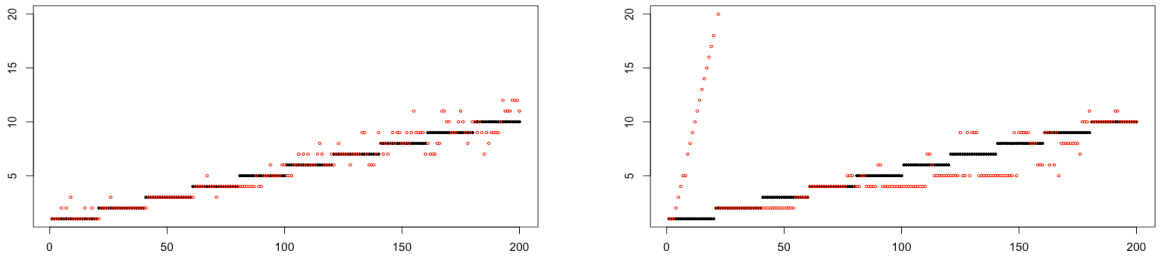
where $c_d = \left(\frac{\Gamma(\frac{3+d}{2})}{\Gamma(\frac{3}{2})\Gamma(\frac{d}{2}+1)}\right)^{1/d}$.

Proof. The proof is similar to the proof of Corollary 2.5, the only difference lies in the fact that (2.45) is replaced by

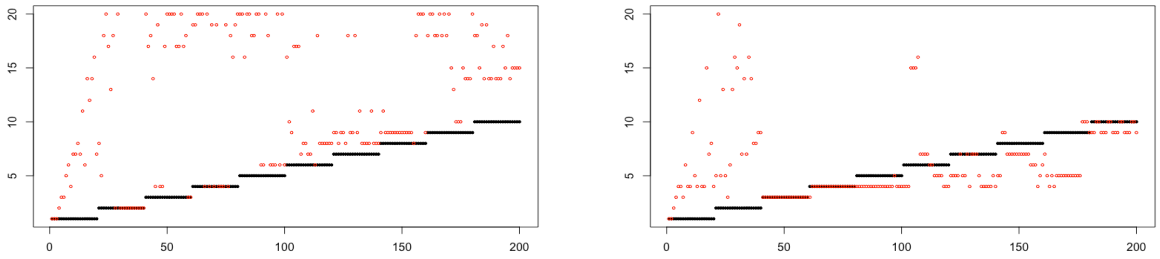
$$\mathbb{E}_{(\hat{\rho}_1,\ldots,\hat{\rho}_T)}\mathbb{E}_{\mathbf{c}\sim\rho_k}\sum_{t=1}^T \frac{\lambda_{t-1}}{2}[\ell(\mathbf{c}, x_t) - \ell(\hat{\mathbf{c}}_t, x_t)]^2 \leq C_1^2\sqrt{T\log T}.$$

□

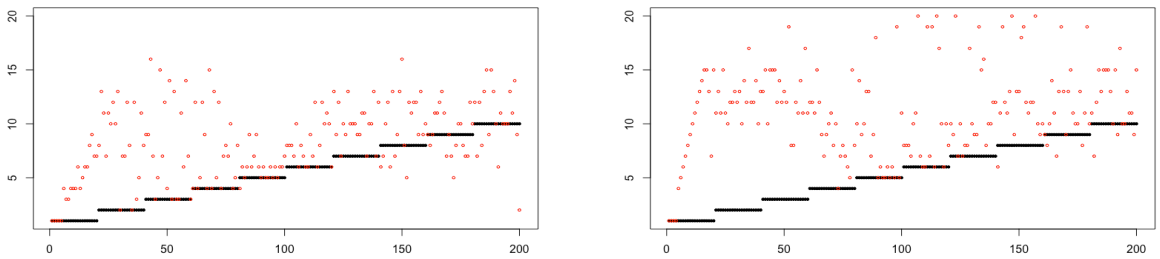
For the sake of completeness, we present in Figure 2.6 the performance of PACBO and its seven competitors for estimating the true number $k_t^\star$ of clusters along time. We acknowledge that no theoretical guarantee is derived for the estimation of $k_t^\star$ yet the practical behavior is remarkable.



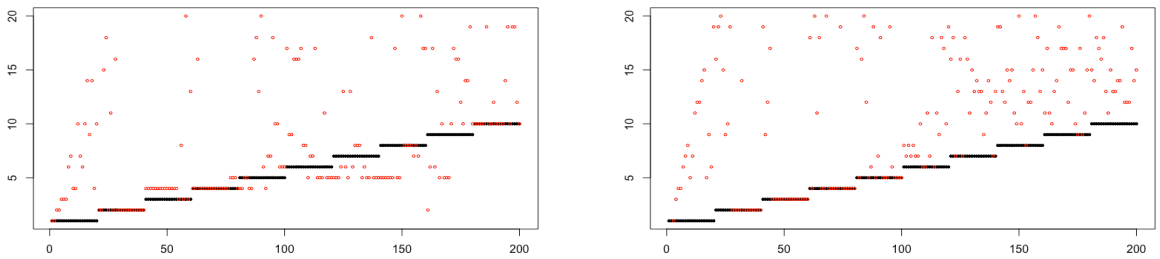
(a) PACBO (left) and Silhouette (right)



(b) Calinski (left) and Hartigan (right)



(c) Djump (left) and DDSE (right)



(d) Lai (left) and Gap (right)

Figure 2.6 – True (black) and estimated (red) number of clusters as functions of t.

3

Sequential Learning of Principal Curves

We consider the learning of principal curves sequentially for arbitrary bounded deterministic sequences. The sequential principal curve seeks to represent a sequence of data by a continuous polygonal curve instead of several cluster centers that has been studied in Chapter 2. To this aim, we introduce a new and adaptive algorithm to learn sequentially principal curves. Inspired by the quasi-Bayesian idea, this algorithm relies on the mode of Gibbs-posterior, with a dynamic estimation of unknown number of segments of principal curves. We prove the regret bounds for the algorithm and give an implementation which is based on the local greedy search to find the optimal curve. We also illustrate its performance on both toy and real dataset.

Contents

3.1 Introduction

Numerous methods have been proposed in the statistics and machine learning literature to sum up information and represent data by condensed and simpler to understand quantities. Among those methods, Principal Component Analysis (PCA) aims at identifying the maximal variance axes of data. This serves as a way to represent data in a more compact fashion and hopefully reveal as well as possible their variability. PCA has been introduced by Pearson (1901) and Spearman (1904) and further developed by Hotelling

(1933). This is one of the most widely used procedures in multivariate exploratory analysis targeting dimension reduction or features extraction. Nonetheless, PCA is a linear procedure and the need for more sophisticated nonlinear techniques has led to the notion of principal curve. Principal curves may be seen as a nonlinear generalization of the first principal component. The goal is to obtain a curve which passes "in the middle" of data, as illustrated by Figure 3.1. This notion has been at the heart of numerous applications in many different domains, such as physics (Friedsam and Oren, 1989; Brunsdon, 2007), character and speech recognition (Reinhard and Niranjan, 1999; Kégl and Krzyżak, 2002), mapping and geology (Banfield and Raftery, 1992; Stanford and Raftery, 2000; Brunsdon, 2007), to name but a few.

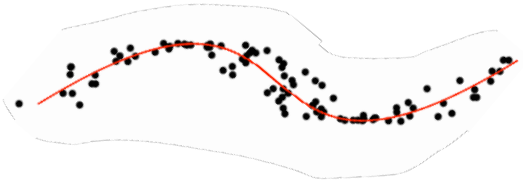


Figure 3.1 – An example of principal curve.

3.1.1 Earlier works on principal curves

The original definition of principal curve dates back to Hastie and Stuetzle (1989). A principal curve is a smooth (C^∞) parameterized curve $\mathbf{f}(s) = (f_1(s), \ldots, f_d(s))$ in $\mathbb{R}^d$ which does not intersect itself, has finite length inside any bounded subset of $\mathbb{R}^d$ and is self-consistent. This last requirement means that $\mathbf{f}(s) = \mathbb{E}[X | s_{\mathbf{f}}(X) = s]$, where $X \in \mathbb{R}^d$ is a random vector and the so-called projection index $s_{\mathbf{f}}(x)$ is the largest real number s minimizing the squared Euclidean distance between $\mathbf{f}(s)$ and x, defined by

$$s_{\mathbf{f}}(x) = \sup \left\{ s : \|x - \mathbf{f}(s)\|_2^2 = \inf_\tau \|x - \mathbf{f}(\tau)\|_2^2 \right\}.$$

Self-consistency means that each point of $\mathbf{f}$ is the average (under the distribution of X) of all data points projected on $\mathbf{f}$, as illustrated by Figure 3.2. However, an unfortunate

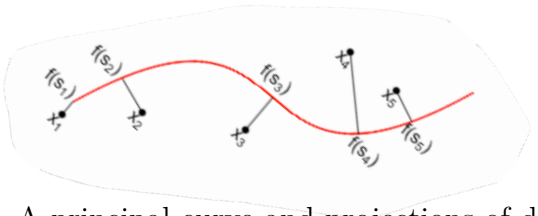


Figure 3.2 – A principal curve and projections of data onto it.

consequence of this definition of principal curve is that its existence is not guaranteed in general for a particular distribution, let alone for an online sequence for which no probabilistic assumption is made. Kégl (1999) proposed a new concept of principal curves which ensures the existence for a large class of distributions. Principal curves $\mathbf{f}^\star$ are defined as the curves minimizing the expected squared distance over a class $\mathscr{F}_L$ of curves whose length is smaller than $L > 0$, namely,

$$\mathbf{f}^\star \in \arg\inf_{\mathbf{f} \in \mathscr{F}_L} \Delta(\mathbf{f}),$$

where

$$\Delta(\mathbf{f}) = \mathbb{E}[\Delta(\mathbf{f}, X)] = \mathbb{E}\left[\inf_s \|\mathbf{f}(s) - X\|_2^2\right].$$

If $\mathbb{E}\|X\|_2^2 < \infty$, $\mathbf{f}^\star$ always exists but may not be unique. In practical situation where only i.i.d copies $X_1, \ldots, X_n$ of X are observed, Kégl (1999) considers classes $\mathscr{F}_{k,L}$ of all polygonal lines with k segments and length not exceeding L, and chooses an estimator $\hat{\mathbf{f}}_{k,n}$ of $\mathbf{f}^\star$ as the one within $\mathscr{F}_{k,L}$ which minimizes the empirical counterpart

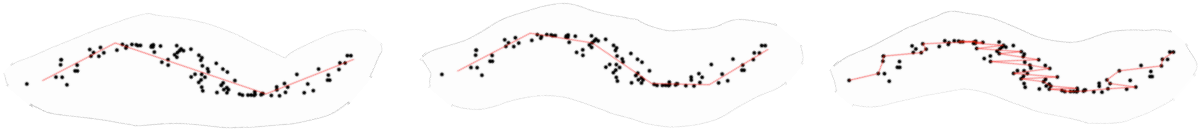
$$\Delta_n(\mathbf{f}) = \frac{1}{n}\sum_{i=1}^{n} \Delta(\mathbf{f}, X_i)$$

of $\Delta(\mathbf{f})$. It is proved in Kégl *et al.* (2000) that if X is almost surely bounded and k is proportional to $n^{1/3}$, then

$$\Delta\left(\hat{\mathbf{f}}_{k,n}\right) - \Delta\left(\mathbf{f}^\star\right) = \mathscr{O}\left(n^{-1/3}\right).$$

As the task of finding a polygonal line with k segments and with length at most L that minimizes $\Delta_n(\mathbf{f})$ is computationally costly, Kégl *et al.* (2000) proposes the Polygonal Line algorithm. This iterative algorithm proceeds by fitting a polygonal line with k segments and considerably speeds up the exploration of vertices of polygonal line by resorting to gradient descent. The two steps (projection and optimization), contained in this algorithm, are similar to what is done in the k-means algorithm. However, the Polygonal Line algorithm is not supported by theoretical bounds and leads to variable performance depending on the distribution of the observations.

As the number k of segments plays a crucial role (a too small k leads to a rough summary of data while a too large k yields overfitting, see Figure 3.3), Biau and Fischer (2012) aim to fill the gap by selecting an optimal k from both theoretical and practical perspectives.



(a) A too small k. (b) An appropriate k. (c) A too large k.

Figure 3.3 – Principal curves with different number k of segments.

Their approach relies strongly on the theory of model selection by penalization introduced by Barron *et al.* (1999) and further developed by Birgé and Massart (2007). By considering countable classes $\{\mathscr{F}_{k,\ell}\}_{k,\ell}$ of polygonal lines with k segments, total length $\ell \leq L$ and whose vertices are on a lattice, the optimal $(\hat{k}, \hat{\ell})$ is obtained as the minimizer of the criterion

$$\mathrm{crit}(k, \ell) = \Delta_n\left(\hat{\mathbf{f}}_{k,\ell}\right) + \mathrm{pen}(k, \ell),$$

where

$$\mathrm{pen}(k, \ell) = c_0\sqrt{\frac{k}{n}} + c_1\frac{\ell}{n} + c_2\frac{1}{\sqrt{n}} + \delta^2\sqrt{\frac{w_{k,\ell}}{2n}}$$

is a penalty function where δ stands for the diameter of observations, $w_{k,\ell}$ denotes the weight attached to class $\mathscr{F}_{k,\ell}$ and constants c_0, c_1, c_2 depend on δ, the maximum length L and the dimension of observations. Biau and Fischer (2012) then prove that

$$\mathbb{E}\left[\Delta(\hat{\mathbf{f}}_{\hat{k},\hat{\ell}}) - \Delta(\mathbf{f}^\star)\right] \leq \inf_{k,\ell}\left\{\mathbb{E}\left[\Delta(\hat{\mathbf{f}}_{k,\ell}) - \Delta(\mathbf{f}^\star)\right] + \mathrm{pen}(k,\ell)\right\} + \frac{\delta^2\Sigma}{2^{3/2}}\sqrt{\frac{\pi}{n}}, \tag{3.1}$$

where Σ is a numerical constant. The access risk of the final polygonal line $\hat{\mathbf{f}}_{\hat{k},\hat{\ell}}$ is close to the minimal access risk achievable over $\mathscr{F}_{k,\ell}$ up to a remainder term decaying as $1/\sqrt{n}$.

3.1.2 Motivation

The big data paradigm—where collecting, storing and analyzing massive amounts of large and complex data becomes the new standard—commands to revisit some of the classical statistical and machine learning techniques. The tremendous improvements of data acquisition infrastructures generates new continuous streams of data, rather than batch datasets. This has drawn a large interest to sequential learning. Extending the notion of principal curves to the sequential settings opens immediate practical application possibilities. As an example, path planning for passengers' location may help taxi companies to better optimize their fleet. Online algorithms that could yield instantaneous path summarization would be adapted to the sequential nature of geolocalized data. Existing theoretical works and practical implementations of principal curves are designed for the batch setting (Kégl, 1999; Kégl *et al.*, 2000; Kégl and Krzyżak, 2002; Sandilya and Kulkarni, 2002; Biau and Fischer, 2012) and their adaptation to the sequential setting is not a smooth process. As an example, consider the algorithm in Biau and Fischer (2012). It is assumed that vertices of principal curves are located on a lattice, and its computational complexity is of order $\mathcal{O}(nN^p)$ where n is the number of observations, N the number of points on the lattice and p the maximum number of vertices. When p is large, running this algorithm at each epoch yields a monumental computational cost. In general, if data is not identically distributed or even adversary, algorithms that originally worked well in the batch setting may not be ideal when cast onto the online setting (see Cesa-Bianchi and Lugosi, 2006, chapter 4). To the best of our knowledge, very little effort has been put so far into extending principal curves algorithms to the sequential context (to the notable exception of Laparra and Malo, 2016, in a fairly different setting and with no theoretical results). The present chapter aims at filling this gap: our goal is to propose an online perspective to principal curves by automatically and sequentially learning the best principal curve summarizing a data stream. Sequential learning takes advantage of the latest collected (set of) observations and therefore suffers a much smaller computational cost.

Sequential learning operates as follows: a blackbox reveals at each time t some deterministic value $x_t, t = 1, 2, \ldots$, and a forecaster attempts to predict sequentially the next value based on past observations (and possibly other available information). The performance of the forecaster is no longer evaluated by its generalization error (as in the batch setting) but rather by a regret bound which quantifies the cumulative loss of a forecaster in the first T rounds with respect to some reference minimal loss. In sequential learning, the velocity of algorithms may be favored over statistical precision. An immediate use of aforecited techniques (Kégl *et al.*, 2000; Sandilya and Kulkarni, 2002; Biau and Fischer, 2012) at each time round t (treating data collected until t as a batch dataset) would result in a monumental algorithmic cost. Rather, we propose a novel algorithm which adapts

to the sequential nature of data, *i.e.*, which takes advantage of previous computations. We refer the reader to the monograph Cesa-Bianchi and Lugosi (2006) for a thorough introduction to sequential learning.

The contributions of this chapter are twofold. We first propose a sequential principal curves algorithm, for which we derive regret bounds. We then move towards an implementation, illustrated on a toy dataset and a real-life dataset (seismic data). The sketch of our algorithm procedure is as follows. At each time round t, the number of segments of k_t is chosen automatically and the number of segments k_{t+1} in the next round is obtained by only using information about k_t and a small amount of past observations. The core of our procedure relies on computing a quantity which is linked to the mode of the so-called Gibbs quasi-posterior which is inspired by quasi-Bayesian learning. The use of quasi-Bayesian estimators is especially advocated by the PAC-Bayesian theory which originates in the machine learning community in the late 1990s, in the seminal works of Shawe-Taylor and Williamson (1997) and McAllester (1999b,a). In this manuscript, Chapter 2 also discusses the use of PAC-Bayesian tools for sequential learning.

The rest of the chapter proceeds as follows. Section 3.2 presents our notation and our online principal curve algorithm, for which we provide regret bounds with sublinear remainder terms in Section 3.3. A practical implementation is proposed in Section 3.4 and we illustrate its performance on a toy dataset and seismic data in Section 3.5. Finally, we collect in Section 3.6 proofs to original results claimed in this chapter.

3.2 Notation

A parameterized curve in $\mathbb{R}^d$ is a continuous function $\mathbf{f} : I \longrightarrow \mathbb{R}^d$ where $I = [a,b]$ is a closed interval of the real line. The length of $\mathbf{f}$ is given by

$$\mathscr{L}(\mathbf{f}) = \lim_{M\to\infty} \left\{ \sup_{a=s_0<s_1<\cdots<s_M=b} \sum_{i=1}^{M} \|\mathbf{f}(s_i) - \mathbf{f}(s_{i-1})\|_2 \right\},$$

where $s_0, s_1, \ldots, s_M$ are any possible divisions of interval I.

Let $x_1, x_2, \ldots, x_T \in B(\mathbf{0}, \sqrt{d}R) \subset \mathbb{R}^d$ be a sequence of data, where $B(\mathbf{c}, R)$ stands for the ℓ_2-ball centered in $\mathbf{c} \in \mathbb{R}^d$ with radius $R > 0$. Let $\mathscr{Q}_\delta$ be a grid over $B(\mathbf{0}, \sqrt{d}R)$, *i.e.*, $\mathscr{Q}_\delta = B(\mathbf{0}, \sqrt{d}R) \cap \Gamma_\delta$ where Γ_δ is a lattice in $\mathbb{R}^d$ with spacing $\delta > 0$. Let $L > 0$ and define for each $k \in [\![1, p]\!]$ the collection $\mathscr{F}_{k,L}$ of polygonal lines $\mathbf{f}$ with k segments whose vertices are in $\mathscr{Q}_\delta$ and such that $\mathscr{L}(\mathbf{f}) \leq L$. Denote by $\mathscr{F}_p = \cup_{k=1}^{p} \mathscr{F}_{k,L}$ all polygonal lines with at most p number of segments , whose vertices are in $\mathscr{Q}_\delta$ and whose length is at most L. Finally, let $\mathscr{K}(\mathbf{f})$ denote the number of segments of $\mathbf{f} \in \mathscr{F}_p$. An example of lattice and grid is illustrated by Figure 3.4.

Our goal is to learn a time-dependent polygonal line which passes through the "middle" of data and gives a summary of all available observations $x_1, \ldots, x_{t-1}$ (denoted by $(x_s)_{1:(t-1)}$ hereafter) before time t. Our output at time t is a polygonal line $\hat{\mathbf{f}}_t \in \mathscr{F}_p$ depending on past information $(x_s)_{1:(t-1)}$ and past predictions $(\hat{\mathbf{f}}_s)_{1:(t-1)}$. When x_t is revealed, the instantaneous at time t is computed as

$$\Delta\left(\hat{\mathbf{f}}_t, x_t\right) = \inf_{s\in I} \|\hat{\mathbf{f}}_t(s) - x_t\|_2^2. \tag{3.2}$$

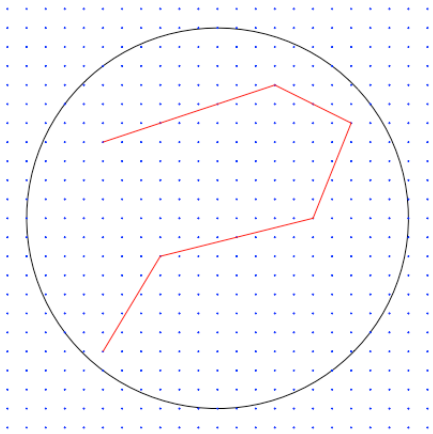


Figure 3.4 – An example of a lattice Γ_δ in $\mathbb{R}^2$ with $\delta = 1$ (spacing between blue points) and $B(0, 10)$ (black circle). The red polygonal line is composed with vertices in $\mathscr{Q}_\delta = B(0,10) \cap \Gamma_\delta$.

Similar to the loss $\ell(\hat{\mathbf{c}}_t, x_t)$ defined in Chapter 2, the loss $\Delta\left(\hat{\mathbf{f}}_t, x_t\right)$ here measures the distance between x_t and a predicted principal curve which is constructed on all the past information. In what follows, we investigate regret bounds for the cumulative loss $\sum_{t=1}^{T} \Delta\left(\hat{\mathbf{f}}_t, x_t\right)$, an analogue of the risk in batch setting, that measures the quality of our algorithm in predicting principal curves in online setting. In addition, due to the online to batch conversion (Littlestone, 1989, Helmbold and Warmuth, 1995, Dekel and Singer, 2006, Audibert, 2009, Gerchinovitz, 2011), we can construct an algorithm, on the basis of $(\hat{\mathbf{f}}_1, \hat{\mathbf{f}}_2, \ldots, \hat{\mathbf{f}}_T)$, which is able to yield an estimator of principal curves in batch setting.. Given a measurable space Θ (embedded with its Borel σ-algebra), we let $\mathscr{P}(\Theta)$ denote the set of probability distributions on Θ, and for some reference measure π, we let $\mathscr{P}_\pi(\Theta)$ be the set of probability distributions absolutely continuous with respect to π.

For any $k \in [\![1, p]\!]$, let π_k denote a probability distribution on $\mathscr{F}_{k,L}$. We define the *prior* π on $\mathscr{F}_p = \cup_{k=1}^{p} \mathscr{F}_{k,L}$ as

$$\pi(\mathbf{f}) = \sum_{k \in [\![1,p]\!]} w_k \pi_k(\mathbf{f}) \mathbb{1}_{\{\mathbf{f} \in \mathscr{F}_{k,L}\}}, \quad \mathbf{f} \in \mathscr{F}_p,$$

where $w_1, \ldots, w_p \geq 0$ and $\sum_{k \in [\![1,p]\!]} w_k = 1$.

A quasi-Bayesian procedure would now consider the Gibbs quasi-posterior (note that this is not a proper posterior in all generality, hence the term "quasi")

$$\hat{\rho}_{t+1}(\cdot) \propto \exp(-\lambda S_t(\cdot)) \pi(\cdot),$$

where

$$S_t(\mathbf{f}) = S_{t-1}(\mathbf{f}) + \Delta(\mathbf{f}, x_t) + \frac{\lambda}{2}\left(\Delta(\mathbf{f}, x_t) - \Delta(\hat{\mathbf{f}}_t, x_t)\right)^2,$$

as advocated both in Chapter 2 and Audibert (2009) who then consider realisations from this quasi-posterior. In the present chapter, we shall rather consider a quantity linked to the mode of this quasi-posterior. Indeed, the mode of the quasi-posterior $\hat{\rho}_{t+1}$ can be written as

$$\operatorname*{arg\,min}_{\mathbf{f} \in \mathscr{F}_p} \Bigg\{ \underbrace{\sum_{s=1}^{t} \Delta(\mathbf{f}, x_s)}_{(i)} + \underbrace{\frac{\lambda}{2} \sum_{s=1}^{t} \left(\Delta(\mathbf{f}, x_s) - \Delta(\hat{\mathbf{f}}_s, x_s)\right)^2}_{(ii)} + \underbrace{\frac{\ln \pi(\mathbf{f})}{\lambda}}_{(iii)} \Bigg\},$$

where *(i)* is a cumulative loss of polygonal line $\mathbf{f}$ in the first t rounds, *(ii)* is a term controlling the variance of the prediction $\mathbf{f}$ to past predictions $\hat{\mathbf{f}}_s, s \leq t$, and *(iii)* can be regarded as a penalty function on the complexity of $\mathbf{f}$ if π is well chosen. This mode hence has a similar flavor to follow the best expert or follow the perturbed leader in the setting of prediction with experts (see Hutter and Poland, 2005 or Cesa-Bianchi and Lugosi, 2006, Chapters 3 and 4) if we consider each $\mathbf{f} \in \mathscr{F}_p$ as an expert which always delivers constant advice. These remarks yield Algorithm 3.1.

Algorithm 3.1 An algorithm to sequentially learn principal curves

1: **Input parameters**: $p > 0, \eta > 0, \pi(z) = \mathrm{e}^{-z}\mathbb{1}_{\{z>0\}}$ and penalty function $h : \mathscr{F}_p \to \mathbb{R}_+^*$
2: **Initialization**: For each $\mathbf{f} \in \mathscr{F}_p$, draw i.i.d $z_{\mathbf{f}} \sim \pi$ and set $\Delta_{\mathbf{f},0} = \frac{1}{\eta}(h(\mathbf{f}) - z_{\mathbf{f}})$
3: **For** $t = 1, \ldots, T$
4: Obtain

$$\hat{\mathbf{f}}_t = \underset{\mathbf{f} \in \mathscr{F}_p}{\arg\inf} \left\{ \sum_{s=0}^{t-1} \Delta_{\mathbf{f},s} \right\},$$

5: Get data x_t and compute $\Delta_{\mathbf{f},s} = \Delta(\mathbf{f}, x_s)$, $s \geq 1$.
6: **End for**

At the beginning, $\hat{\mathbf{f}}_1$ is preferred as the perturbed polygonal line having the smallest penalty $h(\mathbf{f})$ (the least complexity) over all possible polygonal lines. The cumulative loss $\sum_{s=0}^{t-1} \Delta_{\mathbf{f},s}$ defined here hence incorporates both the cumulative loss (like the term (i)) and the complexity of $\mathbf{f}$ (like the term (iii)). This first algorithm does not take into account the variance of predictions. However, it can be done if we constraint the choice of $\hat{\mathbf{f}}_t$ only on a neighbourhood of $\hat{\mathbf{f}}_{t-1}$ rather than $\mathscr{F}_p$ at each time t. This consideration will be illustrated later on in Algorithm 3.3.

3.3 Regret bounds for sequential learning of principal curves

We now present our main theoretical results.

Theorem 3.1. *For any sequence $(x_t)_{1:T} \in B(\mathbf{0}, \sqrt{d}R)$, $R \geq 0$ and any penalty function $h : \mathscr{F}_p \to \mathbb{R}_+^*$, let $\pi(z) = \mathrm{e}^{-z}\mathbb{1}_{\{z>0\}}$. Let $0 < \eta \leq \frac{1}{d(2R+\delta)^2}$, then the procedure described in Algorithm 3.1 satisfies*

$$\sum_{t=1}^{T} \mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t, x_t)\right] \leq (1 + c_0(\mathrm{e}-1)\eta) S_{T,h,\eta} + \frac{1 + c_0(\mathrm{e}-1)\eta}{\eta}\left(1 + \ln \sum_{\mathbf{f} \in \mathscr{F}_p} \mathrm{e}^{-h(\mathbf{f})}\right),$$

where $c_0 = d(2R+\delta)^2$ and

$$S_{T,h,\eta} = \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\substack{\mathbf{f} \in \mathscr{F}_p \\ \mathscr{K}(\mathbf{f}) = k}} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) + \frac{h(\mathbf{f})}{\eta} \right\} \right\}.$$

The expectation of the cumulative loss of polygonal lines $\hat{\mathbf{f}}_1, \ldots, \hat{\mathbf{f}}_T$ is upper-bounded by the smallest penalised cumulative loss over all $k \in \{1, \ldots, p\}$ up to a multiplicative term $(1 + c_0(\mathrm{e}-1)\eta)$ which can be made arbitrarily close to 1 by choosing a small enough η. However, this will lead to both a large $h(\mathbf{f})/\eta$ in $S_{T,h,\eta}$ and a large $\frac{1}{\eta}\left(1 + \ln \sum_{\mathbf{f} \in \mathscr{F}_p} \mathrm{e}^{-h(\mathbf{f})}\right)$.

In addition, another important issue is the choice of the penalty function h. For each $\mathbf{f} \in \mathscr{F}_p$, $h(\mathbf{f})$ should be large enough to ensure a small $\sum_{\mathbf{f}\in\mathscr{F}_p} \mathrm{e}^{-h(\mathbf{f})}$ while not too large to avoid overpenalization and a larger value for $S_{T,h,\eta}$. We therefore set

$$h(\mathbf{f}) \geq \ln(p\mathrm{e}) + \ln\left|\{\mathbf{f} \in \mathscr{F}_p, \mathscr{K}(\mathbf{f}) = k\}\right| \tag{3.3}$$

for each $\mathbf{f}$ with k segments (where $|M|$ denotes the cardinality of a set M) since it leads to

$$\sum_{\mathbf{f}\in\mathscr{F}_p} \mathrm{e}^{-h(\mathbf{f})}) = \sum_{k\in[\![1,p]\!]} \sum_{\substack{\mathbf{f}\in\mathscr{F}_p \\ \mathscr{K}(\mathbf{f})=k}} \mathrm{e}^{-h(\mathbf{f})} \leq \sum_{k\in[\![1,p]\!]} \frac{1}{p\mathrm{e}} \leq \frac{1}{\mathrm{e}}.$$

In Lemma 3.3, we give an explicit form of the penalty function: $h(\mathbf{f}) = c_1\mathscr{K}(\mathbf{f}) + c_2 L + c_3$, where $c_1 = \ln(8V_d) + 3d^{\frac{3}{2}} - d$, $c_2 = \frac{\ln 2}{\delta\sqrt{d}} + \frac{d}{\delta}$, $c_3 = d\ln\left(\frac{\sqrt{d}(2R+\delta)}{\delta}\right) + \ln(2V_d)$ and V_d denotes the volume of unit ball in $\mathbb{R}^d$. This penalty function satisfies (3.3), hence $\left(1 + \ln\sum_{\mathbf{f}\in\mathscr{F}_p} \mathrm{e}^{-h(\mathbf{f})}\right) \leq 0$. We therefore obtain the following corollary.

Corollary 3.1. *Under the assumptions of Theorem 3.1, let* $h(\mathbf{f}) = c_1\mathscr{K}(\mathbf{f}) + c_2 L + c_3$ *and*

$$\eta = \min\left\{\frac{1}{c_0}, \sqrt{\frac{c_1 p + c_2 L + c_3}{c_0(\mathrm{e}-1)\inf_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^T \Delta(\mathbf{f}, x_t)}}\right\}.$$

Then

$$\sum_{t=1}^T \mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t, x_t)\right] \leq \inf_{k\in[\![1,p]\!]}\left\{\inf_{\substack{\mathbf{f}\in\mathscr{F}_p \\ \mathscr{K}(\mathbf{f})=k}}\left\{\sum_{t=1}^T \Delta(\mathbf{f}, x_t) + \sqrt{c_0(\mathrm{e}-1)r_{T,k,L}}\right\}\right\} + \sqrt{c_0(\mathrm{e}-1)r_{T,p,L}} + 2\mathrm{e}c_0(c_1 p + c_2 L + c_3),$$

where $r_{T,k,L} = \inf_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^T \Delta(\mathbf{f}, x_t)(c_1 k + c_2 L + c_3), k = 1, 2, \ldots, p$.

Proof. Note that firstly that with such choice of $h(\mathbf{f})$, one has $1 + \ln\sum_{\mathbf{f}\in\mathscr{F}_p} \mathrm{e}^{-h(\mathbf{f})} \leq 0$ and $\sup_{\mathbf{f}} h(\mathbf{f}) \leq c_1 p + c_2 L + c_3$, then from Theorem 3.1, we have

$$\sum_{t=1}^T \mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t, x_t)\right] \leq S_{T,h,\eta} + \eta c_0(\mathrm{e}-1)\inf_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^T \Delta(\mathbf{f}, x_t) + c_0(\mathrm{e}-1)(c_1 p + c_2 L + c_3).$$

If $\sqrt{\frac{c_1 p + c_2 L + c_3}{c_0(\mathrm{e}-1)\inf_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^T \Delta(\mathbf{f}, x_t)}} \leq \frac{1}{c_0}$, we substitute η in (3.12) with $\sqrt{\frac{c_1 p + c_2 L + c_3}{c_0(\mathrm{e}-1)\inf_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^T \Delta(\mathbf{f}, x_t)}}$ and obtain the following inequality

$$\sum_{t=1}^T \mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t, x_t)\right] \leq \inf_{k\in[\![1,p]\!]}\left\{\inf_{\substack{\mathbf{f}\in\mathscr{F}_p \\ \mathscr{K}(\mathbf{f})=k}}\left\{\sum_{t=1}^T \Delta(\mathbf{f}, x_t) + \sqrt{c_0(\mathrm{e}-1)r_{T,k,L}}\right\}\right\} + \sqrt{c_0(\mathrm{e}-1)r_{T,p,L}} + c_0(\mathrm{e}-1)(c_1 p + c_2 L + c_3). \tag{3.4}$$

If $\sqrt{\frac{c_1 p + c_2 L + c_3}{c_0(\mathrm{e}-1)\inf_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^T \Delta(\mathbf{f}, x_t)}} > \frac{1}{c_0}$, *i.e.*, $(\mathrm{e}-1)\inf_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^T \Delta(\mathbf{f}, x_t) \leq c_0(c_1 p + c_2 L + c_3)$, we can prove similarly that

$$\sum_{t=1}^T \mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t, x_t)\right] \leq \inf_{k\in[\![1,p]\!]}\left\{\inf_{\substack{\mathbf{f}\in\mathscr{F}_p \\ \mathscr{K}(\mathbf{f})=k}}\left\{\sum_{t=1}^T \Delta(\mathbf{f}, x_t)\right\}\right\} + c_0(\mathrm{e}+1)(c_1 p + c_2 L + c_3). \tag{3.5}$$

Combining (3.4) and (3.5) terminates the proof of Corollary 3.1. □

Note that since $k \leq p$, the regret bound on the right hand side of inequality in Corollary 3.1 can be simplified as (similarly for Theorem 3.2)

$$\inf_{k\in[\![1,p]\!]}\left\{\inf_{\substack{\mathbf{f}\in\mathscr{F}_p\\ \mathscr{K}(\mathbf{f})=k}}\left\{\sum_{t=1}^{T}\Delta(\mathbf{f},x_t)\right\}\right\}+2\sqrt{c_0(\mathrm{e}-1)r_{T,p,L}}+2\mathrm{e}c_0(c_1p+c_2L+c_3).$$

However, we keep k in this corollary and in the next theorem only for implementation purpose since the number k of segments is important in implementation for updating sequentially principal curve from a neighborhood of the previous one.

Note that Corollary 3.1 is not usable in practice since the optimal value for η depends on $\inf_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^{T}\Delta(\mathbf{f},x_t)$ which is obviously unknown. We therefore provide an adaptive refinement of Algorithm 3.1 in the following Algorithm 3.2. Note that η_0 should have been

Algorithm 3.2 An adaptive algorithm to sequentially learn principal curves

1: **Input parameters**: $p>0$, $L>0$, π, h and $\eta_0=\frac{1}{c_0}$
2: **Initialization**: For each $\mathbf{f}\in\mathscr{F}_p$, draw $z_{\mathbf{f}}\sim\pi$, $\Delta_{\mathbf{f},0}=\frac{1}{\eta_0}(h(\mathbf{f})-z_{\mathbf{f}})$
3: **For** $t=1,\ldots,T$
4: Obtain

$$\hat{\mathbf{f}}_t=\operatorname*{arginf}_{\mathbf{f}\in\mathscr{F}_p}\left\{\sum_{s=0}^{t-1}\Delta_{\mathbf{f},s}\right\}. \tag{3.6}$$

5: Compute $\eta_t=\min\left\{\frac{1}{c_0},\frac{\sqrt{c_1p+c_2L+c_3}}{c_0\sqrt{(\mathrm{e}-1)t}}\right\}$
6: Get data x_t and compute $\Delta_{\mathbf{f},t}=\Delta(\mathbf{f},x_t)+\left(\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}}\right)(h(\mathbf{f})-z_{\mathbf{f}})$
7: **End for**

of the similar form as η_t, *i.e.*, $\eta_0=\min\left\{\frac{1}{c_0},\frac{\sqrt{c_1p+c_2L+c_3}}{c_0\sqrt{(\mathrm{e}-1)}}\right\}$. Since $(\mathrm{e}-1)<c_1p+c_2L+c_3$ which is clear if one checks the value of c_1 in Lemma 3.3, η_0 equals to $\frac{1}{c_0}$.

Theorem 3.2. *For any sequence $(x_t)_{1:T}\in B(\mathbf{0},\sqrt{d}R),R\geq 0$, let $h(\mathbf{f})=c_1\mathscr{K}(\mathbf{f})+c_2L+c_3$ where c_1, c_2, c_3 are constants depending on R,d,δ. Let $\pi(z)=\mathrm{e}^{-z}\mathbb{1}_{\{z>0\}}$ and*

$$\eta_0=\frac{1}{c_0},\qquad \eta_t=\min\left\{\frac{1}{c_0},\frac{\sqrt{c_1p+c_2L+c_3}}{c_0\sqrt{(\mathrm{e}-1)t}}\right\},\qquad t\geq 1,$$

where $c_0=d(2R+\delta)^2$. Then the procedure described in Algorithm 3.2 satisfies

$$\sum_{t=1}^{T}\mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t,x_t)\right]\leq\inf_{k\in[\![1,p]\!]}\left\{\inf_{\substack{\mathbf{f}\in\mathscr{F}_p\\ \mathscr{K}(\mathbf{f})=k}}\left\{\sum_{t=1}^{T}\Delta(\mathbf{f},x_t)+c_0\sqrt{(\mathrm{e}-1)T(c_1k+c_2L+c_3)}\right\}\right\}$$
$$+2c_0\sqrt{(\mathrm{e}-1)T(c_1p+c_2L+c_3)}+3c_0(c_1p+c_2L+c_3).$$

The message of this regret bound is that the expected cumulative loss of polygonal lines $\hat{\mathbf{f}}_1,\ldots,\hat{\mathbf{f}}_T$ is upper-bounded by the minimal cumulative loss over all $k\in\{1,\ldots,p\}$, up to an additive term which is sublinear in T. The actual magnitude of this remainder term is $\sqrt{kT}$. When L is fixed, the number k of segments is a measure of complexity of the retained polygonal line. This bound therefore yields the same magnitude than (3.1) which is the most refined bound in the literature so far (Biau and Fischer, 2012, where the optimal values for k and L are obtained in a model selection fashion).

3.4 Implementation

The argument of the infimum in Algorithm 3.2 is taken over $\mathscr{F}_p = \cup_{k=1}^{p} \mathscr{F}_{k,L}$ which has a cardinality of order $|\mathscr{Q}_\delta|^p$, where $|\mathscr{Q}_\delta|$ is the number of vertices of grid $\mathscr{Q}_\delta$. This exponential cardinality makes any greedy search largely time-consuming. We instead turn to the following strategy: given a polygonal line $\hat{\mathbf{f}}_t \in \mathscr{F}_{k_t,L}$ with k_t segments, we consider, with a certain proportion, the availability of $\hat{\mathbf{f}}_{t+1}$ within a neighbourhood $\mathscr{U}(\hat{\mathbf{f}}_t)$ (see the formal definition below) of $\hat{\mathbf{f}}_t$. This consideration is suited for principal curve setting since if observation x_t is close to $\hat{\mathbf{f}}_t$, one can expect that the polygonal line which well fits observations $x_s, s = 1, \ldots, t$ lies in a neighbourhood of $\hat{\mathbf{f}}_t$. In other words, if each polygonal line $\mathbf{f}$ is regarded as an action, we no longer assume that all actions are available at all times, and allow the set of available actions to vary at each time. This is a model known as "sleeping experts (or actions)" in prior work (Freund *et al.*, 1997, Auer *et al.*, 2003, Blum and Mansour, 2007, Kleinberg *et al.*, 2008). In this setting, defining the regret with respect to the best action in the whole set of actions in hindsight remains to be difficult since that action might sometimes be unavailable in certain rounds. Hence it is natural to define the regret with respect to the best ranking of all actions in the hindsight according to their losses or rewards, and at each round one chooses among the available actions by selecting the one which ranks the highest in this ordering. Kleinberg *et al.* (2008) introduced this notion of regret and studied both the full-information (best action) and partial-information (multi-armed bandit) settings with stochastic and adversarial rewards and adversarial action availability. They pointed out that the **EXP4** algorithm (Auer *et al.*, 2003) attains the optimal regret in adversarial rewards case but runs in time exponential in the number of all actions. Kanade *et al.* (2009) considered full and partial information with stochastic action availability and proposed an algorithm that runs in polygonal time. In what follows, we shall realise our implementation by resorting to "sleeping experts" *i.e.,* a special available set of actions that adapts to the setting of principal curves.

Let σ denote an ordering of $|\mathscr{F}_p|$ actions, and $\mathscr{A}_t$ an available subset of the actions at round t. We let $\sigma(\mathscr{A}_t)$ denote the highest ranked action in $\mathscr{A}_t$. In addition, for any action $\mathbf{f} \in \mathscr{F}_p$ we define the reward $r_{\mathbf{f},t}$ of $\mathbf{f}$ at round $t, t \geq 1$ by

$$r_{\mathbf{f},t} = c_0 - \Delta(\mathbf{f}, x_t).$$

It is clear that $r_{\mathbf{f},t} \in (0, c_0)$. The convention from losses to gains is done in order to facilitate the subsequent performance analysis. The reward of an ordering σ is the cumulative reward of the selected action at each time

$$\sum_{t=1}^{T} r_{\sigma(\mathscr{A}_t),t},$$

and the reward of the best ordering is $\max_\sigma \sum_{t=1}^{T} r_{\sigma(\mathscr{A}_t),t}$ ($\mathbb{E}\left[\max_\sigma \sum_{t=1}^{T} r_{\sigma(\mathscr{A}_t),t}\right]$ when $\mathscr{A}_t$ is stochastic).

Before giving a formal definition of the neighbourhood $\mathscr{U}(\hat{\mathbf{f}}_t)$, we first introduce our implementation procedure that starts with a **partition** step which aims at identifying the "relevant" neighbourhood of an observation $x \in \mathbb{R}^d$ with respect to a given polygonal line, and then proceeds with the definition of **neighbourhood** of an action $\mathbf{f}$, and finally we give details of implementation and show its regret bound.

Partition For any polygonal line $\mathbf{f}$ with k segments, we denote by $\vec{\mathbf{V}} = (v_1, \ldots, v_{k+1})$ its vertices and by $s_i, i = 1, \ldots, k$ the line segments connecting v_i and v_{i+1}. In the sequel, we

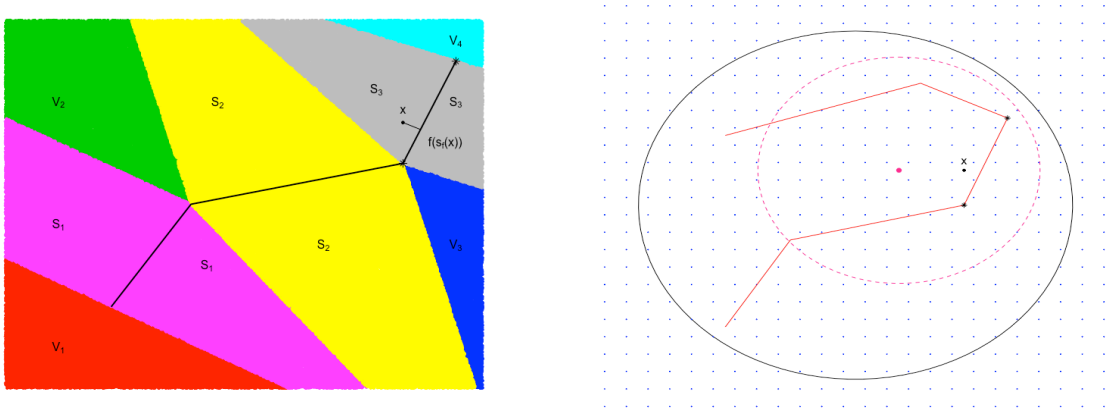
use $\mathbf{f}(\vec{\mathbf{V}})$ to represent the polygonal line formed by connecting consecutive vertices in $\vec{\mathbf{V}}$ if no confusion is possible. In addition, for $x \in \mathbb{R}^d$, let $\Delta(x, v_i) = ||x - v_i||_2^2$ be the squared distance between x and v_i, and let $\Delta(x, s_i)$ be the squared distance between x and line segment s_i, *i.e.*,

$$\Delta(x, s_i) = \begin{cases} ||x - v_i||_2^2 & \text{if } s_{s_i}(x) = v_i \ , \\ ||x - v_{i+1}||_2^2 & \text{if } s_{s_i}(x) = v_{i+1}, \\ ||x - v_i||_2^2 - \left((x - v_i)^T \frac{v_{i+1} - v_i}{||v_{i+1} - v_i||_2}\right)^2 & \text{otherwise,} \end{cases}$$

where $s_{s_i}(x)$ is the projection index of x to line segment s_i. In this step, $\mathbb{R}^d$ can be partitioned into at most $(2k+1)$ disjoint sets $V_i, i = 1, \ldots, k+1$ and $S_i, i = 1, \ldots, k$ defined as

$$V_i = \left\{ x \in \mathbb{R}^d : \Delta(x, v_i) = \Delta(\mathbf{f}, x), x \notin \cup_{s=1}^{i-1} V_s \right\},$$
$$S_i = \left\{ x \in \mathbb{R}^d : \Delta(x, s_i) = \Delta(\mathbf{f}, x), x \notin \cup_{m=1}^{i-1} S_m \cup_{i=1}^{k+1} V_i \right\},$$

where V_i, $i = 1, \ldots, k+1$ is the set of all $x \in \mathbb{R}^d$ whose closest vertex of $\mathbf{f}$ is v_i, and S_i, $i = 1, \ldots, k$ is the set of all $x \in \mathbb{R}^d$ whose closest segment of $\mathbf{f}$ is s_i. Figure 3.5a gives an example of Voronoi partition with respect to a polygonal line with 3 segments. In the sequel, we use $V_{i:j}, i \le j$ (*resp.* $S_{i:j}$ and $v_{i:j}$) to abbreviate the sequence of partitions $V_i, \ldots, V_j$ (*resp.* $S_i, \ldots, S_j$ and $v_i, \ldots, v_j$).



(a) partition step (b) local greedy search step

Figure 3.5 – (a) a Voronoi-like partition given a polygonal line (black solid line); (b) a local greedy search region (blue points inside pink dashed circle), where the pink dashed circle is a ℓ_2 ball with center $\bar{\mathcal{N}}_t(x)$ and with radius $\mathscr{D}(\mathcal{N}_t(x))$; x is an observation in $\mathbb{R}^2$ and its $\mathcal{V}(x)$ is v_3 and v_4, represented by asterisks.

For any $x \in \mathbb{R}^d$, let us denote by $\mathcal{V}(x)$ the set of vertices of $\mathbf{f}$ connecting to the projection

$\mathbf{f}(s_{\mathbf{f}}(x))$ of x onto $\mathbf{f}$ and denote by $\mathscr{N}(x)$ the neighbourhood x with respect to $\mathbf{f}$, *i.e.*,

$$\mathscr{V}(x)=\begin{cases} v_{1:2} & \text{if } x\in V_1\cup S_1,\\ v_{1:3} & \text{if } x\in V_2,\\ v_{i:i+1} & \text{if } x\in S_i, i=2,\dots,k-1,\\ v_{i-1:i+1} & \text{if } x\in V_i, i=3,\dots,k-1,\\ v_{k-1:k+1} & \text{if } x\in V_k,\\ v_{k:k+1} & \text{if } x\in S_k\cup V_{k+1}, \end{cases} \mathscr{N}(x)=\begin{cases} V_{1:2}\cup S_{1:2} & \text{if } x\in V_1\cup S_1,\\ V_{1:3}\cup S_{1:3} & \text{if } x\in V_2,\\ V_{i:i+1}\cup S_{i-1:i+1} & \text{if } x\in S_i, i=2,\dots,k-1,\\ V_{i-1:i+1}\cup S_{i-2:i+1} & \text{if } x\in V_i, i=3,\dots,k-1,\\ V_{k-1:k+1}\cup S_{k-2:k} & \text{if } x\in V_k,\\ V_{k:k+1}\cup S_{k-1:k} & \text{if } x\in S_k\cup V_{k+1}. \end{cases}$$

Note that the definition of $\mathscr{V}(x)$ and $\mathscr{N}(x)$ holds whenever $\mathbf{f}$ has $k\geq 4$ segments. If $1\leq k<4$, $\mathscr{V}(x)$ (*resp.* $\mathscr{N}(x)$) is identical for any x. Next, let

$$\mathscr{N}_t(x)=\left\{x_s,, s=1,\dots,t, x_s\in\mathscr{N}(x)\right\}$$

be the set of observations $x_{1:t}$ belonging to $\mathscr{N}(x)$. We finally let $\bar{\mathscr{N}}_t(x)$ stand for the average of all observations in $\mathscr{N}_t(x)$ and $\mathscr{D}(M)=\sup_{x,y\in M}||x-y||_2$ denotes the diameter of set $M\in\mathbb{R}^d$.

In order to avoid a bad initialization of $\hat{\mathbf{f}}_1$, intuitively, we begin our prediction of principal curve after having at least t_0 (t_0 is very small) observations, and we choose the principal curve $\hat{\mathbf{f}}_1$ as the first component line segment whose vertices are the two farthest projections of data $x_{1:t_0}$ on the first component line. The reward in this setting is therefore $r_{\mathbf{f},t}=c_0-\Delta(\mathbf{f},x_{t_0+t})$ for $t=1,2,\dots$, and $\hat{\mathbf{f}}_t$ is based on $x_1,x_2,\dots,x_{t_0+t-1}$. Given $\hat{\mathbf{f}}_t$, $t\geq 1$ with k_t segments, we obtain firstly the Voronoi-like partition $V_{1:k_t+1}$ and $S_{1:k_t}$, then identify the adjacent vertices set $\mathscr{V}\left(x_{t+t_0}\right)=\left\{v_{i_t:j_t}\right\}$ and set $\mathscr{N}_{t+t_0}\left(x_{t+t_0}\right)$ for observation x_{t+t_0}.

Neighbourhood For each t, let us first define the local grid $\mathscr{Q}_{\delta,t}(x)$ around $x\in\mathbb{R}^d$ as

$$\mathscr{Q}_{\delta,t}(x)=B\left(\bar{\mathscr{N}}_t(x),\mathscr{D}\left(\mathscr{N}_t(x)\right)\cap\mathscr{Q}_\delta,\right.$$

where $B(x,r)$ is a ℓ_2-ball in $\mathbb{R}^d$, centered in x with radius $r>0$. Then for $k\in\{k_t-1,k_t,k_t+1\}$, we define a candidate $\vec{\mathbf{V}}$ (equivalently a candidate $\mathbf{f}(\vec{\mathbf{V}})$) with $k+1$ vertices as

$$\vec{\mathbf{V}}=\left(v_{1:i_t-1},v_{1:m},v_{j_t+1:k_t+1}\right),$$

where $m=k+1-k_t+j_t-i_t$ and $v_{1:m}$ are m distinct vertices belonging to $\mathscr{Q}_{\delta,t+t_0}\left(x_{t+t_0}\right)$ around x_{t+t_0}. In other words, a candidate $\vec{\mathbf{V}}$ is built by keeping all the vertices of $\hat{\mathbf{f}}_t$ that do not belong to $\mathscr{V}\left(x_{t+t_0}\right)$ and by updating the remaining m vertices drawn from $\mathscr{Q}_{\delta,t+t_0}\left(x_{t+t_0}\right)$. When $k=k_t-1$ (*resp.* k_t, k_t+1), it means that the candidate $\vec{\mathbf{V}}$ has one less vertices (*resp.* same, one more) than $\hat{\mathbf{f}}_t$. We define the neighbourhood $\mathscr{U}(\hat{\mathbf{f}}_t)$ of $\hat{\mathbf{f}}_t$ by

$$\mathscr{U}(\hat{\mathbf{f}}_t)=\left\{\mathbf{f}(\vec{\mathbf{V}}),\quad \vec{\mathbf{V}} \text{ such that } v_{1:m}\in\mathscr{Q}_{\delta,t+t_0}\left(x_{t+t_0}\right) \text{ for } k\in\{k_t-1,k_t,k_t+1\}\right\}. \tag{3.7}$$

The cardinality $\left|\mathscr{U}\left(\hat{\mathbf{f}}_t\right)\right|$ is upper bounded by $\left|\mathscr{F}_p\right|^{\frac{3}{p}}$ since all elements in $\left|\mathscr{U}\left(\hat{\mathbf{f}}_t\right)\right|$ differ with $\hat{\mathbf{f}}_t$ up to at most three vertices. Finally, we give our implementation with action availability that may vary at each time in Algorithm 3.3.

The Algorithm 3.3 has an exploration phase (when $I_t=1$) and an exploitation phase ($I_t=0$). In the exploration phase, it is allowed to observe rewards of all possible actions and

Algorithm 3.3 A locally greedy algorithm to sequentially learn principal curves

Input parameters: $p > 0$, $R > 0$, $L > 0$, $\epsilon > 0$, $\alpha > 0$, $1 > \beta > 0$ and any penalty function h

Initialization: Given $(x_t)_{1:t_0}$, obtain $\hat{\mathbf{f}}_1$ as the first principal component

For $t = 1$, we set $\hat{r}_{\mathbf{f},1} = r_{\mathbf{f},1}$ for all $\mathbf{f} \in \mathscr{F}_p$

For $t = 2, \ldots, T$

Draw $I_t \sim Bernoulli(\epsilon)$ and $z_{\mathbf{f}} \sim \pi$ for all $\mathbf{f} \in \mathscr{F}_p$.

Let $\hat{\sigma}^t = \text{sort}\left(\mathbf{f}, \quad \sum_{s=1}^{t-1} \hat{r}_{\mathbf{f},s} - \frac{1}{\eta_{t-1}} h(\mathbf{f}) + \frac{1}{\eta_{t-1}} z_{\mathbf{f}}\right)$, *i.e.,* descending sorting all $\mathbf{f} \in \mathscr{F}_p$ according to their perturbed cumulative reward till $t-1$.

If $I_t = 1$, set $\mathscr{A}_t = \mathscr{F}_p$ and $\hat{\mathbf{f}}_t = \hat{\sigma}^t(\mathscr{A}_t)$ and observe $r_{\hat{\mathbf{f}}_t,t}$

$$\hat{r}_{\mathbf{f},t} = r_{\mathbf{f},t} \quad \text{for} \quad \mathbf{f} \in \mathscr{F}_p.$$

If $I_t = 0$, set $\mathscr{A}_t = \mathscr{U}(\hat{\mathbf{f}}_{t-1})$, $\hat{\mathbf{f}}_t = \hat{\sigma}^t(\mathscr{A}_t)$ and observe $r_{\hat{\mathbf{f}}_t,t}$

$$\hat{r}_{\mathbf{f},t} = \begin{cases} \frac{r_{\mathbf{f},t}}{\mathbb{P}(\hat{\mathbf{f}}_t = \mathbf{f} | \mathscr{H}_t)} & \text{if} \quad \mathbf{f} \in \mathscr{U}(\hat{\mathbf{f}}_{t-1}) \cap cond(t) \quad \text{and} \quad \hat{\mathbf{f}}_t = \mathbf{f}, \\ \alpha & \text{otherwise,} \end{cases}$$

where $\mathscr{H}_t$ denotes all the randomness before time t and $cond(t) = \left\{\mathbf{f} \in \mathscr{F}_p : \mathbb{P}\left(\hat{\mathbf{f}}_t = \mathbf{f} | \mathscr{H}_t\right) > \beta\right\}$. .

End for

to choose an optimal perturbed action from the set $\mathscr{F}_p$ of all actions. In the exploitation phase, only rewards of a part of actions can be accessed to and rewards of others are estimated by a constant, and we update our action from the neighbourhood $\mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right)$ of the previous action $\hat{\mathbf{f}}_{t-1}$. This local update (or search) can hence reduce computation complexity. In addition, this local search will be enough to account for the case when x_t locates in $\mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right)$. However, when x_t is far from x_{t-1}, the algorithm is less likely to perform well since we assume implicitly that the sequence of data is stationary to the extent that a "real principal curve" exists. Otherwise, learning principal curve would be of less interest, think for example when the data is uniformly distributed on a hypercube. The parameter β needs to be carefully calibrated since on the one hand, β should not be too big to make sure that the condition $cond(t)$ is non-empty, otherwise, all rewards are estimated by the same constant and thus leading to the same descending ordering of tuples for both $\left(\sum_{s=1}^{t-1} \hat{r}_{\mathbf{f},s}, \mathbf{f} \in \mathscr{F}_p\right)$ and $\left(\sum_{s=1}^{t} \hat{r}_{\mathbf{f},s}, \mathbf{f} \in \mathscr{F}_p\right)$. Therefore, we may face the risk of having $\hat{\mathbf{f}}_{t+1}$ still in the neighbourhood of $\hat{\mathbf{f}}_t$ even if we are in the exploration phase at time $t+1$; on the other hand, very small β could result in large bias for estimation $\frac{r_{\mathbf{f},t}}{\mathbb{P}(\hat{\mathbf{f}}_t = \mathbf{f} | \mathscr{H}_t)}$ of $r_{\mathbf{f},t}$. In addition, the exploitation phase is similar but different to the label efficient prediction (Cesa-Bianchi *et al.*, 2005, Remark 1.1) since we allow an action at time t to be different from the previous one. Moreover, Neu and Bartók (2013) has proposed *Geometric Resampling* method to estimate the conditional probability $\mathbb{P}\left(\hat{\mathbf{f}}_t = \mathbf{f} | \mathscr{H}_t\right)$ since this quantity often does not have an explicit form. However, due to the simple exponential distribution of $z_{\mathbf{f}}$ chosen in our case, an explicit form of $\mathbb{P}\left(\hat{\mathbf{f}}_t = \mathbf{f} | \mathscr{H}_t\right)$ is straightforward.

Theorem 3.3. *Assume that* $p > 6$, $T \geq \max\left\{2|\mathscr{F}_p|^2, \frac{c_1 p + c_2 L + c_3}{e-1}\right\}$ *and let*

$$\beta = |\mathscr{F}_p|^{-\frac{1}{2}} T^{-\frac{1}{4}}, \qquad \alpha = \frac{c_0}{\beta}, \qquad \hat{c}_0 = \frac{2c_0}{\beta},$$

$$\eta_1 = \eta_2 = \ldots, = \eta_T = \frac{\sqrt{c_1 p + c_2 L + c_3}}{\sqrt{T(e-1)\hat{c}_0}}, \quad \epsilon = 1 - |\mathscr{F}_p|^{\frac{1}{2} - \frac{3}{p}} T^{-\frac{1}{4}}.$$

Then the procedure described in Algorithm 3.3 satisfies the regret bound

$$\sum_{t=1}^{T}\mathbb{E}\left[\Delta\left(\hat{\mathbf{f}}_t, x_{t+t_0}\right)\right] \leq \inf_{\mathbf{f}\in\mathscr{F}_p}\left[\sum_{t=1}^{T}\Delta\left(\mathbf{f}, x_{t+t_0}\right)\right] + \mathscr{O}(T^{\frac{3}{4}}).$$

Proof. With the given value of parameters, the assumption of Lemma 3.4, Lemma 3.5 and Lemma 3.6 are satisfied; Combining their inequalities lead to the following

$$\begin{aligned}
\sum_{t=1}^{T}\mathbb{E}\left[r_{\hat{\mathbf{f}}_t,t}\right] \geq & \mathbb{E}\left[\max_{\sigma}\left\{\sum_{t=1}^{T} r_{\sigma(\mathscr{A}_t),t} - \frac{1}{\eta}h(\sigma(\mathscr{A}_t))\right\}\right] - 2\alpha\beta(1-\epsilon)\sum_{t=1}^{T}\left|\mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right)\right| \\
& - \hat{c}_0^2(\mathrm{e}-1)\eta T - \hat{c}_0(\mathrm{e}-1)(c_1 p + c_2 L + c_3) \\
& - \left(1-\left|\mathscr{F}_p\right|\beta\right)\sqrt{2T\left[\frac{c_0^2}{\beta} + \alpha^2(1-\beta) + (c_0+2\alpha)^2\right]\ln\left(\frac{1}{\beta}\right)} - \left|\mathscr{F}_p\right|\beta c_0 T \\
\geq & \mathbb{E}\left[\max_{\sigma}\left\{\sum_{t=1}^{T} r_{\sigma(\mathscr{A}_t),t} - \frac{1}{\eta}h(\sigma(\mathscr{A}_t))\right\}\right] - (1-\epsilon)\left|\mathscr{F}_p\right|^{\frac{3}{p}} c_0 T \\
& - \hat{c}_0^2(\mathrm{e}-1)\eta T - \hat{c}_0(\mathrm{e}-1)(c_1 p + c_2 L + c_3) \\
& - \left(1-\left|\mathscr{F}_p\right|\beta\right)\sqrt{2T\left[\frac{c_0^2}{\beta} + \alpha^2(1-\beta) + (c_0+2\alpha)^2\right]\ln\left(\frac{1}{\beta}\right)} - \left|\mathscr{F}_p\right|\beta c_0 T \\
\geq & \mathbb{E}\left[\max_{\sigma}\left\{\sum_{t=1}^{T} r_{\sigma(\mathscr{A}_t),t} - \frac{1}{\eta}h(\sigma(\mathscr{A}_t))\right\}\right] - \mathscr{O}\left(\left|\mathscr{F}_p\right|^{\frac{1}{2}} T^{\frac{3}{4}}\right),
\end{aligned}$$

where the second inequality is due to the fact that the cardinality $\left|\mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right)\right|$ is upper bounded by $\left|\mathscr{F}_p\right|^{\frac{3}{p}}$ for $t \geq 1$. One can notice that this is a rather loosen upper bound for $\left|\mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right)\right|$ and this value could be much smaller in practice. In addition, using the definition of $r_{\mathbf{f},t}$ that $r_{\mathbf{f},t} = c_0 - \Delta(\mathbf{f}, x_{t+t_0})$ terminates the proof of Theorem 3.3. □

One can see that the regret is upper bounded by a term of order $\left(\left|\mathscr{F}_p\right|^{\frac{1}{2}} T^{\frac{3}{4}}\right)$, sublinear in T. In addition, note that the quantity $2\alpha\beta(1-\epsilon)\sum_{t=1}^{T}\left|\mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right)\right|$ and the last one appearing on the right hand side of the first inequality of the proof is in fact a measurement of bias of estimations in T rounds (we refer to Lemma 3.4 and Lemma 3.6) as $\hat{r}_{\mathbf{f},t}$ in Algorithm 3.3 is a biased estimation of $r_{\mathbf{f},t}$ when $I_t = 0$ (*i.e.,* when local update happens) but an unbiased one when $I_t = 1$. Hence when $\epsilon = 1$, these two bias-related quantities would disappear and the upper bound reduces to the one in Theorem 3.2. When $\epsilon < 1$, even if our goal is to reduce the computational complexity, we still need to make ϵ sufficiently big to ensure that the bias, or more precisely the regret bound would not be of order of T. The choice ϵ in Theorem 3.3 is therefore a compromise between the computational efficiency and regret bound, and this leads to a regret bound of order of $\left|\mathscr{F}_p\right|^{\frac{1}{2}} T^{\frac{3}{4}}$. In addition, our algorithm achieves an order that is smaller (from the point of view of both the number $\left|\mathscr{F}_p\right|$ of all actions and the total rounds T) than Kanade *et al.* (2009) since at each time, the availability of actions for our algorithm can be either the whole action set or a neighbourhood of the previous action while Kanade *et al.* (2009) considers at each time only partial and independent stochastic available set of actions generated from a predefined distribution.

3.5 Numerical experiments

We illustrate the performance of Algorithm 3.3 on synthetic and real-life data. Our implementation (hereafter denoted by `slpc` is conducted with the R language and thus our most natural competitor is the R package `princurve` (which is the algorithm from Hastie and Stuetzle, 1989). Throughout this section, values for parameters are: $p = 20$, $R = \max_{t=1,\dots,T} ||x||_2/\sqrt{d}$; We set a comparatively large value for maximum length L, *i.e.*, $L = 0.01p\sqrt{d}R$. This value is proportional to the length of the longest p segments polygonal line in $B(\mathbf{0}, \sqrt{d}R)$. The coefficient 0.01 is favored for its practical performance on both synthetic and read data. The spacing δ of the lattice is adjusted with respect to data scale.

Synthetic data We generate a data set $\left\{x_t \in \mathbb{R}^2, t = 1,\dots,100\right\}$ uniformly along the curve $y = 0.05 \times (x-5)^3$, $x \in [0,10]$, *i.e.*,

$$x_t = (x(t), y)^T + \epsilon_t,$$

where values for $x(t)$ are equidistant in $[0,10]$ and ϵ_t are i.i.d uniformly distributed on the cube $[-0.5, 0.5]^2 \subset \mathbb{R}^2$.

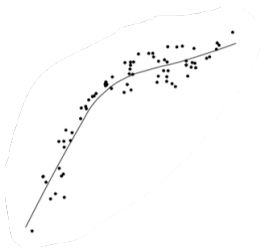
Table 3.1 shows the regret for the ground truth (sum of squared distances of all points to the true curve), `princurve` (sum of squared distances between $t+1$th observation and fitted `princurve` trained on all past t observations) and `slpc` ($\sum_{t=0}^{T-1} \Delta(\hat{\mathbf{f}}_{t+1}, x_{t+1})$). `slpc` greatly outperforms `princurve` on this example, as illustrated by Figure 3.6 and Figure 3.7. Moreover, Figure 3.7 presents the predicted principal curve $\hat{\mathbf{f}}_{t+1}$ for both `princurve` (red) and `slpc` (green). The output of `princurve` yields a curve which does not pass in "the middle of data" but rather bends towards the curvature of the data cloud: `slpc` does not suffer from this behavior. To better illustrate the way `slpc` works between two epochs, Figure 3.7 focuses on the impact of collecting a new data point on the principal curve. We see that only a local vertex is impacted, whereas the rest of the principal curve remains unaltered. This cutdown in algorithmic complexity is one the key assets of `slpc`.

ground truth	`princurve`	`slpc`
0.945 (0)	25.387 (0)	9.893 (0.246)

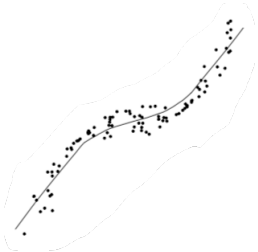
Table 3.1 – Regret (cumulative loss) on synthetic data (average over 10 trials, with standard deviation in brackets). Note: `princurve` is deterministic.

Seismic data Seismic data spanning long periods of time are essential for a thorough understanding of earthquakes. The "Centennial Earthquake Catalog" (Engdahl and Villaseñor, 2002) aims at providing a realistic picture of the seismicity distribution on Earth. It consists in a global catalog of locations and magnitudes of instrumentally recorded earthquakes from 1900 to 1999. Figure 3.8 is taken from the USGS website [1] and gives the global earthquakes locations on the period 1900–1999. The seismic data (latitude, longitude, magnitude of earthquakes, *etc.*) used in this chapter may be downloaded from this website. We focus on a particularly representative seismic active zone (a lithospheric

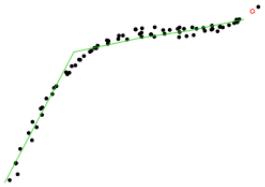
1. https://earthquake.usgs.gov/data/centennial/



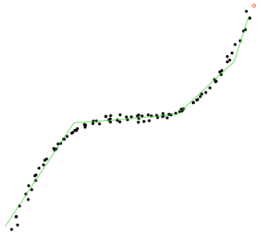
(a) $t = 75$, `princurve`.



(b) $t = 100$, `princurve`.

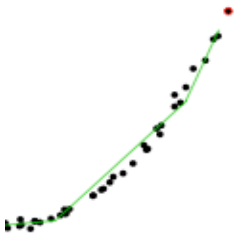


(c) $t = 75$, `slpc`.

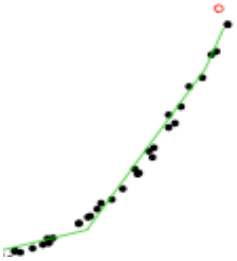


(d) $t = 100$, `slpc`.

Figure 3.6 – Synthetic data - Black dots represent data $x_{1:t}$, red point is the new observation x_{t+1}. `princurve` (solid red) and `slpc` (solid green).



(a) At time $t = 97$.



(b) At time $t = 98$.

Figure 3.7 – Zooming in: how a new data point changes the principal curve.

border close to Australia) whose longitude is between E130° to E150° and latitude between S10° to N20°, with $T = 218$ seismic recordings. As shown in Figure 3.9, `slpc` recovers nicely the tectonic plate boundary. Lastly, since no ground truth is available for real data, we use instead a slightly different version of R^2 coefficient to assess the performance: residuals are replaced by the squared distance between data points and their projections onto the principal curve. The average of R^2 coefficient over 10 trials in this example is 0.990.

Daily commute data The identification of segments of personal daily commuting trajectories can help taxi or bus companies to optimize their fleets and increase frequencies on segments with high commuting activity. Sequential principal curves appear to be an ideal tool to address this learning problem: we test our algorithm on trajectory data from the University of Illinois at Chicago [2]. The data is obtained from the GPS reading systems carried by two of the lab members during their daily commute for 6 months in the Cook county and the Dupage county of Illinois. Figure 3.10 presents the learning curves yielded by `princurve` and `slpc` on geolocalization data for the first person, on

2. https://www.cs.uic.edu/~boxu/mp2p/gps_data.html

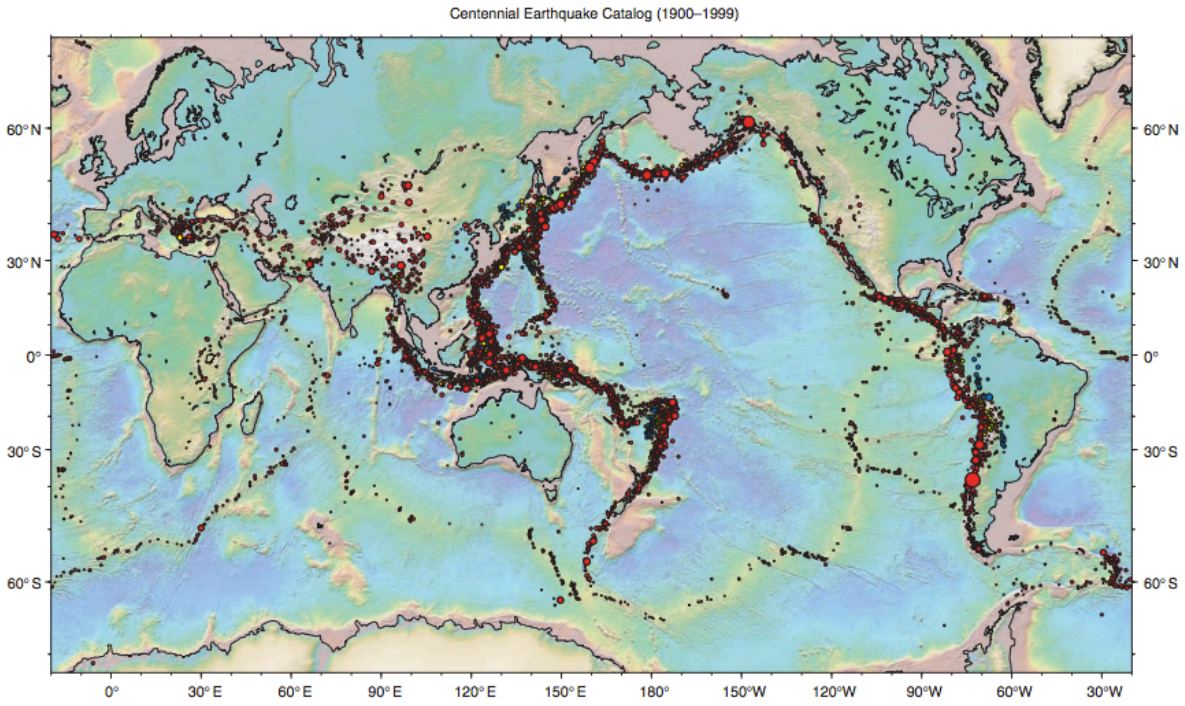


Figure 3.8 – Seismic data from https://earthquake.usgs.gov/data/centennial/

May 30 in the data set. A particularly remarkable asset of `slpc` is that abrupt curvature in the data sequence is perfectly captured, whereas `princurve` does not enjoy the same flexibility. Again, we use the R^2 coefficient to assess the performance (where residuals are replaced by the squared distance between data points and their projections onto the principal curve). The average over 10 trials is 0.998.

3.6 Proofs

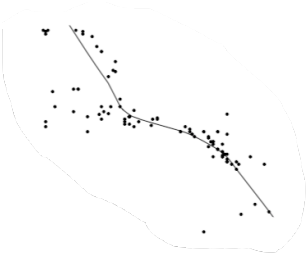
This section contains the proof of Theorem 3.2 (note that Theorem 3.1 is a straightforward consequence, with $\eta_t = \eta$, $t = 0, \ldots, T$) and necessary lemmas for Theorem 3.3. Let us first define for each $t = 1, \ldots, T$ the following forecaster sequence $\left(\hat{\mathbf{f}}_t^\star\right)_{1:T}$

$$\hat{\mathbf{f}}_t^\star = \operatorname*{arg\,inf}_{\mathbf{f}\in\mathscr{F}_p} \left\{ \sum_{s=1}^{t} \Delta(\mathbf{f}, x_s) + \frac{1}{\eta_{t-1}} h(\mathbf{f}) - \frac{1}{\eta_{t-1}} z_{\mathbf{f}} \right\} = \operatorname*{arg\,inf}_{\mathbf{f}\in\mathscr{F}_p} \left\{ \sum_{s=1}^{t} \tilde{\Delta}_{\mathbf{f},s} \right\}, \quad t \geq 1,$$

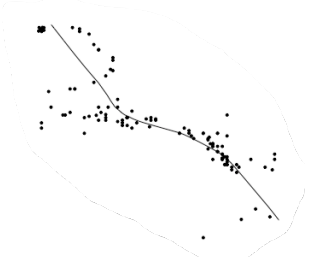
where $\tilde{\Delta}_{\mathbf{f},s} = \Delta(\mathbf{f}, x_s) + \left(\frac{1}{\eta_{s-1}} - \frac{1}{\eta_{s-2}}\right)(h(\mathbf{f}) - z_{\mathbf{f}})$ and by convention $1/\eta_{-1} = 0$. Note that $\hat{\mathbf{f}}_t^\star$ is an "illegal" forecaster since it peeks into the future by observing additional $\Delta(\mathbf{f}, x_t)$. In addition, denote by

$$\mathbf{f}_T^\star = \operatorname*{arg\,inf}_{\mathbf{f}\in\mathscr{F}_p} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) + \frac{1}{\eta_{T-1}} h(\mathbf{f}) \right\} \tag{3.8}$$

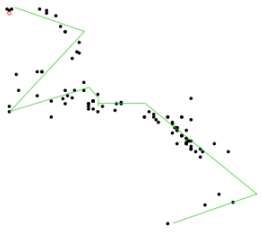
the polygonal line in $\mathscr{F}_p$ which minimizes the cumulative loss in the first T rounds plus a penalty term. $\mathbf{f}_T^\star$ is deterministic while $\hat{\mathbf{f}}_t^\star$ is a random quantity since it depends on $z_{\mathbf{f}}$, drawn from π, for $\mathbf{f} \in \mathscr{F}_p$. If several $\mathbf{f}$ attain the infimum of (3.8), we choose $\mathbf{f}_T^\star$ as the one having the smallest complexity. We now enunciate the first (out of three) intermediary technical result.



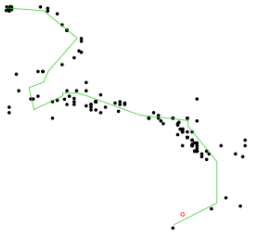
(a) `princurve`, $t = 100$



(b) `princurve`, $t = 125$



(c) `slpc`, $t = 100$



(d) `slpc`, $t = 125$

Figure 3.9 – Seismic data. Black dots represent seismic recordings $x_{1:t}$, red dot is the new recording x_{t+1}.

Lemma 3.1. *For $T \geq 1$ and any sequence $x_1, \ldots, x_T$ in $B(\mathbf{0}, \sqrt{d}R)$,*

$$\sum_{t=1}^{T} \tilde{\Delta}_{\hat{\mathbf{f}}_t^\star, t} \leq \sum_{t=1}^{T} \tilde{\Delta}_{\hat{\mathbf{f}}_T^\star, t}, \qquad \pi\text{-almost surely.} \tag{3.9}$$

Note that this lemma is classical in the follow the perturbed leader setting (chapter 4 of Cesa-Bianchi and Lugosi, 2006). For the completeness of this section, we would still give its proof in the following.

Proof. Proof by induction on T. Clearly (3.9) holds for $T = 1$. Assume that (3.9) holds for $T - 1$, *i.e.*,

$$\sum_{t=1}^{T-1} \tilde{\Delta}_{\hat{\mathbf{f}}_t^\star, t} \leq \sum_{t=1}^{T-1} \tilde{\Delta}_{\hat{\mathbf{f}}_{T-1}^\star, t}.$$

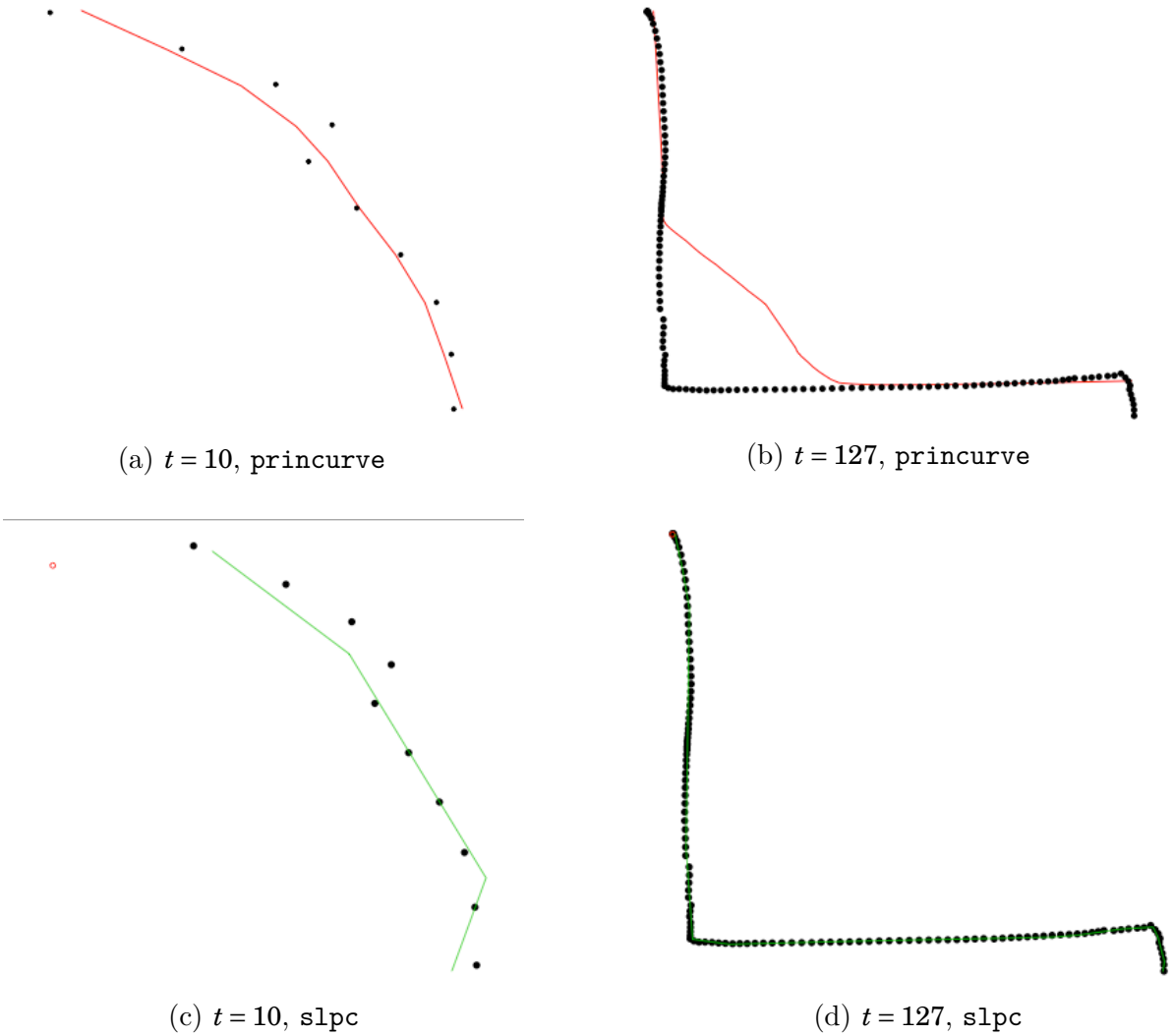
Then

$$\begin{aligned}
\sum_{t=1}^{T} \tilde{\Delta}_{\hat{\mathbf{f}}_t^\star, t} &= \sum_{t=1}^{T-1} \tilde{\Delta}_{\hat{\mathbf{f}}_t^\star, t} + \tilde{\Delta}_{\hat{\mathbf{f}}_T^\star, T} \\
&\leq \sum_{t=1}^{T-1} \tilde{\Delta}_{\hat{\mathbf{f}}_{T-1}^\star, t} + \tilde{\Delta}_{\hat{\mathbf{f}}_T^\star, T} \\
&\leq \sum_{t=1}^{T-1} \tilde{\Delta}_{\hat{\mathbf{f}}_T^\star, t} + \tilde{\Delta}_{\hat{\mathbf{f}}_T^\star, T} = \sum_{t=1}^{T} \tilde{\Delta}_{\hat{\mathbf{f}}_T^\star, t},
\end{aligned}$$

where the first inequality is due to the assumption that (3.9) holds for $T - 1$; the second inequality is by the definition of $\mathbf{f}_{T-1}^\star$ which minimizes $\sum_{s=1}^{T-1} \tilde{\Delta}_{\mathbf{f}, s}$ with respect to $\mathbf{f} \in \mathscr{F}_p$. □

By (3.9), if we rewrite $\tilde{\Delta}_{\hat{\mathbf{f}}_t^\star, t}$ and $\tilde{\Delta}_{\hat{\mathbf{f}}_T^\star, t}$, we have π-almost surely that

$$\sum_{t=1}^{T} \Delta(\hat{\mathbf{f}}_t^\star, x_t) \leq \left(\sum_{t=1}^{T} \Delta(\hat{\mathbf{f}}_T^\star, x_t) + \frac{1}{\eta_{T-1}} h(\hat{\mathbf{f}}_T^\star) - \frac{1}{\eta_{T-1}} z_{\hat{\mathbf{f}}_T^\star} \right) + \sum_{t=1}^{T} \left(\frac{1}{\eta_{t-2}} - \frac{1}{\eta_{t-1}} \right) \left(h(\hat{\mathbf{f}}_t^\star) - z_{\hat{\mathbf{f}}_t^\star} \right)$$



(a) $t = 10$, `princurve` (b) $t = 127$, `princurve`

(c) $t = 10$, `slpc` (d) $t = 127$, `slpc`

Figure 3.10 – Daily commute data - Black dots represent collected locations $x_{1:t}$, red point is the new observation x_{t+1}. `princurve` (solid red) and `slpc` (solid green).

$$\leq \left(\sum_{t=1}^{T} \Delta(\mathbf{f}_T^\star, x_t) + \frac{1}{\eta_{T-1}} h(\mathbf{f}_T^\star) - \frac{1}{\eta_{T-1}} z_{\mathbf{f}_T^\star} \right) + \sum_{t=1}^{T} \left(\frac{1}{\eta_{t-2}} - \frac{1}{\eta_{t-1}} \right) \left(h(\hat{\mathbf{f}}_t^\star) - z_{\hat{\mathbf{f}}_t^\star} \right)$$
$$= \inf_{\mathbf{f} \in \mathscr{F}_p} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) + \frac{1}{\eta_{T-1}} h(\mathbf{f}) \right\} - \frac{1}{\eta_{T-1}} z_{\mathbf{f}_T^\star} + \sum_{t=1}^{T} \left(\frac{1}{\eta_{t-2}} - \frac{1}{\eta_{t-1}} \right) \left(h(\hat{\mathbf{f}}_t^\star) - z_{\hat{\mathbf{f}}_t^\star} \right),$$

where the second inequality is due to the definition of $\hat{\mathbf{f}}_T^\star$ and $\mathbf{f}_T^\star$. Hence

$$\mathbb{E}\left[\sum_{t=1}^{T} \Delta\left(\hat{\mathbf{f}}_t^\star, x_t \right) \right] \leq \inf_{\mathbf{f} \in \mathscr{F}_p} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) + \frac{1}{\eta_{T-1}} h(\mathbf{f}) \right\} - \frac{1}{\eta_{T-1}} \mathbb{E}[z_{\mathbf{f}_T^\star}] + \sum_{t=1}^{T} \mathbb{E}\left[\left(\frac{1}{\eta_{t-1}} - \frac{1}{\eta_{t-2}} \right) \left(-h(\hat{\mathbf{f}}_t^\star) + z_{\hat{\mathbf{f}}_t^\star} \right) \right]$$
$$\leq \inf_{\mathbf{f} \in \mathscr{F}_p} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) + \frac{1}{\eta_{T-1}} h(\mathbf{f}) \right\} + \sum_{t=1}^{T} \left(\frac{1}{\eta_{t-1}} - \frac{1}{\eta_{t-2}} \right) \mathbb{E}\left[\sup_{\mathbf{f} \in \mathscr{F}_p} \left(-h(\mathbf{f}) + z_{\mathbf{f}} \right) \right]$$
$$= \inf_{\mathbf{f} \in \mathscr{F}_p} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) + \frac{1}{\eta_{T-1}} h(\mathbf{f}) \right\} + \frac{1}{\eta_{T-1}} \mathbb{E}\left[\sup_{\mathbf{f} \in \mathscr{F}_p} \left(-h(\mathbf{f}) + z_{\mathbf{f}} \right) \right],$$

where the second inequality is due to $\mathbb{E}[z_{\mathbf{f}_T^\star}] \geq 0$ and $\left(\frac{1}{\eta_{t-1}} - \frac{1}{\eta_{t-2}} \right) \geq 0$ for $t = 1, \ldots, T$ since η_t is non-increasing in t in Theorem 3.2. In addition, for $y \geq 0$, one has

$$\mathbb{P}\left(-h(\mathbf{f}) + z_{\mathbf{f}} > y \right) = \mathrm{e}^{-h(\mathbf{f}) - y}.$$

Hence, for any $y \geq 0$

$$\mathbb{P}\left(\sup_{\mathbf{f}\in\mathscr{F}_p}(-h(\mathbf{f})+z_{\mathbf{f}})>y\right) \leq \sum_{\mathbf{f}\in\mathscr{F}_p}\mathbb{P}(z_{\mathbf{f}}\geq h(\mathbf{f})+y) = \sum_{\mathbf{f}\in\mathscr{F}_p} \mathrm{e}^{-h(\mathbf{f})}\mathrm{e}^{-y} = u\mathrm{e}^{-y},$$

where $u = \sum_{\mathbf{f}\in\mathscr{F}_p}\mathrm{e}^{-h(\mathbf{f})}$. Therefore, we have

$$\begin{aligned}
\mathbb{E}\left[\sup_{\mathbf{f}\in\mathscr{F}_p}(-h(\mathbf{f})+z_{\mathbf{f}})-\ln u\right] &\leq \mathbb{E}\left[\max\left(0,\sup_{\mathbf{f}\in\mathscr{F}_p}(-h(\mathbf{f})+z_{\mathbf{f}}-\ln u)\right)\right] \\
&\leq \int_0^\infty \mathbb{P}\left(\max\left(0,\sup_{\mathbf{f}\in\mathscr{F}_p}(-h(\mathbf{f})+z_{\mathbf{f}}-\ln u)\right)>y\right)\mathrm{d}y \\
&\leq \int_0^\infty \mathbb{P}\left(\sup_{\mathbf{f}\in\mathscr{F}_p}(-h(\mathbf{f})+z_{\mathbf{f}})>y+\ln u\right)\mathrm{d}y \\
&\leq \int_0^\infty u\mathrm{e}^{-(y+\ln u)}\mathrm{d}y = 1.
\end{aligned}$$

We thus obtain

$$\mathbb{E}\left[\sum_{t=1}^T \Delta\left(\hat{\mathbf{f}}_t^\star, x_t\right)\right] \leq \inf_{\mathbf{f}\in\mathscr{F}_p}\left\{\sum_{t=1}^T \Delta(\mathbf{f},x_t)+\frac{1}{\eta_{T-1}}h(\mathbf{f})\right\}+\frac{1}{\eta_{T-1}}\left(1+\ln\sum_{\mathbf{f}\in\mathscr{F}_p}\mathrm{e}^{-h(\mathbf{f})}\right). \tag{3.10}$$

Next, we control the regret of Algorithm 3.2.

Lemma 3.2. *Assume that $z_{\mathbf{f}}$ is sampled from exponential distribution in $\mathbb{R}$,* i.e., *$\pi(z) = \mathrm{e}^{-z}\mathbb{1}_{\{z>0\}}$. Assume that $\sup_{t=1,\ldots,T}\eta_{t-1} \leq \frac{1}{d(2R+\delta)^2}$. Then for any sequence $x_1, x_2, \ldots, x_T \in B(\mathbf{0}, \sqrt{d}R)$,*

$$\sum_{t=1}^T \mathbb{E}\left[\Delta\left(\hat{\mathbf{f}}_t, x_t\right)\right] \leq \sum_{t=1}^T \left(1+\eta_{t-1}c_0(\mathrm{e}-1)\right)\mathbb{E}\left[\Delta\left(\hat{\mathbf{f}}_t^\star, x_t\right)\right]. \tag{3.11}$$

where $c_0 = d(2R+\delta)^2$.

Proof. Before proceeding to the proof, let us first give some notation that would be useful for proof convenience. For $\mathbf{f}\in\mathscr{F}_p$, denote by

$$A_{t-1}^{\mathbf{f}} = \sum_{s=1}^{t-1}\Delta(\mathbf{f},x_s)+\frac{1}{\eta_{t-1}}h(\mathbf{f}), \quad t\geq 1.$$

In particular, we made a convention here and in what follows that $\sum_{s=1}^0 \Delta(\mathbf{f},x_s)\equiv 0$. For some fixed $\mathbf{f}_0\in\mathscr{F}_p$ and a vector $\vec{x} = (x_{\mathbf{f}})_{\mathbf{f}\in\mathscr{F}_p} \in (\mathbb{R}_+^*)^{|\mathscr{F}_p|}$ (recall that $|\mathscr{F}_p|$ is the cardinality of $\mathscr{F}_p$), define for $t\geq 1$

$$A_{t-1}^{-\mathbf{f}_0,\vec{x}} = \min_{\mathbf{f}\neq\mathbf{f}_0}\left\{\sum_{s=1}^{t-1}\Delta(\mathbf{f},x_s)+\frac{1}{\eta_{t-1}}(h(\mathbf{f})-x_{\mathbf{f}})\right\}, \quad A_{\star,t}^{-\mathbf{f}_0,\vec{x}} = \min_{\mathbf{f}\neq\mathbf{f}_0}\left\{\sum_{s=1}^{t}\Delta(\mathbf{f},x_s)+\frac{1}{\eta_{t-1}}(h(\mathbf{f})-x_{\mathbf{f}})\right\}.$$

Now let us give the proof which is similar to that of Hutter and Poland (2005). For $\mathbf{f}\in\mathscr{F}_p, m\in\mathbb{R}$, one has

$$\frac{\mathbb{P}\left(z_{\mathbf{f}} > \eta_{t-1}\left(A^{\mathbf{f}}_{t-1} - m + c_0\right)\right)}{\mathbb{P}\left(z_{\mathbf{f}} > \eta_{t-1}\left(A^{\mathbf{f}}_{t-1} - m\right)\right)} = \begin{cases} \mathrm{e}^{-\eta_{t-1}c_0} & \text{if } m < A^{\mathbf{f}}_{t-1}\,, \\ \mathrm{e}^{-\eta_{t-1}\left(A^{\mathbf{f}}_{t-1} - m + c_0\right)} & \text{if } A^{\mathbf{f}}_{t-1} \le m < A^{\mathbf{f}}_{t-1} + c_0, \\ 1 & \text{if } A^{\mathbf{f}}_{t-1} + c_0 \le m. \end{cases}$$

It is easy to see that $\frac{\mathbb{P}\left(z_{\mathbf{f}} > \eta_{t-1}\left(A^{\mathbf{f}}_{t-1} - m + c_0\right)\right)}{\mathbb{P}\left(z_{\mathbf{f}} > \eta_{t-1}\left(A^{\mathbf{f}}_{t-1} - m\right)\right)} \ge \mathrm{e}^{-\eta_{t-1}c_0}$ holds uniformly for all $m \in \mathbb{R}$. Hence for $\vec{x} \in (\mathbb{R}^*_+)^{|\mathscr{F}_p|}$, we have

$$\begin{aligned}
\mathbb{P}\left(\hat{\mathbf{f}}_t = \mathbf{f}_0 | z_{\mathbf{f}} = x_{\mathbf{f}}, \mathbf{f} \neq \mathbf{f}_0\right) &= \mathbb{P}\left(\sum_{s=1}^{t-1} \Delta(\mathbf{f}_0, x_s) + \frac{1}{\eta_{t-1}}(h(\mathbf{f}_0) - z_{\mathbf{f}_0}) < A^{-\mathbf{f}_0,\vec{x}}_{t-1} \middle| z_{\mathbf{f}} = x_{\mathbf{f}}, \mathbf{f} \neq \mathbf{f}_0\right) \\
&= \mathbb{P}\left(z_{\mathbf{f}_0} > \eta_{t-1}\left(A^{\mathbf{f}_0}_{t-1} - A^{-\mathbf{f}_0,\vec{x}}_{t-1}\right) \middle| z_{\mathbf{f}} = x_{\mathbf{f}}, \mathbf{f} \neq \mathbf{f}_0\right) \\
&= \mathbb{P}\left(z_{\mathbf{f}_0} > \eta_{t-1}\left(A^{\mathbf{f}_0}_{t-1} - A^{-\mathbf{f}_0,\vec{x}}_{t-1}\right)\right) \\
&\le \mathrm{e}^{\eta_{t-1}c_0}\mathbb{P}\left(z_{\mathbf{f}_0} > \eta_{t-1}\left(A^{\mathbf{f}_0}_{t-1} - A^{-\mathbf{f}_0,\vec{x}}_{t-1} + c_0\right)\right) \\
&\le \mathrm{e}^{\eta_{t-1}c_0}\mathbb{P}\left(z_{\mathbf{f}_0} > \eta_{t-1}\left(A^{\mathbf{f}_0}_{t-1} + \Delta(\mathbf{f}_0, x_t) - A^{-\mathbf{f}_0,\vec{x}}_{\star,t}\right)\right) \\
&= \mathrm{e}^{\eta_{t-1}c_0}\mathbb{P}\left(z_{\mathbf{f}_0} > \eta_{t-1}\left(A^{\mathbf{f}_0}_{t-1} + \Delta(\mathbf{f}_0, x_t) - A^{-\mathbf{f}_0,\vec{x}}_{\star,t}\right) \middle| z_{\mathbf{f}} = x_{\mathbf{f}}, \mathbf{f} \neq \mathbf{f}_0\right) \\
&= \mathrm{e}^{\eta_{t-1}c_0} P\left(\sum_{s=1}^{t} \Delta(\mathbf{f}_0, x_s) + \frac{1}{\eta_{t-1}}(h(\mathbf{f}_0) - z_{\mathbf{f}_0}) < A^{-\mathbf{f}_0,\vec{x}}_{\star,t} \middle| z_{\mathbf{f}} = x_{\mathbf{f}}, \mathbf{f} \neq \mathbf{f}_0\right) \\
&= \mathrm{e}^{\eta_{t-1}c_0}\mathbb{P}\left(\hat{\mathbf{f}}^\star_t = \mathbf{f}_0 | z_{\mathbf{f}} = x_{\mathbf{f}}, \mathbf{f} \neq \mathbf{f}_0\right),
\end{aligned}$$

where the third equality is due to the independence between $z_{\mathbf{f}}, \mathbf{f} \in \mathscr{F}_p$; the second inequality relies on the fact that $\Delta(\mathbf{f}_0, x_t) + A^{-\mathbf{f}_0,\vec{x}}_{t-1} \le c_0 + A^{-\mathbf{f}_0,\vec{x}}_{\star,t}$. Therefore,

$$\begin{aligned}
\mathbb{P}\left(\hat{\mathbf{f}}_t = \mathbf{f}_0\right) &= \int \mathbb{P}\left(\hat{\mathbf{f}}_t = \mathbf{f}_0 \middle| z_{\mathbf{f}} = x_{\mathbf{f}}, \mathbf{f} \neq \mathbf{f}_0\right) \pi\left(x^{-\mathbf{f}}\right) \mathrm{d}x^{-\mathbf{f}} \\
&\le \mathrm{e}^{\eta_{t-1}c_0} \int \mathbb{P}\left(\hat{\mathbf{f}}^\star_t = \mathbf{f}_0 \middle| z_{\mathbf{f}} = x_{\mathbf{f}}, \mathbf{f} \neq \mathbf{f}_0\right) \pi\left(x^{-\mathbf{f}}\right) \mathrm{d}x^{-\mathbf{f}} = \mathrm{e}^{\eta_{t-1}c_0}\mathbb{P}\left(\hat{\mathbf{f}}^\star_t = \mathbf{f}_0\right),
\end{aligned}$$

where $x^{-\mathbf{f}}$ is the vector obtained by removing the coordinate $x_{\mathbf{f}}$ from $\vec{x}$. The above inequality leads to $\mathbb{E}[\Delta(\hat{\mathbf{f}}_t, x_t)] = \sum_{\mathbf{f}_0 \in \mathscr{F}_p} \Delta(\mathbf{f}_0, x_t)\mathbb{P}\left(\hat{\mathbf{f}}_t = \mathbf{f}_0\right) \le \mathrm{e}^{\eta_{t-1}c_0}\mathbb{E}[\Delta(\hat{\mathbf{f}}^\star_t, x_t)]$. Finally, summing on t and using the elementary inequality $\mathrm{e}^x \le 1 + (\mathrm{e}-1)x$ if $x \in (0,1)$ concludes the proof. □

Lemma 3.3. *For $k \in [\![1, p]\!]$, we control the cardinality of set $\{\mathbf{f} \in \mathscr{F}_p, \mathscr{K}(\mathbf{f}) = k\}$ as*

$$\begin{aligned}
\ln\left|\{\mathbf{f} \in \mathscr{F}_p, \mathscr{K}(\mathbf{f}) = k\}\right| &\le \left(\ln(8V_d) + 3d^{\frac{3}{2}} - d\right)k + \left(\frac{\ln 2}{\delta\sqrt{d}} + \frac{d}{\delta}\right)L + d\ln\left(\frac{\sqrt{d}(2R+\delta)}{\delta}\right) + \ln(2V_d) \\
&\triangleq c_1 k + c_2 L + c_3,
\end{aligned}$$

where V_d denotes the volume of the unit ball in $\mathbb{R}^d$ and its value is $\frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)}$, where Γ denotes the gamma function.

Proof. First, by the last inequality on page 54 and the one at the third line of page 55 of Kégl (1999), one can have that

$$\left|\{\mathbf{f} \in \mathscr{F}_p, \mathscr{K}(\mathbf{f}) = k\}\right| \le 2^{\left\lceil \frac{L}{\sqrt{d}\delta} \right\rceil + 3k} V_d^{k+1} \left(\frac{\sqrt{d}(2R+\delta)}{\delta}\right)^d \left(\frac{L}{k\delta} + 3\sqrt{d}\right)^{kd}.$$

Note that the diameter in Kégl (1999) of the range of $(x_t)_{1:T}$ equals to $2\sqrt{d}R$ in our case since $(x_t)_{1:T}$ are bounded in $B(\mathbf{0}, \sqrt{d}R)$. Hence

$$\begin{aligned}
\ln\left|\{\mathbf{f}\in\mathscr{F}_p, \mathscr{K}(\mathbf{f})=k\}\right| &\le \left(\frac{L}{\sqrt{d}\delta}+3k+1\right)\ln 2+(k+1)\ln V_d + kd\ln\left(\frac{L}{k\delta}+3\sqrt{d}\right)+d\ln\left(\frac{\sqrt{d}(2R+\delta)}{\delta}\right)\\
&\le \left(\frac{L}{\sqrt{d}\delta}+3k+1\right)\ln 2+(k+1)\ln V_d + kd\left(\frac{L}{k\delta}+3\sqrt{d}-1\right)+d\ln\left(\frac{\sqrt{d}(2R+\delta)}{\delta}\right)\\
&\le \left(\ln(8V_d)+3d^{\frac{3}{2}}-d\right)k+\left(\frac{\ln 2}{\sqrt{d}\delta}+\frac{d}{\delta}\right)L+d\ln\left(\frac{\sqrt{d}(2R+\delta)}{\delta}\right)+\ln(2V_d),
\end{aligned}$$

where we apply the inequality $\ln x \le x-1, x>0$ to the term $\ln\left(\frac{L}{k\delta}+3\sqrt{d}\right)$ on the right hand side of the first inequality above. This can bring out L from the logarithmic to form a bound of the shape $c_1k+c_2L+c_3$ for the penalty function h. □

We now have all the ingredients to prove Theorem 3.1 and Theorem 3.2.

First, combining (3.10) and (3.11) yields that

$$\begin{aligned}
\sum_{t=1}^{T}\mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t,x_t)\right] &\le \inf_{\mathbf{f}\in\mathscr{F}_p}\left\{\sum_{t=1}^{T}\Delta(\mathbf{f},x_t)+\frac{1}{\eta_{T-1}}h(\mathbf{f})\right\}+\frac{1}{\eta_{T-1}}\left(1+\ln\sum_{\mathbf{f}\in\mathscr{F}_p}\mathrm{e}^{-h(\mathbf{f})}\right)\\
&\quad + c_0(\mathrm{e}-1)\sum_{t=1}^{T}\eta_{t-1}\mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t^{\star},x_t)\right]\\
&\le \inf_{k\in[\![1,p]\!]}\left\{\inf_{\substack{\mathbf{f}\in\mathscr{F}_p\\ \mathscr{K}(\mathbf{f})=k}}\left\{\sum_{t=1}^{T}\Delta(\mathbf{f},x_t)+\frac{h(\mathbf{f})}{\eta_{T-1}}\right\}\right\}+\frac{1}{\eta_{T-1}}\left(1+\ln\sum_{\mathbf{f}\in\mathscr{F}_p}\mathrm{e}^{-h(\mathbf{f})}\right)\\
&\quad + c_0(\mathrm{e}-1)\sum_{t=1}^{T}\eta_{t-1}\mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t^{\star},x_t)\right]\\
&= S_{T,h,\eta_{T-1}}+\frac{1}{\eta_{T-1}}\left(1+\ln\sum_{\mathbf{f}\in\mathscr{F}_p}\mathrm{e}^{-h(\mathbf{f})}\right)+c_0(\mathrm{e}-1)\sum_{t=1}^{T}\eta_{t-1}\mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t^{\star},x_t)\right],
\end{aligned}$$

where

$$S_{T,h,\eta_{T-1}}=\inf_{k\in[\![1,p]\!]}\left\{\inf_{\substack{\mathbf{f}\in\mathscr{F}_p\\ \mathscr{K}(\mathbf{f})=k}}\left\{\sum_{t=1}^{T}\Delta(\mathbf{f},x_t)+\frac{h(\mathbf{f})}{\eta_{T-1}}\right\}\right\}.$$

Assume that $\eta_t=\eta$, $t=0,\dots,T$, then

$$\begin{aligned}
\sum_{t=1}^{T}\mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t,x_t)\right] &\le S_{T,h,\eta}+\frac{1}{\eta}\left(1+\ln\sum_{\mathbf{f}\in\mathscr{F}_p}\mathrm{e}^{-h(\mathbf{f})}\right)+c_0(\mathrm{e}-1)\eta\sum_{t=1}^{T}\mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t^{\star},x_t)\right]\\
&\le S_{T,h,\eta}+c_0(\mathrm{e}-1)\eta S_{T,h,\eta}+\frac{1+c_0(\mathrm{e}-1)\eta}{\eta}\left(1+\ln\sum_{\mathbf{f}\in\mathscr{F}_p}\mathrm{e}^{-h(\mathbf{f})}\right), \qquad (3.12)
\end{aligned}$$

where the second inequality is obtained by (3.10). We terminate the proof of Theorem 3.1.

Finally, we give the proof of Theorem 3.2. Assume that

$$\eta_0=\frac{1}{c_0}\quad\text{and}\quad\eta_t=\min\left\{\frac{1}{c_0},\frac{\sqrt{c_1p+c_2L+c_3}}{c_0\sqrt{(\mathrm{e}-1)t}}\right\},\qquad t=1,\dots,T.$$

Since $\mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t^\star, x_t)\right] \le c_0$ for any $t = 1, \dots, T$, we have

$$\sum_{t=1}^{T} \mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t, x_t)\right] \le \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\substack{\mathbf{f} \in \mathscr{F}_p \\ \mathscr{K}(\mathbf{f}) = k}} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) + \frac{h(\mathbf{f})}{\eta_{T-1}} \right\} \right\} + \frac{1}{\eta_{T-1}} \left(1 + \ln \sum_{\mathbf{f} \in \mathscr{F}_p} \mathrm{e}^{-h(\mathbf{f})} \right) + c_0^2 (\mathrm{e}-1) \sum_{t=1}^{T} \eta_{t-1}$$
$$\le \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\substack{\mathbf{f} \in \mathscr{F}_p \\ \mathscr{K}(\mathbf{f}) = k}} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) + \frac{h(\mathbf{f})}{\eta_{T-1}} \right\} \right\} + c_0^2 (\mathrm{e}-1) \sum_{t=1}^{T} \eta_{t-1}.$$

If $T \le 1 + \frac{c_1 p + c_2 L + c_3}{\mathrm{e}-1} \triangleq 1 + M_{p,L}$ (*i.e.*, $\eta_1 = \eta_2 = \cdots = \eta_{T-1} = \frac{1}{c_0}$), then

$$\sum_{t=1}^{T} \mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t, x_t)\right] \le \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\substack{\mathbf{f} \in \mathscr{F}_p \\ \mathscr{K}(\mathbf{f}) = k}} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) + c_0 h(\mathbf{f}) \right\} \right\} + c_0 (\mathrm{e}-1) T$$
$$\le \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\substack{\mathbf{f} \in \mathscr{F}_p \\ \mathscr{K}(\mathbf{f}) = k}} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) \right\} \right\} + c_0 (c_1 p + c_2 L + c_3) + c_0 (\mathrm{e} - 1 + c_1 p + c_2 L + c_3)$$
$$\le \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\substack{\mathbf{f} \in \mathscr{F}_p \\ \mathscr{K}(\mathbf{f}) = k}} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) \right\} \right\} + 3 c_0 (c_1 p + c_2 L + c_3), \tag{3.13}$$

where the last inequality is obvious since $\mathrm{e} - 1 \le c_1 p + c_2 L + c_3$. If $T > 1 + M_{p,L}$, then $\eta_{T-1} = \frac{\sqrt{c_1 p + c_2 L + c_3}}{c_0 \sqrt{(\mathrm{e}-1)(T-1)}}$ and

$$c_0^2 (\mathrm{e}-1) \sum_{t=1}^{T} \eta_{t-1} = c_0 (\mathrm{e}-1) \left(\sum_{t \le 1 + M_{p,L}} \frac{1}{c_0} + \sum_{t > 1 + M_{p,L}}^{T} \frac{\sqrt{c_1 p + c_2 L + c_3}}{c_0 \sqrt{(\mathrm{e}-1) t}} \right)$$
$$\le c_0^2 (\mathrm{e}-1) \sum_{t=1}^{T} \frac{\sqrt{c_1 p + c_2 L + c_3}}{c_0 \sqrt{(\mathrm{e}-1) t}} \le 2 c_0 \sqrt{(\mathrm{e}-1) T (c_1 p + c_2 L + c_3)}.$$

Hence, we have

$$\sum_{t=1}^{T} \mathbb{E}\left[\Delta(\hat{\mathbf{f}}_t, x_t)\right] \le \inf_{k \in [\![1,p]\!]} \left\{ \inf_{\substack{\mathbf{f} \in \mathscr{F}_p \\ \mathscr{K}(\mathbf{f}) = k}} \left\{ \sum_{t=1}^{T} \Delta(\mathbf{f}, x_t) + c_0 \sqrt{(\mathrm{e}-1) T (c_1 k + c_2 L + c_3)} \right\} \right\}$$
$$+ 2 c_0 \sqrt{(\mathrm{e}-1) T (c_1 p + c_2 L + c_3)}. \tag{3.14}$$

Combining (3.13) and (3.14) concludes the proof of Theorem 3.2.

Lemma 3.4. *Under the Algorithm 3.3, if* $1 \ge \epsilon > 0$, $1 > \beta > 0$, $\alpha \ge \frac{(1-\beta) c_0}{\beta}$ *and* $\left| \mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right) \right| \ge 2$ *for all* $t \ge 2$, *where* $\left| \mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right) \right|$ *is the cardinality of* $\mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right)$, *then we have*

$$\sum_{t=1}^{T} \mathbb{E}\left[r_{\hat{\mathbf{f}}_t, t} \right] \ge \sum_{t=1}^{T} \mathbb{E}\left[\hat{r}_{\hat{o}^t(\mathscr{A}_t), t} \right] - 2(1-\epsilon) \alpha \beta \sum_{t=1}^{T} \left| \mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right) \right|.$$

Proof. First notice that $\mathscr{A}_t = \mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right)$ if $I_t = 0$, and that for $t \ge 2$

$$\mathbb{E}\left[r_{\hat{\mathbf{f}}_t, t} \middle| \mathscr{H}_t, I_t = 0 \right] = \mathbb{E}\left[r_{\hat{o}^t(\mathscr{A}_t), t} \middle| \mathscr{H}_t, I_t = 0 \right]$$

$$
\begin{aligned}
&= \sum_{\mathbf{f}\in\mathscr{A}_t\cap cond(t)} r_{\mathbf{f},t}\mathbb{P}\left(\hat{\sigma}^t(\mathscr{A}_t)=\mathbf{f}\middle|\mathscr{H}_t\right) + \sum_{\mathbf{f}\in\mathscr{A}_t\cap cond(t)^c} r_{\mathbf{f},t}\mathbb{P}\left(\hat{\sigma}^t(\mathscr{A}_t)=\mathbf{f}\middle|\mathscr{H}_t\right)\\
&\geq \sum_{\mathbf{f}\in\mathscr{A}_t\cap cond(t)} r_{\mathbf{f},t} + \sum_{\mathbf{f}\in\mathscr{A}_t\cap cond(t)^c} \alpha\mathbb{P}\left(\hat{\sigma}^t(\mathscr{A}_t)=\mathbf{f}\middle|\mathscr{H}_t\right)\\
&\quad -(1-\beta)\sum_{\mathbf{f}\in\mathscr{A}_t\cap cond(t)} r_{\mathbf{f},t} - \sum_{\mathbf{f}\in\mathscr{A}_t\cap cond(t)^c}(\alpha - r_{\mathbf{f},t})\mathbb{P}\left(\hat{\sigma}^t(\mathscr{A}_t)=\mathbf{f}\middle|\mathscr{H}_t\right)\\
&= \mathbb{E}\left[\hat{r}_{\hat{\sigma}^t(\mathscr{A}_t),t}\middle|\mathscr{H}_t, I_t=0\right] - (1-\beta)\sum_{\mathbf{f}\in\mathscr{A}_t\cap cond(t)} r_{\mathbf{f},t} - \sum_{\mathbf{f}\in\mathscr{A}_t\cap cond(t)^c}(\alpha - r_{\mathbf{f},t})\mathbb{P}\left(\hat{\sigma}^t(\mathscr{A}_t)=\mathbf{f}\middle|\mathscr{H}_t\right)\\
&\geq \mathbb{E}\left[\hat{r}_{\hat{\sigma}^t(\mathscr{A}_t),t}\middle|\mathscr{H}_t, I_t=0\right] - (1-\beta)c_0|\mathscr{A}_t| - \alpha\beta|\mathscr{A}_t|\\
&\geq \mathbb{E}\left[\hat{r}_{\hat{\sigma}^t(\mathscr{A}_t),t}\middle|\mathscr{H}_t, I_t=0\right] - 2\alpha\beta|\mathscr{A}_t|,
\end{aligned}
$$

where $cond(t)^c$ denotes the complement of set $cond(t)$; the first inequality above is due to the assumption that for all $\mathbf{f}\in\mathscr{A}_t\cap cond(t)$, we have $\mathbb{P}\left(\hat{\sigma}^t(\mathscr{A}_t)=\mathbf{f}\middle|\mathscr{H}_t\right)\geq\beta$. For $t=1$, the above inequality is trivial since $\hat{r}_{\hat{\sigma}^1(\mathscr{U}(\hat{\mathbf{f}}_0)),1}\equiv 0$ by its definition. Hence, for $t\geq 1$, one has

$$
\begin{aligned}
\mathbb{E}\left[r_{\hat{\mathbf{f}}_t,t}\middle|\mathscr{H}_t\right] &= \epsilon\mathbb{E}\left[r_{\hat{\sigma}^t(\mathscr{F}_p),t}\middle|\mathscr{H}_t, I_t=1\right] + (1-\epsilon)\mathbb{E}\left[r_{\hat{\sigma}^t(\mathscr{A}_t),t}\middle|\mathscr{H}_t, I_t=0\right]\\
&\geq \mathbb{E}\left[\hat{r}_{\hat{\mathbf{f}}_t,t}\middle|\mathscr{H}_t\right] - 2\alpha\beta(1-\epsilon)|\mathscr{A}_t|. \qquad (3.15)
\end{aligned}
$$

Summing on both side of inequality (3.15) over t terminates the proof of Lemma 3.4. □

Lemma 3.5. *Let* $\hat{c}_0=\frac{c_0}{\beta}+\alpha$. *If* $0<\eta_1=\eta_2=\cdots=\eta_T=\eta<\frac{1}{\hat{c}_0}$, *then we have*

$$
\mathbb{E}\left[\max_{\hat{\sigma}}\left\{\sum_{t=1}^{T}\hat{r}_{\hat{\sigma}(\mathscr{A}_t),t}-\frac{1}{\eta}h\left(\hat{\sigma}(\mathscr{A}_t)\right)\right\}\right]-\sum_{t=1}^{T}\mathbb{E}\left[\hat{r}_{\hat{\sigma}^t(\mathscr{A}_t),t}\right]\leq\hat{c}_0^2(\mathrm{e}-1)\eta T+\hat{c}_0(\mathrm{e}-1)(c_1 p+c_2 L+c_3).
$$

Proof. By the definition of $\hat{r}_{\mathbf{f},t}$ in Algorithm 3.3, for any $\mathbf{f}\in\mathscr{F}_p$ and $t\geq 1$, we have uniformly

$$
\hat{r}_{\mathbf{f},t}\leq\max\left\{\frac{r_{\mathbf{f},t}}{\mathbb{P}\left(\hat{\mathbf{f}}_t=\mathbf{f}\middle|\mathscr{H}_t\right)},\alpha,r_{\mathbf{f},t}\right\}\leq\max\left\{\frac{c_0}{\beta},\alpha\right\}\leq\hat{c}_0,
$$

where in the second inequality we use that $r_{\mathbf{f},t}\leq c_0$ for all $\mathbf{f}$ and t, and that $\mathbb{P}\left(\hat{\mathbf{f}}_t=\mathbf{f}\middle|\mathscr{H}_t\right)\geq\beta$ when $\mathbf{f}\in\mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right)\cap cond(t)$. The rest of the proof is similar to that of Lemma 3.1 and Lemma 3.2. In fact, if we define by $\hat{\Delta}(\mathbf{f},x_t)=\hat{c}_0-\hat{r}_{\mathbf{f},t}$, then one can easily observe the following relation when $I_t=1$ (similar relation in the case that $I_t=0$)

$$
\begin{aligned}
\hat{\mathbf{f}}_t=\hat{\sigma}^t\left(\mathscr{F}_p\right) &= \arg\max_{\mathbf{f}\in\mathscr{F}_p}\left\{\sum_{s=1}^{t-1}\hat{r}_{\mathbf{f},s}+\frac{1}{\eta}(z_{\mathbf{f}}-h(\mathbf{f}))\right\}\\
&= \arg\min_{\mathbf{f}\in\mathscr{F}_p}\left\{\sum_{s=1}^{t-1}\hat{\Delta}(\mathbf{f},x_s)+\frac{1}{\eta}(h(\mathbf{f})-z_{\mathbf{f}})\right\}.
\end{aligned}
$$

Then applying Lemma 3.1 and Lemma 3.2 on this newly defined sequence $\hat{\Delta}\left(\hat{\mathbf{f}}_t,x_t\right),t=1,\ldots T$ leads to the result of Lemma 3.5.

□

The proof of upcoming Lemma 3.6 requires the following submartingale inequality: let $Y_0,\dots Y_T$ be a sequence of random variable adapted to random events $\mathscr{H}_0,\dots,\mathscr{H}_T$ such that for $1\le t\le T$, the following three conditions hold

$$\mathbb{E}[Y_t|H_t]\le 0,\quad \mathrm{Var}(Y_t|H_t)\le a^2,\quad Y_t-\mathbb{E}[Y_t|H_t]\le b.$$

Then for any $\lambda>0$,

$$\mathbb{P}\left(\sum_{t=1}^{T}Y_t>Y_0+\lambda\right)\le\exp\left(-\frac{\lambda^2}{2T(a^2+b^2)}\right).$$

The proof can be found in Theorem 7.3 of Chung and Lu (2006).

Lemma 3.6. *Assume that* $0<\beta<\frac{1}{|\mathscr{F}_p|},\alpha\ge\frac{c_0}{\beta}$ *and* $\eta>0$*, then we have*

$$\begin{aligned}&\mathbb{E}\left[\max_{\sigma}\left\{\sum_{t=1}^{T}r_{\sigma(\mathscr{A}_t),t}-\frac{1}{\eta}h(\sigma(\mathscr{A}_t))\right\}\right]-\mathbb{E}\left[\max_{\hat\sigma}\left\{\sum_{t=1}^{T}\hat r_{\hat\sigma(\mathscr{A}_t),t}-\frac{1}{\eta}h(\hat\sigma(\mathscr{A}_t))\right\}\right]\\&\qquad\le\left(1-|\mathscr{F}_p|\beta\right)\sqrt{2T\left[\frac{c_0^2}{\beta}+\alpha^2(1-\beta)+(c_0+2\alpha)^2\right]\ln\left(\frac{1}{\beta}\right)}+|\mathscr{F}_p|\beta c_0T.\end{aligned}$$

Proof. First, we have almost surely that

$$\max_{\sigma}\left\{\sum_{t=1}^{T}r_{\sigma(\mathscr{A}_t),t}-\frac{1}{\eta}h(\sigma(\mathscr{A}_t))\right\}-\max_{\hat\sigma}\left\{\sum_{t=1}^{T}\hat r_{\hat\sigma(\mathscr{A}_t),t}-\frac{1}{\eta}h(\hat\sigma(\mathscr{A}_t))\right\}\le\max_{\mathbf{f}\in\mathscr{F}_p}\sum_{t=1}^{T}\left(r_{\mathbf{f},t}-\hat r_{\mathbf{f},t}\right).$$

Denote by $Y_{\mathbf{f},t}=r_{\mathbf{f},t}-\hat r_{\mathbf{f},t}$. Since

$$\mathbb{E}\left[\hat r_{\mathbf{f},t}\middle|\mathscr{H}_t\right]=\begin{cases}r_{\mathbf{f},t}+(1-\epsilon)\alpha\left(1-\mathbb{P}\left(\hat{\mathbf{f}}_t=\mathbf{f}|\mathscr{H}_t\right)\right) & \text{if}\quad \mathbf{f}\in\mathscr{U}(\hat{\mathbf{f}}_{t-1})\cap cond(t),\\ \epsilon r_{\mathbf{f},t}+(1-\epsilon)\alpha & \text{otherwise,}\end{cases}$$

and $\alpha>c_0\ge r_{\mathbf{f},t}$ uniformly for any $\mathbf{f}$ and t, then we have uniformly that $\mathbb{E}[Y_t|\mathscr{H}_t]\le 0$, hence satisfying the first condition.

For the second condition, if $\mathbf{f}\in\mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right)\cap cond(t)$, then

$$\begin{aligned}\mathrm{Var}(Y_t|\mathscr{H}_t)=&\mathbb{E}\left[\hat r_{\mathbf{f},t}^2|\mathscr{H}_t\right]-\left(\mathbb{E}\left[\hat r_{\mathbf{f},t}|\mathscr{H}_t\right]\right)^2\\\le&\epsilon r_{\mathbf{f},t}^2+(1-\epsilon)\left[\frac{r_{\mathbf{f},t}^2}{\mathbb{P}\left(\hat{\mathbf{f}}_t=\mathbf{f}|\mathscr{H}_t\right)}+\alpha\left(1-\mathbb{P}\left(\hat{\mathbf{f}}_t=\mathbf{f}|\mathscr{H}_t\right)\right)\right]\\&-\left[r_{\mathbf{f},t}+(1-\epsilon)\alpha\left(1-\mathbb{P}\left(\hat{\mathbf{f}}_t=\mathbf{f}|\mathscr{H}_t\right)\right)\right]^2\\\le&\frac{r_{\mathbf{f},t}^2}{\beta}+\alpha^2(1-\beta)\le\frac{c_0^2}{\beta}+\alpha^2(1-\beta).\end{aligned}$$

Similarly, for $\mathbf{f}\notin\mathscr{U}\left(\hat{\mathbf{f}}_{t-1}\right)\cap cond(t)$, one can have $\mathrm{Var}(Y_t|\mathscr{H}_t)\le\alpha^2$.

Moreover, for the third condition, since

$$\mathbb{E}\left[Y_{\mathbf{f},t}|\mathscr{H}_t\right]\ge-2\alpha,$$

then

$$Y_{\mathbf{f},t}-\mathbb{E}\left[Y_{\mathbf{f},t}|\mathscr{H}_t\right]\le r_{\mathbf{f},t}+2\alpha\le c_0+2\alpha.$$

Setting $\lambda = \sqrt{2T\left[\frac{c_0^2}{\beta} + \alpha^2(1-\beta) + (c_0+2\alpha)^2\right]\ln\left(\frac{1}{\beta}\right)}$ leads to

$$\mathbb{P}\left(\sum_{t=1}^{T} Y_{\mathbf{f},t} \geq \lambda\right) \leq \beta.$$

Hence the following inequality holds with probability $1 - \left|\mathscr{F}_p\right|\beta$

$$\max_{\mathbf{f}\in\mathscr{F}_p} \sum_{t=1}^{T} \left(r_{\mathbf{f},t} - \hat{r}_{\mathbf{f},t}\right) \leq \sqrt{2T\left[\frac{c_0^2}{\beta} + \alpha^2(1-\beta) + (c_0+2\alpha)^2\right]\ln\left(\frac{1}{\beta}\right)}.$$

Finally, noticing that $\max_{\mathbf{f}\in\mathscr{F}_p} \sum_{t=1}^{T} \left(r_{\mathbf{f},t} - \hat{r}_{\mathbf{f},t}\right) \leq c_0 T$ almost surely, we terminate the proof of Lemma 3.6.

□

4

Text mining, neural networks and chatbot

This chapter of manuscript summarizes the work that has been done within iAdvize. It mainly concerns with text mining and Natural Language Processing (NLP) and can be divided into two parts. The first part is about the sentiment analysis (positive and negative) of French tweets which is essentially a binary classification problem. We introduce at first several important procedures in NLP that are premise for classification methods. Then we show that our tool for identifying the sentiment of tweets outperforms an existing sentiment classifier in `pattern` library of python. The second part concerns with the creation of a conversational agent (known as chatbot) which, given a question, can choose the best response from a set of prefixed responses. We begin with an introduction of deep learning and neural networks including its basic prototype and learning methods. Then we proceed to the recurrent neural network (RNN) and several derivatives such as Long Short Term Memory (LSTM) and Sequence to Sequence (seq2seq) which is the-state-of-art model for building such a chatbot. Finally, we give an example showing the performance of this chatbot.

Contents

4.1 Text mining

4.1.1 Introduction

Text mining, also referred to as text data mining, roughly equivalent to text analytics, is the process of deriving high-quality information from text. High-quality information is typically derived through the devising of patterns and trends through means such as statistical pattern learning. Text mining usually involves the process of structuring the input text (usually parsing, along with the addition of some derived linguistic features and the removal of others, and subsequent insertion into a database), deriving patterns within the structured data, and finally evaluation and interpretation of the output. "High quality" in text mining usually refers to some combination of relevance, novelty, and interestingness. Typical text mining tasks include text categorization, text clustering, concept/entity extraction, sentiment analysis, information retrieval and lexical analysis to study word frequency distributions. It also includes data mining techniques such as link and association analysis, visualization, and predictive analytics. The overarching goal is, essentially, to turn text into data for analysis, via application of Natural Language Processing (NLP) and analytical methods.

Text mining fosters strong connections with Natural Language Processing (NLP), data mining and machine learning. It seeks to extract useful information from unstructured textual data through the identification and exploration of interesting patterns (*i.e.,* sentiment, trends etc hidden in messages). By the exploration of these patterns, messages of certain sentiment, or taking about the same trend, can be sent to proper consultants having the competence to answer them. In addition, administrators or consultants can detect in real time a sudden increase of volume of certain trend.

Pre-processing of textual messages

Textual messages appeared on the internet are often not as formal as that in the book and article. They may contain noises such as poor grammar, spelling and mechanics, abbreviations and Emoji (*e.g.,* your the best!, thx, bjr); They could convey a familiarity that is likely to be jarring or offensive in many situations (*e.g.,* thanks hon!). Pre-processing of textual messages is therefore crucial in text mining since it can help to reduce noises of messages and form a clean textual structure of messages. For example, pre-processing can change abbreviations (thx, bjr) to complete forms (thanks, bonjour); it can also normalize poor spellings in French such as "general" or "géneral" to correct spelling "général". To our knowledge, although there is no standard procedures for pre-processing of texts since they vary with different text mining tasks, regular expression and tokenization proves to be two important global procedures used in many cases. Regular expression (*i.e.,* regex) is a sequence of characters that defines a search pattern. Taking the regex **a.** as an example, **a** is a literal character which matches just letter "a" and . is a meta character which matches every character except a newline. As a consequence, this regex would match for example "a" or "ax" or "a0". Regex is therefore important in pre-processing since any irregular pattern inside messages can be found and replaced by a regular pattern.

Another important procedure, known as tokenization, is the process of breaking a stream of text into words, phrases, symbols, or other meaningful elements called tokens. The tokens then become the most elementary inputs for further processing such as parsing or word representation. A message such as "The weather is good !" would be tokenized into a list of single words and an exclamation: **[The, weather, is, good, !]** if the space were chosen

as the token delimiter. According to different text mining tasks, other pre-processing techniques may include removal of stop words, stemming etc (see Vijayarani, 2015 for more details).

Word representation

In text mining, word representation aims at mapping words from vocabulary to vectors of real numbers such that they can be used later to fulfill tasks such as sentiment analysis, subject classification etc. The most intuitive and simplest way to represent a word from vocabulary is known as the one-hot representation which maps each word of vocabulary by a vector of length equaling to the size of vocabulary. All the components of this vector are 0 except for one component whose value is 1. For example, the words "The" and "weather" are represented respectively by one-hot vectors with different location of component 1.

$$\begin{aligned}&\text{The}:(0,0,0,1,0,0,\ldots,0),\\&\text{weather}:(0,0,\ldots,0,1,0,\ldots,0).\end{aligned}$$

One-hot representation can be easily extended to sentence level: a sentence is represented by a vector whose $i-$th component is a boolean expressing the occurrence or absence of the $i-$th word from the vocabulary. One-hot representation is useful in practice for its simple implement and it can fulfill some text mining tasks such as sentiment analysis with the help of machine learning algorithms (*e.g.,* Support Vector Machine (SVM), Random Forrest (RF) etc). However, it still has at least two disadvantages: the first one is that the vector length of a word or sentence increases with the vocabulary size and most of the components of this vector are 0, *i.e.,* the vector is sparse; the second one is that it cannot express relevant information of words on the semantic level. For example, the word "Paris" and "Beijing" are semantically similar in the following sentences: "Paris is the capital of France" and "Beijing is the capital of China". If they are represented by one-hot vector, we cannot identify this similarity of two words.

Distributed representation, proposed firstly by Harris (1954), aims at quantifying and categorizing semantic similarities between linguistic items based on their distributional properties in large samples of language data. The basic idea underlying it is that words used and occurred in the same contexts tend to have similar meanings. Later, this idea was extended by Firth (1957) that "a word is characterized by the company it keeps". More precisely, distributed representation maps words to vectors of comparatively lower dimensions. The components of vectors are no longer integers but real values. For example, the word "weather" may be represented by a vector of form $(0.792, 0.177, 0.107, 0.109, 0.542, \ldots)$. This way of word representation enables us to project words to a dense vector space such that words with similar semantic meaning are closer in distance. The distributional representation of words is on the basis of statistical language model which characterize sequences of words by probability distributions. More precisely, given a sequence $\boldsymbol{S}$ of m words $x_1, x_2, \ldots, x_m$, the statistical language model estimates the probability $\mathbb{P}(x_1, \ldots, x_n)$ over the whole sequence $\boldsymbol{S}$ to measure its confidence in accordance with grammar and semantic rules of natural language. To calculate this probability, one has the chain rule:

$$\mathbb{P}(x_1, \ldots, x_m) = \mathbb{P}(x_m | x_{m-1}, \ldots, x_1) \times \mathbb{P}(x_{m-1} | x_{m-2}, \ldots, x_1) \times \cdots \times \mathbb{P}(x_2 | x_1) \times \mathbb{P}(x_1),$$

where $\mathbb{P}(x_i | x_{i-1}, x_{i-2}, \ldots, x_1)$ is the conditional probability of $i-$th word given all past words. The probability $\mathbb{P}(x_i | x_{i-1}, \ldots, x_1)$ can be modeled by n-gram model: it can be

approximated by $\mathbb{P}(x_i|x_{i-1},x_{i-2},\ldots,x_{i-n+1})$, the probability of the $i-$th word conditioned on a shorten historical context of previous $n-1$ words (assume that $n \leq i$). The n-gram conditional probability can then be calculated from frequency counts

$$\mathbb{P}(x_i|x_{i-1},x_{i-2},\ldots,x_{i-n+1}) \approx \frac{\text{count}\{x_i,x_{i-1},\ldots,x_{i-n+1}\}}{\text{count}\{x_{i-1},\ldots,x_{i-n+1}\}}.$$

on the basis of a large corpus of data. If a certain n-gram is not seen before (*i.e.,* frequency count is 0), smoothing methods such as add-one estimate can be used to avoid zero probability. Still, n-gram model cannot capture long range dependencies. Simply choosing a big n for the purpose of covering long range dependencies would deteriorate the quality of estimation since many zero frequency counts were likely to appear.

Neural language model, however, can take into account long range dependencies. It uses Neural Networks to model the word as well as its relation with context. It represents words as non-linear combinations of weights in a neural net. The dimensions of weights are often prefixed to a smaller number (50 to 200 in practice) with respect to the vocabulary size. Hence, neural language model avoids the sparsity of vector representation that happens with the augmentation of vocabulary size. To our knowledge, neural language model is firstly investigated by Bengio *et al.* (2003). Later, Mikolov *et al.* (2013) propose two-layer neural networks (word2vec) to reconstruct linguistic contexts of words. In general, word2vecl model is trained and learns to predict the probability of a word x_i in vocabulary given a linguistic context C, *i.e.,* $\mathbb{P}(x_i|C)$. This context C might be a fixed-size window of k previous words (*i.e.,* $C = (x_{i-1},x_{i-2},\ldots,x_{i-k})$) or of both k previous and future words (*i.e.,* $C = (x_{i-k},x_{i-k+1},\ldots,x_{i-1},x_{i+1},x_{i+1},\ldots,x_{i+k})$. This one is known as *continuous bag-of-words* in word2vec. Another one, known as *skip-gram*, inverses the previous problem by learning to predict the probability of a context given a word. After training high dimensional word vectors on a large amount of data, the resulting vectors can be used to answer very subtle semantic relationships between words, such as a city and the country it belongs to, *i.e.,* France is to Paris as China is to Beijing. The word2vec has been subsequently explained and analysed recently (Goldberg and Levy, 2014, Rong, 2014). Under the same idea, Le and Mikolov (2014) extends the neural representation to sentence level (*i.e.,* seq2vec). In what follows, both one-hot representation and seq2vec will be used in the sentiment analysis to be discussed in the next section.

4.1.2 Sentiment analysis

Sentiment analysis is to determine the overall contextual polarity (positive or negative) of a message. It is widely applied to reviews or comments on the internet for a variety of applications, ranging from marketing to custom services. As iAdvize has integrated social media messages such as tweets, facebook comments and feedback etc), knowing the sentiment hidden behind them is important since it reflects customer's altitude towards some topics or products. The goal for us is therefore to create a tool that is able to evaluate the sentiment of messages and identify those with negative emotion such that consultants are able to provide instant and proper services to customers. Otherwise, our clients may suffer from the lose of customers.

From the machine learning point of view, sentiment analysis is a supervised learning task, *i.e.,* to be able to estimate the sentiment label of a message, one needs a set of samples containing not only content of messages but also polarity labels. We use web crawling technique as well as Twitter API to obtain a set of samples $(x_i,y_i), i = 1,2,\ldots,n$, where x_i

denotes a message and y_i sentiment label (positive or negative). We show below several examples of samples:

— ("appartement extrêmement bruyant en raison d'une voie de circulation fréquentée. Impossible de dormir. quasi...", "neg")
— ("Je ne recommanderai ...", "neg")
— ("Bon accueil, personnel à l'ecoute. Tout à fait satisfa...", "pos")
— (" C'est la première fois que je commandais sur votre site. J'ai apprécié la livraison en ce...", "pos")

We pre-process raw messages by removing stop words in French such as "au", "avec", "du" and replacing particular sequence of characters (url, number etc) with special tokens ("URL", "digital_number" etc). Notice that the stop words as well as particular sequences contain little polarity information. In addition, since messages as tweets may contain poorly spelling words ("merciiiii", "lololo" etc), we use regular expression to normalize them. These steps of pre-processing reduces both the noise of messages and the vocabulary size. Sentiment analysis is essentially a binary classification, many machine learning classifiers such as Bernoulli Naive Bayesian (BNB), Support Vector Machine (SVM) and Random Forest (RF) can serve to this task. Further, the observations are divided into two parts: a training part for training the classifiers and a test part for measuring the prediction accuracy of classifiers. The proportion of training and test is **80%** and **20%** and the accuracy is computed as $\sum_{j=1}^{n_2} \mathbb{1}_{\{\hat{y}_j \neq y_j\}}/n_2$, where n_2 is the size of test set. Optimal values for hyperparameters such as Laplace smoothing parameter for BNB (*resp.* penalty parameter and kernel for SVM; number of trees, maximum depth of tree and criterion for measuring the quality of a split for RF) are chosen as follows: we set firstly a parameter grid and then use a module called `RandomizedSearchCV` in `sklearn` library with 2-Fold cross validation to obtain the optimal combination of hyperparameters values. The following Table 4.1 compares the accuracy of three classifiers (SVM, NB and RF) and an existing module called `pattern` in python, based on two word representations (one-hot representation on sentence level and seq2vec). Note that BNB classifier is not compatible with continuous input vectors obtained by seq2vec since it only allows input vectors whose components are of value 0 or 1.

	one-hot	seq2vec	`pattern`
Prediction accuracy	RF = 0.641 SVM = 0.578 BNB = 0.664	RF = 0.567 SVM = 0.558	0.603

Table 4.1 – Prediction accuracy of sentiment analysis for French tweets.

It is noticed that both RF and BNB classifier with one-hot representation outperform the `pattern` classifier in sentiment analysis for French tweets. The reasons might be that our training sample is more tweet-orientated and the classifiers we use are more robust than the one used in `pattern`. In addition, since sentiment of messages depends largely (except for irony) on the occurrence of positive or negative words appearing in the messages, one-hot representation on the sentence level proves to be a direct and an efficient consequence of this characteristic.

4.2 Neural networks and Deep learning

With the development of large scale computing ability, deep learning based on neural networks has shown its potential and powerful capability in past decade in many fields such as speech and pattern recognition, Natural language processing and automatic driving. In what follows, we only focus on deep learning in NLP field and, more concretely, in building a conversational agent (*i.e.,* chatbot) which is able to respond automatically to human messages. The purpose of building such a chatbot is to aid the consultants of our clients since it is founded that a large amount of messages disposed by them are either repeated or semantic-similar. We begin this section with the introduction of neural network including its basic architecture and learning methods. Then we proceed to recurrent neural network (RNN) which takes into account the order of input sequences. We first describe its prototype and then present several particular models such as Long Short-Term Memory (LSTM) and Sequence to Sequence (seq2seq) which are tailored for chatbot building.

4.2.1 Introduction

Neural networks (NNs) are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance) to do tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the analytic results to identify cats in other images. They have found most use in applications difficult to express in a traditional computer algorithm using rule-based programming.

Investigations of neural network can be dated back to McCulloch and Pitts (1943) who created a computational model for neural networks based on mathematics and algorithms called threshold logic. Later Rosenblatt (1958) proposed an artificial neuron called a perceptron which is initially used in binary classification. Today, it's more common to use other models of artificial neurons, in this section, and in much modern work on neural networks, the main neuron model used is the one called the sigmoid neuron.

4.2.2 Architecture of neural networks

The most basic components of neural network are sigmoid neuron units. A sigmoid neuron unit has input $\mathbf{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$, weight $\mathbf{w} = (w_1, w_2, \ldots, w_d) \in \mathbb{R}^d$ for each inputs and a bias $b \in \mathbb{R}$. The output of it is $\sigma(\mathbf{wx} + b)$, where

$$\sigma(z) = \frac{1}{1 + \mathrm{e}^{-z}}, \quad z \in \mathbb{R} \tag{4.1}$$

is the sigmoid function and $\mathbf{wx}$ denotes the inner product between vector $\mathbf{w}$ and $\mathbf{x}$. The sigmoid neuron unit first applies a linear transformation of input and then projects this transformation to an image space $(0, 1)$. It can be regarded as an extension of perceptron which outputs only 0 or 1. In neural network, sigmoid function σ is just one possible candidate for neuron unit, we list here other commonly used candidates

$$\text{rectifier}: \mathbf{Relu}(z) = \mathbf{max}(0, z), \qquad \text{Hypobolic tangent}: \mathbf{tanh}(z) = \frac{1 - \mathrm{e}^{-2z}}{1 + \mathrm{e}^{-2z}} \quad z \in \mathbb{R}. \tag{4.2}$$

Neural network consists of three parts: an input layer, several hidden layers and an output layer. Figure 4.1 illustrates the architecture of a simple neural network two hidden layers. Note that a neural network can be made more complicated by embedding a larger number of hidden layers as well as more neuron units within each hidden layer. For illustration purpose, we only consider a simple neural network here. At input layer, each circle represents a component x_i of input $\mathbf{x}$, $i = 1,2,\ldots,d$ while, at hidden layers, each circle represents a neuron unit. The first hidden layer neurons take inputs from input layer and output values that will be treated as inputs for neurons within the next hidden layer. The output of second hidden layer again will be used as input for output layer neurons. In classification setting, the output of output layer would be a probability distribution over labels. For instance, in handwriting digits identification (Figure 4.2) where one prefers to predict a number digit from image pixels inputs by neural networks, the output would be a probability distribution indicating the probability of each digital label $\mathbf{0,1,2,\ldots,9}$, and the predicted digit is the one with highest probability.

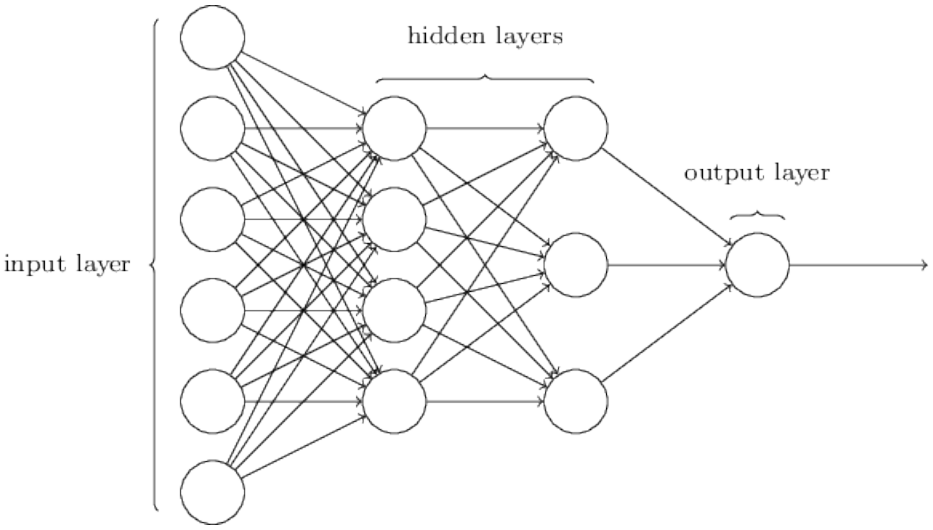


Figure 4.1 – Architecture of Neural network, source [1]

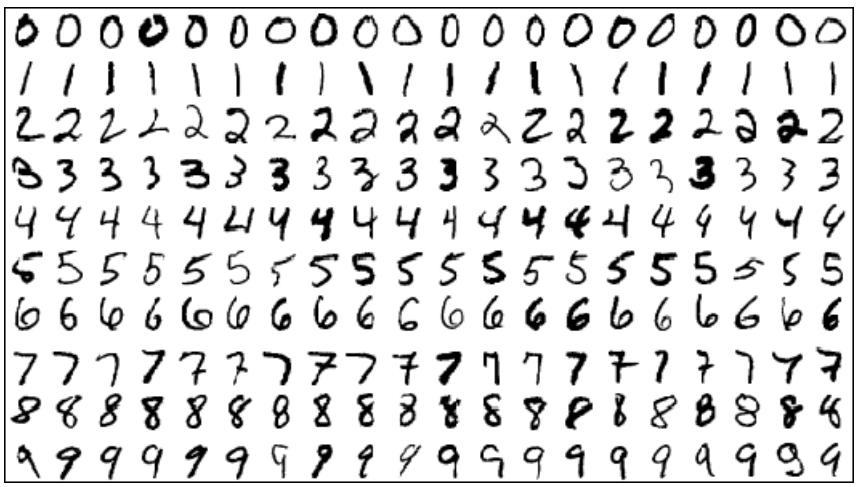


Figure 4.2 – A handwriting example from MNIST database

1. http://www.shivambansal.com/blog/neural_network_1/

More formally, suppose that the first hidden layer contains m ($m \in \mathbb{N}^*$) neuron units and the second hidden layer l ($l \in \mathbb{N}^*$) neuron units. In addition, we dispose n pairs of observations $(\mathbf{x}_s, y_s), s = 1, \ldots, n$ where $\mathbf{x}_s \in \mathbb{R}^d$ is an input for neural network and y_s is the label of $\mathbf{x}_s$ belonging to a set of labels $\{1, 2, \ldots, k\}$, $k \in \mathbb{N}^*$. For each $\mathbf{x}_s$, output $\mathbf{z}_s \in \mathbb{R}^m$ of first hidden layer is

$$\mathbf{z}_s = \sigma\left(W_{I,1}\mathbf{x}_s + \mathbf{b}_{I,1}\right),$$

where $W_{I,1} \in \mathbb{R}^{m \times d}$ is an input weight parameter and $\mathbf{b}_{I,1} \in \mathbb{R}^m$ is a bias parameter associated with first hidden layer. Similarly, output $\mathbf{h}_s \in \mathbb{R}^l$ of second hidden layer is

$$\mathbf{h}_s = \sigma\left(W_{I,2}\mathbf{z}_s + \mathbf{b}_{I,2}\right),$$

where $W_{I,2} \in \mathbb{R}^{l \times m}$ is a weight parameter and $\mathbf{b}_{I,2} \in \mathbb{R}^l$ is a bias parameter associated with second hidden layer. Finally, the output layer first applies a sigmoid function σ and a linear transformation on $\mathbf{h}_s$ to get a intermediate vector $\mathbf{o}_s$, and then transfers it to a probability distribution $\delta(\mathbf{o}_s)$ over labels via the softmax function δ, *i.e.,*

$$\begin{cases} \mathbf{o}_s = \sigma(W_o\mathbf{h}_s + \mathbf{b}_o) \\ \delta(\mathbf{o}_s)_j = \frac{e^{o_{s,j}}}{\sum_{i=1,\ldots,k} e^{o_{s,i}}}, \quad j = 1, \ldots, k, \end{cases}$$

where $W_o \in \mathbb{R}^{k \times l}$ is an output weight parameter, $\mathbf{b}_o \in \mathbb{R}^k$ is an output bias parameter and $\delta(\mathbf{o}_s)$ is a softmax function operating on a k-dimensional vector $\mathbf{o}_s = (o_{s,1}, \ldots, o_{s,k})$. The prediction $\hat{y}_s$ of y_s is

$$\hat{y}_s = \underset{j=1,2,\ldots k}{\arg\max}\left(\delta(\mathbf{o}_s)_j\right). \tag{4.3}$$

Here the prediction $\hat{y}_s$ of y_s is therefore the most probable digit given $\mathbf{x}_s$. One should notice that definition for $\hat{y}_t$ is not unique in the context of neural network, sometimes it can also be defined as a probability distribution over all labels.

4.2.3 Gradient based learning

Given the observations $(\mathbf{x}_s, y_s), s = 1, \ldots, n$ and a designed neural network, the network needs to learn from observations the optimal values of parameters such that predictions are approximating true labels. To quantify the quality of neural network, one needs a cost function

$$C(\theta) = \frac{1}{n}\sum_{s=1,\ldots,n} \ell(\hat{y}_s, y_s),$$

where θ denotes the collection of all weights and all biases in the network (we suppose θ belongs to a parameter space Θ hereafter) and ℓ is a loss function whose definition can be the same as that in Section 1.1.1. For example, if $\hat{y}_s$ is defined as (4.3), then ℓ can be the squared loss; if $\hat{y}_s = (\hat{y}_{1,s}, \ldots, \hat{y}_{k,s})$, a simplex probability distribution over all values $\{1, 2, \ldots, k\}$ of label, where $\hat{y}_{j,s} \in [0, 1]$ corresponds to the probability of label j given the input $\mathbf{x}_s$, then ℓ can be the perplexity of a probability model defined by

$$\ell(\hat{y}_s, y_s) = -\sum_{j=1}^{k} \mathbb{1}_{\{y_s = j\}} \ln \hat{y}_{j,s}. \tag{4.4}$$

Our goal is to find an optimal value of θ minimizing the cost function $C(\theta)$. Gradient descent is a first-order iterative optimization algorithm for finding the minimum of $C(\theta)$

whose steps can be described as follows:

$$\theta^{(n+1)} = \theta^{(n)} - \gamma_n \nabla C\left(\theta^{(n)}\right) \quad n = 1, 2, \ldots, \tag{4.5}$$

where $\theta^{(1)}$ is an initial value for iterations, $\gamma_n > 0$ is known as the learning rate and $\nabla C(\theta)$ is the gradient of $C(\theta)$. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of gradient (or of the approximate gradient) at current value $\theta^{(n)}$. The sequence $\theta^{(n)}, n = 1, \ldots,$ satisfies the following lemma:

Lemma 4.1. *(Goodfellow et al., 2016) If $C(\theta)$ is differentiable on Θ and $\gamma_n > 0$ is small enough for all n, then $C\left(\theta^{(n+1)}\right) \leq C\left(\theta^{(n)}\right)$.*

Proof. By Taylor's expansion,

$$C\left(\theta^{(n+1)}\right) = C\left(\theta^{(n)}\right) - \gamma_n \left\|\nabla C\left(\theta^{(n)}\right)\right\|_2^2 + o(\gamma_n),$$

where $\lim_{\gamma_n \longrightarrow 0} \frac{o(\gamma_n)}{\gamma_n} = 0$. For γ_n small enough, one has $o(\gamma_n) \leq \gamma_n \left\|\nabla C\left(\theta^{(n)}\right)\right\|_2^2$ which terminates the proof. □

To obtain the gradient ∇C (for simplicity, we omit the parameter θ when no confusion will arise), it is required to calculate $\nabla \ell(\hat{y}_s, y_s)$ for each $s = 1, \ldots, n$ and average them. This leads to a challenge in applying directly gradient descent rule (4.5) in practice since the number n of inputs can be very large and learning thus occurs slowly. To conquer it, stochastic gradient descent to be presented in the next section is often used to speed up the learning.

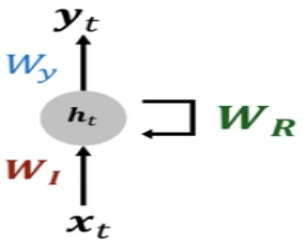
Stochastic gradient descent

The idea of stochastic gradient descent is to estimate the gradient ∇C by computing $\nabla \ell(\hat{y}_s, y_s)$ for only a small sample of randomly chosen training inputs. By averaging over this small sample it turns out that we can quickly get a good estimate of the true gradient ∇C and this helps to speed up gradient descent, and thus learning. More precisely, stochastic gradient descent works by randomly choosing a number of samples from training inputs. Let us denote them by $\{(\mathbf{x}_s, y_s), s \in \mathcal{M}\}$, and refer to them as as mini-batch. Provided that the cardinality $|\mathcal{M}|$ of $\mathcal{M}$ is large enough but comparatively small with respect to n, then we expect that

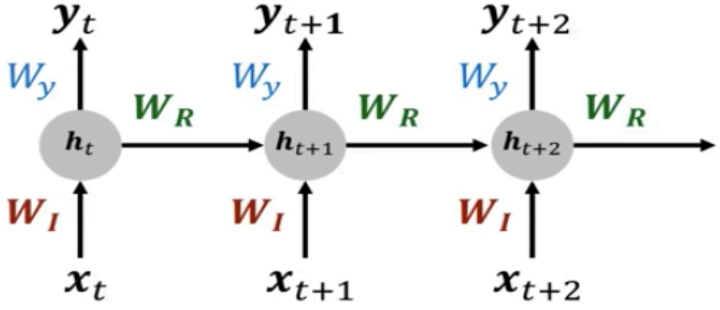
$$\sum_{s \in \mathcal{M}} \frac{\nabla \ell(\hat{y}_s, y_s)}{|\mathcal{M}|} \approx \sum_{s=1}^{n} \frac{\nabla \ell(\hat{y}_s, y_s)}{n} = \nabla C.$$

The convergence of stochastic gradient descent has been analyzed using the theories of convex minimization and stochastic approximation. Generally, when the learning rates γ_n decrease with an appropriate rate, and are subject to relatively mild assumptions, stochastic gradient descent converges almost surely to a global minimum when the objective function is convex or pseudo-convex, and otherwise converges almost surely to a local minimum (see Bottou, 1998 and Kiwiel, 2001 for more details).

The classical neural network does not take into account the order and sequential character of inputs which can be very important in NLP tasks. For instance, the order of words in a sentence decides the meaning of it. In the next section, we introduce recurrent neural network which is created to cope with sequential inputs.



(a) A RNN cell



(b) Unfold of RNNs into full networks

4.3 Recurrent Neural Network

4.3.1 Introduction

Recurrent Neural Networks (RNNs) are popular models that make use of sequential information of inputs and outputs. They have shown great promise in many NLP tasks such as machine translation, speech recognition and conversational agent. They are called recurrent since they perform the same task for each element of sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a "memory" which stores information about what has been calculated so far. More precisely, Figure 4.3a and Figure 4.3b give respectively the structure of a RNN cell and the unrolling of RNN into a full network.

In these two figures, $\mathbf{x}_t$ denotes an input (*e.g*, the $t-$th word of a sentence) at time t and $\hat{y}_t$ the output (*i.e.,* prediction of y_t) of RNN; W_I, W_R and W_y are respectively weight parameters associated to different layers of RNN; $\mathbf{h}_t$ is a hidden state which is calculated based on the previous hidden state $\mathbf{h}_{t-1}$ and the current input x_t. Concretely, one has

$$\begin{aligned} \mathbf{h}_t &= \sigma\left(W_I \mathbf{x}_t + W_R \mathbf{h}_{t-1} + \mathbf{b}_h\right) \stackrel{\Delta}{=} \sigma(\mathbf{s}_t), \\ \hat{y}_t &= \mathbf{softmax}\left(W_y \mathbf{h}_t + \mathbf{b}_y\right) \stackrel{\Delta}{=} \mathbf{softmax}(\mathbf{z}_t), \end{aligned} \tag{4.6}$$

where $\mathbf{b}_h$ and $\mathbf{b}_y$ are bias parameters. We give two intermediate notation $\mathbf{s}_t$ and $\mathbf{z}_t$ simply for the convenience of computing gradients in next section. Moreover, we denote by ℓ_t the loss $\ell(\hat{y}_t, y_t)$ for simplicity and $C(\theta) = \sum_t \ell_t$ with $\theta = \left(W_I, W_R, W_y, \mathbf{b}_h, \mathbf{b}_y\right)$.

Unlike the classical deep neural network which uses different parameters at different layers, RNN shares the same parameters θ across all steps. This reflects the fact that we are

performing the same task at each step, just with different inputs. This greatly reduces the total number of parameters we need to learn. In addition, the prediction $\hat{y}_t$ is a function depending not only on current input $\mathbf{x}_t$ but also on all past inputs $\mathbf{x}_s, s = 1,\ldots,t-1$ in RNN. Moreover, lengths of inputs and outputs inside RNNs can be controlled with different tasks, as illustrated in Figure 4.4.

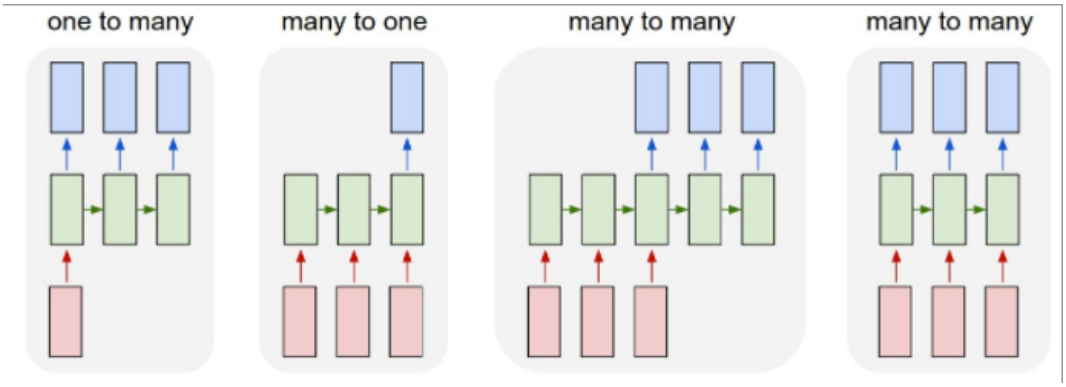


Figure 4.4 – Different RNN models, source [2]

The first plot can be applied to image captioning which takes an image as input and outputs a sentence of words; the second one to sentence analysis which takes a sentence of words as inputs and outputs a sentiment of this sentence; the third one to machine translation (*resp.* conversational agent) which takes a sentence in certain source language (*resp.* question) and outputs a translated one (*resp.* response); The fourth one to video classification where one wishes to label each frame of a video.

Backpropagation Trough Time and vanishing gradients

To learn good values of parameters by Stochastic Gradient Descent, one needs to calculate the gradients of loss $C(\theta)$ with respect to parameters $\theta = (W_I, W_R, W_y, \mathbf{b}_h, \mathbf{b}_y)$. Since $C(\theta)$ is the sum of loss ℓ_t at each time, we consider in the sequel only gradient of ℓ_t with respect to θ. First, the gradient $\partial \ell_t / \partial W_I$ is straight forward. Since

$$\frac{\partial \ell_t}{\partial W_I(i,j)} = \frac{\partial \ell_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial \mathbf{z}_{t,i}} \frac{\partial \mathbf{z}_{t,i}}{\partial W_I(i,j)} = (\hat{y}_{t,i} - y_{t,i})\, \mathbf{h}_{t,j},$$

where $\mathbf{z}_{t,i}$, $\hat{y}_{t,i}$ and $y_{t,i}$ are respectively i-th component of $\mathbf{z}_t, \hat{y}_t$ and y_t; $W_I(i,j)$ is the $(i,j)-$th entry of matrix W_I. We have therefore

$$\frac{\partial \ell_t}{\partial W_I} = (\hat{y}_t - y_t) \otimes \mathbf{h}_t,$$

where $\otimes$ is the outer product of two vectors.

However, the computation of gradient $\partial \ell_t / \partial W_R$ is rather complicated since W_R is attached to all hidden states $\hat{h}_s, s = 1,\ldots,t$. Backpropagation Trough Time (BPTT) is the key algorithm that makes the computation of gradients more tractable. More precisely, noticing by the chain rule that

$$\frac{\partial \ell_t}{\partial W_R(i,j)} = \sum_{k=0}^{t} \mathrm{Tr}\left(\left(\frac{\partial \ell_t}{\partial \mathbf{s}_k} \right)^T \frac{\partial \mathbf{s}_k}{\partial W_R(i,j)} \right)$$

2. http://karpathy.github.io/2015/05/21/rnn-effectiveness/

$$\triangleq \sum_{k=0}^{t} \mathrm{Tr}\left(\left(\delta_k^{(t)}\right)^T \frac{\partial \mathbf{s}_k}{\partial W_R(i,j)}\right),$$

where $\mathbf{Tr}(M)$ denotes the trace of any matrix M and M^T its transpose, and $\delta_k^{(t)} = \partial \ell_t / \partial \mathbf{s}_k$.

The gradient $\partial \ell_t / \partial W_R(i,j)$ is the sum of gradients at each time step since W_R is used in all previous steps. In other words, we need to backpropagate gradients from $k = t$ to $k = 0$. Moreover, since

$$\begin{aligned}\delta_k^{(t)} = \frac{\partial \ell_t}{\partial \mathbf{s}_k} &= \frac{\partial \mathbf{h}_k}{\partial \mathbf{s}_k}\frac{\partial \mathbf{s}_{k+1}}{\partial \mathbf{h}_k}\frac{\partial \ell_t}{\partial \mathbf{s}_{k+1}} \\ &= \mathrm{diag}(\mathbf{1} - \mathbf{h}_k \odot \mathbf{h}_k) W_R^T \delta_{k+1}^{(t)} \\ &= \left(W_R^T \delta_{k+1}^{(t)}\right) \odot (\mathbf{1} - \mathbf{h}_k \odot \mathbf{h}_k) \end{aligned} \tag{4.7}$$

and

$$\begin{aligned}\delta_t^{(t)} = \frac{\partial \mathbf{h}_t}{\partial \mathbf{s}_t}\frac{\partial \mathbf{z}_t}{\partial \mathbf{h}_t}\frac{\partial \ell_t}{\partial \mathbf{z}_t} &= \mathrm{diag}(\mathbf{1} - \mathbf{h}_t \odot \mathbf{h}_t) W_y^T (\hat{y}_t - y_t) \\ &= \left(W_y^T (\hat{y}_t - y_t)\right) \odot (\mathbf{1} - \mathbf{h}_t \odot \mathbf{h}_t), \end{aligned} \tag{4.8}$$

where $\odot$ denotes the element-wise product. It indicates that $\delta_k^{(t)}, k = 0, \ldots, t$ can be computed recursively, *i.e.,* the BPTT algorithm

$$\begin{cases} \delta_t^{(t)} = \left(W_y^T (\hat{y}_t - y_t)\right) \odot (\mathbf{1} - \mathbf{h}_t \odot \mathbf{h}_t), \\ \delta_k^{(t)} = \left(W_R^T \delta_{k+1}^{(t)}\right) \odot (\mathbf{1} - \mathbf{h}_k \odot \mathbf{h}_k), \end{cases} \tag{4.9}$$

and then the gradient $\partial \ell_t / \partial W_R$ can be easily computed as follows

$$\frac{\partial \ell_t}{\partial W_R} = \sum_{k=0}^{t} \delta_k^{(t)} \otimes \mathbf{h}_{k-1},$$

where $\mathbf{h}_{-1} = \mathbf{0}$ and the second equality is due to $\frac{\partial \mathbf{s}_k}{\partial W_R(i,j)}$.

However, the classical RNN presented above has difficulties learning long-range dependencies (interactions between words that are several steps apart) since it suffers from vanishing gradient problem (Hochreiter, 1991, 1998). By (4.7), one has

$$\delta_k^{(t)} = \mathrm{diag}(\mathbf{1} - \mathbf{h}_k \odot \mathbf{h}_k) W_R^T \delta_{k+1}^{(t)} = \left(\prod_{j=k}^{t-1} \mathrm{diag}\left(\mathbf{1} - \mathbf{h}_j \odot \mathbf{h}_j\right) W_R^T\right) \delta_t^{(t)}.$$

If the norm of matrix $\left(\mathbf{1} - \mathbf{h}_j \odot \mathbf{h}_j\right)$ is uniformly upper-bounded by a constant $c_1 < 1$, *i.e.,* $\left\|\mathbf{1} - \mathbf{h}_j \odot \mathbf{h}_j\right\| \leq c_1$ for all $j = 1, t$, and that the norm $\|W_R\|$ of W_R is upper-bounded by a constant $r_2 < 1$, then $\left\|\delta_k^{(t)}\right\|_2 \leq (c_1 c_2)^{t-k} \left\|\delta_t^{(t)}\right\|_2$. It indicates that the gradient are reducing at an exponential speed and eventually vanishing after several time steps. In other words, gradient contributions from "far way " steps become zero and hence information stored in those states has no contribution to the current learning.

A popular solution to vanishing gradient problem is the Long Short-Term Memory (LSTM) model that will be introduced in the next section.

4.3.2 Long short-term memory

LSTM was first proposed by Hochreiter and Schmidhuber (1997) and then improved by Gers *et al.* (2000). It is one of the most widely used models in NLP today. Later, Chung *et al.* (2014) proposes the GRU model, a simplified version of LSTM. Both of these two models are explicitly designed to deal with vanishing gradient problem and efficiently learn long-range dependencies. To understand LSTM, we first illustrate by Figure 4.5 its structure where each rectangular represents a cell at time step t, $\mathbf{x}_t$ (blue circle) the input and $\mathbf{h}_t$ (purple circle) the hidden state.

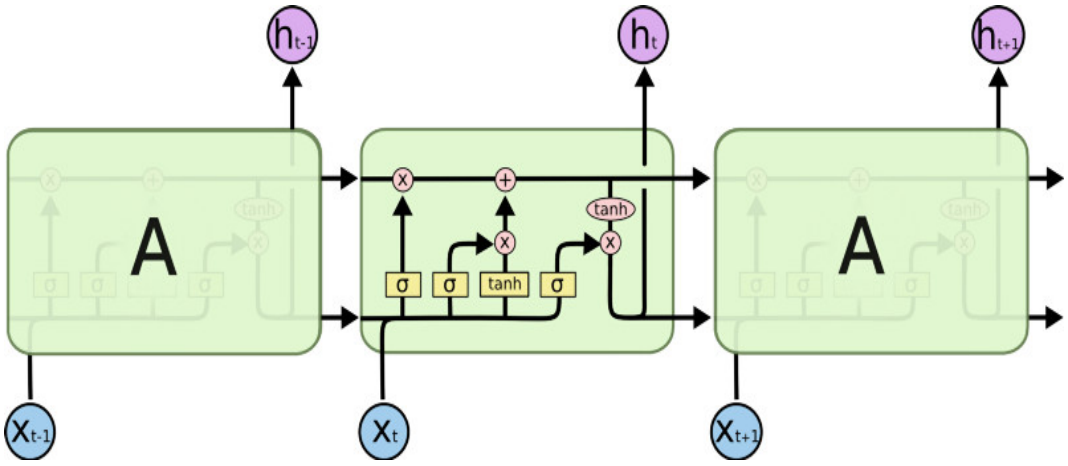


Figure 4.5 – LSTM model, source [3]

It is seen that there are two black lines passing all along between cells of LSTM where the bottom line denotes the transferring of hidden state $\mathbf{h}_t$ between adjacent cells (as same as the what is indicated by purple circle) and the top line denotes the cell state $\mathbf{c}_t$, a new time-varying quantity that does not exist in the classical RNN. The cell state is an internal memory of cell unit that can be updated by combining previous memory with new input. In addition, LSTM differs with classical RNN in the way of computing the hidden state. To understand how hidden states and cell states are computed, we detail below all necessary quantities in LSTM:

$$
\begin{aligned}
\mathbf{f}_t &= \sigma\left(W_f^h \mathbf{h}_{t-1} + W_f^x \mathbf{x}_t + \mathbf{b}_f\right), \\
\mathbf{i}_t &= \sigma\left(W_i^h \mathbf{h}_{t-1} + W_i^x \mathbf{x}_t + \mathbf{b}_i\right), \\
\mathbf{g}_t &= \tanh\left(W_g^h \mathbf{h}_{t-1} + W_g^x \mathbf{x}_t + \mathbf{b}_g\right), \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t, \\
\mathbf{o}_t &= \sigma\left(W_o^h \mathbf{h}_{t-1} + W_o^x \mathbf{x}_t + \mathbf{b}_o\right), \\
\mathbf{h}_t &= \mathbf{o}_t \times \tanh(\mathbf{c}_t),
\end{aligned}
$$

where σ is the sigmoid function in (4.1) applying component-wise on each element of vectors; **tanh** is defined in (4.2), and Ws, $\mathbf{b}$s are LSTM parameters that are independent of t.

The quantities $\mathbf{f}_t$, $\mathbf{i}_t$ and $\mathbf{o}_t$ are respectively known as forget gate, input gate and output gate. Note that they share the exact same equations, but with different parameter matrices. Since all of them are squashed by sigmoid function σ, the value of each component

3. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

of $\mathbf{f}_t$, $\mathbf{i}_t$ and $\mathbf{o}_t$ are between 0 and 1. The forget gate decides the ratio of previous cell state $\mathbf{c}_{t-1}$ to be kept in the current cell state $\mathbf{c}_t$; the input gate controls the ratio of new candidate values $\mathbf{g}_t$ to be added to the current cell state; the output gate defines the extend of internal state to be exposed to the external network. Those gates all have the same dimensions as the hidden states and cell states. Although Figure 4.5 shows an example of LSTM with only one hidden layer, one can consider more complicated LSTM by stacking more hidden layers. In this case, the hidden states of previous hidden layer will play a role of inputs for the next hidden layer.

4.3.3 Sequence to sequence model

Sequence to sequence learning with Neural networks (it is called seq2seq model henceforth) is first proposed by Sutskever *et al.* (2014) and extended by Bahdanau *et al.* (2015) by adding an attention mechanism. It is a model aims at matching two related sequences of different lengths. For example, these two sequences can be a sentence with its translation, an question and a reasonable answer. Seq2seq model uses multi-layered LSTM (or GRU) as cells to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM (or GRU) to decode the target sequence from this vector. Originally used in French to English translation, it has later been applied to the creation of conversational agent (*i.e.,* chatbot) and automated reply of Email (see Vinyal and Le, 2015, Kannan and Kurach, 2016 for more details).

More precisely, Figure 4.6 illustrates the diagram of seq2seq model in its application to conversational agent. It comprises an encoder and a decoder, both of which are composed

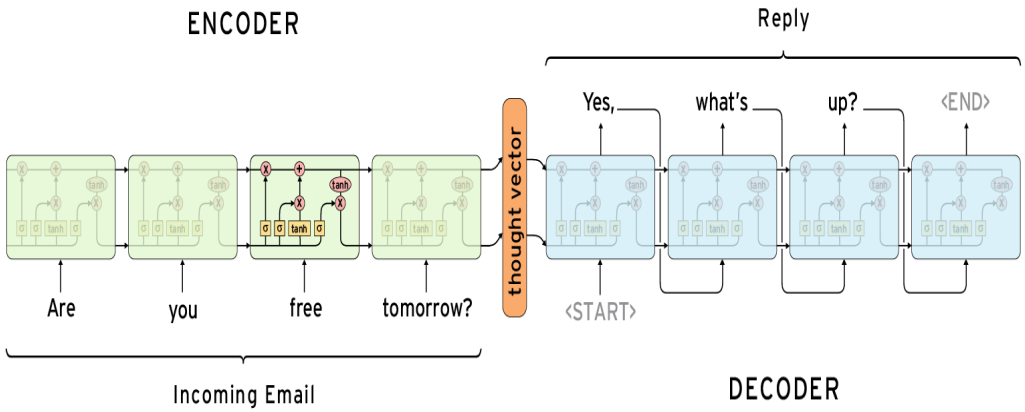


Figure 4.6 – Seq2seq model with encoder and decoder, source [4]

of LSTM cells. In encoder step, a LSTM takes words of input sentence (*e.g.,* a question) as inputs and outputs nothing at each time step. Moreover, the hidden states as well as the cell states are updated along cells and the final cell state (the "thought vector" in Figure 4.6) that have incorporated information of the whole input sentence will be treated as an initial state for the decoder. In decoder step, it begins by receiving a special token "start" and yields a word in vocabulary that has the highest probability, then this word output will be regarded as an input for the next LSTM cell and yields the most probable word. This procedure continues until a special token "end" is yield by the network. The seq2seq model is sometimes referred as a generative model since the final response is generated word by word by the trained model.

4. http://suriyadeepan.github.io/2016-12-31-practical-seq2seq/

In the next section, we will create a chatbot based on the seq2seq model.

4.3.4 A seq2seq-based chatbot

iAdvize company is a conversational platform that integrates different channels (*e.g.,* chat, call, social media etc) for the purpose of connecting customers and consultants with expertise. The connection, which comes into being mostly via conversation, is quite useful for the augmentation of conversion rate on commercial websites since consultants can help their customers solve difficulties that may impede purchase. It is found that a large amount of messages given by customers, especially at the beginning of conversations, are of similar meaning or even repeated sentences. The creation of chatbot, hence, aims at helping consultant by reducing their time on responding to those repeated and simple messages such that they would have more time on treating complicated messages that may have higher value.

A Conversational agent (*i.e.,* chatbot) can interact with customers by having natural conversations indistinguishable from human. The types of chatbot can be very rich depending on different tasks that a chatbot to achieve, different models on which it is created etc. For example, from domain perspective, the chatbot can be generally divided into two categories: closed-domain and open-domain. The formal one tries to achieve a very specific goal such as only responding to questions concerning a certain scenario (*e.g.,* delivery or payment); The latter one aims at handling conversations with open subject. From model perspective, chatbots based on retrieval model will choose a response from a prefixed set of responses given a question. Such kind of chatbot will not give responses with grammar mistakes but may work badly on unseen cases for which no pre-defined responses exist; chatbots based on generative model (*i.e.,* seq2seq model) can generate response from scratch. It has an ability to "remember" past information and to cope with new cases but may suffer from grammar mistakes and the training of such model requires a huge amount of training samples.

As the seq2seq model has great potential in chatbot creation, as shown in Vinyal and Le (2015), we will create our chatbot by this model. However, instead of generating a response by seq2seq model, we prefer to choose a response from a prefixed set of responses since responses with no grammar mistakes are of top priority in practice. To this aim, we first define a score to measure the consistency of a response given a message. Let us denote by $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ a message with n words and by $\mathbf{y} = (y_1, \ldots, y_m)$ a response with m words. The set of prefixed responses is denoted by $\mathscr{Y}$. We define the score of $\mathbf{y}$ given $\mathbf{x}$ by the negative log probability of $\mathbf{y}$ given $\mathbf{x}$, *i.e.,*

$$\text{score}(\mathbf{y}|\mathbf{x}) = -\ln \mathbb{P}_\theta(\mathbf{y}|\mathbf{x}) = -\frac{1}{m}\sum_{j=1}^{m} \ln \mathbb{P}_\theta\left(y_j | x_1, \ldots, x_n, y_1, y_2, \ldots, y_{j-1}\right),$$

where the probability $\mathbb{P}_\theta$ is based on a seq2seq model. Note that a response with higher consistency to a message will have smaller score.

Dataset and pre-processing steps

The raw dataset used for training seq2seq model contains 30694 pairs of question and response extracted from a log of conversations on the delivery subject between customers and consultants. Before training the model, several pre-processing steps are needed to clean up the raw dataset. Generally, each message is tokenized to a sequence of single

words and characters and the most 10000 frequent tokens are chosen as our vocabulary. Tokens not belonging to the vocabulary are replaced by special token "UNK". In addition, we make use of bucketing, which is a method to efficiently handle sentences of different lengths. Let us first clarify the problem. For pairs of observations, we will have question sentences of different lengths $\boldsymbol{L_1}$ on input, and response sentences of different lengths $\boldsymbol{L_2}$ on output. Since the question sentence is passed as encoder inputs in encoder, and the response sentence comes as decoder inputs (prefixed by another special token "BOS", meaning begin of sentence), we should in principle create a seq2seq model for every pair $\boldsymbol{(L_1, L_2+1)}$ of lengths of a question and response. This would result in an enormous computation graph consisting of many very similar sub-graphs. On the other hand, we could just pad every sentence with a special token "PAD". Then we would need only one seq2seq model, for the padded lengths. But for shorter sentence our model would be inefficient, encoding and decoding many "PAD" tokens that are useless. As a compromise between constructing a computation graph for every pair of lengths and padding to a single length, we use a number of buckets and pad each sentence to the length of the bucket above it. We use default buckets, a list of 4 tuples $\boldsymbol{[(5,10),(10,15),(20,25),(40,50)]}$. It means that if the input is a question with 3 tokens, and the corresponding output is a response with 6 tokens, then they will be put in the first bucket and padded to length 5 for encoder inputs, and length 10 for decoder inputs. If the question is with 8 tokens and the corresponding response has 18 tokens, then they will not fit into the $\boldsymbol{(10,15)}$ bucket, but into $\boldsymbol{(20,25)}$ bucket, *i.e.,* the question will be padded to 20, and the corresponding response to 25. We limit maximum length for question and response to 40 and 50 since more than **95%** pairs of observations have lengths below these limits.

Training the model and results

The seq2seq model is trained with following parameter values: the embedding size for each token is set to be 256; the batch size for Stochastic Gradient descent 128; the number of hidden layers 2; the learning rate 0.5. Since running seq2seq model is time consuming, we have considered only batch size and learning rate as hyperparameters to adjust (for others, we have used their default values). We set candidate value 128 and 256 for batch size and 0.01, 0.1, 0.5 for learning rate. In addition, since we did not find a suitable ***Q&A*** benchmark dataset for French that enables us to compute the F1 score, values for the hyperparameters are therefore decided by checking the coherence of answers generated by the model given some questions. The full code which is implemented in python with `Tensorflow` library is on Github: [5]. We test our trained model on several test questions and Table 4.2 shows for each question the top 3 responses with lowest scores (*i.e.,* three most consistent responses) within a set of 12 prefixed responses of 6 categories listed in Table 4.3, where the category of each response is indicated in the parenthesis. It is seen that the first response well corresponds to the first message and its score is lower than that of two other responses which seem to be inconsistent. However, they are both of "délais livraison" category which still have connection with the information expressed by the first message; For the second message whose meaning is quite clear and completely independent with other categories, the first two responses correspond well to the message and their score are comparatively much lower than that of the third response; For the third and fourth message, only the top response is in accordance with the message. The remaining two responses that have higher scores prove to be improper responses.

5. https://github.com/iadvize/data-delivery-chatbot-service

Question	Responses (score)
“quel mode de livraison m’assure de l’avoir rapidement”	1, “il faut choisir le mode livraison chronopost pour le recevoir samedi.” (15.5) 2, “ce produit n’est pas livrable en point relais.” (17.1) 3, “le delais de livraison c’est d’autour de 10 jours pour des articles en stock.” (17.2)
“bonne journée!”	1, “à vous également à bientôt.” (11.9) 2, “bonne journée.” (13.6) 3, “auriez vous la référence de cette table?” (21.3)
“bonjour, je n’arrive pas à créer un compte.”	1, “bonjour, quelle est l’etape qui vous pose problème?” (18.6) 2, “il faut choisir le mode livraison chronopost pour le recevoir samedi.” (21.1) 3, “à vous également à bientôt.” (22.1)
“je souhaite commander une table.”	1, “auriez vous la référence de cette table?” (14.2) 2, “ce produit n’est pas livrable en point relais.” (15.7) 3, “le delais de livraison c’est d’autour de 10 jours pour des articles en stock.” (15.7)

Table 4.2 – Test of chabot on delivery scenario

Responses
1, “bonjour quels types d’articles souhaitez vous commander?” (mode livraison) 2, “il faut choisir le mode livraison chronopost pour le recevoir samedi.” (mode livraison) 3, “tout dépend si le ou les modèle(s) que vous choisissez sont en stock et livrable sous 2 jours.” (délais livraison) 4, “le delais de livraison c’est d’autour de 10 jours pour des articles en stock.” (délais livraison) 5, “un message d’erreur s’affiche t-il?” (créer un compte) 6, “bonjour, quelle est l’etape qui vous pose problème ?” (créer un compte) 7, “auriez vous la référence de cette table.” (commander une table) 8, “cette table est en stock sur notre site. si vous la commandez via le web, la livraison se fera à votre domicile.” (commander une table) 9, “je suis désolée mais votre commande n’entre pas dans le cadre de la livraison en point relais.” (point relais) 10, “ce produit n’est pas livrable en point relais.” (point relains) 11,“bonne journée.” (au revoir) 12, “à vous également à bientôt.” (au revoir)

Table 4.3 – A prefixed set of responses

List of Tables

References

P. Alquier and G. Biau. Sparse single-index model. *Journal of Machine Learning Research*, 14:243–280, 2013.

P. Alquier and B. Guedj. An Oracle Inequality for Quasi-Bayesian Non-Negative Matrix Factorization. *Mathematical Methods of Statistics*, 26(1):55–67, 2017.

P. Alquier and K. Lounici. PAC-Bayesian theorems for sparse regression estimation with exponential weights. *Electronic Journal of Statistics*, 5:127–145, 2011.

P. Alquier. *Transductive and Inductive Adaptive Inference for Regression and Density Estimation*. PhD thesis, Université Paris 6, 2006.

J.-Y. Audibert. Aggregated estimators and empirical complexity for least squared regression. *Annales de l'institut Henri Poincaré : Probabilités et Statistiques*, 40:685–736, 2004.

J.-Y. Audibert. *Une approche PAC-bayésienne de la théorie statistique de l'apprentissage*. PhD thesis, Université Paris 6, 2004.

J.-Y. Audibert. Fast learning rates in statistical inference through aggregation. *The Annals of Statistics*, 37(4):1591–1646, 2009.

P. Auer, N. Cesa Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal of Computing*, 32(1):48–77, 2003.

K. S. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001.

D. Bahdanau, K.H. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. ICLR 2015, pages 3104–3112, Cambridge, MA, USA, 2015.

J. D. Banfield and A. E. Raftery. Ice floe identification in satellite images using mathematical morphology and clustering about principal curves. *Journal of the American Statistical Association*, 87(417):7–16, 1992.

W. Barbakh and C. Fyfe. Online clustering algorithms. *International Journal of Neural Systems*, 18(3):185–194, 2008.

A. Barron, L. Birgé, and P. Massart. Risk bounds for model selection via penalization. *Probability Theory and Related Fields*, 113:301–413, 1999.

P. L. Bartlett, T. Linder, and G. Lugosi. The minimax distortion redundancy in empirical quantizer design. *IEEE Transactions on Information Theory*, 44(5):1802–1813, 1998.

J.-P. Baudry, C. Maugis, and B. Michel. Slope heuristics: overview and implementation. *Statistics and Computing*, 22(2):455–470, 2012.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

G. Biau and A. Fischer. Parameter selection for principal curves. *IEEE Transactions on Information Theory*, 58(3):1924–1939, 2012.

L. Birgé and P. Massart. Minimal penalties for gaussian model selection. *Probability Theory and Related Fields*, 183:33–73, 2007.

A. Blum and Y. Mansour. From external to internal regret. *Journal of Machine Learning Research*, 8:1307–1324, 2007.

L. Bottou. *Online Algorithms and Stochastic Approximations*. Cambridge University Press, Cambridge, UK, 1998.

C. Brunsdon. Path estimation from GPS tracks. In *Proceedings of the 9th International Conference on GeoComputation*, National Centre for Geocomputation, National University of Ireland, Maynooth, Eire, 2007.

R. B. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3:1–27, 1974.

O. Cappé, E. Moulines, and T. Rydén. *Inference in Hidden Markov Models*. Springer, 2005.

B. P. Carlin and S. Chib. Bayesian model choice via markov chain monte carlo methods. *Journal of the Royal Statistical Society*, Series B(57):473–483, 1995.

O. Catoni. *Statistical Learning Theory and Stochastic Optimization*. École d'Été de Probabilités de Saint-Flour 2001. Springer, 2004.

O. Catoni. *PAC-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning*, volume 56 of *Lecture notes – Monograph Series*. Institute of Mathematical Statistics, 2007.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning and Games*. Cambridge University Press, New York, 2006.

N. Cesa-Bianchi, P. M. Long, and M. K. Warmuth. Worst-case quadratic loss bounds for prediction using linear functions and gradient descent. *IEEE Transactions on Neural Networks*, 7(3):604–619, 1996.

N. Cesa-Bianchi, D. Helmbold, Y. Freund, Y. Haussler, and M. K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, 1997.

N. Cesa-Bianchi, G. Lugosi, and G. Stoltz. Minimizing regret with label-efficient prediction. *EEE Transactions on Information Theory*, 51:2152–2162, 2005.

N. Cesa-Bianchi, Y. Mansour, and G. Stoltz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66(2):321–352, 2007.

N. Cesa-Bianchi. Analysis of two gradient-based algorithms for on-line regression. *Journal of Computer and System Sciences*, 59(3):392–411, 1999.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

A. Choromanska and C. Monteleoni. Online clustering with experts. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 227–235, 2012.

F. Chung and L. Lu. Concentration inequalities and martingale inequalities: A survey. *Internet Mathematics*, 3:79–127, 2006.

J. Y. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. https://arxiv.org/abs/1412.3555, 2014.

I. Csiszár. I-divergence geometry of probability distributions and minimization problems. *Annals of Probability*, 3:146–158, 1975.

A. S. Dalalyan and A. B. Tsybakov. Aggregation by exponential weighting and sharp oracle inequalities. In *Learning theory (COLT2007), Lecture Notes in Comput. Sci., Vol. 4539*, pages 97–111, 2007.

A. S. Dalalyan and A. B. Tsybakov. Aggregation by exponential weighting, sharp PAC-Bayesian bounds and sparsity. *Machine Learning*, 72:39–61, 2008.

A. S. Dalalyan and A. B. Tsybakov. Mirror averaging with sparsity priors. *Bernoulli*, 18(3):914–944, 2012.

A. S. Dalalyan and A. B. Tsybakov. Sparse regression learning by aggregation and Langevin Monte-Carlo. *Journal of Computer and System Sciences*, 78(5):1423–1443, 2012.

O. Dekel and Y. Singer. Data-driven online to batch conversions. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 267–274. MIT Press, 2006.

P. Dellaportas, J. J. Forster, and I. Ntzoufras. On Bayesian model and variable selection using MCMC. *Statistics and Computing*, 12(1):27–36, 2002.

C. Dhanjal, R. Gaudel, and S. Clémençon. Incremental Spectral Clustering with the Normalised Laplacian. In *DISCML - 3rd NIPS Workshop on Discrete Optimization in Machine Learning*, 2011.

E. R. Engdahl and A. Villaseñor. 41 global seismicity: 1900–1999. *International Geophysics*, 81:665–690, 2002.

J. R. Firth. A synopsis of linguistic theory 1930–55. 1952–59:1–32, 1957.

A. Fischer. On the number of groups in clustering. *Statistics and Probability Letters*, 81:1771–1781, 2011.

A. Fischer. Selecting the length of a principal curve within a gaussian model. *Electronic Journal of Statistics*, 7:342–363, 2013.

Y. Freund, R. E. Schapire, Y. Singer, and M. K. Warmuth. Using and combining predictors that specialize. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 334–343, 1997.

H. Friedsam and W. A. Oren. The application of the principal curve analysis technique to smooth beamlines. In *Proceedings of the 1st International Workshop on Accelerator Alignment*, 1989.

A. Gelman and D. B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457 – 511, 1992.

S. Gerchinovitz. *Prédiction de suites individuelles et cadre statistique classique : étude de quelques liens autour de la régression parcimonieuse et des techniques d'agrégation*. PhD thesis, Université Paris-Sud, 2011.

F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451— 2471, 2000.

Y. Goldberg and O. Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. 2014. http://arxiv.org/abs/1402.3722.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

A. D. Gordon. *Classification*, volume 82 of *Monographs on Statistics and Applied Probability*. Chapman Hall/CRC, Boca Raton, 1999.

P. J. Green. Reversible Jump Markov Chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

B. Guedj and P. Alquier. PAC-Bayesian estimation and prediction in sparse additive models. *Electronic Journal of Statistics*, 7:264–291, 2013.

B. Guedj and L. Li. Sequential Learning of Principal Curves. preprint, 2018.

B. Guedj and S. Robbiano. Pac-bayesian high dimensional bipartite ranking. *Journal of Statistical Planning and Inference*, 196:70 – 86, 2018.

S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):511–528, 2003.

L. Györfi and G. Ottucsák. Sequential prediction of unbounded stationary time series. *IEEE Transactions on Information Theory*, 53(5):1866–1872, 2007.

A. György, T. Linder, G. Lugosi, and G. Ottucsák. The on-line shortest path problem under partial monitoring. *Journal of Machine Learning Research*, 8:2369–2403, 2007.

Z. S. Harris. *Distributional Structure*, volume 10. WORD, 1954.

J. A. Hartigan. *Clustering Algorithms*. Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, New York, 1975.

T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84:502–516, 1989.

W. K. Hasting. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97 – 109, 1970.

D. P. Helmbold and M. K. Warmuth. On weak learning. *Journal of Computer and System Sciences*, 50:551–573, 1995.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735 — 1780, 1997.

S. Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen*. PhD thesis, Technische Univ. Munich, 1991.

S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998.

H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417—441, 1933.

M. Hutter and J. Poland. Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research*, 6:639–660, 2005.

A. Kalai and S. Vempala. Efficient algorithms for universal portfolios. *Journal of Machine Learning Research*, 2:423–440, 2002.

A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

V. Kanade, B. McMahan, and B. Bryan. Sleeping experts and bandits with stochastic action availabilibty and adversarial rewards. *AISTATS*, 3:1137–1155, 2009.

A. Kannan and K. Kurach. Smart reply: Automated response suggestion for email. 2016. https://arxiv.org/abs/1606.04870.

L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Mathematical Statistics. Wiley-Interscience, Hoboken, 1990.

B. Kégl and A. Krzyżak. Piecewise linear skeletonization using principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):59–74, 2002.

B. Kégl, A. Krzyżak, T. Linder, and K. Zeger. A polygonal line algorithm for constructing principal curves. In *Proceedings of the 11th International Conference on Neural Information Processing Systems*, pages 501–507. MIT Press, 1998.

B. Kégl, A. Krzyzak, T. Linder, and K. Zeger. Principal curves: Learning and convergence. In *Information Theory, 1998. Proceedings. 1998 IEEE International Symposium on*, page 387. IEEE, 1998.

B. Kégl, A. Krzyżak, T. Linder, and K. Zeger. Learning and design of principal curves. *IEEE transactions on pattern analysis and machine intelligence*, 22(3):281–297, 2000.

B. Kégl. *Principal curves: learning, design, and applications*. PhD thesis, Concordia University Montreal, Quebec, 1999.

J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

J. Kivinen and M. K. Warmuth. Averaging expert predictions. In *Computational Learning Theory: 4th European Conference (EuroCOLT '99)*, pages 153–167. Springer, 1999.

K. C. Kiwiel. *Convergence and efficiency of subgradient methods for quasiconvex minimization*, volume 90. Springer, Berlin, Heidelberg, 2001.

R. D. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. *Regret Bounds for Sleeping Experts and Bandits*. In COLT. Springer, 2008.

A. N. Kolmogorov and V. M. Tikhomirov. ϵ-entropy and ϵ-capacity of sets in function spaces. *American Mathematical Society Translations*, 17:277–364, 1961.

S. Kotz and S. Nadarajah. *Multivariate T-Distributions and Their Applications*. Cambridge University Press, 2004.

W. J. Krzanowski and Y. T. Lai. A criterion for determination the number of clusters in a data set. *Biometrics*, 44:23–34, 1988.

V. Laparra and J. Malo. Sequential principal curves analysis. https://arxiv.org/abs/1606.00856, 2016.

Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. http://arxiv.org/abs/1405.4053, 2014.

L. Li, B. Guedj, and S. Loustau. A Quasi-Bayesian Perspective to Online Clustering. *Electronic Journal of Statistics*, 12(5), 2018.

L. Li. *PACBO: PAC-Bayesian Online Clustering*, 2016. R package version 0.1.0.

E. Liberty, R. Sriharsha, and M. Sviridenko. An algorithm for online k-means clustering. In *Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 81–89. SIAM, 2016.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–216, 1994.

N. Littlestone. From on-line to batch learning. In *Proceedings of the Second Annual Workshop on Computational Learning Theory*, COLT '89, pages 269–284, 1989.

D. A. McAllester. PAC-Bayesian model averaging. In *Proceedings of the 12th annual conference on Computational Learning Theory*, pages 164–170. ACM, 1999.

D. A. McAllester. Some PAC-Bayesian theorems. *Machine Learning*, 37(3):355–363, 1999.

W. S. McCulloch and W. Pitts. *A logical calculus of the ideas immanent in nervous activity*, volume 5 of *Bulletin of Mathematical Biology*. Springer, 1943.

K. L. Mengersen and R. L. Tweedie. Rates of convergence of the hastings and metropolis algorithms. *The Annals of Statistics*, 24(1):101 – 121, 1996.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, and A. H. Telle. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6), 1953.

S. Meyn and R. Tweedie. *Markov Chains and Stochastic Stability*. Springer, London, 1993.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50:159–179, 1985.

G. Neu and G. Bartók. An efficient algorithm for learning with semi-bandit feedback. In *Lecture Notes in Computer Science*, volume 8139, pages 234–248. Springer, Berlin, Heidelberg, 2013.

K. Pearson. On lines and planes of closest fit to systems of point in space. *Philosophical Magazine*, 2(11):559–572, 1901.

P. H. Peskun. Optimum monte-carlo sampling using markov chains. *Biometrika*, 60(3):607 – 612, 1973.

A. Petralias and P. Dellaportas. An MCMC model search algorithm for regression problems. *Journal of Statistical Computation and Simulation*, 83(9):1722–1740, 2013.

D. Pollard. Strong consistency of k-means clustering. *Annal of Statistics*, 9(1):135–140, 1981.

A. S. Pérez. *Aggregation of time series predictors, optimality in a locally stationary context*. PhD thesis, Télécom ParisTech, 2015.

A. E. Raftery and S. M. Lewis. Comment: One long run with diagnostics: Implementation strategies for markov chain monte carlo. *Statistical Science*, 7(4):493 – 497, 1992.

A. E. Raftery and S. M. Lewis. Convergence assessment techniques for markov chain monte carlo. *Statistics and Computing*, 8:319 – 335, 1998.

K. Reinhard and M. Niranjan. Parametric subspace modeling of speech transitions. *Speech Communication*, 27:19–42, 1999.

P. Rigollet and A. B. Tsybakov. Sparse estimation by exponential weighting. *Statistical Science*, 4(27):558–575, 2012.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, New York, 2004.

C. P. Robert and G. Casella. *Introducing Monte Carlo Methods with R*. Springer-Verlag, 2009.

G. O. Roberts and J. S. Rosenthal. Harris Recurrence of Metropolis-Within-Gibbs and Trans-Dimensional Markov Chains. *Annals of Applied Probability*, 16(4):2123–2139, 2006.

X. Rong. word2vec parameter learning explained. 2014. http://arxiv.org/abs/1411.2738.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386 – 408., 1958.

S. Sandilya and S. R. Kulkarni. Principal curves with bounded turn. *IEEE Transactions on Information Theory*, 48:2789–2793, 2002.

M. Seeger. PAC-Bayesian generalization bounds for gaussian processes. *Journal of Machine Learning Research*, 3:233–269, 2002.

M. Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, 2003.

J. Shawe-Taylor and R. C. Williamson. A PAC analysis of a Bayes estimator. In *Proceedings of the 10th annual conference on Computational Learning Theory*, pages 2–9. ACM, 1997.

C. Spearman. "General Intelligence", Objectively Determined and Measured. *The American Journal of Psychology*, 15(2):201–292, 1904.

D. C. Stanford and A. E. Raftery. Finding curvilinear features in spatial point patterns: principal curve clustering with noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):601–609, 2000.

L. Sutskever, O. Vinyal, and Q.V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS 14, pages 3104 – 3112, Cambridge, MA, USA, 2014. MIT Press.

R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic. *Journal of the Royal Statistical Society*, 63:411– 423, 2001.

S. Vijayarani. Preprocessing techniques for text mining - an overview. *International Journal of Computer Science & Communication Networks*, 5(1):7–16, 2015.

O. Vinyal and Q.V. Le. A neural conversational model. 2015. https://arxiv.org/abs/1506.05869.

V. Vovk. Competitive on-line statistics. *International Statistical Review*, 69(2):213–248, 2001.

O. Wintenberger. Optimal learning with Bernstein online aggregation. *Machine Learning*, 106(1):119–141, 2017.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

université angers

Titre : Algorithmes stochastiques en ligne .....
....

Mots clés : Apprentissage en ligne, Clustering en ligne, Quasi-bayésian, Borne de regret minimax, Reversible Jump Markov Chain Monte Carlo, Courbe principale séquentielle.

Résumé : Cette thèse travaille principalement sur trois sujets. Le premier concentre sur le clustering en ligne dans lequel nous présentons un nouvel algorithme stochastique adaptatif pour regrouper des ensembles de données en ligne. Cet algorithme repose sur l'approche quasi-bayésienne, avec une estimation dynamique (*i.e.*, dépendant du temps) du nombre de clusters. Nous prouvons que cet algorithme atteint une borne de regret de l'ordre $\sqrt{TlnT}$ et que cette borne est asymptotiquement minimax sous la contrainte sur le nombre de clusters. Nous proposons aussi une implémentation par RJMCMC. Le deuxième sujet est lié à l'apprentissage séquentiel des courbes principales qui cherche à résumer une séquence des données par une courbe continue. Pour ce faire, nous présentons une procédure basée sur une approche maximum a posteriori pour le quasi-posteriori de Gibbs. Nous montrons que la borne de regret de cet algorithme et celui de sa version adaptative est sous-linéaire en l'horizon temporel T. En outre, nous proposons une implémentation par un algorithme glouton local qui intègre des éléments de sleeping experts et de bandit à plusieurs bras. Le troisième concerne les travaux qui visent à accomplir des tâches pratiques au sein d'iAdvize, l'entreprise qui soutient cette thèse. Il inclut l'analyse des sentiments pour les messages textuels et l'implémentation de chatbot dans lesquels la première est réalisé par les méthodes classiques dans la fouille de textes et les statistiques et la seconde repose sur le traitement du langage naturel et les réseaux de neurones artificiels.

Title : Online stochastic algorithms ....
.....

Keywords : Online learning, Online clustering, Quasi-Bayesian, Minimax regret bound, Reversible Jump Markov Chain Monte Carlo, Sequential principal curve.

Abstract : This thesis works mainly on three subjects. The first one is online clustering in which we introduce a new and adaptive stochastic algorithm to cluster online dataset. It relies on a quasi-Bayesian approach, with a dynamic (*i.e.*, time-dependent) estimation of the (unknown and changing) number of clusters. We prove that this algorithm has a regret bound of the order of $\sqrt{TlnT}$ and is asymptotically minimax under the constraint on the number of clusters. A RJMCMC-flavored implementation is also proposed. The second subject is related to the sequential learning of principal curves which seeks to represent a sequence of data by a continuous polygonal curve. To this aim, we introduce a procedure based on the MAP of Gibbs-posterior that can give polygonal lines whose number of segments can be chosen automatically. We also show that our procedure is supported by regret bounds with sublinear remainder terms. In addition, a greedy local search implementation that incorporates both sleeping experts and multi-armed bandit ingredients is presented. The third one concerns about the work which aims to fulfilling practical tasks within iAdvize, the company which supports this thesis. It includes sentiment analysis for textual messages by using methods in both text mining and statistics, and implementation of chatbot based on nature language processing and neural networks.