Structured prediction for sequential data

Rémi Lajugie

École Doctorale de Sciences Mathématiques de Paris Centre

THÈSE DE DOCTORAT

présentée par

Rémi LAJUGIE

Pour obtenir le grade de docteur en sciences
Spécialité : Informatique

Prédiction structurée pour l'analyse de données séquentielles

Structured prediction for sequential data

Directeurs de thèse : Sylvain ARLOT et Francis BACH
Rapporteurs : Florence D'ALCHÉ-BUC et Fei SHA

Soutenance prévue le 18 septembre 2015 devant la commission d'examen composée de :

Mme Florence D'ALCHÉ-BUC	Télecom ParisTech	rapporteur
M. Sylvain ARLOT	École normale supérieure	directeur
M. Francis BACH	École normale supérieure	directeur
M. Arshia CONT	IRCAM	examinateur
M. Jean PONCE	École normale supérieure	examinateur
M. Stéphane ROBIN	Agro ParisTech	examinateur

INRIA 5eme étage
23 avenue d'Italie
75 013 Paris

UPMC
Ecole Doctorale de Sciences
Mathématiques de Paris Centre
4 place Jussieu
75252 Paris Cedex 05
Boite courrier 290

Résumé

Dans cette thèse nous nous intéressons à des problèmes d'apprentissage automatique dans le cadre de sorties structurées dans le cas particulier impliquant une structure séquentielle.

D'une part, nous considérons le problème de l'apprentissage de mesure de similarité pour deux tâches impliquant une telle structure: (i) la détection de rupture dans des signaux multivariés et (ii) le problème de déformation temporelle entre des paires de signaux. Les méthodes généralement utilisées pour résoudre ces deux problèmes dépendent fortement d'une mesure de similarité afin de comparer les points composant les signaux à chaque instant, par exemple la distance Euclienne. Nous proposons d'apprendre une mesure de similarité à partir de données totalement étiquettées, c'est à dire des signaux segmentés ou des paires de signaux pour lesquels la déformation optimale est connue. En nous appuyant sur des méthodes précédemment développées pour la prédiction structurée, nous présentons des moyens algorithmiquement efficaces pour effectuer l'apprentissage. Ces méthodes permettent notamment l'utilisation de fonctions de pertes spécifiquement adaptées aux tâches considérées. Nous validons notre approche sur des données réelles venant de divers domaines : vision, musique et bioinformatique.

D'autre part, nous nous intéressons au problème de la *faible supervision* où les données séquentielles ne sont pas totalement étiquetées. Nous nous intéressons au problème d'alignement d'une séquence sur sa représentation symbolique, en nous focalisant en particulier sur le problème d'alignement d'un enregistrement musical sur la partition jouée. Nous considérons la représentation symbolique comme donnant (i) une information complète sur l'ordre des symboles ou notes et (ii) une information approximative sur la forme de l'alignement attendu. Nous proposons une approche discriminative à ce problème utilisant ces deux types d'informations et apprenons, à partir de ces données faiblement supervisées, un classifieur pour chaque note. Nous développons une méthode d'apprentissage fondée sur l'optimisation d'une fonction de coût convexe qui tire profit de la faible supervision et de la structure séquentielle des données. Nous démontrons la validité de l'approche sur des données musicales.

Abstract

In this manuscript, we consider structured machine learning problems and consider more precisely the ones involving sequential structure.

In a first part, we consider the problem of similarity measure learning for two tasks where sequential structure is at stake: (i) the multivariate change-point detection problem and (ii) the time warping of pairs of time series problem. The methods generally used to solve these problems crucially rely on a good similarity measure to compare points at different timestamps, e.g., the Euclidean distance. We propose to learn a similarity measure from fully labelled data, i.e., signals already segmented or pairs of signals for which the optimal time warping is known. Based on techniques coming from the structured prediction field, we present algorithmically efficient ways for learning. These methods allows us to use loss functions that we specifically designed for the tasks considered. We eventually validate our approach on real-world vision, bioinformatics and musical data.

In a second part, we focus on the problem of *weak supervision*, in which sequential data are not totally labeled. We conduct our study on the problem of aligning a time series to its symbolic representation, using as a leading example the problem of aligning an audio recording with the score or partition played. We consider the symbolic representation as two fold: (i) it gives a complete information about the order of events or notes played and (ii) it gives an approximate idea about the expected shape of the alignment. We propose a discriminative approach taking into account these two kinds of information and we learn, from these weakly labeled data, classifiers for each of the possible notes. Our learning problem is based on the optimization of a convex cost function that takes advantage of the weak supervision and of the sequential structure of data. Our approach is validated through experiments on the task of audio-to-score on real musical data.

Contents

Contributions and outline of the thesis

Chapter 1: In this first introductory chapter, we present the core of this thesis: the structured prediction framework. It is a supervised classification framework for labels belonging to a *structured set*. Such a set is generally finite of big cardinality, preventing for performing an exhaustive search in it. It can be made of complex or composite outputs such as graphs or sequences. We present the usual large-margin approach to deal with it: the structured support vector machines. We stress the need to be able to solve the *loss-augmented decoding* efficiently. We also present the convex optimization tools that are needed in the following chapters.

Chapter 2: In this contribution, we present a structured prediction approach for multi-label classification that goes beyond the binary relevance approach which consists in learning a classifier for each possible label and then doing prediction independently. We propose an approach that takes into account interactions between labels in the form of pairwise affinities or repulsions. These can be encoded through an affinity matrix that we learn directly from the data while learning classifiers for each label. We cast the learning as a structured prediction one with losses properly designed for the task: the F_1 and the Hamming loss. We show that, depending on the sign of the entries of the affinity matrix, the structured prediction problem should be addressed in different ways. If this matrix only encodes affinities, the problem reduces to a standard structured prediction objective that can be solved approximately using graph cut techniques. If the matrix encodes affinities and repulsions, the *loss-augmented decoding* is not tractable and is related to the max-cut problem. We provide algorithmically efficient semidefinite and spectral relaxations for this case. Eventually, we demonstrate on real-world dataset the benefits of the proposed approach.

Chapter 3: In this second introductory chapter, we present the *sequential* structure. We present the dynamic programming framework that let us deal with it computationally. We focus on three applications: (i) the *change-point detection* that is the segmentation of a time series, (ii) the *alignment* problem that is the building of a mapping between a time series and a reference signal, (iii) the *time warping* that puts in correspondence pairs of timestamps in two different time series with a contiguity constraint. In the second part of this chapter, we present the similarity measure learning problem by focusing on the learning of *Mahalanobis quasimetric*.

Chapter 4: In this contribution, we propose to learn a similarity measure for the specific task of change-point detection and more generally for partitioning problems where potential constraints are involved. We consider methods that rely on a Mahalanobis quasimetric. We propose a formulation that turns out to cast the partitioning problem as *linear* in the matrix B associated to the Mahalanobis quasimetric. We cast the learning of B as a large-margin structured prediction problem and show that the loss-augmented decoding can be solved efficiently. We then improve our approach by going beyond fully-supervised datasets and propose an extension of our technique that can learn a quasimetric from partially labeled datasets. For general segmentation problems, we propose a way to relax the structured prediction optimization problem into a tractable one using a spectral relaxation. Eventually, we demonstrate the benefits of our approach, on video segmentation and segmentation of genomic time series tasks as well as on a real-world image segmentation task.

Chapter 5: In this contribution, we consider the learning of a Mahalanobis quasimetric for another task where sequential structure is at stake: the time warping problem that puts in correspondence pairs of time series. This problem is ubiquitous in many applications, for instance in music: we can think to two interpretations of the same musical piece that share the global structure but with local tempo differences. We consider the case where the problem rely on a Mahalanobis pairwise similarity measure. We propose to learn it from pairs of signals for which the optimal warping is known. To that extent, we propose a representation of warpings as binary matrices and build a new loss between these objects. We also propose a new approach to deal with large-margin structured prediction by relaxing the set of binary matrices into its convex hull and derive an efficient conditional algorithm to solve the optimization problem associated. We eventually apply our approach on real musical data coming from artificially warped data as well as the Bach's chorales dataset.

Chapter 6: In this contribution we propose to make use of weakly supervised sequential data at training time and focus on the task of aligning a time series (e.g, a musical recording) to a symbolic representation (e.g, the partition). Indeed, in applications, manually timestamped data on which the approaches of the two previous chapters are built are rarely available. But it is often easy to gather partial information, for instance the musical score for a music. Alignment uses a similarity measure between timestamps of the time series and symbols of the representation. We propose to learn it using only weakly-supervised data. We cast this problem as a structured prediction task using the Hamming loss and use a square surrogate to it to derive a *convex* relaxation of the empirical loss minimization problem that we can efficiently solve. We then apply our generic approach on the audio-to-score task. We show that it is possible to use the score specifically and to include musically meaningful penalties directly in the empirical loss minimization problem. We eventually provide experiments on real-world audio-to-score datasets. Our first results, on monophonic data show that our approach let us learn good similarity measure between symbols of the partition and musical recordings by making

use of only few supervision.

We list there the publications related to this manuscript:

- Chapter 2 is based on our preprint: Semidefinite and Spectral Relaxations for Multi-Label Classification, R. Lajugie, P. Bojanowski, S. Arlot, F. Bach [1].
- Chapter 4 is based on the article: Large-margin Metric Learning for Constrained Partitioning Problems, R. Lajugie, S. Arlot, F. Bach, In *Proc. ICML* 2014.
- Chapter 5 is based on the article: Metric Learning for Temporal Sequence Alignment, R. Lajugie, D. Garreau, In *Proc. NIPS* 2014.
- The core material Chapter 6 is a joint work with Piotr Bojanowski, Philippe Cuvillier, Sylvain Arlot and Francis Bach. It is under submission.
- In Chapter 6, we briefly recall the main results of our ECCV paper: Weakly Supervised Action Labelling in Videos Under Ordering Constraints, P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, J.Sivic, In. *Proc. ECCV*, 2014. However the results of this paper as well as the ones of our preliminary report: Weakly-Supervised Alignment of Video With Text, P. Bojanowski, R. Lajugie, E. Grave, F. Bach, I. Laptev, J. Ponce, C. Schmid [2] are not discussed in this manuscript and will be extensively detailed in the manuscript of P. Bojanowski.

1. Available on arXiv: http://arxiv.org/abs/1506.01829.
2. Available on arXiv: http://arxiv.org/abs/1505.06027.

Notations

Mathematical notations

$(e_1, \ldots, e_p)$	canonical basis of $\mathbb{R}^p$
$\mathbb{E}_X[f(X)]$	expectation of the function f under the distribution of X
Id_p	Identity matrix of dimension p
$\langle \cdot, \cdot \rangle$	Euclidean dot product for real vector spaces, namely Frobenius dot product for matrices: $\forall X, X' \in \mathbb{R}^{p \times K}$, $\langle X, X' \rangle \mathrm{Tr}(X^\top X)$ and standard dot product for vectors $\forall x, x' \in \mathbb{R}^p$, $\langle x, x' \rangle = x^\top x'$
$\mathbf{1}_p$	vector of $\mathbb{R}^p$ made of all ones, that is $\mathbf{1}_p = \sum_{i=1}^p e_i$
$\mathbb{R}^{p \times k}$	space of real matrices of p rows and k columns
$\mathbb{R}^p$	real vector space of dimension p
$\succeq$	partial order on the set of symmetric matrices, $\forall A, B \in S_p$, $A \succeq B$ if, and only if $B - A$ is positive semidefinite
A^B	For two sets A and B, set of functions from B to A
I_C	Convex indicator of the set C, that is $\forall x \in C, I_C(x) = 0$, and $+\infty$ otherwise
S_p^{++}	set of symmetric positive matrices of size $p \times p$
S_p^{+}	set of symmetric semidefinite positive matrices of size $p \times p$
S_p	set of symmetric matrices of size $p \times p$
X	upper case letters denote matrices
x	lower case letters denote vectors
x_i	i-th component of a vector
$X_{\cdot,j}$	j-th column of a matrix
$X_{i,\cdot}$	i-th row of a matrix

Abbreviations

a.k.a	also known as
e.g.	*exampli gratia*, for instance
i.e.	*id est*, meaning

Conventions

N	number of training instances
w or W	denotes the parameter to learn of a method
x or X	denotes the input of an algorithm
x^i	Superscripts refer to the position of a training instance in the sample
y or Y	denotes the output of an algorithm

Chapter 1

Introduction and Related Work

In a very broad perspective, our work addresses some issues coming from machine learning. This field of research aims at extracting meaningful rules from observed data that can be further generalized to interpret new data. We focus in this thesis on the specific problems induced by structured outputs, when the object involved in a machine learning task has a complex shape. We start our presentation by a first structured machine learning problem: the multi-class classification problem where the goal is to predict several outputs from some set to an instance. This can typically take the form of image tagging, that is to describe all the objects present in an image. Then we emphasize over a specific kind of structure: the sequential one which is ubiquitous in signal processing and more generally in all data-dependent fields. We present our two structured machine learning approaches to learn specific similarity measures for the change-point detection and the time warping problems. However the approaches we develop suffer from the fact they require fully-supervised data which are in practice hard to get, even if they exist in some specific cases we present in the corresponding chapters. To overcome this issue, we propose to go beyond existing approaches. To that extent, we consider musical data for which it is easy to gather weak-supervision thanks to the partition. We propose to only make use of this little amount of information to build a model for each of the possible notes.

In this introductory chapter, we introduce the notations used in this manuscript as well as the major concepts we use.

1.1 Background

Notations. Throughout this manuscript, lower case letters denote vectors, upper case ones matrices. When working with a generic Euclidean space, we refer to its standard dot product using the notation $\langle \cdot, \cdot \rangle$. These finite dimensional spaces are always real and they can be either vector space of dimension p, that we denote $\mathbb{R}^p$ or space of matrices of dimension p_1 lines and p_2 columns that we denote $\mathbb{R}^{p_1 \times p_2}$. For vector spaces, the dot product between x, x' in $\mathbb{R}^p$ corresponds to $\langle x, x' \rangle = x^\top x'$. For a matrix space, this dot product corresponds to $\langle X, X' \rangle = \mathrm{Tr}(X^\top X')$ for two matrices $X, X' \in \mathbb{R}^{p_1 \times p_2}$.

The norm associated to this dot product is denoted using the symbol $\|\cdot\|_2$. Training instances are indexed by the superscript i. Subscripts refers to indices within a vector or a matrix. Namely, for a vector x, x_j denotes its j-th component. For a matrix X, $X_{j,k}$ is the element indexed by the j-th row and the k-th column. For a square matrix $W \in \mathbb{R}^{p\times p}$, we use the symbol $\succeq$ to denote the partial order induced by the semidefinite cone. That is, if W and W' are two semidefinite positive matrices, we note $W \succeq W'$ if $W - W'$ is positive semidefinite. For a matrix $W \succeq 0$, we also make use of the notation $W \in \mathcal{S}^+$. If moreover, W is positive definite (if W has no eigenvalue equals to 0), we use the notation $W \succ 0$ or $W \in \mathcal{S}^{++}$. For matrices, we adopt also the following conventions: $X_{.,k}$ is the vector corresponding to the k-th column of X and $X_{k,.}$ the vector corresponding to the k-th row.
The p-dimensional vector composed of ones is noted $\mathbf{1}_p$. The dependency in p might be omitted when it is clear from the context. For a finite set $\mathcal{Y}$, the symbol $|\mathcal{Y}|$ stands for the cardinality of the set.
Let A and B be two sets, we denote by A^B, the set of functions from B to A.

1.1.1 Definitions and concepts

Let us now describe some of the major concepts that are at stake in this thesis.

A fundamental dichotomy in machine learning lies in the difference between *supervised* and *unsupervised* learning (see e.g, [Hastie et al., 2009]).
Supervised learning aims at learning a *decision function* f in some generic function set $\mathcal{F}$. This associates an input instance x in some input set $\mathcal{X}$ to an output y in some output set $\mathcal{Y}(x)$. It is important to notice that the output set $\mathcal{Y}(x)$ can depend on the training input. To avoid cumbersome notations, we introduce $\mathcal{Y} = \bigcup_{x\in\mathcal{X}} \mathcal{Y}(x)$, the set of all possible outputs.
A decision function is learned using N training pairs composed of one input signal $x^i \in \mathcal{X}$ together with the associated response $y^i \in \mathcal{Y}$. To set an analogy with human learning, we can think to some teacher showing pairs of one object x^i with the corresponding label y^i. Depending on the set $\mathcal{Y}$, different problems, or *tasks*, can be considered:

- If $\mathcal{Y}$ is a finite set of categorical variables (for example labels), the problem is known as *classification*. To fix ideas, let us consider $\mathcal{X} = \mathbb{R}_+$ the set of all possible light intensities recorded by some sensor and $\mathcal{Y}(x) = \text{“}day\text{”}, \text{“}night\text{”}$. The output of a classification algorithm is a binary-valued function that, to an illumination x, associates $f(x) \in \mathcal{Y}$, corresponding to “day” or “night”. In this setting, the decision function is often referred to as the *classifier*.
- If $\mathcal{Y}$ is a continuous space, the problem is known as the one of regression. With the previous example, if we take $\mathcal{Y} = [0, 24]$ representing the time of the day, the goal of regression is to build a function that, to an illumination x associates the time of the day. [1]

On the opposite, *unsupervised* learning aims at extracting rules from data with no exterior knowledge given by any teacher or expert. That is, there is no training

1. In this toy example, the regression problem might be very hard to solve, since at dawn and twilight, the illumination is the same.

output example y^i anymore. Such an algorithm outputs a transformation of the data like a partition of them, a projection of these on a lower dimensionnal space and so on.
In this manuscript, we mainly consider *supervised classification*.

Batch supervised classification. More precisely, we assume to be given N training pairs (x^i, y^i) composed of one input and the corresponding output. Moreover, we assume that all the N pairs can be accessed at any stage during the machine learning procedure for learning a classifier. This is known as the *batch* or *offline* setting: a prediction function is learned once for all and is never modified afterwards. So, there is two stages clearly defined:

1. the *training* stage in which a classifier is learned from data,
2. the *testing* stage in which the classifier is used on new data.

The batch setup is opposed to the *online* in which data comes sequentially. In this setting, testing and training stages are mixed: an input x^i is presented, the current classifier makes a prediction, the output y^i is revealed, the machine learning procedure modifies the classifier based on this feedback and then applies it on a new input x^{i+1}.

Input representation. In this thesis, we always assume that, in a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$, the input x belongs to some finite dimensional real vector space. There either exists some p such that $\mathcal{X} \subset \mathbb{R}^p$ or p and q such that $\mathcal{X} \subset \mathbb{R}^{p \times q}$. The inputs are thus denoted by x or X, depending whether we consider a vector or a matrix. We do not provide specific attention to the design of inputs. This is however a central topic in the signal processing literature, since the good representation of inputs, a.k.a. the design of features, has a major impact over the performances of machine learning techniques. Roughly speaking, if the descriptors we use for representing inputs miss a crucial aspect of a problem, it is hopeless to solve it. Imagine that, rather than building a day/night classifier, we want to build a dawn/twilight one. The use of light intensity alone will not allow us to build a better classifier than a random one. But, if we include the crucial aspect of the dynamic of the light intensity in our input, such a classifier will be easy to build. Beyond this toy example, in computer vision, the scale-invariant feature transform (SIFT) descriptors [Lowe, 1999] and their variants has drastically improved the performance of object classification tasks during the 2000's [Mikolajczyk and Schmid, 2005] [2]. Fine engineering of features followed by the use of these as input of a standard classification technique is a very common way to address object recognition. This leads to state-of-the-art results on most of the benchmark datasets in computer vision such as PASCAL [Everingham et al., 2010] or ImageNet [Russakovsky et al., 2015]. In music, for genre classification of songs, the representation of the signal has a high impact over the accuracy. For a fixed machine learning technique, different feature representation of musical signals can lead to very different classification performances [Li et al., 2003]. Engineering of inputs is thus very important in machine

2. Nowadays, the standard state-of-the-art features are built using neural networks.

learning and in all data-oriented fields.
Nevertheless, the central topic of the manuscript is *outputs*. More precisely, we focus on finite outputs sets $\mathcal{Y}$ with a complex underlying structure. For example $\mathcal{Y}$ can be a set of sequences, alignments or graphs.

Output sets and loss functions. The structure of an output set $\mathcal{Y}(x)$ is encoded through a *loss function* that measures how far two elements $y, y' \in \mathcal{Y}(x)$ are. That is, for each $x \in \mathcal{X}$, there exists a function

$$\ell_x : \mathcal{Y}(x) \times \mathcal{Y}(x) \to \mathbb{R}^+.$$

For simplicity and since it is always the case in the rest of the manuscript, we assume that if

$$y, y' \in \mathcal{Y}(x) \quad \text{and} \quad y, y' \in \mathcal{Y}(x'),$$

then

$$\ell_x(y, y') = \ell_{x'}(y, y').$$

That way, we can extend the family of losses $(\ell_x)_{x \in \mathcal{X}}$ to a loss over the whole set $\mathcal{Y}$. We define:

$$\begin{aligned} \ell : & \qquad \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+ \\ \ell(y, y') & = \begin{cases} \ell_x(y, y') \text{ if } \exists x, y \in \mathcal{Y}(x), \quad y' \in \mathcal{Y}(x), \\ +\infty \text{ otherwise.} \end{cases} \end{aligned} \tag{1.1}$$

For the classifier between day and night we have seen so far, one can imagine to define a loss that is equal to 1 if $y \neq y'$ and to 0 otherwise. The loss function is of crucial importance for machine learning tasks since it allows us to compare a true output y^i with a prediction made by a decision function.

1.1.2 Standard approaches to the supervised batch classification problem

In this work, we follow the discriminative (a.k.a conditional) approach to the classification problem. We assume that the N training pairs $(x^i, y^i) \in \mathcal{X} \times \mathcal{Y}$ are the independent and identically distributed realizations of random variables (X, Y) [3] that have some probability distribution p over $\mathcal{X} \times \mathcal{Y}$. Classifiers are built in the following way:

1. we find a model $\widehat{p}$ for the conditional probabilities $p(Y|X)$ using training data,
2. the classifier is the function associating to an input x, the output y that maximizes the model $\widehat{p}(y|x)$.

3. In this paragraph, and only in this, the upper case letters do not denote matrices but the random variable as opposed to its realization which is in lower case.

Intuitively, step 1 corresponds to building a function that scores all the possible outputs $y \in \mathcal{Y}$ for a given input $x \in \mathcal{X}$ [4], and step 2 corresponds to taking the highest scored output as the prediction of the classifier. Note that this approach is opposed to the *generative* one that tries to build a model for the joint probability $p(X, Y)$ [Ng and Jordan, 2002].

Two discriminative approaches. In the discriminative setting, there are two major ways to build such a model. The first one builds a conditional model of probabilities using the maximum likelihood principle. The second one learns a parameter by minimizing some empirical loss undergone over the training instances. It turns out that, in the case where the training pairs are assumed to be independent and identically distributed [5] using the maximum likelihood principle is equivalent to minimizing a specific empirical loss. See Sec. 1.2.7 for more details.

Assessing the quality of classifiers. Once a classifier has been learned, it is important to measure its quality. The standard way to do so is to consider new data, the *test* data, that have the same distribution as the train data and to use them to estimate the mean loss using the classifier. Namely, if $f : \mathcal{X} \to \mathcal{Y}$ is a classifier, the performance of a classifier is assessed by:

$$\mathbb{E}_{(X,Y)}\Big[\ell(f(X), Y)\Big],$$

which should be as small as possible. Generally, since we have only access to a finite number of test data [6], we assess the quality of the classifier by computing the empirical loss over the N_T test data $(x^{N+1}, y^{N+1}), \ldots, (x^{N+N_T}, y^{N+N_T})$:

$$\frac{1}{N_T} \sum_{i=1}^{N_T} \ell\big(f(x^{N+i}), y^{N+i}\big).$$

1.2 The classical binary classification problem

To fix ideas, let us present the simplest machine learning problem: the binary classification [Bishop et al., 2006, Hastie et al., 2009].

1.2.1 Parameterization of the problem

Let $\mathcal{Y}$ be an output set with two elements that may consist of two mutually exclusive labels. We parametrize it by the injection:

$$P : \mathcal{Y} \to \{-1, 1\}$$

4. In the rest of the manuscript such a function is called a *discriminative* function.
5. This is always implicitly the case in this manuscript.
6. That are pairs of data (x^i, y^i).

that sends one of the label to 1 and the other to -1. To avoid heavy notations, in the rest of the section, we identify $\mathcal{Y}$ to $P(\mathcal{Y}) = \{-1, 1\}$ [7]. The input set $\mathcal{X}$ is included in a vector space of dimension d, namely $\mathcal{X} \subset \mathbb{R}^d$. The goal of binary classification is to learn a decision function $f \in \mathcal{F} \subset \{-1, 1\}^{\mathcal{X}}$. Equivalently, we can assume that f is the sign of a *discriminative function*, $f(x) = \text{sign}(F(x))$, for some $F : \mathcal{X} \to \mathbb{R}$ (here to avoid problems of definition, we consider that $\text{sign}(0) = 1$).

Moreover, let us consider the following parametric linear model for F, we consider that there exists some w in $\mathbb{R}^p$ such that:

$$\forall x \in \mathbb{R}^p, F(x) = F(x; w) = \langle w, x \rangle. \tag{1.2}$$

Our overall model for the decision function is thus [8]

$$f(x) = f(x; w) = \text{sign}(F(x; w)). \tag{1.3}$$

Note that most of the techniques we present in this manuscript can be extended to the case where w is a parameter of arbitrary dimension using the theory of kernels [Schölkopf and Smola, 2002]. However, we stick the presentation to the finite dimensional case as it is the only relevant in this manuscript.

In the subsequent sections, we propose to learn the parameter w. Suppose that we are given N pairs of training instances $(x^1, y^1) \dots (x^N, y^N) \in \mathbb{R}^p \times \{-1, 1\}$. These instances are supposed to be drawn independently with a common distribution p over $\mathbb{R}^p \times \mathcal{Y}$.

Now let us see to what corresponds the building of conditional probabilities and the empirical loss minimization framework, in that case.

1.2.2 The logistic regression model

Logistic regression [Hastie et al., 2009] learns a probabilistic model based on a discriminative function. In the binary case, the conditional probabilities are of the following form:

$$p(y = 1|x; w) = \frac{\exp\left(\frac{\langle w,x\rangle}{2}\right)}{\exp\left(-\frac{\langle w,x\rangle}{2}\right)+\exp\left(\frac{\langle w,x\rangle}{2}\right)} = \frac{1}{1+\exp\left(-\langle w,x\rangle\right)},$$
$$p(y = -1|x; w) = \frac{\exp\left(\frac{\langle w,x\rangle)}{2}\right)}{\exp\left(\frac{\langle w,x\rangle}{2}\right)+\exp\left(\frac{\langle w,x\rangle}{2}\right)} = \frac{1}{1+\exp\left(\langle w,x\rangle\right)}.$$

Thus we have the following compact expression for the conditional probabilities:

7. Note however that this injection is arbitrary and that we could have chosen an other one, for instance by sending one input on 0 and the other on 1.

8. Following the standard notation of Tsochantaridis et al. [2005], when a function over the input space $\mathcal{X}$ depends on a parameter w, we separate parameters from variables using a semicolon.

$$p(y|x;w) = \frac{1}{1+\exp(-y\langle w, x\rangle)}. \tag{1.4}$$

The idea is then to adjust w in this model using the maximum likelihood principle over the training instances, that is, to maximize the conditional probabilities of the observed training labels given the training inputs, i.e.,

$$\underset{w\in\mathbb{R}^p}{\text{maximize}} \quad \Pi_{i=1}^N p(y^i|x^i;w). \tag{1.5}$$

This principle has been popularized by Fisher [Fisher, 1915] and can be traced back to Laplace and Bernoulli [9]. The usual way to solve the problem of Eq. (1.5), is to consider the log-likelihood, that is, in the case of logistic regression, to maximize:

$$\sum_{i=1}^N \log\Big(p(y^i|x^i)\Big) = \sum_{i=1}^N \log\Big(p(y^i|x^i)\Big) = -\sum_{i=1}^N \log\Big(1+\exp\big(-y^i\langle w, x^i\rangle\big)\Big). \tag{1.6}$$

Efficient algorithms exist to optimize this quantity. When data are low-dimensional, i.e., p is small, they are based on the Newton-Raphson method and called iteratively reweighted least squares in that case. They make use of the Hessian matrix of the expression (1.6) [10] and require its inversion. Thus, one iteration of the Newton-Raphson method is of complexity $O(p^3)$. In high dimension, this iteration cost is prohibitive, and we can only afford the computation of a gradient of the expression of Eq. (1.6) which is of complexity $O(p)$. In this situation, gradient descent methods are privilegied (see Sec. 1.6 for further details).

1.2.3 The $0-1$ loss

Now, let us consider the $0-1$ loss minimization approach. To learn the parameter w of the discriminative function, the intuition is to do so by minimizing the resulting error. Such an error should be 0 when the predicted label agrees with the actual training one and greater than 0 in other cases. The most natural notion of error in the binary classification problem is the $0-1$ loss. This corresponds to counting the number of mistakes done over the training examples. Between a prediction $f(x^i;w)$ and the actual label, we consider the following loss function:

$$\ell\big(y^i, f(x^i;w)\big) = \mathbb{1}_{y^i\neq f(x^i;w)}.$$

Asymmetric losses. Note that this loss is, up to a scaling, the only loss that is symmetric [11] over $\{-1,1\}$. However, if predicting 1 instead of -1 (false positive) is different from predicting -1 instead of 1 (false negative) we can define asym-

9. See [Stigler et al., 2007] for the whole history.
10. The Hessian is the matrix of second order partial derivatives.
11. We recall that a function $g : \mathcal{Y}\times\mathcal{Y}\to\mathbb{R}$ is symmetric if, and only if,

$$\forall y_1, y_2 \in \mathcal{Y}, g(y_1,y_2) = g(y_2,y_1).$$

metric losses [Viola and Jones, 2001, Masnadi-Shirazi and Vasconcelos, 2010]. For instance, in biomedicine the cost of false detection of a disease is usually smaller than a missing detection. Asymmetric classifiers have for instance been successfully applied to the detection of prostate cancers [Artan et al., 2010] with a significantly better accuracy than when using a symmetric loss function. However, if asymmetric cost functions are at stake, tuning the ratio between false positive cost and false negative is of crucial importance. This can be done using an exhaustive procedure [Masnadi-Shirazi and Vasconcelos, 2010] or directly when learning a classifier [Bach et al., 2006]. In this manuscript, we only consider symmetric losses.

1.2.4 Minimization of the empirical loss

Learning the optimal parameter w can be done by minimizing the empirical loss over the training examples. This is the approach proposed by Vapnik [1998]. In the binary classification setting, this corresponds to finding w in such a way that it minimizes the following quantity:

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{y^i \neq f(x^i;w)}. \tag{1.7}$$

This quantity is known as the empirical $0-1$ loss. If the pairs (x^i, y^i) are i.i.d, and if $m = \mathbb{E}[\ell(y^i, f(x^i; w))]$ exists, then the law of large numbers guarantees that the empirical loss converge to m when $N \to \infty$.

1.2.5 Preventing overfitting

Let us assume that there exists some computationally efficient way to deal with the minimization of Eq. (1.7). We will face an issue known as the problem of *overfitting*. This phenomenon occurs when we have learned "too well" on our training data and when the decision rule learned is so complex that it does not have good performance on new data. Indeed, we are not actually interested in having a small loss over the training examples. Instead, our goal is to have a small error on data that we have never seen before. Such data are commonly referred to as *test* data. Namely, the criterion we would really like to minimize would be the following expected loss:

$$\mathbb{E}_{y,x}\left[\mathbb{1}_{y \neq f(x;w)}\right].$$

We would like to find a parameter w such that the expected error is minimal on data that are drawn according to the same distribution than training instances. The minimization of Eq. (1.7) does not lead to good generalization performances, especially when the number of dimension of w is large. The issue is that, if the class of possible discriminative functions is too rich, it will be possible to stick to the training data. This phenomenon is what we call the overfitting problem. On the opposite, if the class of possible discriminative functions is too small, the

In the binary case, keep in mind that a loss ℓ is totally defined by the values $\ell(-1, 1)$ and $\ell(1, -1)$.

prediction function learned will be too far from the data. This problem is known as the underfitting problem. Therefore, we have to adjust the tradeoff between too simple discriminative functions and too complex ones [Vapnik, 1998, Bishop et al., 2006, Hastie et al., 2009]. For the loss minimization method we have presented so far, the usual way to prevent overfitting is to penalize vectors w that have a large norm. The idea is thus to replace the minimization of the empirical loss by the minimization of the *regularized* empirical loss:

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{y^i \neq f(x^i;w)} + \lambda\Omega(f). \tag{1.8}$$

The function Ω is called the *regularizer* and the parameter $\lambda \in \mathbb{R}_+$ is called the *regularization parameter*. We refer to the term $\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{y^i \neq f(x^i;w)}$ of Eq. (1.8) as the *data fitting* term.

Choosing Ω . The most commonly used choice for the penalization function Ω is the squared L_2 norm (by abuse of language, we simply talk about the L_2 regularization). As stated in Eq. (1.3), the prediction function is the sign of some discriminative function F which is linearily parameterized by a parameter w belonging to some vector space. The L_2 regularization in that setup corresponds to penalizing the empirical loss minimization by the quantity $\lambda\|w\|_2^2$ [Hoerl and Kennard, 1970]. This is a special case [12] of the control of the "complexity" of a function through L_2 norm in reproducing kernel Hilbert spaces [Schölkopf and Smola, 2002].
Indeed, the speed of variation of the discriminative function can be seen as a good measure of the complexity of the classifier in our case. If we let the discriminative function vary a lot, it will be possible for it to stick very well to training data, simply because it will be possible for it to change its sign quickly. The penalization by the norm of w enforces to have discriminative functions that do not vary too fast. To check that, let $x, x' \in \mathbb{R}^p$. Then using Eq. (1.2), we have that:

$$|F(x;w) - F(x';w)| \leq \|w\|_2(x - x').$$

Thus $\|w\|$ quantifies the regularity of the function F. In our work, we generally use the L_2 regularization. However, a lot of work has been done about choosing Ω. To enforce sparsity in the learned vector w (roughly speaking this means that the vector w has a lot of zeros), the L_1 penalization is commonly used. For instance, for regression task, this yields to the Lasso (or basis pursuit) [Tibshirani, 1996, Chen et al., 1998, Zhao and Yu, 2006] methods. Some other approaches to promote group sparsity have been recently designed. They make use of structured norms like Bach et al. [2012] or combine L_1 and L_2 regularization for the "elastic net" method [Zou and Hastie, 2005]. These approaches allow us to enforce task-

12. When we use a parametric model for F such as the one introduced in Eq. (1.2), we are implicitly working in a Hilbert space of finite dimension. In the kernel literature point of view, our setup corresponds to the "linear kernels"[Shawe-Taylor and Cristianini, 2000, Schölkopf and Smola, 2002].

dependent sparsity patterns in the parameter to learn. For instance, for the analysis of functional magnetic resonance imaging (fMRI) data, group sparsity is a way to induce the selection of predictive regions of the brain at different scales. This has improved the performance for the task of brain reading, that consists in inferring what a person is thinking about from active brain regions [Jenatton et al., 2012].

Choosing λ. Once the penalization function is chosen, we need to adjust the regularization parameter λ. It intuitively allows us to control the tradeoff between the regularization term (that leads to underfitting) and the data-fitting term (that leads to overfitting). This can be done by using a validation procedure [Hastie et al., 2009]:

1. among our data, we select a fraction of these that we left aside, these are called the *validation* data, the rest of the data are *training* data,
2. for each possible λ, we minimize the regularized empirical loss on the training data,
3. we select the λ that have the smallest loss over validation data.

However data are often scarce and we cannot really afford to set aside a certain amount of data for the validation. We would like to put as few instances as possible in the validation set. Ideally, we would like to be able to train using all the data without loosing the benefits of the validation procedure. This concern has led to a smarter way to do this validation procedure. This is the so-called *cross-validation* that has received a large interest since its introduction in the seventies [Stone, 1974, Geisser, 1975] approach. Instead of cutting the data in one train/validation split, we can split data into K folds (generally $K = 5$ or $K = 10$ is enough [Hastie et al., 2009, Arlot and Lerasle, 2012]). Successively, we select one of the K folds as a validation set. For each λ we compute the value of the regularized empirical loss. Then, we aggregate the results obtained for each of the K folds and select our parameter λ.

Now, let us move on more algorithmic concerns.

1.2.6 Convexification of the loss minimization problem

Our goal is now to solve the regularized empirical loss minimization problem. As suggested before, we consider now the regularization function Ω to be set as the squared L_2 norm. So, the problem we aim at solving is to:

$$\underset{w \in \mathbb{R}^p}{\text{minimize}} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{y^i \neq f(x^i; w)} + \lambda \|w\|_2^2. \tag{1.9}$$

This problem is known to be NP-hard [Zhang, 2004]. It is also known that convex problems are solvable in polynomial time using for instance the ellipsoid method [Nesterov et al., 1994]. Thus a way to deal with the minimization of Eq. (1.9) is to approximate it by a convex relaxation.

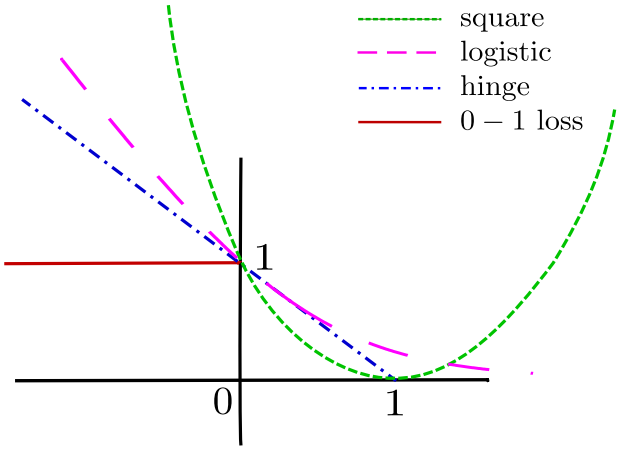


Figure 1.1: Some possible surrogates to the $0-1$ classification loss as functions of the confidence score $y\langle w, x\rangle$.

Convex surrogates. The most popular approach to replace the $0-1$ loss by a convex approximation is to consider a quantity, that we call the *margin* [Cortes and Vapnik, 1995]. This gives a confidence score for a generic output pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$. It is defined for binary classification as $y\langle w, x\rangle$. The convex approximation C to the loss is usually set as a convex function of the margin and satisfies the two following properties:

- $\forall y, w, x,\ C\big(y\langle w, x\rangle\big) \geq \mathbb{1}_{y \neq \text{sign}(\langle w, x\rangle)}$,
- C is differentiable in 0 and $C'(0) < 0$.

Such a function C is generally referred to as a convex surrogate to the $0-1$ loss. Figure 1.1 depicts some of the more common surrogates. This leads to the following *convexified empirical loss minimization* problem:

$$\underset{w \in \mathbb{R}^p}{\text{minimize}} \sum_{i=1}^{N} C\big(y\langle w, x\rangle\big) + \lambda \|w\|_2^2, \tag{1.10}$$

which is now a convex problem.

Replacing the $0-1$ loss by a surrogate satisfying the two aforementioned properties has been shown to be statistically consistent [Zhang, 2004, Bartlett et al., 2006]. This means that if we consider the case where $\lambda = 0$ in the problem of Eq. (1.10) and let N tend towards infinity, the minimizer of the regularized loss minimization problem of Eq. (1.7) and the one of Eq. (1.10) converges to the same values.

1.2.7 Standard examples

Following the approach previously described, several surrogates to the $0-1$ loss have been designed. Some of these are depicted in Fig. 1.1.

Smooth surrogates. Two smooth surrogates functions are widely used: the logistic surrogate and the squared one. The square surrogate replaces the $0-1$ loss

by

$$\big(1 - y\langle w, x\rangle\big)^2,$$

whereas the logistic one replace it by

$$\frac{1}{\log(2)} \log\Big(1 + \exp\big(-y\langle w, x\rangle\big)\Big).$$

The latter function is sometimes called the soft-max. Note that the empirical loss minimization problem associated to this loss is exactly the problem of Eq. (1.6). The logistic surrogate is the pink curve in Fig. 1.1. The square one corresponds to the green curve.

Non-smooth surrogate: the hinge loss. The hinge loss surrogate replaces the $0-1$ loss by the quantity $\max\big(0, 1 - y\langle w, x\rangle\big)$. This is the dashed blue curve in Fig. 1.1. This loss is very popular since it gave birth to the technique of support vector machines (SVMs) [Vapnik, 1998, Shawe-Taylor and Cristianini, 2000, Schölkopf and Smola, 2002, Steinwart and Christmann, 2008]. These are nowadays widespread in applications. In computer vision they are used for image classification and are the state-of-the-art method on some standard benchmark datasets including the ImageNet challenge [Russakovsky et al., 2015]. In biomedicine, they are used for protein classification [Leslie et al., 2004] or for gene expression classification [Rapaport et al., 2007].

Let us now move to the case where the output set $\mathcal{Y}$ is not binary anymore but is a big space with a potential *structure* to take into account.

1.3 Classification in discrete output spaces

The leading example for this section is an instance of the sequence labelling problem. We consider an optical character recognition (OCR) task [Taskar et al., 2003]. OCR aims at automatically recognizing characters in handwritten text. Given a text string of T characters, the goal is to find the sequence of characters that have been written. Assigning the sequence of recognized characters to the input data corresponds to what we call later the *decoding* of our structured model. This example has a natural structure. Indeed, not all sequences of characters are possible. For instance, sentences should be sequences of words in a dictionary and in a natural language some strings are impossible or very unlikely. In English "bnd" never occurs whereas "and" occurs very often.

We can then imagine two different but unsatisfying approaches to this problem:
(i) The first one casts the decoding as solving T successively independent single character recognition tasks. This is not the optimal way to do because we have a strong prior that finding the character that is at some position t heavily depends on the character at position $t-1$ and the one at position $t+1$.
(ii) The second one is to encode each word of the dictionary as a specific output,

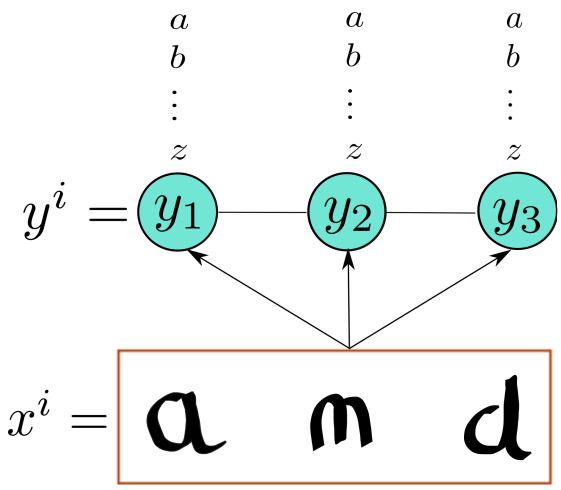


Figure 1.2: An example of a chain typically used for dealing with the OCR model [Taskar et al., 2003]. It is a good compromise between (i) the multiclass approach of Fig. 1.3 that becomes quickly intractable due to the huge set on which decoding is done and (ii) the independent model which does the predictions independently for each characters of the sequence, corresponding removing the horizontal arrows in that figure.

namely to consider the problem as a multiclass classification one with D classes, where D is the size of the vocabulary considered. This leads to a computationally intractable multiclass classification problem consisting in learning one prediction function for each of the possible word in the dictionnary. This model would correspond to the model depicted in Fig. 1.3.

Beyond the independent and the purely multiclass approches, properly modeling the OCR problem requires to take the sequential structure of the data into account. Practitionners generally consider an output model that is depicted in Fig. 1.2.
In the following sections, using this real life example we see how structure can be exploited to derive efficient machine learning techniques to address this kind of problems.

1.3.1 Notion of structured output space

In this section, we describe the structured prediction framework. For simplicity all variables x and parameters w are supposed to be vectorial. Given an input $x^i \in \mathbb{R}^p$, we recall that we define the corresponding structured output space as a pair $(\mathcal{Y}(x^i), \ell_{x^i})$. The output set $\mathcal{Y}(x^i)$ is finite and $\ell_{x^i} : \mathcal{Y}(x^i) \times \mathcal{Y}(x^i) \to \mathbb{R}$ is the corresponding loss function. We recall that this output set depends a priori on the input. For instance, for OCR, the space of possible output words is not the same if a word has 4 characters or 11. Following the convention we have established in previous sections, we can however consider the set $\mathcal{Y}$ as the union of all the $\mathcal{Y}(x^i)$ [13]

13. To avoid any problem and keep $|Y|$ finite, we do not consider unbounded sequences of characters assume that we have a prior over the maximal length of elements in $\mathcal{Y}$, for instance the size

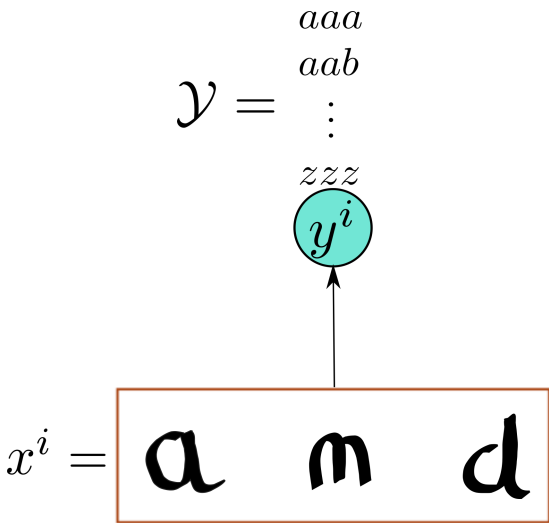


Figure 1.3: Example of the output model behind a purely multiclass approach to the OCR problem.

and extend the losses ℓ_{x^i} in a single loss ℓ over $\mathcal{Y}$ as in Eq. (1.1). In what follows, we consider that the set $\mathcal{Y}$ does not depend on the input x anymore since, using the same trick as the one we used for the loss, we can always reduce a structured prediction problem to this case.

Prediction and discriminative functions. Following the standard framework of structured prediction [Tsochantaridis et al., 2005, Taskar et al., 2003], we extend the notions of prediction and discriminative functions used implicitly or explicitly in the binary case. We consider a structured output set $\mathcal{Y}$ and a loss ℓ on it. Given input data $x \in \mathcal{X} \subset \mathbb{R}^p$, we consider the task of building a *prediction function*

$$f : \mathcal{X} \to \mathcal{Y}.$$

Following Tsochantaridis et al. [2005], Taskar et al. [2003], we assume that this function f is of the form:

$$f(x) \in \operatorname*{argmax}_{y \in \mathcal{Y}} F(x, y). \tag{1.11}$$

The function F is usually called the *discriminative function*. Intuitively, this function scores all possible outputs in $\mathcal{Y}$ given a certain x. In our OCR example, for a sequence of T characters, we can see this function F as scoring all the possible T-uples of characters. In most applications, F is assumed to be linearly parameterized by some $w \in \mathbb{R}^d$ and some *joint feature map* $\phi : \mathbb{R}^p \times \mathcal{Y} \to \mathbb{R}^d$. This corresponds to modelizing the discriminative function as:

$$F(x, y; w) = \langle w, \phi(x, y) \rangle. \tag{1.12}$$

of the longest word of the dictionnary.

Recall that the goal of the learning procedure is to learn w.

***Decoding* problem.** For a given input x, and a given parameter w, the decoding is the task of evaluating the function f of Eq. (1.11) [14]. The name might be surprising at first sight but can be justified for at least one historical reason. One of the first approaches to structured prediction, the max-margin-Markov networks of Taskar et al. [2003], were considering a probabilistic model for the joint feature map $\phi(x, y)$. This was decomposable over some graph with pairwise interactions (for instance the chain of letters depicted in Fig. 1.2). Mathematically, this means that, assuming an element of the set $\mathcal{Y}$ can be decomposed into T variables, namely $y = (y_1, \dots, y_T)$, the function ϕ has some decomposition:

$$\forall x, y, \qquad \phi(x, y) = \sum_{t=1}^{T} \sum_{t_1=1}^{T} \phi_{t,t_1}(x, y_t, y_{t_1}) \tag{1.13}$$

Thus, the task of finding the argmax of F consists in finding the maximum of some potential function that can be decomposed over a graph. In the graphical models literature, this is referred to as *decoding* [Wainwright and Jordan, 2008]. Indeed, it can be interpreted in a communication setting as the decoding of the some information that is sent over a communication channel [Shannon, 1948]. Note that in the literature, some people use the term of "inference" for the operation we call decoding. In this manuscript we reserve this name for another operation that will be described later.

1.3.2 Two examples of structured outputs sets

Let us now look at two important examples of structured output spaces, and the corresponding joint feature maps.

Binary classification as degenerate case of the structured setting. Note that the previous derivation extends the one done in Sec. 1.2. Indeed, binary classification falls into the structured setting. It suffices to consider a feature function $\phi(x, y) = x$ if $y = 1$, $-x$ if $y = -1$.

Feature map and the OCR example. A natural way to deal with the OCR example [Taskar et al., 2003] is to represent the outputs on a chain. A word of length T is represented by T variables y_t, each of these corresponding to one character. For a given input word x, the joint feature map is supposed to be organized on a chain built over the variables y_t [15]. In other words, there is a decomposition of the joint

14. This corresponds to the evaluation of the argmax over $\mathcal{Y}$ of Eq. (1.11).
15. Such a chain is depicted in Fig. 1.2.

feature map $\phi(x, y)$ of the following form:

$$\forall y \in \mathcal{Y}, \qquad \phi(x, y) = \sum_{t=1}^{T-1} \phi_t(x, y_t, y_{t+1}). \tag{1.14}$$

Performing the decoding task, for a given parameter of the discriminative function is the decoding of a chain graphical model that can be done using the standard Viterbi algorithm [Viterbi, 1967]. Note that, if we had chosen a more complex structure for the graph over which the joint feature map ϕ decomposes, such as the one depicted in Fig. 1.3, the decoding in the underlying graphical model would have been more difficult.

Learning w. Like for binary classification, there are two main ways to learn such a parameter w given N pairs of training examples (x^i, y^i). The first one is to build a model for the probabilities $p(y|x)$ using the maximum likelihood principle. The second one requires to build a loss function over the structured set $\mathcal{Y}$. Note that, contrary to the binary classification setting, there is a lot of freedom to design such a loss.

The first method we present is a generalization of the logistic regression model and does not require any loss to compare elements of $\mathcal{Y}$. That is why we postpone the design of a suitable loss function to further sections.

1.3.3 The conditional random field (CRF) model

The conditional random field model [Lafferty et al., 2001] is one of the first techniques to address structured prediction that have been proposed. This is an extension of the logistic regression model in the case where the output y is not in a set of classes but belongs to some structured set. Indeed, in the case of binary outputs, this model reduces exactly to logistic regression.

CRF model. Like logistic regression, the CRF model aims at building a conditional model of the output $p(y|x)$ of the following form:

$$p(y|x) \propto \exp\big(\langle w, \phi(x, y)\rangle\big), \tag{1.15}$$

where ϕ is a joint feature map of inputs and outputs. The computational tractability of the CRF model heavily depends upon the form of this function ϕ. More precisely, the crucial point is to define on which kind of graph the joint feature map can be decomposed. For a better understanding of this question, we invite the reader to have a look at Fig. 1.2 and Fig. 1.3. They both depicts an OCR task, but make two different models concerning the graphical representation of the joint feature functions. The second figure depicts the output model that casts the prediction task as a single multiclass problem over a pretty big output space whereas the first one depicts the case where outputs can be decomposed over a chain. Namely in the latter case, we consider the building of a conditional probability model based

on the joint feature map of the same form as in Eq. (1.14). In the rest of this section, we present the CRF model in the case where outputs y have naturally a sequential structure. Namely, we assume that elements y of $\mathcal{Y}$ can be represented as a T-uple of individual variables $(y_1 \dots y_T)$. In the specific case where the structure is a chain, the conditional probabilities are of the following form:

$$p(y|x;w) \propto \exp\Big(\sum_{t=1}^{T-1} \big\langle w_t, \phi_t(x, y_t, y_{t+1})\big\rangle\Big). \tag{1.16}$$

Link with the discriminative function. Like in the binary case, once w has been learned, one gets a classifier $f : \mathcal{X} \to \mathcal{Y}$:

$$\forall x \in \mathcal{X}, \qquad f(x) \in \operatorname*{argmax}_{y\in\mathcal{Y}} p(y|x).$$

This is equivalent [16] to:

$$\forall x \in \mathcal{X}, \qquad f(x) \in \operatorname*{argmax}_{y\in\mathcal{Y}} \big\langle w, \phi(x, y)\big\rangle. \tag{1.17}$$

This classifier is exactly of the form of the one associated to Eq. (1.12).

From an algorithmic point of view, there are two important steps when we deal with a conditional random field: the training and the decoding. Given annotated training pairs (x^i, y^i), training aims at learning the best parameters w_k of the model. Decoding consists, once the parameter w is learned, to associate to a novel input x the corresponding output $y \in \mathcal{Y}$.

Decoding in a CRF. Decoding is the task of evaluating, for some given x, $f(x)$ defined by Eq. (1.17). Due to the remark made in the previous paragraph, this notion corresponds to the decoding notion that we have seen in the previous section for general classifiers built on discriminative functions.
Computationally, the decoding in a CRF model is as hard as the decoding in the graphical model associated to the output. In the specific case of a chain CRF, the decoding can be thus performed using the Viterbi algorithm [Viterbi, 1967].

Training of a CRF. Imagine that we are given N training pairs (x^i, y^i), the training of a CRF consists in minimizing the training negative log-likelihood of the graphical model of Eq. (1.15), that is to minimize over the parameter w:

$$\frac{1}{n}\sum_{i=1}^{N} -\log\big(p(y|x^i;w)\big) + \lambda\Omega(w), \tag{1.18}$$

where Ω is typically a convex regularizer (generally the L_2 squared norm) to avoid overfitting.

16. We take the log, which is an increasing function, of the previous equation.

This optimization program is convex and is typically solved using quasi-Newton algorithms [Wallach, 2002, Sha and Pereira, 2003] or stochastic gradient methods [Vishwanathan et al., 2006]. The state-of-the-art technique [Schmidt et al., 2015] optimizes this objective using a slightly modified version of stochastic average gradient [Roux et al., 2012].

Let us mention some fields in which conditional random fields have received attention: (i) computer vision [Kumar and Hebert, 2006, Nowozin and Lampert, 2011] where they can be used in various settings, especially for image segmentation [Kolesnikov et al., 2014] and (ii) natural language processing where they can be used for topic modelling [Zhu and Xing, 2010], semantic role labelling [Cohn and Blunsom, 2005], named-entity recognition [Settles, 2004] or shallow parsing [Sha and Pereira, 2003].

The CRF model however does not take into account at any moment the structure of the space, that can be given by a loss. Let us now move to another approach for building a classifier that takes explicitly the loss ℓ over $\mathcal{Y}$ into account.

1.4 The standard structured SVM framework

Following the same ideas that have led us to the formulation of the regularized loss minimization problem of Eq. (1.8), we can derive an analogous optimization objective for the structured prediction case.

1.4.1 Empirical loss minimization.

We consider the learning of a parameter w for the discriminative function F involved in Eq. (1.11) by minimizing the following empirical loss:

$$\frac{1}{N}\sum_{i=1}^{N}\ell\big(y^i, f(x^i; w)\big) + \frac{\lambda}{2}\|w\|_2^2. \tag{1.19}$$

This is equivalent to minimizing:

$$\frac{1}{N}\sum_{i=1}^{N}\ell\big(y^i, \operatorname*{argmax}_{y\in\mathcal{Y}} F(x^i, y; w)\big) + \frac{\lambda}{2}\|w\|_2^2.$$

This minimization problem generalizes the one of Eq. (1.7).

1.4.2 Convexification of the empirical loss minimization

In the binary classification setting, when $\mathcal{Y} = \{-1, 1\}$ and the loss is the $0-1$ loss, the minimization of Eq. (1.19) for $w \in \mathbb{R}^p$ is NP-hard [Zhang, 2004]. So, this problem is also hard for general sets $\mathcal{Y}$ and losses ℓ and we have to find a tractable approximation of it.

Following the same approach than for binary classification, we replace the non-convex function $w \to \ell(y, f(x; w))$ by a convex surrogate C. The most popular

of these in the structured setting is the structured hinge loss. It has received a lot of interest [Taskar et al., 2003, Tsochantaridis et al., 2005, Yu and Joachims, 2009, Nowozin and Lampert, 2011] and extends the binary hinge loss presented in Sec. 1.2.7. The idea is to extend the concept of *margin* of Sec. 1.2. Given a training pair (x^i, y^i) and an output $y \in \mathcal{Y}$, we define the margin between y and y^i as the difference of the two scores of the discriminative function, that is:

$$m(x^i, y^i, y; w) = \big\langle w, \phi(x^i, y^i) - \phi(x^i, y) \big\rangle.$$

The intuition behind structured SVM [17] is to build a convex surrogate that generalizes the binary hinge loss. Two approaches have been proposed by Tsochantaridis et al. [2005] to generalize the Hinge loss:

– the slack rescaling approach that replaces $\ell(y^i, f(x^i; w))$ by:

$$\max_{y \in \mathcal{Y}} \Big\{ \ell(y^i, y) \Big(1 - \big\langle w, \phi(x^i, y^i) - \phi(x^i, y) \big\rangle \Big) \Big\},$$

– the margin rescaling approach that replaces $\ell(y^i, f(x^i; w))$ by:

$$\max_{y \in \mathcal{Y}} \Big\{ \ell(y^i, y) - \big\langle w, \phi(x^i, y^i) - \phi(x^i, y) \big\rangle \Big\}.$$

Both of these approaches reduce to the binary SVM when the output set is made of two outputs [18]. We focus more on the second approach since in this thesis, due to computational issues, this is the only one we can consider.

1.4.3 Loss-augmented decoding

From now on, we consider that we replace the non convex loss ℓ by its margin rescaling surrogate. We want to minimize with respect to $w \in \mathbb{R}^p$: [19]

$$H_N(w) = \frac{1}{N} \sum_{i=1}^{N} \max_{y \in \mathcal{Y}} \Big\{ \ell(y^i, y) - \big\langle w, \phi(x^i, y^i) - \phi(x^i, y) \big\rangle \Big\} + \frac{\lambda}{2} \|w\|_2^2. \tag{1.20}$$

To be able to minimize this function we need at least, to be able to evaluate it for any $w \in \mathbb{R}^p$. Contrary to the binary SVM, the maximization in the sum of Eq. (1.20) is not necessarily easy to solve. It depends on the structure of ϕ (like the decoding) but also on the loss ℓ over $\mathcal{Y}$. The task of maximizing with respect to $y \in \mathcal{Y}$,

$$\ell(y^i, y) - \big\langle w, \phi(x^i, y^i) - \phi(x^i, y) \big\rangle,$$

17. Or structural SVM.

18. If with $\mathcal{Y} = \{-1, +1\}$, and $\phi(x, y) = y\frac{x}{2}$, we recover exactly the hinge loss $\max(0, 1 - y\langle w, x \rangle)$ for both margin rescaling and slack rescaling approaches.

19. Note that to have the formulation of Eq. (1.20) well defined, we need to extend the feature map $\phi(x, y)$ in a similar way we extended the loss from $\mathcal{Y}(x)$ to $\mathcal{Y}$ in Eq. (1.1), by defining $\phi(x, y) = -\infty$ if $y \notin \mathcal{Y}(x)$ Moreover we need to take the convention $+\infty - \infty = 0$.

is called the *loss-augmented decoding*.

Let us see in the case of OCR how we can build a loss that leads to a tractable loss-augmented decoding by reducing it to a simple decoding.

Loss over the set of outputs: the example of OCR. Now, let us build a loss function over the set $\mathcal{Y}$ for the OCR task. The most standard loss in that context is the natural loss from information theory: the Hamming loss [Hamming, 1950]. Let y and y' be two possible outputs for OCR. The Hamming loss counts the number of characters that differ between y and y'. Formally, this is the sum of individual errors done over each character. For two output words $y = (y_1, \dots y_T)$ and $y' = (y'_1, \dots y'_T)$ of the same length T, the Hamming loss is defined by:

$$\ell(y, y') = \sum_{t=1}^{T} \mathbb{1}_{y_t \neq y'_t}. \tag{1.21}$$

Recall that this loss is not used by the CRF model we have seen in Sec. 1.3.3.

It is a sum over each node of the graph of Fig. 1.2. Therefore we can write the loss-augmented decoding for a pair of training example (x^i, y^i), using the same notations as in Eq. (1.13) as the maximization over $\mathcal{Y}$ of:

$$\sum_{t=1}^{T-1} \left\langle w_t, \phi_t(x^i, y_t, y_{t+1}) - \phi_t(x^i, y^i_t, y^i_{t+1}) \right\rangle + \sum_{t=1}^{T} \mathbb{1}_{y_t \neq y'_t}. \tag{1.22}$$

The maximization of this quantity can be done by a decoding in the graphical model of Fig. 1.2 where pairwise potentials are $\left\langle w_t, \phi_t(x^i, y_t, y_{t+1}) - \phi_t(x^i, y^i_t, y^i_{t+1}) \right\rangle$ and unary ones $\mathbb{1}_{y_t \neq y'_t}$. Therefore the loss-augmented decoding can be performed through a decoding step over a graph whose potentials are slightly modified as noted by Taskar et al. [2003].

1.4.4 Optimization of the structured SVM

For background material about optimization and further references we refer the reader to Sec. 1.6. Our goal in this section is to explain how it is possible to optimize the structured SVM minimization problem. We consider the algorithmic problem of reaching the minimum of the function of Eq. (1.20) for $w \in \mathbb{R}^p$.

Plain and stochastic subgradient techniques. The first idea might be to compute a subgradient of the function to minimize (the structured hinge loss is not differentiable). Then we make a step using a stepsize decaying in $O(1/k)$ (if k is the iteration counter) in the opposite of the direction of the subgradient. This procedure is known to converge to the optimum of the function. For SVM this algorithm is often called Pegasos [Shalev-Shwartz et al., 2011]. More details about convergence are in Sec. 1.6. However, due to the need of carefully tuning the stepsize, these techniques are empirically slow compared for instance to cutting plane or Frank-Wolfe methods.

Cutting plane techniques. One of the most popular procedures to optimize structured SVM is the one proposed by Tsochantaridis et al. [2005] and Joachims et al. [2009]. This technique is known as the "cutting-plane method" and is currently implemented in the popular package SVMstruct. The idea is to minimize at each iteration, a more and more accurate surrogate to the objective function.
Iteratively, given the objective of minimizing the function H_N of Eq. (1.20), a cutting plane method:

1. finds a subgradient g of the function at the current point $w_k \in \mathbb{R}^p$,
2. gets an affine lower bound to the objective function by

$$h_k : \ w \to g^\top w + H(w_k),$$

   item builds a convex lower bound to H_N by

$$L_k = \max_{j \leq k} h_j(w),$$

3. minimizes L_k to get a new point w_{k+1}.

Dual of the structured SVM problem. After having reviewed algorithms that perform the minimization of the structured SVMs in the primal, let us now derive a dual problem. We recall that the primal problem of structured SVM is:

$$\underset{w \in \mathbb{R}^p}{\text{minimize}} \ \frac{1}{N} \sum_{i=1}^{N} \max_{\bar{y}^i \in \mathcal{Y}} \Big\{ \ell(\bar{y}^i, y^i) - \big\langle w, \phi(x^i, y^i) - \phi(x^i, \bar{y}^i) \big\rangle \Big\} + \frac{\lambda}{2} \|w\|_2^2.$$

This problem is a convex optimization problem. Let us introduce a parametrization for the output space. We embed each $y \in \mathcal{Y}$ on the extreme points of the $|\mathcal{Y}|$ dimensional simplex. Namely, each element $y \in \mathcal{Y}$ is mapped to a vector of dimension $|\mathcal{Y}|$ with exactly one non-zero entry equal to one. In the rest of this paragraph, we identify y to its vectorial representation in the corresponding output space of dimension $|\mathcal{Y}|$. This representation allows us to see also $\ell(\bar{y}^i, y_i)$ as a dot product in the $|\mathcal{Y}|$ dimensional space: let us stack all the possible loss $\ell(\bar{y}^i, y^i)$ into a single vector L_i. We then have that:

$$\ell(\bar{y}^i, y^i) = \langle \bar{y}^i, L_i \rangle.$$

Similarly, we stack all the possible vectors $\phi(x^i, y^i) - \phi(x^i, \bar{y}^i)$ in a matrix Δ^{ϕ_i} of size $p \times |\mathcal{Y}|$ whose columns correspond to a $\phi(x^i, y^i) - \phi(x^i, \bar{y}^i)$. Thus the the output set $\mathcal{Y}$ can be identified to extreme points of the $|\mathcal{Y}|$ simplex, namely

$$|\mathcal{Y}| = \{0, 1\}^{|\mathcal{Y}|}, \langle \bar{y}^i, \mathbf{1}_{|\mathcal{Y}(x^i)|} \rangle = 1.$$

Using these notations, we get the following equivalent formulation of the primal problem:

$$\underset{w\in\mathbb{R}^p}{\text{minimize}}\ \frac{1}{N}\sum_{i=1}^{N}\max_{\bar{y}^i\in\mathcal{Y}}\Big\{\langle\bar{y}^i, L_i\rangle - \langle w, \Delta^{\phi_i}\bar{y}^i\rangle\Big\} + \frac{\lambda}{2}\|w\|_2^2.$$

We introduce the vector $Z \in \{0,1\}^{N|\mathcal{Y}|}$ as the vertical concatenation of all the $\bar{y}^i$. To avoid cumbersome notations, we denote by $\mathcal{Z}$ the set where authorized Z lies [20], L the concatenation of the L_i and the matrix $\Delta^\phi \in \mathbb{R}^{Np\times N|\mathcal{Y}|}$ the block-diagonal matrix whose blocks are the Δ^{ϕ_i}. All these transformations allow us to see the structured SVM problem as involving only one training example. With these notations, the primal can be compactly written as:

$$\underset{w\in\mathbb{R}^p}{\text{minimize}}\ \max_{Z\in\mathcal{Z}}\Big\{\frac{1}{N}\langle Z, L\rangle - \langle w, \Delta^\phi Z\rangle\Big\} + \frac{\lambda}{2}\|w\|_2^2.$$

Note that, with respect to Z, the inner loop is the maximization of a linear form over extremal points of a huge cartesian product of simplices. By the so-called "fundamental theorem of linear programming", this is equivalent to:

$$\underset{w\in\mathbb{R}^p}{\text{minimize}}\ \max_{Z\in\overline{\mathcal{Z}}}\Big\{\frac{1}{N}\langle Z, L\rangle - \frac{1}{N}\langle w, \Delta^\phi Z\rangle\Big\} + \frac{\lambda}{2}\|w\|_2^2, \tag{1.23}$$

where $\overline{\mathcal{Z}}$ is the convex hull of $\mathcal{Z}$. This problem is already in its primal/dual form. Since we are in a situation where we can use a saddle-point theorem (Prop. 5.5.7 (3) of [Bertsekas, 2015]), we get the dual by switching the maximization and the minimization. Now let us call $\bar{y}^i$ the individual block of variables of Z corresponding to the simplex associated to example i. Setting the gradient with respect to w to 0 yields to the following relation between primal and dual variables:

$$w = \frac{1}{\lambda N}\sum_{i=1}^{N}\Delta^\phi Z = \frac{1}{\lambda N}\sum_{i=1}^{N}\Delta^{\phi_i} y_i. \tag{1.24}$$

And the dual program can be written in the following form:

$$\underset{(\bar{y}^1,\ldots\bar{y}^N)\in\mathcal{Y}\times\ldots\times\mathcal{Y}}{\text{maximize}}\ -\frac{1}{2\lambda N^2}\|\sum_{i=1}^{N}\Delta^{\phi_i}\bar{y}^i\|_2^2 + \frac{1}{N}\sum_{i=1}^{N}\langle\bar{y}^i, y^i\rangle. \tag{1.25}$$

We recover the dual of structured SVM in the form it is usually cast [Taskar et al., 2003, Lacoste-Julien et al., 2013]. However, the way we derived it, using a saddle-point theorem is different from the usual way that makes use of Lagrange multipliers [Lagrange, 1811].

Now, let us derive some optimization techniques directly from this dual [21].

20. That corresponds in fact the Cartesian products of the $N\ |\mathcal{Y}|$ dimensional simplices.

21. Note that, historically, the first optimization techniques for structured SVMs started from the saddle-point form of the problem of Eq. (1.23) and makes use the dual extragradient method of Nesterov [2009]. More details can be found in [Taskar et al., 2006].

Frank-Wolfe techniques in the dual. The dual program of structured SVMs is the maximization of a concave quadratic function over a compact set $\overline{\mathcal{Z}}$. It is thus a convex problem. However the number of variables involved can be big. It is $N|\mathcal{Y}|$. This dual seems thus hard to deal with. However, Frank-Wolfe methods [Frank and Wolfe, 1956, Jaggi, 2013] allows us to cope with this dual thanks to a little trick. First, let us briefly recall the principle of the algorithm.
Roughly speaking, Frank-Wolfe algorithm minimizes convex differentiable functions over compact and convex sets (that is the case for $\overline{\mathcal{Z}}$) by iteratively [22]:

1. computing a linear approximation to the function using its gradient,
2. minimizing this linear approximation, getting a point z^s of the optimization set, here $\overline{\mathcal{Z}}$,
3. doing a convex combination between the current point and z^s using some combination parameter γ.

The key insight allowing us to use Frank-Wolfe technique for the dual problem of Eq. (1.25) is that performing one Frank-Wolfe step is exactly equivalent to solving loss-augmented decoding for all the examples as shown by [Lacoste-Julien et al., 2013]. Indeed, for each example i, we recall that we denote the *block* of variables in Z as $\bar{y}^i$. The corresponding entries in the gradient of the objective of the problem of Eq. (1.25) are:

$$g_i(\bar{y}^i) = -\frac{1}{\lambda N^2} \sum_{j=1}^{N} \Delta^{\phi_j^\top} \Delta^{\phi_i} \bar{y}^j + \frac{1}{N} \sum_{i=1}^{N} y^i, \tag{1.26}$$

But since we have the primal/dual relation between variables, this can be rewritten as:

$$g_i(\bar{y}^i) = -\frac{1}{N} \sum_{j=1}^{N} \Delta^{\phi_j \top} w + \frac{1}{N} \sum_{i=1}^{N} y^i. \tag{1.27}$$

For each block i, a Frank-Wolfe step corresponds to maximizing the linear form $\langle y, g_i(\bar{y}^i) \rangle$ over y belonging in the i-th simplex. Due to the form of Eq. (1.27), this is exactly a loss-augmented decoding step. More details can be found in Sec. 2 of [Lacoste-Julien et al., 2013].

Because the optimization set $\overline{\mathcal{Z}}$ is a Cartesian product of simplices, each corresponding to one training example, a natural extension of the Frank-Wolfe method have been to do the Frank-Wolfe updates in a block-coordinate manner [23] [Lacoste-Julien et al., 2013]. This has led to state-of-the-art results concerning optimization of structured SVMs. This is due to the fact that the dual objective of structural SVM is quadratic. Indeed, when dealing with a quadratic program, Frank-Wolfe techniques have a major advantage over others like stochastic gradient: there is no need to adjust any internal parameter of the algorithm. In practice, tuning the parameters of stochastic gradient techniques requires high-level knowledge and experience [Bottou, 2012]. Frank-Wolfe techniques in this specific case are parameter-free.

22. This algorithm is described more in details in Sec. 1.6.

23. That is, the gradient is updated for only one of the simplices corresponding to a block of variables $\bar{y}_i$ at each time.

1.5 Differences between binary and structured prediction

We have already seen, on the OCR example, that there are several ways to represent the output space $\mathcal{Y}$ for structured prediction, contrary to the case of binary classification [24]. All the representations are not equally relevant. In this section we describe formally what makes the quality of a representation.

1.5.1 Parametrization of output spaces

In Sec 1.4.2, we have seen that the key ingredient for being able to solve the structured SVM problem is the ability to efficiently solve the loss-augmented decoding. This is a major difference with binary classification where this is an obvious step. To address this issue, we need to build a good *parametrization* (or embedding) of the output set, i.e., to find an injective mapping: $P : \mathcal{Y} \to \mathbb{R}^p$. We illustrate this section using the running OCR example.

Simplest embedding. In the previous section, to derive the dual of the structured SVM problem, we built such a mapping. To this end, we defined the injection P which associates to an element $y \in \mathcal{Y}$, a corner of the $|\mathcal{Y}|$-dimensional simplex. This is an indicator vector which has $|\mathcal{Y}|$ entries, all component equals to 0 except one which is equal to 1.
However, for most applications, this parametrization is not informative. For the OCR example this embedding would correspond to embed each of the possible sequence of T characters in a K^T dimensional space, where K is the number of characters in the considered alphabet. This approach to the OCR problem corresponds to the one that is depicted in Fig. 1.3.

Properties of a good embedding. Embeddings should allow us to formulate the loss-augmented decoding in a computationally tractable way. Namely, this means that the loss can be expressed in a natural way in this space. In general the embedding presented in the previous paragraph does not let us formulate the loss in a simple and tractable manner.

Natural embedding for the OCR example. For the OCR example, instead of considering a K^T-dimensional space, a natural way to encode $\mathcal{Y}$ is to consider the TK dimensional concatenation of the individual indicator vectors for each character. In practice this is what is done [Taskar et al., 2003]. As we have seen before, and as depicted by Fig. 1.2, we can represent each character y_t in the sequence, by its indicator vector. The cartesian product of such vectors gives a natural representation for any sequence of T characters. From now on, we consider that y_t is the indicator

24. For binary classification, we always have $\mathcal{Y}$ equivalent to take $\mathcal{Y} = \{-1, 1\}$.

vector of character y_t and y refers to the concatenation of these vectors. Using this embedding, the Hamming loss can be expressed in a simple way:

$$\ell(y, y') = \sum_{t=1}^{T} \mathbb{1}_{y_t \neq y'_t} = \|y - y'\|_2^2. \tag{1.28}$$

This allows us to express the OCR problem in a tractable way. In particular, as seen in Sec. 1.4.3 the loss-augmented decoding [Taskar et al., 2003] can be naturally expressed in that space.

1.5.2 Notion of weak supervision

All that we have presented so far, sticking to the previous approaches to structured prediction [Tsochantaridis et al., 2005, Taskar et al., 2003], is based upon the assumption of *full supervision*. We assume that we are provided training pairs of input x together with a corresponding exact output y. However, in structured prediction, it is possible to have data that are weakly-supervised, to have only access to partial information about the actual label. For instance, in the case of OCR, we can imagine to only know a few characters of the sequence. We can also imagine to know the order of characters occuring in the sequence, for instance that a "r" comes after a "t" but that there might be other characters in between. This information, however not complete, should be taken into account during learning. In this thesis, to handle weak supervision, we propose to see this kind of partial information as an uncertainty over the label. The training instances are not pairs of input/outputs but pairs $(x^i, \mathcal{Y}^i)$ of an input together with the subset $\mathcal{Y}^i$ of $\mathcal{Y}$ which is the set of all possible outputs that satisfy the conditions given by the available partial information. For instance, for OCR, if we know that a sequence starts by the character "a", $\mathcal{Y}^i$ is the subset of all words starting by this character. Obviously this cannot happen for binary classification since $|\mathcal{Y}| = 2$: either we know y^i exactly, either we do not it at all.

Empirical loss minimization In this setting, it is natural to consider a variant of the empirical loss minimization of Eq. (1.19), that is to consider the following problem:

$$\underset{w \in \mathbb{R}^p}{\text{minimize}} \min_{y^i \in \mathcal{Y}^i} \frac{1}{N} \sum_{i=1}^{N} \ell\big(y^i, f(x; w)\big). \tag{1.29}$$

If we introduce the set of all subset of the finite set $\mathcal{Y}$ and denote it by $2^{\mathcal{Y}}$, this is equivalent to considering the loss $\tilde{\ell} : \mathcal{Y} \times 2^{\mathcal{Y}} \to \mathbb{R}$ defined by $\tilde{\ell}(y, \mathcal{Y}^i) = \inf_{y^i \in \mathcal{Y}^i} \ell(y^i, y')$ and to look for minimizing the empirical loss using this loss. The intuition is to jointly find a decision rule together with the most likely label. Such a weakly-supervised approach is depicted in Fig. 1.4.

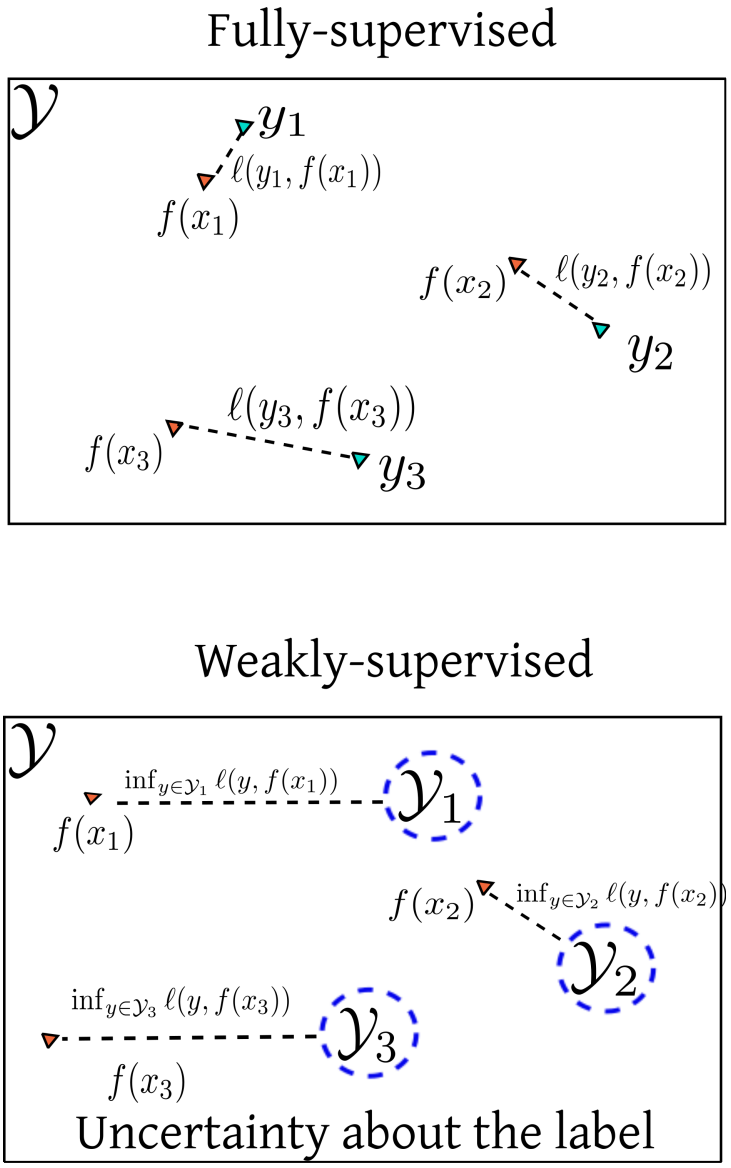


Figure 1.4: A cartoon figure depicting the idea of weak supervision and the difference with full supervision. In a structured prediction task, it is possible to have partial information about a label. In the bottom figure this is represented through a circle corresponding to the subset $\mathcal{Y}_i$. In a weakly supervised setting, the goal is to minimize the loss between the prediction $f(x_i)$ and the subset of labels corresponding to the weak supervision. The point realizing the infimum between a prediction and the corresponding set $\mathcal{Y}_i$ is the *rounded* solution.

1.6 Tools from convex optimization

Most of the machine learning algorithms derived in this manuscript involve the minimization of some function. In this section, we consider the minimization of a generic real-valued function defined on $\mathbb{R}^d$, namely: $f : \mathbb{R}^d \to \mathbb{R}$. All the functions used in this section are convex.

1.6.1 Elements of convex analysis

Convex set. A subset S of $\mathbb{R}^d$ is said *convex* if:

$$\forall x, y \in S,\ \forall t \in [0,1], \qquad (1-t)x + ty \in S. \tag{1.30}$$

Convex functions. Intuitively, a convex function is a function whose graph is, for each pair of points $(x^i, x^j) \in \mathbb{R}^p \times \mathbb{R}^p$, under the chord $[f(x^i), f(x^j)]$. More formally, a *convex* function is a function f defined on a convex set $C \subset \mathbb{R}^p$ satisfying:

$$\forall x, y \in C, \quad \forall \alpha \in [0,1], \quad f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y).$$

The function is strictly convex if the following property holds:

$$\forall x, y \in C, \forall \alpha \in]0,1[, \quad f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha) f(y).$$

In all this section, we assume that the set C is the whole space $\mathbb{R}^d$. This assumption is not restrictive, because if we introduce the convex indicator function I_C which is equal to 0 over C and $+\infty$ on the rest of the space, we can extend the function f to a convex function $f + I_C$ over the whole space [Boyd and Vandenberghe, 2004]. This extended function whose values are on the extended real line $\overline{\mathbb{R}}$ has the same minimum than the original one.
Note also that a function f is said to be *concave* if $-f$ is convex.

Convex functions are common in mathematics. For instance, in this thesis we use the functions defined by $\forall x \in \mathbb{R}, f(x) = (x-1)^2$ or $f(x) = \max(0, 1-x)$ which are convex. These two functions are depicted in Fig. 1.1.

Subgradient, gradient and Hessian. We say that g is in the subgradient to the convex function f at x, if the following holds:

$$\forall y \in \mathbb{R}^d, \quad f(y) - f(x) \geq \langle g, y - x \rangle.$$

The subgradient is a set generally denoted by $\partial f(x)$. One of the key properties of convex functions is that they always admit a non-empty subgradient set [Rockafellar, 1970].

Assume that $f : \mathbb{R}^d \to \mathbb{R}$ is a continuously differentiable function. We denote by $\nabla f(x)$ the d dimensional column vector whose i-th component is the partial derivative $\frac{\partial f(x)}{\partial x_i}$ of f at x along the i-th vector of the canonical basis. This vector is called the *gradient* of f at x. The gradient of f at x can be thought as the steepest

ascent direction of f around x. Note that if a convex function f is differentiable at x, $\partial f(x) = \{\nabla f(x)\}$.

For a twice differentiable function f, we call the matrix H_x whose (i,j)-th component is equal to $\frac{\partial^2 f(x)}{\partial x_i \partial x_j}$ the Hessian of f at x. If f is twice continuously differentiable, this matrix is symmetric.

Note that for a function f twice continuously differentiable at x, we have the following Taylor expansion around any point x:

$$f(y) = f(x) + \nabla f(x)^\top (y - x) + (y - x)^\top H_x (y - x) + o(\|y\|_2^2).$$

Note also that in that case, a function is convex if, and only if, H_x is positive semidefinite for all x in $\mathbb{R}^d$.

Minimum of convex functions. The fundamental fact about convex function [Rockafellar, 1970, Boyd and Vandenberghe, 2004, Bertsekas, 1999], is that any local minimum is a global one. This property is crucial for practitioners since any minimization algorithm will always converge to a global minimizer of the function.

In terms of subgradients, x is a minimum of a convex function f if and only if $0 \in \partial f(x)$.

Fenchel conjugate. For a function f, we can define its Fenchel conjugate f^* [Fenchel, 1949], a.k.a its Legendre transform, by:

$$f^*(x) = \sup_{w \in \mathbb{R}^d} \langle w, x \rangle - f(w). \tag{1.31}$$

Note that f^* is always a convex function, as a supremum of affine functions. Intuitively, this corresponds to encoding a function through its tangent hyperplanes. Let us consider the simplest case: a one-dimensional convex function f, differentiable, whose derivative function f' is bijective from $\mathbb{R}$ to $\mathbb{R}$. In this case, the Fenchel conjugate of f at x corresponds graphically to the intercept of the line tangent to f with slope x (bijectivity of f' guarantees existence of such a point where the derivative of f is x). This is depicted in Fig. 1.5. More formally, the Fenchel conjugate is found over one fundamental idea: a convex function is a function whose epigraph [25] is convex. The convex conjugate encodes this convex set in terms of its supporting hyperplanes.
Under mild conditions [26], that are always met in that manuscript, the biconjugate of a convex function is the function itself.

25. See Fig. 1.5 for an intuition about epigraph. This is the zone in blue above the graph of a function

26. That is, if the function f is proper and closed. A convex function is proper if it is never equals to $-\infty$ and if it is finite for at least one x. See e.g, Th. 12.2 of [Rockafellar, 1970]. A convex function is closed if its epigraph is closed or equivalentally if the level sets $L_\alpha = \{x \in \mathbb{R}^d, f(x) \leq \alpha\}$ are closed for every $\alpha \in \mathbb{R}$.

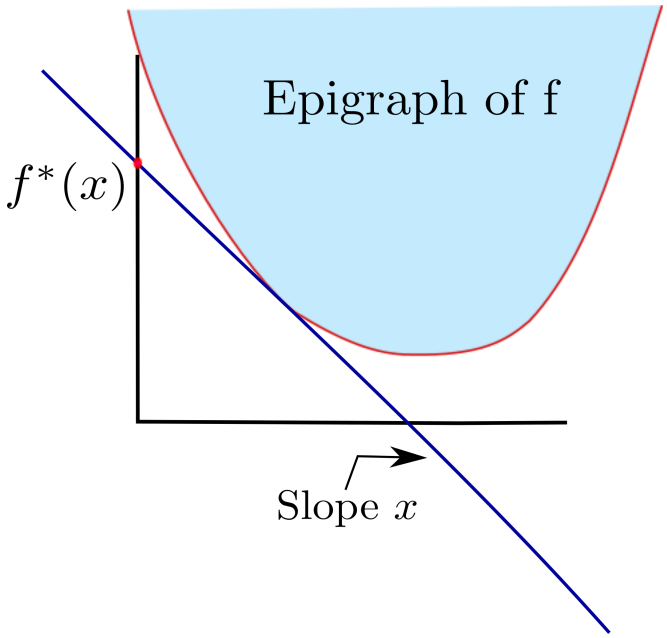


Figure 1.5: Representation of a convex function from $\mathbb{R}$ to $\mathbb{R}$. The convex conjugate value $f^*(x)$ can be read as the intercept of the tangent to f whose slope is x. The blue zone corresponds to the "epigraph" of f that is convex for convex functions.

Convex problems. A problem is convex if it is the minimization of a convex function f over a convex set $C \in \mathbb{R}^d$:

$$\underset{x \in C}{\text{minimize}}\, f(x). \tag{1.32}$$

The function to minimize is generally referred to the *objective* function of the optimization problem. As mentioned earlier, this can always be extended over the whole space $\mathbb{R}^d$ by introducing the convex indicator of C.

The formulation of Eq. (1.32) means that we aim at finding $x \in C$ such that the value of f is minimal. The function f is called the *objective function* of the minimization problem. We have to stress the difference between the problem of Eq. (1.32) and the expression

$$\min_{x \in C} f(x),$$

which denotes the minimal value of f. If the function f to minimize is an affine function and if the optimization domain C is a polytope, then the optimization problem is referred to as a *linear program* (LP). When f is quadratic, this is a *quadratic program*.
Note that we also call a problem convex if it is the maximization of a concave function over a convex set C.

Optimization problems in machine learning. Recall that machine learning problems generally come from the convexification of the empirical loss minimization

as seen in Sec. 1.4.2. Thus the objective function f has some structure. In this manuscript, we always consider the case where f is the sum of a data-fitting term g and a convex regularizer Ω. Namely, the generic form of minimization problems we consider is:

$$\underset{x\in\mathbb{R}^d}{\text{minimize}}\ g(x)+\Omega(x). \tag{1.33}$$

Note that the regularizer Ω is (almost) always a convex increasing function of the norm $\|\cdot\|_2$ in our work.

Dual of a convex problem. A central notion in convex optimization is duality. The idea is to find a non-trivial and informative lower bound to the minimal value of the problem that we are solving. We want this lower bound to be as tight as possible. The problem of Eq. (1.33) is referred to as the *primal* problem. A corresponding *dual* problem is a maximization problem of a concave function whose values are always lower or equal to the ones of the primal. There are two major ways to find a dual problem to a primal one: using Lagrangian duality [Boyd and Vandenberghe, 2004] [27] or using Fenchel conjugate [Rockafellar, 1970]. We describe the latter approach since it is specifically well-suited in the case of problems in composite form like in Eq. (1.33).

The dual problem of Eq. (1.33) is:

$$\underset{y\in\mathbb{R}^d}{\text{maximize}}\ -g^*(y)-\Omega^*(y), \tag{1.34}$$

where Ω^* and g^* are the Fenchel conjugate of Ω and g respectively. The dual objective function is thus: $-g^*-\Omega^*$. It is easy to see that for all pairs $(x,y)\in\mathbb{R}^d\times\mathbb{R}^d$, we have the following relation (that is often called the weak duality relation):

$$-g^*(y)-\Omega^*(y)\leq g(x)+\Omega(x). \tag{1.35}$$

This holds for any pair (x,y), thus we have

$$\forall x,y\in\mathbb{R}^d\quad -g^*(y)-\Omega^*(y)\leq \min_{z\in\mathbb{R}^d}\big\{g(z)+\Omega(z)\big\}\leq g(x)+\Omega(x).$$

Thus for all pairs of primal/dual variables (x,y), we have that

$$g(x)+\Omega(x)+g^*(y)+\Omega^*(y)\geq g(x)+\Omega(x)-\min_{z\in\mathbb{R}^d}\big\{g(z)+\Omega(z)\big\}\geq 0.$$

The quantity $g(x)+\Omega(x)+g^*(y)+\Omega^*(y)$ is called the *duality gap* associated to the primal/dual pair (x,y). It is always a majorizer to the gap between the value of the function $g+\Omega$ at x and the value of its minimum. For convex functions, the duality gap satisfies also an other appealing property. Indeed, if Ω is a proper con-

27. This duality comes from the techniques of Lagrange multiplier that have been introduced to deal originally with mechanical problems and can be traced back to Lagrange [1811].

vex function [28], the dual satisfies the strong Fenchel duality theorem [Rockafellar, 1970], that is, the reverse inequality:

$$\min_{x\in\mathbb{R}^d}\{g(x)+\Omega(x)\} = \max_{y}\{-g^*(y)-\Omega^*(y)\}. \tag{1.36}$$

This is much harder to prove than the inequality (1.35) and holds only in some cases [29] whereas weak duality is always true for any problem. This means that the dual optimal value and the primal one are the same. That is, if the primal has a minimizer x^* and the dual a maximizer y^*, then the duality gap associated to these is equal to 0. That way, if we use a minimization algorithm to solve Eq. (1.33) that produces a sequence of iterates $x_1, \ldots, x_k$, and that in parallel we run the same algorithm to maximize the dual and get a sequence $y_1, \ldots, y_k$, the duality gaps associated to the pair (x_i, y_i) provides us a good way to monitor the convergence of the algorithm since the gap between the optimal primal and dual value is 0.

1.6.2 Algorithmic approaches

We now focus on the algorithmic way to minimize convex functions. This problem has been shown to be solvable in polynomial time [Nesterov et al., 1994] and many polynomial time algorithms have been designed. Among these we can distinguish several trends:

- second-order methods that make use of the Hessian, like Newton-based algorithms [Boyd and Vandenberghe, 2004, Nesterov, 2004]. They are well-suited for problems in small dimension (d small),
- first-order methods that only makes use of the gradient or of subgradients that scale to large d,
- zero-th order methods that only makes use of function values (see chapter 9 of [Nemirovskii]) and that a priori can scale to huge d.

Machine learning tasks involve large-dimensional parameters. Algorithms need to be memory efficient and computationally cheap. For instance, we cannot afford to compute the Hessian of our objective function. This prevents us from using Newton-like or interior point techniques. We thus focus on the first-order methods that only make use of gradient or subgradient of the function to minimize.

Batch first-order techniques. first-order techniques only make use of the gradient of the function to optimize. A gradient algorithm [Cauchy, 1847] is generally of the form of Alg. 1. The idea is, at each iteration, to do a step from the current point in the opposite of the direction of the gradient (which is the steepest descent direction). The length of this step, that is called the *stepsize*, has to be set carefully. We stop the descent as soon as we have completed some terminating condition E. This can be, for instance, a maximum number of iterations.

28. This is always the case in the problems we consider.
29. That include most convex problems in general and all the one we consider in this manuscript in particular.

Algorithm 1 Generic gradient algorithm to minimize $f(x)$

Input: $x_0 \in \mathbb{R}^d$, $\forall t \in \mathbb{N}^*$, stepsize α_t, a terminating condition E
$t = 0$
repeat
  $x_{t+1} = x_t - \alpha_t \nabla f(x_t)$
  $t = t + 1$
until $E =$ true
Output: a minimizer x^*

In the case where the function is not differentiable, we can replace the gradient by a subgradient of the function f. In that case, we speak of a subgradient technique.

Standard stochastic first-order techniques. Stochastic optimization techniques are particularly well-suited for minimizing functions that are finite sums [Robbins and Monro, 1951, Bottou, 1998], and have thus received a lot of interest in the machine learning literature. They tackle an optimization problem of the form:

$$\underset{x \in \mathbb{R}^d}{\text{minimize}} \; f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x).$$

They rely on the very simple idea of replacing the full gradient of f by an approximation given by the gradient of one of the f_i. If the sampling of the f_i is uniform, the gradient of f_i yields to an unbiased estimate of the actual gradient of f.
In machine learning because of the very nature of the empirical loss minimization problem of Eq. (1.19), most of the optimization problems we meet are in the form of finite sums. To go from a batch method like the one of Alg. 1 to a stochastic version, one has just to replace by the gradient of one of the f_i taken uniformly at random. These techniques are widely used in practice due to their scalability [Bousquet and Bottou, 2008].

Projected version of gradient techniques. All the previous derivations assume that the optimization is conducted over the whole space $\mathbb{R}^d$ (that is we are in the unconstrained setup) and that the gradient could be computed for any point in $\mathbb{R}^d$. However, it is often that the function f is properly defined only over a convex subset $C \subset \mathbb{R}^d$ (the function might have been extended over the whole space but f takes finite values only on the subset C). The most natural way to handle this problem is to add to Alg. 1, a projection step. This leads to Alg. 2. The projection step is typically a minimization problem of the form:

$$\underset{x \in C}{\text{minimize}} \; \|x_{t+1} - x\|_2^2. \tag{1.37}$$

This optimization problem is quadratic and might be hard to solve: it is very likely that the projection step is as expensive as the original optimization program. In

Algorithm 2 Generic projected gradient algorithm to minimize $f(x)$ over a compact domain C

Input: $x_0 \in C$, $\forall t \in \mathbb{N}^*$, stepsize α_t, a terminating condition E
$t = 0$
repeat
  $x_{t+1} = x_t - \alpha_t \nabla f(x_t)$
  Project back onto C and set x_{t+1} to be this projection
  $t = t + 1$
until E = true
Output: a minimizer x^*

Algorithm 3 Generic Frank-Wolfe algorithm to minimize $f(x)$ over a convex and compact domain C

Input: $x_0 \in C$, $\forall t \in \mathbb{N}^*$, convex combination parameters γ_t , a terminating condition E
$t = 0$
repeat
  Compute gradient of f at x_t, $\nabla f(x_t)$
  Compute $x_s \in \text{argmin}_{x \in C} \langle \nabla f(x_t), x \rangle$.
  Set $x_{t+1} = (1 - \gamma_t)x_t + \gamma_t x_s$
  $t = t + 1$
until E = true

such situations, it might be possible that we can only afford to minimize linear forms over C (we assume C to be convex and compact for simplicity). Such a linear minimization can be used instead of the projection step. This is the idea behind the Frank-Wolfe algorithm, also called the conditional gradient algorithm.

The Frank-Wolfe algorithm. The Frank-Wolfe [Frank and Wolfe, 1956, Jaggi, 2013] algorithm is a first-order method to minimize convex objectives over compact sets. The idea is to iteratively approximate the function f by its linear approximation at the current point x. Then we minimize this linear approximation over the compact set, yielding to some point x_s. Using a convex combination parameter γ_k (set for instance using the universal step-size $\gamma_k = \frac{2}{k+2}$ where k is the iteration counter), this algorithm converges to a minimum of the convex function. Pseudo code is provided in Alg. 3. For the simplicity of notations, we consider that the function to optimize is differentiable.

1.6.3 Some convergence results for standard first-order techniques

In this section, we mention some of the speeds of standard first-order optimization algorithms. Let us fix the vocabulary.
Let $\mathcal{A}$ be a generic iterative algorithm for solving Eq. (1.32). This algorithm produces a sequence of iterates $(x_t)_{t \in \mathbb{N}}$. We define the *accuracy* at time t as the differ-

ence between $f(x_t)$ and the minimal value f^* of f. The speed can be given in two ways: (i) given a target accuracy ϵ, saying that the convergence is in $O(g(\epsilon))$ means that we need $O(g(\epsilon)$ iterations of the algorithm to get the accuracy ϵ, (ii) given a fixed number of iterations k, saying that convergence is $O(g(k))$ means that after k iteration, accuracy is of order $O(g(k))$. The convergence rates we give here are of the second form. Generally, a rate is said to be slow if it is slower than $O(\frac{1}{k})$ and fast if it is above.

Lipschitz continuity, smoothness and strong convexity. For the analysis of convex optimization methods, two standard different assumptions have a drastic influence over the convergence results:

1. the first one is the *smoothness* (or Lipschitz gradient [30]) assumption, namely if f is differentiable at every point, f is β-smooth if and only if,

$$\exists \beta > 0, \forall x, y \in \mathbb{R}^d, \ \|\nabla f(x) - \nabla f(y)\|_2 \leq \beta \|x - y\|_2.$$

   For a twice differentiable function, this corresponds to having a uniform upper bound over the eigenvalues of the Hessian,

2. the second one is the *strongly convex* assumption. We say that f is α-strongly convex if and only if

$$\exists \alpha > 0, \forall x, y \in \mathbb{R}^p, \|f(x) - f(y)\|_2 \geq \alpha \|x - y\|.$$

   For a twice differentiable function, this corresponds to having a uniform lower bound over the eigenvalues of the Hessian.

Convergence rates of convex optimization techniques. We recall in Table 1.1 some of the major convergence results concerning first-order batch optimization techniques for the minimization problem (1.32). Stochastic rates are in Table 1.2. More precise results can be found in the relevant litterature [Bertsekas, 1999, Bubeck, 2014].

We recall that the convergence rates in Tables 1.2 and 1.1 correspond to the order of magnitude of the error at iteration k during an optimization procedure. For instance and for a batch method, a convergence rate in $O(1/k)$ means that after k iterations, the difference $f(x_k) - f^*$ is of order $O(1/k)$. For a stochastic method, a $O(1/k)$ convergence rate means that after k iterations, the expected difference $\mathbb{E}\left[f(x_k) - f^*\right]$.
It is interesting to note that the smoothness does not help in the stochastic case. Fast rates for smooth functions do not transfer from the batch to the stochastic case. This problem has recently received a lot of attention. Several algorithms

30. We recall that a function f is smooth if there exists a constant $L > 0$ such that,

$$\forall x, y \in \mathbb{R}^d, \ \|f(x) - f(y)\|_2 \leq L \|x - y\|_2$$

.

Assumptions	Method	Convergence rate
convex	subgradient descent	$O(\frac{1}{\sqrt{k}})$
convex	projected subgradient	$O(\frac{1}{\sqrt{k}})$
smooth	subgradient descent	$O(\frac{1}{k})$
strongly convex	subgradient descent	$O(\frac{1}{k})$
strongly convex	projected subgradient	$O(\frac{1}{k})$
differentiable	gradient descent	$O(\frac{1}{k})$
differentiable	projected gradient	$O(\frac{1}{k})$
smooth	Frank-Wolfe	$O(\frac{1}{k})$
smooth	accelerated gradient descent	$O(\frac{1}{k^2})$
smooth + strongly convex	accelerated gradient descent	$O(\exp(-k))$

Table 1.1: Some convergence results for batch optimization methods. "convex" stands for functions that are convex with no other assumption, "smooth" for smooth functions and "strongly convex" for strongly convex ones. In this table, all functions are supposed to be convex.

have been designed to try to bridge this gap, by using the specific structure of machine learning problems. State-of-the-art achieves exponential rates for smooth and strongly convex functions:

1. The stochastic average gradient descent (SAG) [Roux et al., 2012] that keeps track of the gradients previously seen to build a more robust stochastic estimate of it. Some variants have been recently designed [Defazio et al., 2014]. They are as fast as SAG but their theoretical analysis is easier.
2. The stochastic variance reduced gradient method (SVRG) [Johnson and Zhang, 2013] achieves the same fast rates as SAG. The idea is to adjust the bias/variance tradeoff of the gradient estimate to converge faster to the optimal value.
3. The stochastic descent algorithm (SDCA) [Shalev-Shwartz and Zhang, 2013] is built explicitely on the assumption of strong convexity [31]. The idea is thus to solve the problem in the dual. Then, the authors manage to derive an efficient coordinate ascent technique in the dual. This method also has the fast rates of SAG and SVRG.

31. On the opposite, SAG or SVRG can easily be used in non-strongly convex cases but without the guarantee of achieving a fast convergence rate.

Assumptions	Method	Convergence rate
convex	subgradient descent	$O(\frac{1}{\sqrt{k}})$
smooth	subgradient descent	$O(\frac{1}{\sqrt{k}})$
convex	projected subgradient	$O(\frac{1}{\sqrt{k}})$
strongly convex	subgradient descent	$O(\frac{1}{k})$
strongly convex	projected subgradient	$O(\frac{1}{k})$
smooth	accelerated gradient	$O(\frac{1}{\sqrt{k}})$
smooth + strongly convex	projected subgradient	$O(\frac{1}{k})$
smooth + strongly convex	SAG/SVRG/SDCA	$O(\exp(-k))$

Table 1.2: Some convergence results for stochastic methods. In this table, all functions are supposed to be convex.

Chapter 2

Tractable Multi-label Classification Using Quadratic Priors

Abstract

In this chapter, we address the problem of multi-label classification which can take, for instance, the form of the automatic tagging of images. We propose to learn a prior over the space of labels to directly improve the performance of these methods. This prior takes the form of a quadratic function of the labels indicator vector and is able to encode both attractive and repulsive relations between labels. We cast this problem as a structured prediction one, aiming at optimizing either the accuracies of the predictors or directly the F_1-score. This leads to an optimization problem closely related to the max-cut problem, which leads to semidefinite and spectral relaxations. Experiments on standard benchmarks demonstrate the benefits of the proposed approach.

This work is based on our preprint: Semidefinite and Spectral Relaxations for Multi-Label Classification, R. Lajugie, P. Bojanowski, S. Arlot, F. Bach [1].

2.1 Introduction

Multi-label classification is an extension of multiclass classification where several labels can be affected to the same example [Tsoumakas and Katakis, 2007, Zhang and Zhou, 2013]. This setting is ubiquitous in real-world applications. For example, it can take the form of image or text tagging, where the goal is to assign instances to categories [Joachims, 1998]. One also can think to a similar problem to the one of Xiao et al. [2010], who proposes to consider the problem of labelling scenes in which several objects appear. The goal is there to predict which objects, among several possible are present in the image. In the multi-label setup we consider a label set $\mathcal{V}$ of cardinal V. Multi-label classification techniques aim at

1. Available on arXiv: http://arxiv.org/abs/1506.01829.

predicting a subset of $\mathcal{V}$. Namely they perform a classification task over the set of subsets of $\mathcal{V}$, that we denote $2^{\mathcal{V}}$. Elements of this set are called *labellings*. One of the main difficulties of this problem lies in the fact that the space of potential labellings $\mathcal{Y}$ is exponentially bigger than the set of labels $\mathcal{V}$. Doing an exhaustive search over the space of labellings is thus not tractable. A first approach that is broadly used consists in training one classifier per possible label and to predict independently the presence or the absence of each label depending of the scores given by classifiers. This setting is often referred to as the *binary relevance* approach [Tsoumakas and Katakis, 2007, Zhang and Zhou, 2013]. This approach, however, does not take into account the structure of the space of potential labellings $\mathcal{Y} = 2^{\mathcal{V}}$. Indeed, some labels might have affinities or repulsions. Moreover these affinities or repulsions can be induced by some context or latent variables. For instance, in image tagging, if it is very likely to see a zebra and a lion in the same image, it is rather not probable to see a reindeer with a lion. A prior over labels could have, for instance, penalized the joint prediction of a reindeer with the lion. On the opposite, if a zebra and a lion are detected in an image, it is very likely that the context label "savana" should be detected as well. Thus labels can have *affinities* or *repulsions*.
This intuition suggests it is possible to go beyond the simple approach of training independently V classifiers.
Including structure into the label set can be done a priori by assuming labels are organized in a certain hierarchy [Rousu et al., 2006, Bi and Kwok, 2011, Deng et al., 2014]. Hariharan et al. [2010] proposes to use prior knowledge when training the classifiers, leading to the learning of correlated classifiers. However this prior is not used at testing time and thus does not affect the way predictions are done on new data.

Our goal is to learn a prior over labels directly from data, at the same time that classifiers are learned. This idea has already been tackled by Petterson and Caetano [2011] who restricted their study to the specific case of *positive affinities* between labels. We go beyond this approach and propose a model allowing to take into account affinities and *incompatibilities* between labels.

Related work. A large part of the recent literature considers the case where the number of different labels $v \in \mathcal{V}$ is moderately large (order of hundreds). In this setting it is possible to learn a specific classifier for each label separately but the space of labellings $\mathcal{Y}$ (of cardinal 2^V) is too large to considering the learning of one classifier for each labelling. One way to train such classifiers is the well-known one-versus-rest technique (a.k.a. binary relevance technique [Tsoumakas and Katakis, 2007]).

Within this setting, some approaches make use of the structured prediction framework of Taskar et al. [2003], Tsochantaridis et al. [2005] as we do. This corresponds to consider the task of prediction as a task over the huge output space $\mathcal{Y}$. For instance, Lampert [2011], Dembczynski et al. [2013] have proposed to plug a structured prediction model to directly optimize a multi-label performance measure within a structured SVM, but consider the prior knowledge between labels as fixed a priori, whereas we aim at learning it. The loss used by Lampert [2011] is

called the "max loss" and is closely related to what we call the Hamming loss in this chapter, whereas Dembczynski et al. [2013] optimize what we call the F_1-loss, which is a tradeoff between precision (number of labels correctly predicted) and recall (number of labels correctly retrieved). They manage to tackle the standard bottlenecks of structured SVMs presented in chapter 1 such as loss-augmented decoding, avoiding an exhaustive search over the power set $\mathcal{Y}$.
Another trend in the recent literature dealing with multi-label classification is to consider the case where the space of labels $\mathcal{V}$ itself is huge [Hsu et al., 2009, Bi and Kwok, 2013]. In these papers, the goal is to use the fact that only few labels are present in an instance. This allows to reduce the dimension of the prediction space and reduces the multi-label problem as a prediction problem over a low dimensional space. The approach we propose in this chapter could be combined with these approaches.

Contributions. The contributions of this chapter are four-fold:

1. we propose a model with priors for multi-label classification allowing attractive as well as repulsive weights,
2. we cast the learning of this model into the framework of structured prediction using either Hamming of F_1 losses and propose an approach for solving exactly the loss-augmented decoding using the F_1 loss,
3. we propose semidefinite and spectral relaxations to efficiently solve the resulting structured prediction problem,
4. we show on real datasets how the learning of such a general prior can improve the multi-label prediction over the models where no prior is learnt or when only attractive weights are allowed.

2.2 Structured prediction for multi-label classification

In this section, we review several classical ways to perform the multi-label classification task when a prior over the labels is fixed. In this chapter, following the terminology introduced in Chapter 1, we refer to the *decoding* as the task of assigning potentially several labels to a data point belonging to some feature space. We then discuss how to learn the parameters of the predictive function. In the rest of the chapter we denote our feature space by $\mathcal{X} \subset \mathbb{R}^p$.

2.2.1 The multi-label classification problem

Let us consider a set of possible labels $\mathcal{V}$ of cardinal V. The set of *labellings*, can be identified to the set of binary vectors $\mathcal{Y} = \{-1, 1\}^V$. This set $\mathcal{Y}$ is the one on which we perform our structured prediction.

Let us assume that for each possible label v, we are given a linear classifier parametrized by $w_v \in \mathbb{R}^p$. We denote by $W \in \mathbb{R}^{p \times V}$, the matrix whose columns

are the vectors w_v. In the multi-label setting, the decoding problem is often cast as:

$$\widehat{y}(x; W) \in \operatorname*{argmax}_{y \in \mathcal{Y}} D(x, y; W) := y^\top W^\top x. \tag{2.1}$$

This is usually referred to as the binary relevance method for multi-label learning [Tsoumakas and Katakis, 2007].
The aforementioned approach does not take into account any dependency between the different labels. A way to do so is to consider a decoding model which is a sum of a term corresponding to independent predictions and one that couples the labels. The latter term can be seen as a penalty F depending on the subset of predicted labels and the input data. The decoding in this chapter is thus the task of finding

$$\widehat{y} \in \operatorname*{argmax}_{y \in \mathcal{Y}} D(x, y; W, F), \tag{2.2}$$

where

$$D(x, y; W, F) = y^\top W^\top x - F(y, x).$$

However, not all functions F are admissible. We can only use the ones that lead to a tractable decoding in Eq. (2.2) (we recall that $|\mathcal{Y}| = 2^V$).

Penalization functions. A class of penalization functions F that is well-suited for our problem is the class of submodular functions [Petterson and Caetano, 2011, Bach, 2013].
When F is submodular, the decoding becomes the maximization of a supermodular function, that is the maximization of a modular [2] minus a submodular function. This is known to be tractable, and solvable in polynomial time in V [Fujishige, 2005]. In this chapter, we focus on a specific family of submodular functions: the ones based on graph-cuts.

Graph-cut based penalty. Following Petterson and Caetano [2011] we first propose in this chapter a graph-cut based penalty F. Such a penalty is typically submodular although we do not need to use this general fact to deal with this penalty. The idea is to consider an undirected graph G whose V nodes are the individual labels. Between each node, there is an edge with a weight. These weights can be included in a matrix C with non-negative entries such that $C_{i,j}$ is the weight of the edge between i and j. The matrix C is the adjacency matrix of the graph.
It is often more convenient to consider the Laplacian matrix associated to the graph,

$$A = \operatorname{Diag}(C^\top \mathbf{1}_V) - C.$$

Due to its spectral properties [3], this matrix is used for spectral clustering or graph based image segmentation [Shi and Malik, 1997, Von Luxburg, 2007].
A labelling y can be seen as splitting the set of labels in two subsets: the set of

2. Any linear function of y is a modular function.
3. This matrix is positive semidefinite since it is a diagonal dominant matrix.

labels present $\mathcal{P} = \{i, y_i = 1\}$ and the one of labels not present $\mathcal{N} = \{i, y_i = -1\}$. The graph cut penalty of Petterson and Caetano [2011] is defined by:

$$\sum_{i \in \mathcal{P}, j \in \mathcal{N}} C_{i,j}.$$

As underlined by Bach [2013], such a cut is submodular as a function of the binary variable y and can be written using only the Laplacian matrix and the vector $y \in \{-1, 1\}^V$ that represents the labelling as:

$$y^\top A y.$$

Adding a linear term in y does not change the fact that the decoding is tractable since such a term is submodular as noted by Bach [2013]. From a graph-cut point of view, it corresponds to adding weights not only on edges but also on vertices. These graphs are often considered in computer vision [Greig et al., 1989, Boykov and Kolmogorov, 2004]. Thus we can consider penalty functions of the form:

$$F(y) = y^\top A y - y^\top d,$$

where $d \in \mathbb{R}^V$ and A has negative off-diagonal entries. We do not impose any constraint on the diagonal terms since these are constant for all $y \in \{-1, 1\}^V$ ans our major concern is to minimize this penalty with respect to such a binary y.

General quadratic penalties. Laplacian matrices are only defined for non-negative weights, that is, for A with non-positive off-diagonal entries. However, penalties of this form are unsatisfactory in practice because they do not let us make use of repulsions between labels. In this chapter, we are also interested in the case where the matrix A can be any matrix, yielding to considering penalties of the form

$$F(y) = y^\top A y - y^\top d$$

where $d \in \mathbb{R}^V$ and $A \in \mathbb{R}^{V \times V}$.

Like in the previous paragraph, such a matrix would have a graph interpretation as encoding not only affinities (the negative off-diagonal entries) between the labels but also repulsions (the positive off-diagonal entries). When A has positive off-diagonal entries, we can write

$$A = A_+ - A_-$$

where

$$A_+ = \max(A, 0)$$

pointwise and

$$A_- = \max(-A, 0).$$

Thus the decoding problem can be cast as maximizing over y:

$$y^\top W^\top x - y^\top A_+ y + y^\top A_- y.$$

As mentioned earlier, we do not impose any constraint on the diagonal of A. In the previous paragraph $y^\top A_+ y$ was equal to 0. But now it is not. The problem of

$$\underset{y \in \{-1,1\}}{\text{minimize}}\, y^\top A_+ y$$

is known as the max-cut problem, see e.g, Eq. (8) of Aspremont and Boyd [2003]. This problem is NP-hard [Johnson, 1982]. It is even APX-hard, meaning that it is even hard to approximate [Papadimitriou and Yannakakis, 1988]. In top of that, if the unique games conjecture is true, it is not possible to get a better approximation ratio for this problem than 0.878 [Khot et al., 2007]. This ratio is reached by the approximation of Goemans and Williamson [1995]. In Sec. 2.5.2 we review this convex relaxation as well as a cheaper one to approximate this problem.

Using these general quadratic penalties, our decoding becomes:

$$\widehat{y}(x; W, A, b) \in \underset{y \in \mathcal{Y}}{\text{argmax}} \left\{ y^\top W^\top x + y^\top d - y^\top A y \right\}. \tag{2.3}$$

As mentioned in the two previous paragraphs, when A has all its off-diagonal entries non-positive, such a function can be optimized in polynomial time. We detail the optimization procedure in Sec. 2.5. In the general case, the decoding task is as hard as a max-cut problem. We provide in Sec. 2.5 efficient relaxations to deal with this. For now, let us focus on how to learn the parameters W, A and d.

2.2.2 Learning the parameters W, d and A

In the previous section we have assumed that we are given V linear classifiers

$$w_v \in \mathbb{R}^p, v \in \{1, \dots, V\},$$

a linear prior $d \in \mathbb{R}^V$ and a matrix $A \in \mathbb{R}^{V \times V}$. Thus, the discussed decoding problem can be seen as parametrized by W, d and A. Its difficulty depends on the subset $\mathcal{A}$ of $\mathbb{R}^{V \times V}$ in which A lies.

Suppose that we are given N examples $(x^i, y^i) \in \mathcal{X} \times \mathcal{Y}, i = 1, \dots, N$, and consider a loss function between two labellings $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$. Ideally, given this loss, we would like to minimize the following regularized empirical loss over $W \in \mathbb{R}^{p \times V}$, $A \in \mathcal{A}$ and $d \in \mathbb{R}^V$. For now, we do not precise the optimization set $\mathcal{A} \subset \mathbb{R}^{V \times V}$ in which A lies. We consider the minimization of

$$\frac{1}{N} \sum_{i=1}^{N} \ell\big(\widehat{y}(x^i; W, A, d), y^i\big) + \lambda \Omega(W, A), \tag{2.4}$$

where Ω is a convex regularizer [4] over the parameter space. This is a hard combinatorial problem that thus needs to be relaxed. Following Taskar et al. [2003], Tsochantaridis et al. [2005], Nowozin and Lampert [2011], we define the structural hinge loss H as:

$$H(x^i, y^i, W, A, d) = \max_{y \in \mathcal{Y}} \big\{ \ell(y, y^i) + D(x^i, y; W, A, d) - D(x^i, y^i, W, A, d) \big\}.$$

Finding the maximizer y, i.e, finding

$$y \in \operatorname*{argmax}_{y \in \mathcal{Y}} \big\{ \ell(y, y^i) + D(x^i, y; W, A, d) - D(x^i, y^i, W, A, d) \big\}$$

in the aforementioned definition is what we refer to as the *loss-augmented decoding*. Note that this loss is a generalization of the standard hinge loss as we have seen in Chapter 1. Indeed, one can notice that if we take

$$\mathcal{Y} = \{-1, 1\},$$

this loss reduces to the standard hinge loss of SVMs up to a rescaling of the data. We estimate parameters W^*, d^* and A^* by solving the following problem:

$$\operatorname*{minimize}_{W \in \mathbb{R}^{V \times p}, A \in \mathcal{A}, d \in \mathbb{R}^V} \frac{1}{N} \sum_{i=1}^{N} H(x^i, y^i; W, A, d) + \lambda \Omega(W, A). \tag{2.5}$$

2.3 Performance measures and losses for multi-label tasks

In order to set up our problem, we need to define a suitable loss function ℓ. Following the classical approaches to multi-label classification, several choices are possible [Tsoumakas and Katakis, 2007, Petterson and Caetano, 2011, Zhang and Zhou, 2013]. Let us review two of the most widely used losses.

Normalized Hamming loss. The simplest loss we can think about is based on accuracy, which is defined as:

$$a(y, y^i) = \frac{V + y^\top y^i}{2V} \in [0, 1],$$

The loss associated to accuracy is the so-called Hamming loss [Hamming, 1950, Zhang and Zhou, 2013]. It is defined as a linear function of the binary label vector

4. In this Chapter we always consider a weighted version of the squared ℓ_2-norm, that allows us to weight differently the parameters A and W but keep in mind that some others can be used for instance depending on some sparsity patterns that are wanted. Note also that in practice, we do not regularize the bias of the model b as commonly done [Hastie et al., 2009].

y by:

$$\ell(y, y^i) = \frac{1}{2V}\left(V - y^{i\top} y\right) = 1 - a(y, y^i) \in [0, 1], .$$

Since y is in $\{-1, 1\}^V$, we have the following equivalent formulation:

$$\ell(y, y^i) = \left\| \frac{1}{2\sqrt{V}}(\mathbf{1} - y) - \frac{1}{2\sqrt{V}}(\mathbf{1} - y^i) \right\|_2^2,$$

where $\mathbf{1}$ is the V-dimensional vector with ones. This loss is proportional to the cardinality of the symmetric difference between two sets

$$A \Delta B = (A \cup B) \setminus (A \cap B).$$

Note also that, if we consider that not all the errors are equivalent, one can use a weighted version instead.

F_1 loss. A common choice in the multi-label learning literature is the F_β loss [Tsoumakas and Katakis, 2007, Petterson and Caetano, 2011]. Let us introduce the two performances measures on which the F_β score is based: the precision p and the recall r. These are respectively defined, with respect to a training labelling $y^i \in \mathcal{Y}$ as:

$$p = \frac{(1 + y^i)^\top (1 + y)}{(1 + y)^\top (1 + y)}, \; r = \frac{(1 + y^i)^\top (1 + y)}{(1 + y^i)^\top (1 + y^i)}.$$

Then, for every $\beta > 0$, the general F_β score is defined as

$$F_\beta(y, y^i) = \frac{(1 + \beta^2)\; p\; r}{\beta^2\; (p + r)} \in [0, 1]\,.$$

The most widely used is the F_1 score (which turns out to be the harmonic mean of precision and recall), and the associated loss is $\ell(y, y^i) = 1 - F_1(y, y^i)$. More precisely:

$$\ell(y, y^i) = \frac{V - y^\top y^i}{2V + y^{i\top} \mathbf{1} + y^\top \mathbf{1}} \in [0, 1]\,. \tag{2.6}$$

Note the non linear dependency of this loss in y.

This loss is a function of precision and recall and has some important advantages over the Hamming loss. In the common situations where each instance has only few labels among all the ones that are possible, the F_β loss penalizes a lot the solution $(-1, \ldots, -1)^\top$ while the Hamming does not [5].

5. For instance, with five possible labels, the Hamming loss between $(-1, 1, -1, -1, -1)$ and $(1, -1, -1, -1, -1)$ is $\frac{2}{5}$ whereas the F_1 loss of Eq. (2.6) is 1.

2.4 Loss-augmented decoding

We propose to derive a structured-SVM-like optimization objective, following Tsochantaridis et al. [2005]. As mentioned earlier, we want to learn the parameters of our predictive function using annotated data. Following the definition of H, we can write the complete optimization problem (2.5) as:

$$\underset{W\in\mathbb{R}^{V\times p},A\in\mathcal{A},b\in\mathbb{R}^V}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N}\Big[\max_{y\in\mathcal{Y}}\big\{\ell(y^i,y)+y^\top W^\top x^i+y^\top d-y^\top Ay\big\} \\ -y^{i\top}W^\top x^i-y^{i\top}d+y^{i\top}Ay^i\Big]+\frac{\lambda_W}{2}\|W\|_2^2+\frac{\lambda_A}{2}\|A\|_2^2. \quad (2.7)$$

Note that the objective function of the optimization is jointly convex in A and W but not smooth.

To solve this convex optimization problem, we need to be able to solve the inner sum maximization, that is the loss-augmented decoding. Let us now see what these problems become when we consider the two losses introduced so far.

Using the Hamming loss. If we use the Hamming loss for ℓ, i.e.,

$$\ell(y^i,y)=\frac{1}{2V}\left(V-y^\top y^i\right),$$

then the loss augmented decoding problem is equivalent to:

$$\max_{y\in\mathcal{Y}}\Big\{y^\top\big(W^\top x^i+d-\frac{1}{2V}y^i\big)-y^\top Ay\Big\}. \quad (2.8)$$

This is the maximization of a quadratic function in y.

Using the F_1 loss. If in turn we decide to use the F_1 loss, the loss-augmented problem becomes harder because of the vector y in the denominator. For one training instance the loss augmented decoding amounts at solving:

$$\underset{y\in\mathcal{Y}}{\text{maximize}}\Big\{\frac{V-y^\top y^i}{2V+y^{i\top}\mathbf{1}+y^\top\mathbf{1}}+y^\top W^\top x^i+y^\top d-y^\top Ay\Big\}. \quad (2.9)$$

To derive a unified view over the loss-augmented decodings using the F_1 and the Hamming loss, let us now reduce this loss-augmented decoding problem as a sequence of maximization of quadratic problems over the labellings y. We define the set $\mathcal{Y}_k$ as the set of labellings such that k entries are positive:

$$\forall k\in\{0,\dots V\},\ \ \mathcal{Y}_k=\Big\{y\in\mathcal{Y}\ ,\ y^\top\mathbf{1}=2k-V\Big\}.$$

As is often done when optimizing the F_1 score, which is a contingency-table based loss [Joachims, 2005], we can divide the initial problem into $V+1$ subproblems by

replacing $y^\top \mathbf{1}$ by $2k - V$. To this end we define the function $\zeta_i(y, k)$ as follows:

$$\zeta_i(y,k) = y^\top \Big(\frac{y^i}{V + y^i \mathbf{1} + 2k} + W^\top x^i + d \Big) - y^\top A y. \tag{2.10}$$

The loss-augmented decoding problem is then equivalent to:

$$\max_{k \in \{0,\dots,V\}} \Big[\frac{V}{V + y^i \mathbf{1} + 2k} + \max_{y \in \mathcal{Y}_k} \zeta_i(y,k) \Big]. \tag{2.11}$$

That way we have cast the latter decoding as the maximization of a finite sequence of quadratic problems in y.

Solving the loss-augmented decoding. The loss-augmented decoding problems of Eq. (2.8) and Eq. (2.9) as well as the decoding problem of Eq. (2.3) require us to be able to solve quadratic optimization problems for $y \in \mathcal{Y}$ under the potential constraint that $y \in \mathcal{Y}_k$. This heavily depends on the sets $\mathcal{A}$ to which A belongs. In the following we consider two different cases:

- when A belongs to a set $\mathcal{A}$ that is made of matrices with non-positive off-diagonal entries, that is when A is the Laplacian of a graph, we show that quadratic minimization problems can be solved using graph cuts and that the loss augmented decoding problem can be solved *exactly* for the Hamming loss and *approximately* for the F_1-loss,
- in the general case, that is when $A \in \mathbb{R}^{V \times V}$, we propose an efficient way to relax the loss-augmented decoding, and solve *exactly* this relaxation.

In the following section, we propose relaxations of these problems leading to a tractable loss-augmented decoding with no restriction over the matrix A.

2.5 Optimization in y

So far, we have written problems that we may not be able to solve efficiently. The first one is the general decoding of Eq. (2.3). The other ones are the subproblems of Eq. (2.8) and Eq. (2.11). All of these are quadratic boolean optimization problems and are closely linked to the max-cut problem [6]. These can be written in the following canonical form:

$$\underset{\substack{u \in \{-1,1\}^V \\ L(u)=0}}{\text{maximize}} \; u^\top b - u^\top A u, \tag{2.12}$$

where $A \in \mathbb{R}^{V \times V}$, $b \in \mathbb{R}^V$ and L is an affine function:

$$L(u) = u^\top \alpha - \beta,$$

6. See, e.g., Boyd and Vandenberghe [2004, Sec. 5.1.5]

where $\alpha \in \mathbb{R}^V$ and $\beta \in \mathbb{R}$. The additional constraint $L(u) = 0$ is only needed for the problem mentioned in Eq. (2.11). For the two other problems, one can simply ignore it. Eq. (2.12) allows us to tackle three problems in a unified framework.

2.5.1 First case: A has negative off-diagonal entries

We first investigate the case where A has negative off-diagonal entries. This is exactly the case considered by Petterson and Caetano [2011]. However our approach is slightly different since, for the F_1-loss, we propose an *exact* technique to solve the loss-augmented decoding.

Let us now see that the problem of Eq. (2.11), in the case where there is no linear constraint, is a variant of the min-cut problem which is known to be solvable in polynomial time using a min-cut/maximum-flow algorithm [Edmonds and Karp, 1972].

Links with graph-cuts Let us consider a graph G with V nodes $\mathcal{V} = \{1, \dots, V\}$ and a set of edges $\mathcal{E}$ that is a collection of pairs

$$e = (v_1, v_2) \in \mathcal{V} \times \mathcal{V},$$

where $v_1 \neq v_2$. We assume that each edge is associated to a weight c_{v_1,v_2}. The adjacency matrix C of the undirected graph G is thus the matrix such that $C_{v_1,v_2} = c_{v_1,v_2}$ if there exists an edge, meaning that $(v_1, v_2) \in \mathcal{E}$. Moreover, we assume that each node of the graph is associated to a real-valued capacity that is encoded in a *capacity vector* $c \in \mathbb{R}^V$.

The canonical min-cut problem [Ford and Fulkerson, 1956, Edmonds and Karp, 1972, Cormen et al., 2001, Boykov and Kolmogorov, 2004, Bach, 2013] can be written as the following optimization problem:

$$\operatorname*{minimize}_{z \in \{0,1\}^V} \sum_{j=1}^{V} \sum_{i=1}^{j-1} C_{i,j}|z_i - z_j| + c^\top z, \tag{2.13}$$

where $C \in \mathbb{R}_+^{V \times V}$ and $c \in \mathbb{R}^V$. By making the change of variables $z = \frac{y+1}{2}$ and carrying on some calculations, we get the following equivalent min-cut problem:

$$\operatorname*{minimize}_{y \in \{-1,1\}^V} 2y^\top c - y^\top C y. \tag{2.14}$$

The key observation is that these optimization programs over discrete sets can be relaxed on the convex hulls of the optimization sets, by considering their Lovasz extension (see e.g, Bach [2013]). That is, the min-cut problem is equivalent to solving the following convex optimization programs over $[0, 1]^V$:

$$\operatorname*{minimize}_{z \in [0,1]^V} \sum_{j=1}^{V} \sum_{i=1}^{j-1} C_{i,j}|z_i - z_j| + c^\top z,$$

or $\overline{\{-1,1\}^V} = [-1,1]^V$:

$$\underset{y\in[-1,1]^V}{\text{minimize}}\ 2y^\top c - y^\top C y. \tag{2.15}$$

Note also that when the matrix A has negative off-diagonal entries, the problem formulated in Eq. (2.12) with no linear constraints can be solved using min-cut / max-flow with standard toolboxes by providing the matrix $C = -A$ and $c = \frac{1}{2}b$.

Approximately solving the constrained problem of Eq. (2.12) **with graph cuts.** When dealing with the problem of Eq. (2.12) with a linear constraint in the variable $u \in \overline{\mathcal{Y}}$, a classical trick to reduce this problem to an unconstrained one is to dualize the constraint $L(u) = 0$, that is to introduce a Lagrange multiplier associated to it [Lagrange, 1811, Boyd and Vandenberghe, 2004]. Thus the constrained problem of Eq. (2.12) is equivalent to:

$$\underset{u\in[-1,1]^V}{\text{minimize}}\ \max_{\mu\in\mathbb{R}} u^\top A u - u^\top b + \mu L(u). \tag{2.16}$$

This problem is still hard to solve. However, we can relax it by switching the min and the max:

$$\underset{\mu\in\mathbb{R}}{\text{maximize}}\ \min_{u\in[-1,1]^V} u^\top A u - u^\top b + \mu L(u). \tag{2.17}$$

Note that the optimal value of the problem of Eq. (2.17) is always a lower bound to the optimal one of the problem of Eq. (2.16), this is known as weak duality [Bertsekas, 1999, Boyd and Vandenberghe, 2004]. When strong duality holds, for instance when the problem is convex, we have equality of the optima. However, for arbitrary A the strong Lagrangian duality theorem does not hold thus solving problem of Eq. (2.17) only yields to an approximate problem of the constrained problem of Eq. (2.12).

Problem (2.17) is the maximization of a unidimensional function of μ whose evaluation is a graph-cut. It can be done using the dichotomy method for instance [Boyd and Vandenberghe, 2004]. Such a method is described in Alg. 4. At the optimum μ^*, the u that

$$\underset{u\in[-1,1]^V}{\text{minimize}}\ u^\top A u - u^\top b + \mu(\beta - u^\top \alpha)$$

is exactly the solution of the relaxed problem of Eq. (2.17) we are looking for.

Other approach to solve the constrained problem of Eq. (2.12) **for negative** A**.** In this paragraph, we show how we can solve the constrained problem of Eq. (2.17) by relating it to the well-studied total variation denoising problem [Chambolle and Darbon, 2009, Bach, 2013].
Here, we consider that the constraint of Eq. (2.12) is simply a cardinality con-

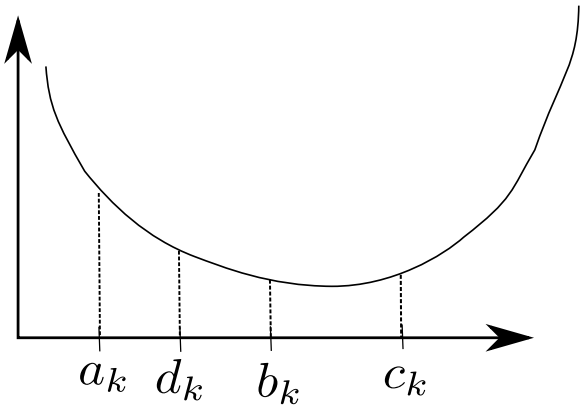


Figure 2.1: An illustration of the key observation of the dichotomy. Among the four points that are kept during Alg. 4 to minimize a convex function, there will always be three among $\{a_k, b_k, c_k, d_k\}$, let us say (u, v, w), in such a way that (i) $u < v < w$, (ii) $f(u) > f(v)$ and $f(v) < f(w)$ and $w - u < c_k - a_k$.

straint [7], namely that the constraint is of the form $u^\top \mathbf{1}_V = \beta$ for a certain $\beta = 2k - V$ and $k \in \{1 \dots V\}$. We start from Eq. (2.17).

By considering the variable $z = \frac{1+u}{2} \in [0,1]^V$ we get that problem (2.17) is equivalent to the following problem:

$$\underset{\mu \in \mathbb{R}}{\text{maximize}}\, \mu(-\beta - V) + \min_{z \in [0,1]^V} \left\{ 4z^\top A z - z^\top (4A\mathbf{1}_V + 2b) + 2\mu z^\top \mathbf{1}_V \right\}. \tag{2.18}$$

The problem of Eq. (2.18) is a separable submodular optimization problem [Bach, 2013]. Thus, solving it can be done by considering the associated proximal problem. More precisely, if we introduce the Choquet integral [8] of the cut $J(u)$ (which is in this case often referred to as the "co-area formula" [Chambolle and Darbon, 2009]), the generic proximal problem associated to any cut problem is:

$$\underset{u \in \mathbb{R}^V}{\text{minimize}}\, \frac{1}{2}\|u - g\|_2^2 + J(u). \tag{2.19}$$

where g in our case is exactly $4A\mathbf{1}_V + 2b$.

This problem is the well known total variation denoising problem. There exists several efficient algorithms to deal with it, especially the ones relying on parametric max-flow techniques. Once problem (2.19) has been solved and that we recovered its solution u^* [9], we get all the candidates for being a solution of (2.18) by considering the families of indicator vector $(\mathbb{1}_{u \geq -\mu})_{u \in \mathbb{R}}$ [Bach, 2013]. This family is finite and has V elements at most since u is of dimension V and μ is a real. Then, we just have to compute the V values associated to the objective function

7. This is always the case in this manuscript.
8. Or Lovasz extension, see e.g, Bach [2013].
9. Note that this solution is unique if α is non-negative [Chambolle and Darbon, 2009].

Algorithm 4 Binary search algorithm to find the optimum of a convex function f on $\mathbb{R}$. For simplicity, we assume the function to have a unique optimum x^*. The key of the approach is that it is always possible to do the *core step*. For an intuition of it, have a look at Fig. 2.1.

Input Target precision ϵ, function f
Find a segment $[a_0, c_0]$ such that $x^* \in [a_0, c_0]$
Find $b_0 \in (a_0, c_0)$ such that $f(b_0) < f(c_0)$ and $f(b_0) < f(a_0)$
$d_0 = \frac{a_0+b_0}{2}$,
while $c_k - a_k > \epsilon$ **do**
  core step: Find $a_{k+1} < b_{k+1} < c_{k+1}$ among $\{a_k, d_k, b_k, c_k\}$ such that (a) $f(a_{k+1}) > f(b_{k+1})$, $f(b_{k+1}) < f(c_{k+1})$ and (b) $c_{k+1} - a_{k+1} < c_k - a_k$
  $k = k + 1$
end while
Output: b_k

and pick the best one. Note however that we did not use this approach yet in our experiments.

2.5.2 Classical semidefinite relaxation for max-cut

Now, let us consider the general case where A can be any matrix.

The family of problems presented in Eq. (2.12) is known as the two-way partitioning problems. They are a generalization of the max-cut problem, with potentially negative entries in A. This problem is known to be NP-complete [Johnson, 1982].

There exists a classical semidefinite relaxation that is known to reach the optimal approximation ratio if $P \neq NP$ and the unique games conjecture is true [Khot et al., 2007]. Let us describe it. Following Boyd and Vandenberghe [2004] or Aspremont and Boyd [2003], we use the relaxation introduced by Goemans and Williamson [1995] to approximate the max-cut problem. We introduce a new variable

$$U = uu^\top \in \mathbb{R}^{V \times V},$$

that we refer to as the *equivalence matrix* associated to the labelling $u \in \mathcal{Y}$. Using this notation we can re-write the term $u^\top Au$ as $\mathrm{Tr}\,(AU)$. Then, using a set of constraints that is equivalent to $U = uu^\top$, the problem (2.12) can be cast as:

$$\underset{\substack{u \in \mathcal{Y} \\ U \in \mathbb{R}^{V \times V}}}{\text{maximize}} \quad u^\top b - \mathrm{Tr}(AU) \quad \text{such that} \quad \begin{cases} \mathrm{Diag}(U) = \mathbf{1}, \\ \mathrm{Rank}(U) = 1, \\ U \succeq uu^\top, \\ L(u) = 0. \end{cases} \tag{2.20}$$

Following Boyd and Vandenberghe [2004], the convex relaxation of this problem is obtained by removing the rank constraint. We use the Schur complement trick

(see, e.g., Boyd and Vandenberghe [2004]) and define a matrix M as:

$$M = \begin{pmatrix} U & u \\ u^\top & 1 \end{pmatrix}.$$

Using e_{V+1}, the $V+1$-th vector of the canonical basis of $\mathbb{R}^{V+1}$, our relaxed problem of (2.12) can be re-written as:

$$\underset{M \in \mathbb{R}^{V \times V}}{\text{maximize}} \ \text{Tr}\left[M \begin{pmatrix} -A & \frac{1}{2}b \\ \frac{1}{2}b & 0 \end{pmatrix} \right] \ \text{such that} \ \begin{cases} \text{Diag}(M) = \mathbf{1}, \\ M \succeq 0, \\ \alpha^\top M e_{V+1} = \beta. \end{cases} \tag{2.21}$$

Problem (2.21) can be solved using any standard semidefinite programming solver at least for small V (< 100). When V is large, one can use specific techniques relying explicitly on the fact the solution is expected to be low-rank; see, e.g., Journée et al. [2010] and references therein.

The vector u is not a vector of $\{-1, 1\}^V$ anymore and, at testing time [10] we need to get such a solution. That is why we need a rounding scheme.

Rounding scheme. We follow Boyd and Vandenberghe [2004] to round the relaxed solution, *i.e.*, get back to some admissible solution of (2.12). We notice that for the optimal pair (u, U) solving Eq. (2.21), $U \succeq uu^\top$ implies that $U - uu^\top$ is positive semidefinite and can be seen as a covariance matrix. Therefore, we (i) sample several

$$v \sim \mathcal{N}(u, U - uu^\top)$$

from a normal distribution, (ii) round the solution by taking the signs and choose the best one in terms of the objective function. This procedure leads to good feasible points in our experiments.

2.5.3 Spectral relaxation

Now, let us see another relaxation possible. It is computationally less costly than the SDP relaxation [11]. The generic problem in Eq. (2.12) can be relaxed by replacing the boolean constraint $u \in \mathcal{Y}$ with a quadratic equality $u^\top u = V$. Please note this makes the problem of Eq. (2.12) non-convex. Using the same expression for $L(u)$ as in the previous section leads to the following optimization problem:

$$\underset{u \in \mathbb{R}^V}{\text{maximize}} \ \left\{ u^\top b - u^\top A u \right\} \ \text{such that} \ \begin{cases} u^\top u = V \\ u^\top \alpha = \beta. \end{cases} \tag{2.22}$$

Note that when optimizing the Hamming loss or when we are not subject to

10. That is when the problem of Eq. (2.12) is the decoding of Eq. (2.3).

11. The SDP relaxation typically do not scale when the label set becomes large, of order of hundred. Such an SDP can be solved using interior point methods [Overton and Wolkowicz, 1997].

the constraint $u^\top \alpha = \beta$, this problem is just a generalized eigenvalue problem [12].

Let us see two approaches to solve this problem when the linear constraint is present.

Binary search approach. As in Sec. 2.5.1, we can deal with the linear constraint by dualizing it, yielding the following problem:

$$\underset{\mu\in\mathbb{R}}{\text{minimize}} \left\{ \mu\beta + \max_{\substack{u\in\mathbb{R}^V \\ u^\top u = V}} u^\top (b - \mu\alpha) - u^\top A u \right\}. \tag{2.23}$$

Note that this problem is not equivalent to the original one for the same reason than in Sec. 2.5.1. This can be solved by performing a binary search, a.k.a. a dichotomy method, over μ. Such an algorithm is described in Alg. 4.

The inner loop maximization problem is classical in optimization, in particular in trust-region methods [Forsythe and Golub, 1965, Spjøtvoll, 1972]. It reduces—using the Lagrange multiplier technique—to solving a quadratic eigenvalue problem [Tisseur and Meerbergen, 2001]. Solving the inner maximization problem of Eq. (2.23) is equivalent to finding the minimal eigenvalue of the problem:

$$\left(\lambda^2 \mathrm{Id}_V - 2\lambda A + A^2 - \tfrac{1}{4V}(b - \mu\alpha)(b - \mu\alpha)^\top\right) u = 0, \tag{2.24}$$

where Id_V denotes the $V \times V$ identity matrix. Following [Tisseur and Meerbergen, 2001], the problem above is solved efficiently by performing the eigendecomposition of the matrix S:

$$S = \begin{pmatrix} A & -\mathrm{Id}_V \\ -\frac{1}{4V}(b - \mu\alpha)(b - \mu\alpha)^\top & A \end{pmatrix}. \tag{2.25}$$

Once this is done, we get the desired solution by taking

$$u = \frac{1}{2}(A - \lambda \mathrm{Id}_V)^{-1}(b - \mu\alpha),$$

where λ is the smallest non-zero eigenvalue of S.

So, we are able to evaluate the function

$$\gamma(\mu) = \mu\beta + \max_{\substack{u\in\mathbb{R}^V \\ u^\top u = V}} u^\top (b - \mu\alpha) - u^\top A u.$$

We can thus solve the problem by binary search.

12. This can be seen by admitting the fact that this problem satisfies strong duality although it is non-convex [Boyd and Vandenberghe, 2004]. Thus, we can introduce a Lagrange multiplier for the constraint $u^\top u = V$ and obtain the equivalent unconstrained problem:

$$\underset{u\in\mathbb{R}^V,\, \lambda\in\mathbb{R}}{\text{minimize}}\; u^\top b - u^\top A u - \lambda(u^\top u - V),$$

that can be solved by considering the generalized eigenvalue problem associated to A and b.

However the use of the dichotomy method is costly in practice. We show in Sec. 2.5.4 how we can deal with the problem of Eq. (2.22) in a more efficient way.

2.5.4 Solving the spectral relaxation subject to equality constraint with a single eigenvalue decomposition.

In this section we present an other way to deal with the spectral relaxation, inspired by Gander et al. [1989]. The proposed method is (i) more efficient computationally than the one of the previous section since it gets rid of the binary search problem over the Lagrange multiplier μ and (ii) an *exact* approach.

We start from the problem of Eq. (2.22). By the change of variables $v = \begin{pmatrix} u \\ 1 \end{pmatrix}$ and $B = \begin{pmatrix} -A & b/2 \\ b/2 & 0 \end{pmatrix}$ and by introducing $D = \begin{pmatrix} \mathrm{Id}_V & 0 \\ 0 & 0 \end{pmatrix}$ (Id_V is the V dimensional identity matrix) we can write the problem as:

$$\begin{aligned} \max_{v \in \mathbb{R}^{V+1}} \quad & v^\top B v \\ \text{such that} \quad & v^\top D v = V \text{ and } v^\top \begin{pmatrix} \alpha & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \beta \\ 1 \end{pmatrix}. \end{aligned} \tag{2.26}$$

Following Gander et al. [1989], let us introduce the QR factorization of the matrix

$$\begin{pmatrix} \alpha & 0 \\ 0 & 1 \end{pmatrix},$$

that is

$$Q^\top \begin{pmatrix} \alpha & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R \\ 0 \end{pmatrix},$$

where $Q \in \mathbb{R}^{(V+1)\times(V+1)}$ is an orthogonal matrix and $R \in \mathbb{R}^{2\times 2}$ is an upper triangular matrix, nonsingular since $\begin{pmatrix} \alpha & 0 \\ 0 & 1 \end{pmatrix}$ has full rank. Let us now introduce

$$U = \begin{pmatrix} U_1 \\ U_2 \end{pmatrix} = Q^\top V.$$

Note that $U_1 \in \mathbb{R}^2$ and $U_2 \in \mathbb{R}^{V-1}$. We can think about U_1 as the variables corresponding to the subspace that is "fixed" by the equality constraint.

Eq. (2.26) can be rewritten as:

$$\begin{aligned} \underset{U \in \mathbb{R}^{V+1}}{\text{maximize}} \quad & U^\top Q^\top B Q U \\ \text{such that} \quad & U^\top D U = V \quad \text{and} \quad U_1^\top R = \begin{pmatrix} \beta \\ 1 \end{pmatrix}. \end{aligned} \tag{2.27}$$

Note that the last constraint allows us to set the variable $U_1^\top = R^{-1}\begin{pmatrix}\beta\\1\end{pmatrix}$.

Let us define $Q^T B Q = \begin{pmatrix}\Delta & \Gamma/2\\ \Gamma^\top/2 & C\end{pmatrix}$, with $\Delta \in \mathbb{R}^{2\times 2}$, $\Gamma \in \mathbb{R}^{T\times 2}$ and $C \in \mathbb{R}^{(V-1)\times(V-1)}$. Using the previous notations, we get:

$$U^\top Q^\top B Q U = U_2^T C U_2 + U_1^\top \Gamma U_2 + U_1^\top \Delta U_1.$$

Let us also introduce $S = V - U_1^\top U_1$. Since U_1 is entirely determined, the problem of Eq. (2.26) is equivalent to:

$$\begin{aligned}\underset{U_2\in\mathbb{R}^{V-1}}{\text{maximize}} \quad & U_2^\top C U_2 + R^\top \begin{pmatrix}\beta\\1\end{pmatrix}\Gamma U_2\\ \text{such that} \quad & U_2^\top D U_2 = S.\end{aligned} \tag{2.28}$$

This problem is equivalent to a generalized eigenvalue problem and can be solved in $O\big((2V)^3\big)$.

We slightly abuse notations since D is restricted to its lower corner right $V-1$ corner.

Now, we have all the tools that we need to properly relax the structured prediction objective of Eq. (2.5) and get a solution of it.

2.6 Optimization in W and A

Let us go back to the optimization of Eq. (2.7) with stochastic subgradient descent. We have to consider at least two cases.

First case: A has only non-positive off-diagonal entries. In this case, we have seen that the loss-augmented decoding could be solved efficiently exactly in the case of the Hamming loss, and approximately using the Lagrange multiplier relaxation of Sec. 2.5.1 in the F_1 loss case. This allows us to find the maximizer of the inner sum maximization of Eq. (2.7) *exactly*. That is we are able to compute a element of the subgradient set of our objective function that we call g for simplicity. If y^i is the solution of the loss-augmented decoding for training instance i, such a subgradient is:

$$\begin{aligned}\partial_W g(W, A, d) &= \lambda_W W + \frac{1}{N}\sum_{i=1}^N x^i(\bar{y}^i - y^i)^\top,\\ \partial_d g(W, A, d) &= \frac{1}{N}\sum_{i=1}^N (\bar{y}^i - y^i),\\ \partial_A g(W, A, d) &= \lambda_A A + \frac{1}{N}\sum_{i=1}^N -\bar{y}^i\bar{y}^{i\top} + y^i y^{i\top}.\end{aligned}$$

Second case: A has general values. When we relax the inner optimization problem in $y \in \mathcal{Y}$ we change the optimization set for the loss-augmented decoding.

For the SDP relaxation, we are working on

$$\mathcal{U} = \{(u, U), U \in \mathbb{R}^{V \times V}, u \in \mathbb{R}^V, U \succeq uu^\top, \text{Diag}(U) = \mathbf{1}_V\},$$

this changes the objective function into:

$$\frac{1}{N}\sum_{i=1}^{N}\Big[\max_{u,U\in\mathcal{U}}\Big\{u^\top\Big(W^\top x^i + d - \frac{1}{2V}y^i\Big) - \text{Tr}(AU)\Big\} - y^{i\top}W^\top x^i - y^{i\top}d + y^{i\top}Ay^i\Big] + \frac{\lambda_W}{2}\|W\|_2^2 + \frac{\lambda_A}{2}\|A\|_2^2,$$

for the Hamming loss, and:

$$\frac{1}{N}\sum_{i=1}^{N}\Big[\max_{k\in\{0,\dots,V\}}\max_{(u,U)\in\mathcal{U}_k}\Big\{u^\top\Big(W^\top x^i + d - \frac{y^i}{2V + y^i\mathbf{1} + k}\Big) - \text{Tr}(AU)\Big\} - y^{i\top}W^\top x^i - y^{i\top}d + y^{i\top}Ay^i\Big] + \frac{\lambda_W}{2}\|W\|_2^2 + \frac{\lambda_A}{2}\|A\|_2^2, \tag{2.29}$$

for the F_1 loss. Note that we have defined

$$\mathcal{U}_k = \{(u, U), U \in \mathbb{R}^{V \times V}, u \in \mathbb{R}^V, U \succeq uu^\top, \text{Diag}(U) = \mathbf{1}_V\, u^\top \mathbf{1}_V = 2k - V\}.$$

For the spectral relaxation, the optimization set becomes the sphere of radius V of $\mathbb{R}^V$, i.e, the set $\{u \in \mathbb{R}^V, u^T u = V\}$. The objective function becomes:

$$\frac{1}{N}\sum_{i=1}^{N}\Big[\max_{u,u^\top u=V}\Big\{u^\top\Big(W^\top x^i + d - \frac{1}{2V}y^i\Big) - u^\top Au\Big\} - y^{i\top}W^T x^i - y^{i\top}d + y^{i\top}Ay^i\Big] + \frac{\lambda_W}{2}\|W\|_2^2 + \frac{\lambda_A}{2}\|A\|_2^2, \tag{2.30}$$

and

$$\frac{1}{N}\sum_{i=1}^{N}\Big[\max_{k\in\{0,\dots,V\}}\max_{u\in\mathcal{U}_k}\Big\{u^\top\Big(W^\top x^i + d - \frac{y^i}{2V + y^i\mathbf{1} + k}\Big) - u^\top Au)\Big\} - y^{i\top}W^\top x^i - y^{i\top}d + y^{i\top}Ay^i\Big] + \frac{\lambda_W}{2}\|W\|_2^2 + \frac{\lambda_A}{2}\|A\|_2^2, \tag{2.31}$$

for the F_1 loss.

Once we have solved the loss-augmented decoding and get a pair (u, U) in the SDP case, or a vector u and set the matrix $U = uu^\top$, we get elements of the

subgradient by the following relations:

$$\partial_W g(W, A, d) = \lambda_W W + \frac{1}{N} \sum_{i=1}^{N} x^i (u - y^i)^\top,$$

$$\partial_d g(W, A, d) = \frac{1}{N} \sum_{i=1}^{N} (u - y^i),$$

$$\partial_A g(W, A, d) = \lambda_A A + \frac{1}{N} \sum_{i=1}^{N} -U + y^i y^{i\top}.$$

To summarize, when A has negative off-diagonal entries, we solve approximately the objective of Eq. (2.5) whereas in the general case, we solve exactly the objective of Eq. (2.29)–(2.31).

2.7 Experimental Evaluation

We now assess the performances of the proposed approach on some standard benchmarks.

Baselines. We compare our approaches to the one of Petterson and Caetano [2011] and to an approach that we call *one-versus-rest*. For the latter method:

- during the training step, we use the N training pairs (x^i, y^i) to learn one classifier for each of the V classes,
- during the testing step, we stack all the individual classifiers in a matrix W and we predict using the binary relevance decoding technique of Eq. (2.1).

In this experimental section we first describe the used datasets and discuss the baselines to which we compare them.

2.7.1 Datasets.

We validate our approach on four datasets. Following Petterson and Caetano [2011], we picked our datasets from the *mulan* [13] repository. We picked the *yeast* [Elisseeff and Weston, 2001], *enron*, *medical* [Pestian et al., 2007] and *bibtex* [Katakis et al., 2008] datasets. The datasets are of various sizes and natures: *yeast* only has 14 labels while *bibtex* has 159. All of them also present different challenges (different structures, label occurrence patterns, etc.).

These datasets are given with a *training / testing* split. We further split the training set to generate a validation set. We select all relevant parameters by plain validation on this set. We report all performances on the actual test set as given in the dataset. Characteristics of these datasets are given in Table 2.1.

13. http://mulan.sourceforge.net/datasets.html

	Instances	Features (p)	Labels (V)
yeast	2417	103	14
enron	1702	1001	53
medical	978	1449	45
bibtex	7395	1836	159

Table 2.1: Standard characteristics of the datasets used.

2.7.2 Binary relevance results.

In Table 2.2 we report the performance of the one-versus-rest approach for all the datasets. For every label, we train a linear classifier using a standard SVM toolbox [Fan et al., 2008]. We select the parameters of the SVM method by validation on a held-out part of the training set. We compare three criteria for choosing the optimal set of regularization parameters. We can either select a common regularization parameter for all classes ("Single λ" column), chosen with the Hamming loss (which decouples over classes), or one per class ("Multiple λ" column). When choosing a common λ for all classes, one can choose it according to the F_1 or Hamming loss on the validation set.

	Single λ		Multiple λ
	F_1	Hamming	Hamming
yeast	0.39	0.40	0.54
enron	0.48	0.49	0.46
medical	0.29	0.29	0.28
bibtex	0.61	0.66	0.66

Table 2.2: Linear SVM performance on the considered datasets. We report the average F_1 loss for various schemes for choosing the regularization parameter λ. The first row indicates if validation is performed with one common λ to all classes or if the λ can be different, the second corresponds to the validation performance measure to select this λ.

Table 2.2 shows that on the *bibtex* dataset, it is important to use the relevant loss as a criterion to select hyperparameter. This corresponds to a case where the number of labels V is big. In that case, as discussed in Sec. 2.3, the Hamming loss behaves more and more differently from the F_1 loss.

One would also expect that picking one parameter per label would lead to better performance. But the benefits from selecting a specific parameter per class is offset by the fact that one cannot use the F_1 loss in this case. In all our remaining simulations, we use a single λ for all classes.

2.7.3 Our model and comparison to Petterson and Caetano [2011].

We run our algorithm—with ℓ equal to the Hamming loss—on all four datasets and compare to the available implementation of Petterson and Caetano [2011]. For all methods we select all hyper-parameters based on the performance in terms of F_1 loss on the validation set. Because of the challenging number of labels, for *bibtex*, we were able to run neither the code from Petterson and Caetano [2011], nor the SDP, in reasonable time.

				SDP			Spectral		
	OvR	P. et al.	MC	$A \leqslant 0$	$A \geqslant 0$	Any A	$A \leqslant 0$	$A \geqslant 0$	Any A
yeast	0.39	**0.36**	0.40	0.40	0.39	0.39	0.39	0.37	0.37
enron	0.48	**0.45**	0.47	0.47	0.47	**0.45**	0.48	0.49	0.49
medical	0.29	0.33	0.29	0.31	0.29	0.24	0.30	**0.21**	0.24
bibtex	0.61	N/A	0.61	N/A	N/A	N/A	0.62	**0.57**	0.60

Table 2.3: Comparison between Petterson and Caetano [2011] and different variants of our method. OvR denotes the one-versus-rest approach. MC is our algorithm with the inner loop solved using min-cut / max-flow. SDP is the semidefinite relaxation of the inner loop. Spectral is the spectral relaxation of the inner loop. All methods optimize the Hamming loss except P. et al. which optimizes the F_1 loss. Results are all F_1 loss.

Table 2.3 compares the one-versus-rest approach, the approach described in Petterson and Caetano [2011] and variants of our method. We compare the two relaxations we proposed while optimizing the Hamming loss. Please recall that the min-cut (MC) solution implies that $A \leqslant 0$ (non-positive entries).

When $A \leqslant 0$, we can measure the tightness of the proposed relaxations. We see that the various relaxations, SDP then spectral, do not degrade performances over the exact approach MC (which cannot be run for general A).

We also notice that using a negative matrix A is a strong limitation. The performance observed when A is unconstrained or non-negative is better. This motivates our formulation and shows that repulsive weights between labels are relevant.

	OvR	Our Hamming	Our F_1
yeast	0.39	0.43	0.43
enron	0.48	0.47	0.47
medical	0.29	0.28	0.28
bibtex	0.61	0.60	0.60

Table 2.4: Comparison of F_1 losses when optimizing the F_1 loss versus the Hamming loss.

2.7.4 The Hamming loss and the F_1 loss.

In this experiment we do not make use of the quadratic prior, so $A = 0$. Table 2.4 gives the F_1 loss we obtain by optimizing either the F_1 loss or the Hamming loss. We compare the implementation of the F_1 score minimization in [Petterson and Caetano, 2011] (carried out using a greedy technique). In that table, "Our F_1" is our own implementation of the support vector technique for F_1-loss [Joachims, 2005] using the optimization described in Section 2.4. This is an exact optimization technique. We also report the results obtained by training SVMs, using the one-versus-rest scheme. It appears that, on these standard datasets of medium size ($V \approx 10 - 50$), optimizing the F_1 loss does not yield to better performances than optimizing the Hamming loss.

2.8 Conclusion

We have proposed a framework to learn a prior for improving the performances of multi-label classification tasks. This prior takes the form of a quadratic function over the space of labels and allows us to use both affinities and repulsions. Existing work [Petterson and Caetano, 2011] only takes into account positive affinities between labels. We provide semidefinite and spectral relaxations of the learning problem, yielding to an efficient optimization scheme. In particular the spectral relaxation permits to deal computationally with datasets rather large ($V > 150$) whereas existing algorithms cannot (since the loss-augmented decoding problems have to be solved many times).

It would be interesting to see how it is possible to leverage the range of applicability of the semidefinite relaxations which is, for now, limited to multi-label problems for which V is of the order of hundreds. To that extent, we could use techniques from matrix optimization theory, taking into account the fact that the solution we aim at finding has low rank [Journée et al., 2010].

Recent theoretical advances [Dembczynski et al., 2013] have suggested that including dependence in data directly in the non linear part of the penalty could drastically improve the performance of the multi-label classification task. In our case this can be done pretty easily by considering a non-linear part of the penalty of the form:

$$\sum_i x_i y^\top A_i y,$$

where x_i is some 1-dimensional data dependent feature, for instance the projection on the first dimension obtained by doing a PCA on data.

Chapter 3

Sequential Structures in Real World Data and Mahalanobis Similarity Measure Learning

In this thesis, we focus on problems where *sequential structure* is involved. We consider data that come with a natural order. To fix ideas let us start by presenting two major examples of sequences.

Spatial sequences. The first kind of sequences are *spatial* ones. These can be met in various fields of science. For instance, in computational biology, a very large source of data comes from the study of the number of replications of genes or nucleotides along DNA. They are called arrays of comparative genomic hybridization (aCGH) [Kallioniemi et al., 1992, du Manoir et al., 1995] and are used for the detection or classification of cancers [Schleiermacher et al., 2010, Hocking et al., 2013]. These data have a spatial sequential structure given by the order along the DNA. These data can thus been represented as a vector $x \in \mathbb{R}^T$ where x_i is the number of copies of DNA at location i. The example of OCR that have driven us through structured prediction in Chapter 1 has also an obvious sequential structure. Characters are naturally ordered on a sheet of paper.

Temporal sequences. Another large class of sequential data comes from temporal signals. They are ubiquitous in many applications. For instance, in audio, sounds, music, speeches come with a temporal order which is the one of the recording. These temporal signals are of two kinds: either they are naturally indexed by discrete times, like financial series [Bai and Perron, 1998, Bolton and Hand, 2002], either they are recordings of natural phenomena such as audio signals in speech [Gold and Morgan, 2000] or music [Downie, 2008]. In the latter case, time is continuous and need to be sampled. That way, we can represent the signal in a computer. There is a whole literature about conditions for a sampling to preserve all the information in the signal during the discretization process. The most famous result is the one of Shannon [Shannon, 1948] and Nyquist [Nyquist, 1928]. It states that, for a of a signal whose maximal frequency is f, it is necessary to sample

it at rate $2f$ to have a non-ambiguous representation of it. In our work, we consider that time has been discretized in a proper manner. In all our applications, we assume to have sampled uniformly the continuous time into T *temporal intervals*. All sampling points are thus evenly distributed over the continuous time. In this manuscript, we always assume that T, the *temporal horizon*, of the sequence is finite.

Note that our vocabulary is driven by the temporal case for the sake of simplicity. A temporal sequence as well as a spatial sequence is referred to as a *time series*. Each element $t \in \{1, \dots, T\}$ corresponds to a *timestamp* or index of the time series.

3.1 Problems involving sequential structure

Two main families of problems involve sequential structures: the first ones are offline. We suppose that the whole sequence has been observed and that the task at hand is solved afterwards. For instance, for detecting cancers, "aCGH" are often considered as batch sequences in which breakpoints should be detected retrospectively. Indeed, types of cancers and their prognosis are closely related to the number of alterations along the DNA sequence [Schleiermacher et al., 2010, Hocking et al., 2013]. However, due to the long length of the sequence, change-points [1] in the number of replicates along the DNA cannot be manually found and the sequence have to be processed automatically. This leads to a computational problem we detail in Sec. 3.2.
The second ones are online or real-time in which the task should be performed "on the fly". Namely, observations are done sequentially and so is the task. For instance, this can be the score-following task which aims at synchronizing online an interpreter to a recorded accompaniment [Cont, 2010]. In this thesis, we only consider the batch setting.

In this manuscript, we mainly focus on problems involving the underlying sequential structure of the data. To fix ideas, let us briefly introduce the three of these we deal with in this thesis:

1. the *change-point detection*, which aims at finding the best piecewise-constant fit to a time series such as the best segmentation in K segments of a noisy audio recording of a succession of K successive notes,
2. the *alignment* that aims at aligning a signal on a reference template such as an audio recording to a partition,
3. the *warping* that aims at building correspondences between two signals that might differ in speed, such as two audio recordings of the same musical pieces.

More formally, let T be the temporal horizon. In this chapter, we define an *atomic sequential problem* as:

1. In this work, we mainly use the term *change-point*. However, in many applications these are called breakpoints.

1. a set of *authorized decisions* $\mathcal{Y}_t$, for each timestamp t. We call $\mathcal{Y}$ the Cartesian product of individual authorized decisions $\mathcal{Y}_1 \times, \ldots \times \mathcal{Y}_t$. If $\mathcal{Y}_t$ depends upon others sets, it must only depends on the past $\mathcal{Y}_{t'}$ then $t' < t$.
2. a terminating payoff function $V_T : \mathcal{Y}_T \to \mathbb{R}$ and, $\forall t \in \{1, \ldots, T\}$ *payoffs functions* $V_t : \mathcal{Y}_t \times \mathcal{Y}_{t+1} \to \mathbb{R}$.
3. an *objective* of the form:

$$\underset{y \in \mathcal{Y}}{\text{maximize}} \sum_{t=1}^{T-1} V_t(y_t, y_{t+1}) + V_T(y_T). \tag{3.1}$$

For these, we define, for every $t < T$ the value functions restricted to timestamp t_0 by $v_{t_0} : \mathcal{Y}_{t_0} \to \mathbb{R}$ which is such that $v_{t_0}(y_{t_0})$

$$\underset{y \in \{\mathcal{Y}_{t_0+1} \times \ldots \times \mathcal{Y}_T\}}{\text{maximize}} \sum_{t=t_0}^{T-1} V_t(y_t, y_{t+1}) + V_T(y_T), \tag{3.2}$$

and for $t_0 = T$, we define $v_T = 0$. These kind of problems can be solved using the generic dynamic programming framework of Bellman [1956]. We present there a more restrictive setup but that is sufficient for the applications we met in this manuscript.

3.1.1 Dynamic programming algorithm

Dynamic programming algorithms provide a general framework for solving a problem by recursively dividing it into smaller ones [Bellman, 1956, Bertsekas, 1995, Cormen et al., 2001, Carlier, 1997]. Contrary to other approaches such as divide-and-conquer, the subproblems recursively solved are not disjoint but overlap.

Principle of optimality. When the problem has a sequential structure as before, the key observation for deriving the dynamic programming technique is the following principle of optimality [Bellman, 1956, Bertsekas, 1995]:

Principle. If the sequence of decisions $(y_1, \ldots, y_T) \in \mathcal{Y}$ is optimal for the problem of Eq. (3.1), then for any $t \in \{1, \ldots, T-1\}$, $(y_t, \ldots, y_T)$ is optimal for the restricted objective

$$\underset{\mathcal{Y}_t \times, \ldots, \times \mathcal{Y}_T}{\text{maximize}} \sum_{t'=t}^{T-1} V_t(y_{t'}, y_{t'+1}) + V_T(y_T).$$

This is rather intuitive, suppose $(y_t, \ldots, y_T)$ were not optimal for a certain t for Eq. (3.2). Let us call the optimal solution $(y'_t, \ldots, y'_T)$. Then $(y_1, \ldots, y_{t-1}, y'_t, \ldots, y_T)$ would have a greater objective value than $(y_1, \ldots, y_t, y_{t+1}, \ldots, y_T)$ for Eq. (3.1). Let us now introduce

$$\forall y_t \in \mathcal{Y}_t, v_t(y_t) = \max_{\mathcal{Y}_{t+1} \times, \ldots, \times \mathcal{Y}_T} V_t(y_t, y_{t+1}) + \sum_{t'=t+1}^{T-1} V_t(y_{t'}, y_{t'+1}) + V_T(y_T),$$

the value of the restricted objective starting at timestamp t with the action y_t. Using the aforementioned principle, it is possible to show [Bellman, 1956, Carlier, 1997] that the optimal solution of the problem of Eq. (3.2) satisfies the following recursion called the *Bellman update equation*:

$$v_t(y_t) = \max_{y_{t+1} \in \mathcal{Y}_{t+1}} V_t(y_t, y_{t+1}) + v_{t+1}(y_{t+1}). \tag{3.3}$$

Generic dynamic programming algorithm. The recursive formulation allows us to use a generic algorithm to solve this problem. It operates in two steps:

1. we compute all the possible values for the value functions v_t using the Bellman update equation of Eq. (3.3),
2. we find the optimal sequence of decisions $(y_1, \ldots, y_T)$ by tracking back the solutions and find at each timestamp $t \in \{1, \ldots, T-1\}$,

$$y_{t+1} \in \operatorname*{argmax}_{y \in \mathcal{Y}_{t+1}} \Big\{ V_t(y_t, y) + v_{t+1}(y) \Big\}.$$

If we take as unity of complexity the evaluation of our payoff function, the complexity of the first step is $\sum_{t=1}^{T} |\mathcal{Y}_t|$, the sum of the cardinalities of the sets of authorized decisions $\mathcal{Y}_t$ at time t. The second step has the same time complexity.

Dynamic programming let us thus deal with problems of the form of the ones of Eq. (3.1) in polynomial time. Note that without the sequential underlying structure of the problem, a brute-force algorithm would have led to an algorithm exponential in time [2].

3.2 Change-point detection problem

Now, let us move to our first sequential problem: the change-point detection. We start by presenting it using the standard statistical approach before introducing more algorithmic aspects and casting it as a sequential problem.

Change-point detection is the problem of detecting abrupt changes in the mean of time series [3] of finite temporal horizon T [Basseville and Nikiforov, 1993]. Time series are assumed to be the realization of some noisy process whose mean is piecewise-constant during time, but with K abrupt changes. Since time has been discretized, the notion of a piecewise-constant mean might be fuzzy. Let us be more precise about this notion. A function of discrete time $\mu : \{1, \ldots, T\} \to \mathbb{R}^p$ is said piecewise-constant with K *segments* if

$$\big|\{t \in \{1, \ldots, T-1\},\ \mu(t) \neq \mu(t+1)\}\big| = K - 1.$$

2. Let us suppose that at each timestamp K decision are possible and the horizon is T, exploring the space of solution with an exhaustive search approach is of size K^T.

3. More generally, change-point detection problems aim at looking for changes in the *distributions* of time series, but in our work we restrict to the commonly met case of change-point detection in the mean.

The function changes of value $K-1$ times. In Fig. 3.1, in the upper right corner, such a piecewise-constant function is depicted for $K=3$. Note that it is equivalent to say that the function is a K piecewise-constant signal or has $K-1$ *breakpoints* or *change-points*. The set of such piecewise-constant functions with exactly K segments, is referred to as $\mathcal{M}_K$. We also denote by

$$\mathcal{M} = \bigcup_{K \in \{0,\dots,T\}} \mathcal{M}_K.$$

This problem is ubiquitous in many real-world applications. For instance in bioinformatics, it is of high importance to be able to detect breakpoints along the DNA copy number. Indeed, as mentioned earlier, the number of copies along the DNA (also known as CGH profiles) are heavily linked to cancer prognosis [Picard et al., 2005, Shah et al., 2007, Hocking et al., 2013]. In finance, detecting abrupt changes among temporal series have been somehow studied, especially for detecting fraud [Bai and Perron, 1998, Bolton and Hand, 2002].

3.2.1 Statistical approach

In this section, we assume a multivariate time series is given. This can be seen as p unidimensional time series of temporal horizon T. Thus, we can represent it by a matrix $X \in \mathbb{R}^{T \times p}$.

Underlying model. The change-point detection problem is based on the following observation model:

$$X_t = \mu^*(t) + \epsilon_t, \tag{3.4}$$

where ϵ_t are independent and identically distributed noise variables with zero mean. The function $\mu^* : \{1,\dots,T\} \to \mathbb{R}^p$ is a piecewise-constant function of the time $\{1,\dots,T\}$ with exactly K segments.

The model is depicted by Fig. 3.1. The change-point detection problem aims at recovering the piecewise-constant function μ^*. With $\mathcal{M}_K$ is the set of all possible piecewise functions μ with K segments, the output $\widehat{\mu}$ is found using a least-squares approach:

$$\widehat{\mu} \in \operatorname*{argmin}_{\mu \in \mathcal{M}_K} \sum_{t=1}^{T} \|X_{t,\cdot} - \mu(t)\|_2^2. \tag{3.5}$$

Model selection. Considering the change-point detection problem leads to a major concern: the choice of the number of breakpoints or segments of the function μ. In other words, how do we choose $\mathcal{M}_K$? The minimization problem (3.5) naturally enforces $\widehat{\mu}$ to lie in a set $\mathcal{M}_K$ with the highest possible number $K-1$ of change-points. In the extreme case, where we consider to minimize the least-squares criterion of Eq. (3.5) over all the set $\mathcal{M} = \bigcup_{K \in \{1,\dots T\}} \mathcal{M}_K$, the function $\widehat{\mu}$ selected will be the one that associates $X_{t,\cdot}$ to $t \in \{1 \dots T\}$. This is a case of overfitting. To circumvent this issue, some works assume the number of changes to be given in

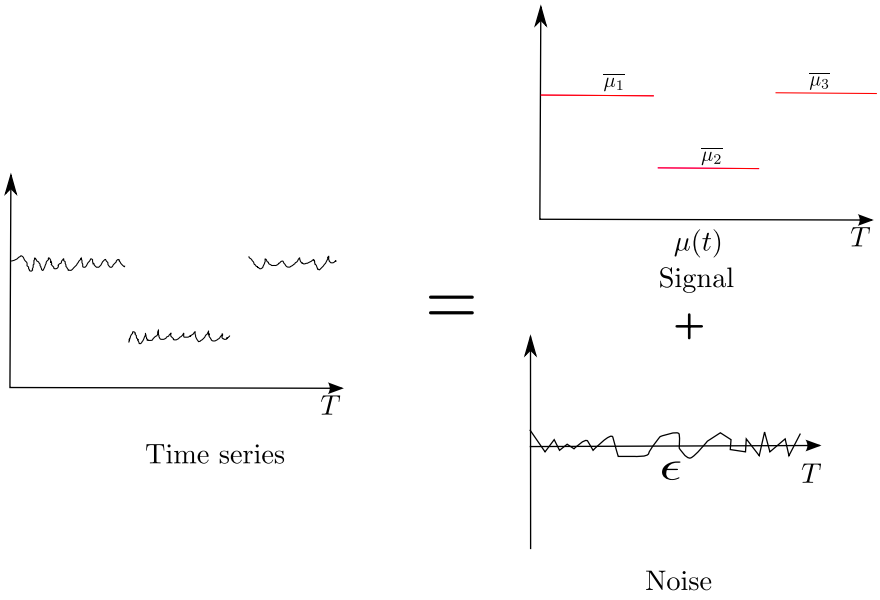


Figure 3.1: An example of change-point detection problem in dimension 1, the upper right plot is the mean function μ we aim to find.

advance and thus constrain the model to be chosen in a subset of $\mathcal{M}$ [Picard, 1985, Harchaoui et al., 2009].
However, this approach might not be totally satisfactory in practice since the number of change-points is not always known. This is why the question can be adressed using the theory of model selection [Schwarz, 1978, Lebarbier, 2005, Massart, 2007]. The idea is to replace the criterion of Eq. (3.5) by the following penalized one:

$$\sum_{t=1}^{T} \|X_{t,\cdot} - \mu(t)\|_2^2 + \text{pen}(\mu), \tag{3.6}$$

and to minimize this criterion over the whole set $\mathcal{M}$. Several penalty functions have been proposed. We denote by $K(\mu)$ the function, that to a piecewise-constant function $\mu \in \mathcal{M}$ associates the number of different segments in it. The simplest approach is the one based on the Akaike Information Criterion (AIC) [Akaike, 1973] that sets $\text{pen}(\mu)$ to be equal to $\kappa K(\mu)$ for some $\kappa > 0$.

It has been shown [Lebarbier, 2005] that, for a temporal horizon T and in the case where the noise ϵ_t is Gaussian with variance σ^2, a good penalty leading to an oracle inequality is of the form:

$$\kappa\sigma^2 K + \eta\sigma^2 \log(\frac{T}{K}).$$

More details about this penalty can be found in Proposition 2.4.1 of Lebarbier [2002].

The problem of adjusting these parameters η and κ is however challenging and is of high practical importance [Hocking et al., 2013]. Some heuristics and techniques exist to directly tune these [Lebarbier, 2005] but they are not really fully

satisfactory in practice. In Chapter 4, we propose a way to adjust the parameter κ of the AIC criterion in a specific case: the one where we are provided supplementary information in the form of manually segmented training data.

3.2.2 Algorithmic approach

From now on, we only consider the problem when the number of segments K is set in advance. In this section, we cast the problem of change-point detection as a partitioning problem.

We call *partitioning problem* any problem that aims at finding a partition of the T datapoints into K subsets that are often referred to as *clusters*. Typically a partitioning problem consists in finding an *assignment* matrix $Y \in \{0,1\}^{T\times K}$ such that $Y_{i,k} = 1$ if and only if datapoint number i is belongs to cluster k. For a partition of 5 datapoints $(1,2,3,4,5)$ into the two clusters $(1,2,4)$, $(3,5)$, the corresponding assignment matrix Y is:

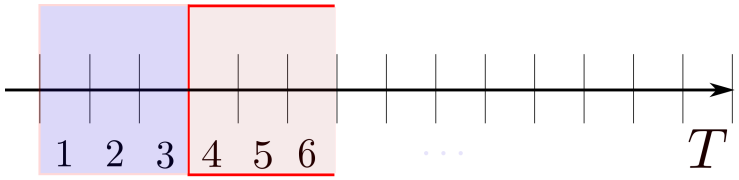
$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Parametrization of μ. Please keep in mind that μ is a piecewise-constant function from $\{1,\dots,T\}$ to $\mathbb{R}^p$, with K segments. It can be represented using the assignment matrix $Y \in \{0,1\}^{T\times K}$ and a mean matrix $\overline{\mu} \in \mathbb{R}^{K\times p}$ whose k-th row $\overline{\mu}_{k,\cdot}$ is exactly the vector corresponding to the $\mu(t)$. That way μ can be identified to the product $Y\overline{\mu}$ which is a p-dimensional matrix such that its t-th row is equal to the mean vector $\mu(t)^\top$. Thus, we have cast the change-point detection problem as a partitioning one. It aims at clustering the data into K subsets under a temporal constraint that enforces the contiguity of the segments. Let us denote by $\mathcal{E}_k$ the subset of assignment matrices that respect the contiguity constraints. For instance, the matrix of the previous paragraph does not satisfy the contiguity constraints, but the matrix:

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix},$$

does. The constraints defining the matrices Y of the set $\mathcal{E}_k$ can be summarized as follows:

1. Y is binary: $Y \in \{0,1\}^{T\times k}$,
2. Y is stochastic: $Y\mathbf{1}_k = \mathbf{1}_T$ (rows sum to one),
3. Y is of rank k (no empty clusters),
4. if $Y_{i,j} = 1$ then $Y_{i',j'} = 0$ for all $j' > j$, $i' \leq i$.



$$\mathrm{Ind}_{1,6} = 3$$

Figure 3.2: An example of the quantity Ind maintained during Alg. 5. this matrix contains the index of the last change-point detected, in that example, at timestamp 6, the optimal segmentation with one change-point partitions data between timestamps 1,2,3 and 4,5,6.

With these notations, the problem of finding the optimal segmentation of the signal can be seen as the following matrix factorization problem:

$$\underset{Y \in \mathcal{E}_k, \overline{\mu} \in \mathbb{R}^{K \times p}}{\text{minimize}} \; \|X - Y\mu\|_2^2.$$

The general problem of partitioning data according to a least-squares criterion into K clusters is known to be NP-hard in the general case [Aloise et al., 2009]. This leads to several possible approximation algorithms based either on the K-means technique [Steinhaus, 1957] or spectral clustering [Ng et al., 2002, Bach and Jordan, 2003, Von Luxburg, 2007]. However, for change-point detection, the temporal structure reduces the number of authorized partitionings. A dynamic programming algorithm allows us to recover the best possible segmentation in $O(T^2K)$ where T is the length of the time series and K the number of segments. This algorithm is extended in Chapter 4 to deal with the case where the number segments K is not known in advance and where we optimize the penalized criterion of Eq. (3.6).

Algorithm for change-point detection when the number of clusters is known. In Alg. 5, we present a classical algorithm (see, e.g, Guthery [1974]) in $O(T^2K)$ allowing us to recover the optimal piecewise-constant function μ. The idea is to compute recursively a cost matrix C of dimension $K \times T$ whose (k, t)-th element corresponds to the cost of the optimal segmentation with k segments that stops at time t. Alongside with this matrix, we keep track of the timestamps that have led to the minimum of the elements of C [4]. Then, we perform a backtracking step. Namely, starting from the end of the time series, we find the timestamp corresponding to the beginning of the last segment, and we iterate until having recovered the whole segmentation.

In Chapter 4, we present an other way of looking at the change-point detection algorithm. Some refinements (based on the iterative pruning of solutions) lead to a drastically faster algorithm in practice [Rigaill, 2010, Killick et al., 2012]. The mean

4. The matrix Ind is the matrix such that $\mathrm{Ind}_{k,t}$ is the timestamp corresponding to the last jump leading to the value $C_{k,t}$. See Fig. 3.2

Algorithm 5 Dynamic programming algorithm to solve the change-point detection problem with a fixed number of $K-1$ change-points

Input $X \in \mathbb{R}^{T\times p}, K \in \{2,\dots,T\}$
Set $C \in \mathbb{R}^{K\times T}$, $\text{Ind} \in \mathbb{R}^{K\times T}$
Initialize $\forall j, C_{1,j} = \frac{1}{j}\sum_{i=1}^{j}\|X_{i,.} - \frac{1}{j}\sum_{t=1}^{j} X_{t,.}\|_2^2, \text{Ind}(1,j) = 1,$
for $k = 2,\dots,K$ **do**
  for $t = 1,\dots T$ **do**
    for $j = k,\dots t$ **do**
      Set $\alpha(j) = C(k-1,j-1) + \frac{1}{t+1-j}\sum_{i=j}^{t}\|X_{i,.} - \frac{1}{t+1-j}\sum_{i=j}^{t} X_{i,.}\|_2^2$
    end for
    $C(k,t) = \min(\alpha)$
    $\text{Ind}(k,t) = \text{argmin}(\alpha)$
  end for
end for
Backtracking step: Set $\Delta(K) = T$
for $i = K,\dots,2$ **do**
  $\Delta(i-1) = \text{Ind}(i-1, \Delta(i))$
end for
Output: Δ

complexity of these refinements is in $O(KT)$ but the worst case is quadratic in time. However in this manuscript we do not make use of these improvements.

Link with dynamic programming. The algorithm we have presented in the previous paragraph is the dynamic programming one of Sec. 3.1.1 applied to a specific function value. More formally, let us define $\mathcal{Y}_k$ as the set of authorized localizations for change-point k. These decision sets can thus be identified to the sets of timestamps $\{k,\dots,T-(K-k)\}$ (the k-th segment cannot start before the k-th timestamp and a symmetric condition holds concerning the end of the segment). The payoffs functions are the negative intra-segment variances, namely:

$$\forall k < K, \forall (i,j) \in \mathcal{Y}_k \times \mathcal{Y}_{k+1}, V_k(i,j) = -\frac{1}{j+1-i}\sum_{t=i}^{j}\Big\|X_{t,.} - \frac{1}{j+1-i}\sum_{t'=i}^{j} X_{t',.}\Big\|_2^2 - I_{j\geq i},$$

and for $k = K$,

$$V_K(i) = I_T(i),$$

where I_C is the convex indicator of the set C. The K-th change-point is in fact a "dummy" change-point since it is always put at the end of the time series.

3.3 Time warping and alignment problems

Let us detail two problems with sequential structure: time warping and alignment.

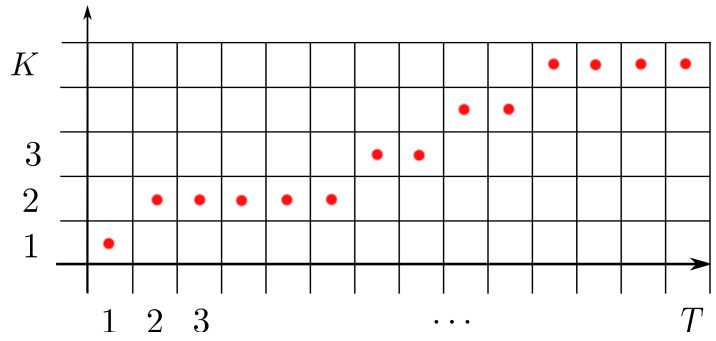


Figure 3.3: An alignment can be seen as an increasing mapping m from $\{1 \dots T\}$ to $\{1, \dots K\}$ such that $m(t+1) = m(t)$ or $m(t+1) = m(t)+1$. The only moves llowedin the matrix of size $T \times K$ depicted here are the horizontal and the diagonal.

3.3.1 The alignment problem

We first describe the alignment problem. This is a specific instance of the time warping (TW) problem that will be exposed in Sec. 3.3.2, when one of the signals is taken as a "reference". Namely, it is the problem of aligning a time series with an ordered *template* that may be, for instance, a symbolic ordered sequence representing events occuring in the signal. This problem is ubiquitous: for instance in computer vision, it is used to localize temporally actions in videos [Bojanowski et al., 2014]; in music information retrieval, a major problem is the score following one (or audio-to-partition problem) where the goal is to align an interpretation to the partition that has been played [Dannenberg, 1984, Vercoe, 1985, Cont, 2010]. Keep in mind throughout this section that the template signal consists in an ordered list of symbolic events.

Mathematically, the goal is to find a non-decreasing mapping m from the set of indexes of the reference signal to set of the indices (timestamps) of the other one. An example of such an alignment is depicted in Fig. 3.3.

Due to the asymmetry in the roles between the two signals, we propose to denote the reference template by $X^A \in \mathbb{R}^{K \times p_1}$ and the signal to align by $X^B \in \mathbb{R}^{T \times p_2}$. The term *timestamp* thus only applies to the temporal elements of the second signal. Elements of the first signal are just called *elements*. Aligning is finding a mapping

$$m : \{1, \dots, T\} \to \{1, \dots, K\}$$

that respects the following constraints:

1. $\forall t \in \{1, \dots, T\}$, $m(t+1) = m(t)$ (horizontal move in Fig. 3.3) or $m(t+1) = m(t)+1$ (diagonal move in Fig. 3.3),
2. $m(1) = 1$, (alignment of the two beginnings of sequences)
3. $m(T) = K$ (alignment of the end of the two sequences).

Note that an alignment between a signal of length T and a reference of length K can be represented by an indicator matrix $Y \in \{0,1\}^{T \times K}$ such that $Y_{ij} = 1$ if and only if $m(i) = j$ of the other signal. This indicator matrix can be visualized

in Fig. 3.3. It corresponds to a graphical representation of the mapping m. Note that the parameterization through Y is the same as the one we have proposed for the segmentation outputed by the change-point detection algorithm of Sec. 3.2. Indeed, aligning a signal on a symbolic sequence of length K is purely equivalent to segmenting this signal into K segments while respecting the contiguity constraints of Sec. 3.2. However, in the context of alignment, this matrix Y is referred to as an *alignment* matrix rather than an assignment one (this term denotes an other object). The set of possible alignments has thus the same cardinality as the set of possible segmentations with K segments. This is the number of possibilities to choose $K-1$ breakpoints among $T-1$ timestamps, that is $\binom{T-1}{K-1}$.

Pairwise affinities. Alignment is done by considering pairwise similarities between elements $X^A_{i,\cdot}$ and $X^B_{j,\cdot}$. Such a similarity can be measured through

$$S : \mathbb{R}^{p_1} \times \mathbb{R}^{p_2} \to \mathbb{R}.$$

We represent pairwise affinities in a matrix C such that $C_{ij} = S(X^A_{i,\cdot}, X^B_{j,\cdot})$. These affinities are often non-positive in applications [Müller, 2007]. Note however that we do not enforce the similarity measure S to produce non-positive affinities. The goal of the alignment procedure is to recover a mapping $m : \{1, \dots, T\} \to \{1, \dots, K\}$ respecting the constraints of the previous paragraph while maximizing the affinity cost defined by:

$$\sum_{t=1}^{T} C_{t,m(t)}.$$

Finding this mapping is thus exactly maximizing the value function $\sum_{t=1}^{T} C_{t,m(t)}$ while having m respecting temporal contiguity constraints. This leads to an efficient dynamic programming algorithm to deal with the temporal structure.

Algorithmic approach. Alignment is generally performed using the technique described in Alg. 6 Note that its complexity is $O(TK)$. This means in particular that the comlexity is *linear* in the time series length T^A This algorithm is a dynamic programming one. Indeed, let $I_C(x)$ be the convex indicator[5] of the set C and define the set of authorized decisions for all $t \in \{1, \dots, T\}$ as $\mathcal{Y}_t = \{1, \dots, K\}$. We also define the payoff functions $\forall t \in \{1, \dots, T\}, \forall y_t \in \mathcal{Y}_t, y_{t+1} \in \mathcal{Y}_{t+1}$:

$$V_t(y_t, y_{t+1}) = -I_{t \geq y_t} + \max(C_{t,y_t} - I_{y_{t+1}-1}(y_t), C_{t,y_t} - I_{y_{t+1}}(y_t)),$$

and for $t = T$, we define the function

$$\forall y_T \in \mathcal{Y}_T, \quad V_T(y_T) = C_T(T, y_T) - I_{y_t}(K).$$

5. That is $I_C(x) = 0$ if $x \in C$ and $+\infty$ elsewhere.

Algorithm 6 Dynamic programming algorithm for aligning temporal sequence. It recovers m the mapping corresponding to the alignment and that maximizes the similarity criterion.

Input: C the cost matrix associated to the temporal signals to align.
Set $D \in \mathbb{R}^{T \times K}$, $m \in \mathbb{R}^{T}$
for $t = 1, \ldots, T$ **do**
  for $k = 1, \ldots, K$ **do**
    $D_{kt} = \max(D_{k-1,t-1} + C_{k,t}, D_{k,t-1} + C_{k,t})$
  end for
end for
Recover m by backtracking: $m(T^A) = K$
while $m(t) > 1$ or $t > 1$ **do**
  if $m(t) = t$ **then**
    $m(t-1) = t - 1$
  else
    $v = \text{argmax}_{i \in \{m(t), m(t)-1\}}(D_{i,t-1})$
    $m(t-1) = v$
  end if$t = t - 1$
end while
Output: Optimal mapping m.

Thus, the goal of the alignment procedure is exactly to maximize the cost:

$$\sum_{t=1}^{T-1} V_t(y_t, y_{t+1}) + V_T(y_T).$$

Doing so using the generic dynamic programming algorithm presented in Sec. 3.1.1, leads to Alg 6.

3.3.2 The general time warping problem

Time warping problem is at stake when there is no signal of reference and we want to build a correspondence between elements of two different time series. This occurs in music, where two audio interpretations of the same piece need to be aligned as a front-end task of any audio editor [Müller, 2007]. It also occurs in video, where two video samples can depict the same action and need to be put in correspondence. The original time warping problem has been introduced for spoken word recognition in speech processing [Sakoe and Chiba, 1978]. More generally, finding a warping of time can be a way to find patterns in time series improving the understanding we have of them, and allows us to extract meaningful information [Berndt and Clifford, 1994, Noma, 2002, Cuturi et al., 2007].

Let us suppose we are given two temporal series $X^A \in \mathbb{R}^{T^A \times p^A}$ and $X^B \in \mathbb{R}^{T^B \times p^B}$. Thus, both of these are sequences of timestamps. The dynamic time warping problem aims at finding pairs of indexes (t_A, t_B) that are in correspon-

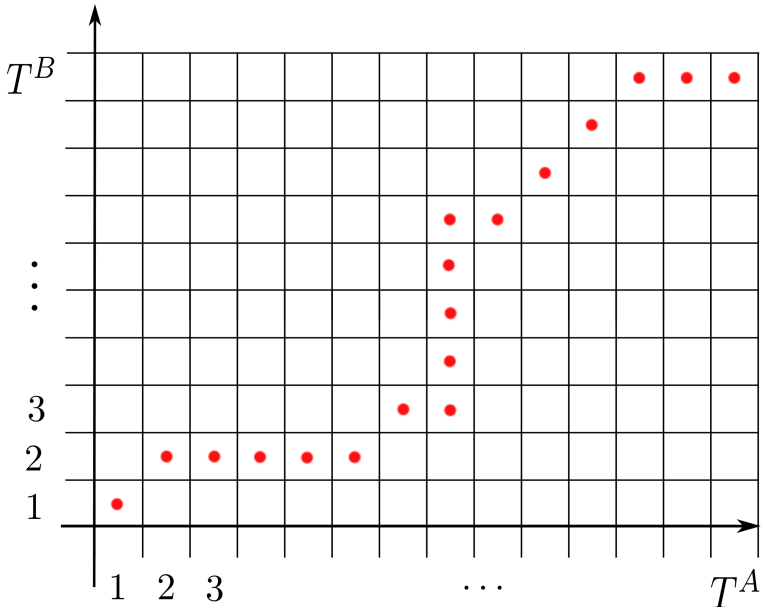


Figure 3.4: Example of a warping between indices of two time series. Note that, contrary to alignment we authorize vertical moves. Thus, a warping is a path from $(1,1)$ to (T^A, T^B) with horizontal, diagonal and vertical moves.

dence while respecting a certain underlying temporal constraint of contiguity of the successive pairs. The approach to tackle this problem is very similar to the one of alignment. The only difference lies in the fact that one actually does not want is an alignment (that is a mapping between timestamps and an order set of symbolic events) but rather a *warping* (or warp). This object can be thought as a bidirectional deformation of the indexes of the two temporal series. More formally, we look for a common temporal abscissa for re-indexing the time series. Namely, we are looking for T such that $\max(T^A, T^B) \leq T \leq T_A + T_B$ and a mapping

$$\begin{aligned} m : \{1, \ldots, T\} &\to \mathbb{R}^2, \\ m(t) &= (\alpha(t), \beta(t)), \end{aligned}$$

such that:

1. $m(1) = (1,1)$, (first timestamps of the temporal series are warped together)
2. $m(T) = (T^A, T^B)$, (last timestamps of the temporal series are warped)
3. $\forall t \in \{1, \ldots, T-1\}, m(t+1) = m(t) + (1,1)$ (diagonal move) or $m(t+1) = m(t) + (0,1)$ (vertical move) or $m(t+1) = m(t) + (1,0)$ (horizontal move).

An example of warping is depicted in Fig. 3.4.

Warping and pairwise affinities. As in the alignment case, we assume to have some similarity measure $S : \mathbb{R}^{T^A} \times \mathbb{R}^{T^B} \to \mathbb{R}$. The comparison scores between each indexes t^A and t^B of the two time series are encoded in a pairwise affinity matrix C. These scores are often non-positive, for instance in music where S is

often set to be the negative Euclidean distance between timestamps [Müller, 2007]. However, except when we explicitly say it, we never assume non-positivity of the entries of C. The standard dynamic time warping problem aims to maximize:

$$\sum_{t=1}^{T} C_{m(t)}.$$

Note that this is really close to the criterion maximized for alignments. Like for alignments, there is an efficient algorithm letting us able to solve this problem is $O(T^A T^B)$. Note that this complexity is linear in the respective lengths of the first time series. This algorithm is detailed more in details in Chapter 5.

Further refinements. We can often have a strong prior concerning the shape of the warping because of the task at hand. For instance, the "diagonal" warping that corresponds to an uniform warping [6] of the two temporal series is widely used in speech [Myers and Rabiner, 1981]. To favor such solutions, it has been proposed to constrain the space of possible warpings so that a warping path cannot be too far from the diagonal. Practically this corresponds to only authorize paths within the so-called "Sakoe and Chiba band" [Sakoe and Chiba, 1978] or the "Itakura parallelogram" [Itakura, 1975].

The shape of the output of the alignment algorithm 6 and DTW algorithm 8 depends upon the matrix C that encodes pairwise affinities. This matrix is set using the similarity measure S yet. However the choice of S has not been discussed in this section. It is of crucial practical importance since different S might lead to totally different outputs for the algorithm. Tuning such a similarity measure by hand is often hard and requires a lot of expert knowledge. The need for adjusting similarity measures goes far beyond the cases of alignment, change-point detection or time warping. A whole field of machine learning is dedicated to design good algorithms to learn similarity measures for various tasks. The next section reviews some of the common ways to address the problem of learning such a similarity measure.

3.4 Similarity measure learning

Similarity measures are fundamental tools for machine learning, data mining and signal processing. Given an input set $\mathcal{X}$ in which datapoints lie, these measures can be seen as functions $S : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that score pairs of inputs. Indeed, the need for comparing datapoints or events happening at different timestamps is ubiquitous. For instance, clustering tasks rely on minimizing a similarity measure between points [Jain and Dubes, Berkhin, 2006]. For example, given T datapoints $x^i \in \mathbb{R}^p$, the K-means algorithm aim at finding subsets of data $C_1, C_2, \ldots, C_K$ such

6. This alignment is the one that is the closest of the straight line from $(1,1)$ to (T^A, T^B) in Fig. 3.4.

that:

$$\sum_{k=1}^{K} \sum_{x^i, x^j \in C_k} \|x^i - x^j\|_2^2,$$

is minimal. The similarity underlying K-means is thus the squared L_2 distance between datapoints. The choice of an other similarity measure yields to other algorithms. For instance using the L_1 distance leads to the K-medians [Bradley et al., 1997].

For multiclass classification, the widely used K-Nearest neighbour classifier [Fix and Hodges Jr, 1951, Cover and Hart, 1967] heavily relies on the choice of a good similarity measure.

In information retrieval, the need for similarity measure arises everywhere. Given a query and D documents, we need to score the relative relevance between these. Generally this problem is addressed using ranking algorithms [Baeza-Yates and Ribeiro-Neto, 1999, Liu, 2009].

These crucially depends on the similarity measure used to compare objects. We could find a lot of other examples in various areas of science and signal processing. In the two next chapters, we will focus on the design of these for the change-point detection task, the warping and alignment problems. We can then wonder if it is possible to design automatically such similarity measures. A large amount of work have been done in that direction. In the next section we propose to detail some underlying principles of the similarity measure learning field.

3.4.1 General setting

In this section, we describe formally how it is possible to learn a similarity measure from data. Before going further, let us describe some mathematical vocabulary that we will stick to in the rest of this chapter.

Some definitions. In the similarity measure learning litterature, the word metric is often used to name the similarity measure learned. However, in some important cases the similarity measure learned is not a metric strictly speaking. To avoid any confusion, we propose here to fix the vocabulary we use in this manuscript. Let S be a function from $\mathcal{X} \times \mathcal{X}$ to $\mathbb{R}$, where $\mathcal{X}$ is the input space. We call S :

– a *similarity measure* if S is symmetric, i.e.,

$$\forall x, y \in \mathcal{X}, \quad S(x, y) = S(y, x),$$

– a *non-negative similarity measure* if $S(x, y)$ is symmetric, non-negative for all pairs (x, y),

– a *quasimetric* if S is non-negative, symmetric, and satisfy triangle inequality. Namely if

$$\forall x, y, z, \; S(x, z) \leq S(x, y) + S(y, z),$$

– a *semimetric* if S is non-negative, symmetric and satisfy the separation axiom: $S(x, y) = 0$ if and only if $x = y$,

– a *metric* if S is both a quasimetric and a semimetric (if S is a distance).

Most of the so-called metric learning algorithms learn in fact a quasimetric, which satisfies all the axioms of the definition of a distance except the separation one: two distinct points can have similarity 0.

Now, let us suppose to have at hand some task and data on which we want to perform it. We also assume to have some information about which datapoints should be close and which ones should not. We assume throughout this section our data lies in some vector space. Namely, $\mathcal{X} \subset \mathbb{R}^p$.

Vocabulary and data. In similarity measure learning, we can be provided several kind of data:

1. The first one are *fully labelled* datasets for the task at hand. In the case of nearest neighbours for instance, this can be whole multiclass datasets. For ranking, it can be a whole dataset with fully ranked objects etc.
2. The *must-link* constraints set $\mathcal{S}$ is the set of all pairs of inputs (x^1, x^2) that should be close according to the similarity learned. Such a constraint is denoted by $x^1 \sim x^2$.
3. The *cannot-link* constraints set $\mathcal{D}$ is the set of pairs that should be dissimilar.We denote such a constraint by $x^1 \nsim x^2$.
4. The set $\mathcal{R}$ is the set of triplets such that (x^1, x^2, x^3) such that x^1 should be more similar to x^2 than x^3. These are often called the *relative constraints* [Bellet et al., 2013].

Note that must-link/cannot-link constraints are different in essence from labeled data for the task at hand. From labeled data, it is often possible to build sets $\mathcal{S}$ and $\mathcal{D}$, but there is often more in the fully labeled data. For instance, for the alignment problem seen in the previous section, a fully time-stamped time series provides must-link/cannot-link constraints, but this information loses the sequential aspect of data.

Among the standard similarity measure algorithms, we can distinguish two major families depending on the input data they require.

Generic similarity measure learning techniques. The first ones are agnostic to the task at hand. They only make use of the sets of $\mathcal{D}$, $\mathcal{R}$, $\mathcal{S}$. These methods learn a metric by adjusting a cost that jointly enforces must link pairs to be similar and cannot-link to be dissimilar. Among these, we can mention the approach of information theoretical metric learning (ITML) [Davis et al., 2007] or the relevant component analysis of [Bar-Hillel et al., 2005, Shental et al., 2002].

Task-dependent similarity measure learning. The second ones are specifically designed for learning a task-specific similarity measure. They thus make use of fully labeled datasets for the task at hand in top of the sets $\mathcal{D}$, $\mathcal{R}$, $\mathcal{S}$. Some specific techniques have been designed for instance for the K-means algorithm [Xing et al., 2002], the K-nearest neighbours [Goldberger et al., 2004, Weinberger and

Saul, 2009], and ranking ones [Mcfee and Lanckriet, 2010, McFee et al., 2012]. The approaches that we propose in Chapters 4 and 5 fall in that category.

3.4.2 Mahalanobis (quasi)metric learning

As underlined by recent and comprehensive surveys about similarity measure learning like Bellet et al. [2013] or Kulis [2012], among the several forms of similarity measure, the most widely learned is the Mahalanobis similarity measure. In this manuscript, we focus on this specific case. For two datapoints x and x' in $\mathbb{R}^p$, the Mahalanobis similarity measure parametrized by the semidefinite positive matrix $W \succeq 0$ is the function:

$$S(x, x'; W) = \sqrt{(x - x')^\top W (x - x')}.$$

Note that in this context we are thus learning either a *quasinorm* or a norm. If we introduce the matrix L as the semidefinite positive square root of W, we can see the use of the Mahalanobis differently. It is equivalent to taking the standard Euclidean distance after having applied to the original data x, x' the linear mapping corresponding to L. Mahalanobis learning measure can be seen as the learning of the optimal linear transformation of data for the task at hand.

In the following, we describe generic algorithms for learning such a quasimetric that do not depend of the task at hand. Then we will move on task-specific examples and present some approach to deal with the K-nearest neighbours classification. Note that all the algorithms we describe in this section learns the squared Mahalanobis quasimetric

$$(x - x')^\top W (x - x'),$$

which is *linear* in W leading to tractable convex optimization problems.

Some general similarity measure learning algorithms.

Given sets $\mathcal{S}, \mathcal{D}, \mathcal{R}$ defined as in Sec. 3.4.1, there are algorithms that, independently from any task at hand, can learn a Mahalanobis quasimetric. We propose to explore two of the most famous ones: the relevant component analysis (RCA) [Bar-Hillel et al., 2005, Shental et al., 2002] and the original information-theoretic similarity measure learning (ITML) [Davis et al., 2007, Jain et al., 2012].

RCA. The original RCA algorithm [Shental et al., 2002, Bar-Hillel et al., 2005] only makes use of must-link equivalence relations, that is the only input is the set $\mathcal{S}$. From the must-link constraints, RCA starts by building K subsets of data called *chunklets* $(C_1, \ldots, C_K)$ from the must-link constraints. These are the transitive closure of the relations (if $x^1 \sim x^2$ and $x^2 \sim x^3$ then x^1, x^2 and x^3 are in the same chunklet). Then the algorithm computes the sum of within chunklet variance, that

is, if m_k is the mean vector of the k-th chunklet:

$$\Sigma = \frac{1}{N}\sum_{k=1}^{K}\sum_{x^i \in C_k}(x^i - m_k)(x^i - m_k)^\top.$$

The semidefinite positive matrix associated to the metric of RCA is then just the inverse of Σ ($W = \Sigma^{-1}$, or the pseudo inverse in the case where Σ is not invertible). This matrix can be interpreted as the maximum likelihood estimate of the inverse covariance matrix in a Gaussian model in which each chunklet would be one class and in which all the covariance matrices are equal. In practice, RCA is used for dimensionality reduction [Bar-Hillel et al., 2005] since it allows us to use task-dependent knowledge, in the form of must-link constraints to define the chunklets and thus build a notion of "relevant variability" for the task. RCA has been extended to handle must-not link constraints. However this algorithm is less used than the standard RCA one due to its practical cost [Yeung and Chang, 2006].

ITML. Given sets $\mathcal{S}$ and $\mathcal{D}$, the idea behind this algorithm is to consider the problem of learning a squared Mahalanobis quasimetric by solving an optimization problem of the form:

$$\begin{aligned} \underset{W\succeq 0, \xi_{i,j}\geq 0}{\text{minimize}} \quad & \textstyle\sum_{i,j}\xi_{i,j} + \lambda\Omega(W, W_0) \\ \text{s.t.} \quad & (x^i - x^j)^\top W(x^i - x^j) \leq u + \xi_{i,j}, \forall (x^i, x^j) \in \mathcal{S} \\ & (x^i - x^j)^\top W(x^i - x^j) \geq v - \xi_{i,j}, \forall (x^i, x^j) \in \mathcal{D} \end{aligned} \tag{3.7}$$

where parameters λ, u and v have to be set carefully. The u parameter tunes the behaviour of the pairs of datapoints that should be similar whereas the v one makes sure that dissimilar pairs are far away according to the similarity measure induced by W. The main contribution of [Davis et al., 2007], have been to consider for the regularizer Ω the log-det divergence. This Bregman divergence is defined, for a semidefinite positive matrix $W \succeq 0$, and a positive definite matrix $W_0 \succ 0$ [7] as:

$$\Omega(W, W_0) = \mathrm{Tr}(WW_0^{-1}) - \log\det(WW_0^{-1}).$$

This divergence justify the name Information Theoretic Metric Learning (ITML). The log-det divergence corresponds to the Kullback-Leibler divergence between two multivariate Gaussian probability distributions of the same mean but one with covariance W_0 and the other one W. This divergence encodes a prior knowledge about the form of the Mahalanobis quasimetric to learn. It keeps the quasimetric "not far" from the metric induced by the matrix W_0 [8]. In many applications, taking W_0 close to the identity of $\mathbb{R}^p$ is sounded when data are not too noisy [Davis et al.,

7. The matrix W_0 is thus associated to an actual metric.

8. Simply taking the matrix gradient of this divergence to 0 yields to see that this divergence is minimal for $W = W_0$.

2007].

Let us now describe how a similarity measure learning can be specifically designed for a task. We focus in the next section on an historical and important example: the nearest-neighbour classification problem.

Important example: the nearest neighbours classification

For the sake of completeness, we recall here the standard algorithm of K-nearest neighbours classification. We are given training instances (x^i, y^i). To classify a new instance x, the idea is to predict by doing a majority vote. The new datapoint x will receive the same label as the majority label among its K-nearest neighbours the $(x^1, \ldots, x^N)$. The notion of neighbourhood crucially depends on the similarity measure at stake. Neighbours are generally computed according to the standard Euclidean norm $\|\cdot\|_2$. In case of equality in the votes, we predict randomly one of the majority labels.

In this algorithm, there is a need for adjusting the similarity measure. For characters recognition [Simard et al., 1993, Belongie et al., 2002] it has been shown that fine engineering of a similarity measure can drastically improve the performance of nearest neighbours classifiers. Classification depends on the geometry of the space. The idea behind algorithms that learns the similarity measure it is to replace the standard Euclidean metric by a Mahalanobis (quasi)one of the form $\forall x, y \in \mathbb{R}^p, S(x, y) = \sqrt{(x-y)^\top W(x-y)}$. Different approaches have been proposed for this learning problem. Two of the most used ones are the large-margin nearest neighbour (LMNN) [Weinberger and Saul, 2009] approach and the neighborhoud component analysis (NCA) [Goldberger et al., 2004]. Recent works try to go beyond these approach, especially by learning local similarity measure for different regions of the space [Weinberger and Saul, 2008, Zhan et al., 2009, Shi et al., 2014].

The RCA as well as LMNN proposes to learn a square root of the semidefinite positive matrix $W \in \mathbb{R}^{p\times p}$ rather than W directly (leading to a non-convex program but allowing to control the rank r of the semidefinite positive matrix). Let us call $L \in \mathbb{R}^{p\times r}$ one of the minimal rank matrices such that $W = LL^\top$.

Large-margin nearest neighbours (LMNN). The first method tries to learn L in such a way that for each training input x^i and associated label y^i, its K nearest neighbours have the same label. The intuition is depicted in Fig. 3.5. This intuition can be formalized mathematically by considering the two following sets of must-link/relative constraints:

- $S = \{(x^i, x^j), y_i = y_j\}$.
- $R = \{(x^i, x^j, x^k), (x^i, x^j) \in S, y_i \neq y_k\}$.

Using these notations, [Weinberger and Saul, 2009] proposes to learn L using a non-convex [9] objective function that realizes a compromise between one grouping term for the "target neighbours" of the training instances and a margin term trying

9. Note that, however, the formulation using W is.

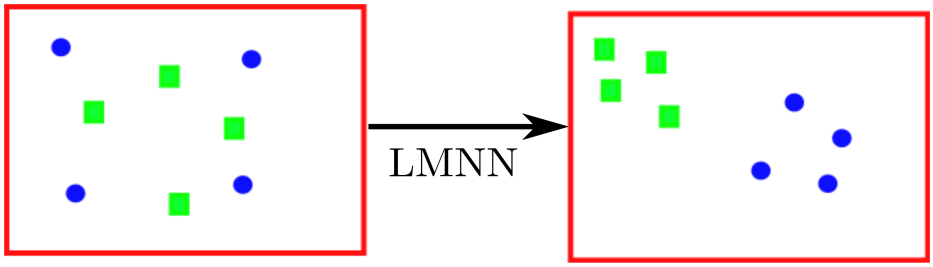


Figure 3.5: An example of how a typical metric learning algorithm works. We consider there the large-margin nearest neighbours one. In this examples the training data are in dimension 2 and there are two labels: green squares and blue points. The goal is to change the geometry of the space so that similar examples are close according to the distance. On the left, we have the original space, and on the right the space after the linear transformation induced by L, the output of LMNN.

to pull away the examples that have not the same label. This leads to the following large-margin minimization program:

$$\begin{aligned} \underset{L\in\mathbb{R}^{p\times r}}{\text{minimize}} \quad & \lambda \sum_{i,j,k} \xi_{ijk} + (1-\lambda) \sum_{(x_i,x_j\in S)} (x^i - x^j)^\top L^\top L (x^i - x^j), \\ & (x^i - x^k)^\top L^\top L (x^i - x^k) - \xi_{ijk} \geq 1 - (x^i - x^j)^\top L^\top L (x^i - x^j). \end{aligned}$$

Neighbourhood component analysis. The second method, the neighbourhood component analysis [Goldberger et al., 2004], seeks to optimize the following program:

$$\underset{L\in\mathbb{R}^{p\times r}}{\text{maximize}} \sum_i \sum_{j, y_j = y_i} \frac{\exp(-\|Lx^i - Lx\|_2^2)}{\sum_{k\neq i} \exp(-\|Lx^i - Lx^k\|_2^2)}.$$

We can interpret this nonconvex objective as looking for a linear transformation of the space that minimizes the classification error using a stochastic classifier [10] of the form of the one in the cost function.

Note also that more generic similarity measure learning techniques such as RCA one or ITML can also be used to learn a Mahalanobis similarity measure in that context.

In Chapter 4, we focus on a specific type of task where the need for a good similarity measure arises: partitioning ones. Namely, the goal is to group the data into clusters such that the sum of some intra-class similarities is the lowest. For the specific cases of change-point detection and K-means distortion (see e.g, [Hastie et al., 2009]), we propose to learn a metric in a fully supervised way.

In Chapter 5, we focus on the way to learn a good similarity measure for the time warping problem

10. That is a classifier, that gives to a new instance x the same label as a training instance x^i drawn according to $p_x(x^i) = \frac{\exp(-\|Lx^i - Lx\|_2^2)}{\sum_{k\neq i} \exp(-\|Lx^i - Lx^k\|_2^2)}$.

Chapter 4

Similarity Measure Learning for Constrained Partitioning Problems

Abstract

We consider unsupervised partitioning problems based explicitly or implicitly on the minimization of Euclidean distortions, such as clustering, image or video segmentation, and other change-point detection problems. We emphasize on cases with specific structure, which include many practical situations ranging from mean-based change-point detection to image segmentation problems. We aim at learning a Mahalanobis metric for these unsupervised problems, leading to feature weighting and/or selection. This is done in a supervised way by assuming the availability of several (partially) labeled datasets that share the same metric. We cast the similarity measure learning problem as a large-margin structured prediction problem, with proper definition of regularizers and losses, leading to a convex optimization problem which can be solved efficiently. Our experiments show how learning the metric can significantly improve performance on bioinformatics, video or image segmentation problems.

Most of the material of this chapter is based on the conference paper: R. Lajugie, S. Arlot, F. Bach, Large-Margin Metric Learning for Constrained Partitioning Problems, In *Proc. ICML*, 2014.

4.1 Introduction

Unsupervised partitioning problems are ubiquitous in machine learning and other data-oriented fields such as computer vision, bioinformatics or signal processing. Their goal is to produce a partition of the data such that the subsets [1] have some high-level interpretation. They include:

1. That are often referred to as *clusters*.

1. traditional *unsupervised clustering* problems, with the classical K-means algorithm, hierarchical linkage methods [Gower and Ross, 1969] and spectral clustering [Ng et al., 2002],
2. *unsupervised image segmentation* problems where two neighbouring pixels are encouraged to be in the same cluster, with mean-shift, techniques [Cheng, 1995] or normalized cuts [Shi and Malik, 1997],
3. *change-point detection* problems adapted to multivariate sequences (such as video) where segments are composed of contiguous elements, with typical window-based algorithms [Desobry et al., 2005] and various methods looking for a change in the mean of some features [Chen and Gupta, 2011].

All the algorithms mentioned above rely on a specific distance (or more generally a similarity measure) on the space of configurations. Using a good distance is crucial to their performance, especially in high-dimensional settings where many dimensions may be irrelevant to the partitioning task. Its choice is heavily problem-dependent. While the design of such a similarity measure has originally been tackled manually (often by trial and error), recent work has considered learning such similarity measure directly from data. Without any supervision, the problem is ill-posed and methods based on generative models may learn a metric or reduce dimensionality (see, e.g., [De la Torre and Kanade, 2006]), but in general with no guarantees that they lead to better partitions. In this chapter, we consider the same goal as Bar-Hillel et al. [2005], Xing et al. [2002] ,Finley and Joachims [2005] and , Bach and Jordan [2006] that is learning a similarity measure for one or several partitioning problems sharing a common geometry, assuming that one or several fully (or partially) labeled partitioned datasets are available during the learning phase. While such labeled datasets are expensive to produce, there are several scenarios where those datasets have already been built, often for evaluation purposes. These occur for video segmentation tasks (see Section 4.5.4), image segmentation tasks (Section 4.6.2) as well as change-point detection tasks in bioinformatics (see [Hocking et al., 2013] and Section 4.4.2). This global framework is sometimes referred to as *supervised clustering* [Finley and Joachims, 2005, 2008].

Related work. The need for similarity measure learning goes far beyond unsupervised partitioning problems. Weinberger et al. [2006] proposed a large-margin framework for learning a quasimetric [2] specifically designed for nearest-neighbours algorithms based on sets of must-link/cannot-link constraints, while Goldberger et al. [2004] considered a probability-based non-convex formulation. For these frameworks, a single dataset is fully labeled and the goal is to learn a metric leading to good testing performance on unseen data. Similarity measure learning has also been considered in semi-supervised clustering of a single dataset, where some partial constraints are given. This includes the works of Bar-Hillel et al. [2005] and the non-convex approach of Xing et al. [2002]. As shown in Section 5.5, these can be used in our setting as well by stacking several datasets into a single one. However,

2. A quasimetric satisfies all the axioms of a distance except the separation one. Please note that the vocabulary of similarity measure learning is recalled in Sec. 3.4.1 of chapter 3.

our discriminative large-margin approach outperforms these, because we consider explicitly the clustering performance, for each dataset. Moreover, these approaches cannot readily use additional prior knowledge on the partitions such as its potential sequential structure.

The task of learning how to partition has also been tackled by Bach and Jordan [2006] for spectral clustering. The problem set-up is the same (availability of several fully partitioned datasets). However, their formulation is non-convex and relies on the unstable optimization of eigenvectors.

Finley and Joachims [2005] considered the same convex large-margin set-up as ours but for correlation clustering, a clustering method based on greedy algorithms or convex relaxations. This approach can be seen as special case of our work, when the number of subsets of the partition is known in advance and when the optimization is conducted in an approximate manner [3].

Other approaches do not require any supervision [De la Torre and Kanade, 2006], and perform simultaneous dimensionality reduction and clustering, by alternating between the computation of a low-rank matrix and clustering of the data using the corresponding metric. However, they cannot take advantage of the labeled information that we use.

Our approach can also be related to the work of Szummer and Hoiem [2008]: given a small set of labeled instances, they use a similar large-margin framework, inspired by Tsochantaridis et al. [2005] and Taskar et al. [2003], to learn parameters of Markov random field, using graph cuts for solving the *loss-augmented decoding* problem of structured prediction. However, their segmentation framework does not apply to unsupervised segmentation, which is one of the goals of this chapter. In this chapter, we present a supervised learning framework aiming at learning how to perform an unsupervised task.

Structured SVM have been used to solve other learning problems, for instance to learn weights for graph matching [Caetano et al., 2009] or a metric for ranking tasks [Mcfee and Lanckriet, 2010]. In computer vision, it has also been used to build task-driven image representations [Kim et al., 2013].

Beyond existing approaches. The existing approaches for learning the metric for unsupervised partitioning problems in a supervised way have two main drawbacks:

1. they are unable to deal clearly with the common case in which the number of clusters in the data is unknown a priori except in the case of Finley and Joachims [2005] for which unknown number of cluster is indirectly taken into account,
2. they cannot make use of any prior knowledge concerning the shape of partitions, which is a significant limitation because in most applications, such an extra prior information [4] is available.

3. See Sec. 4.6.2 for a more detailed comparison

4. For instance, in image segmentation this can be the spatial contiguity of the subsets, and for detecting changes in the distribution of time series, such a prior is the sequential contiguity of subsets.

Dealing with unknown number of clusters. None of the aforementioned methods is suited for learning a penalty term for selecting the number of clusters, as they do not include any model selection term. However, the scenario where the number of clusters is unknown is in practice very common. For instance in bioinformatics for a-CGH segmentation [Hocking et al., 2013], it is unrealistic to assume to know a priori in how many segments a sequence should be split. The same remark holds for the segmentation of long videos: at test time, it is not realistic to assume the number of segments is known. We explore these applications in Sections 4.5.4 and 4.5.5.

Hard priors. A common prior is the sequential structure that can be found everywhere in signal processing, from audio [Gillet et al., 2007] to bioinformatics with a-CGH or EEG segmentation [Hocking et al., 2013, Brodsky and Darkhovsky, 1993]. Our work focuses on hard-coding such a prior by restricting the set of authorized partitions. This leads to the well-known change-point detection problem. In particular, we show that in this case and using a similar structured SVM as in Finley and Joachims [2008], the loss-augmented decoding problem can be solved *exactly* in polynomial time using a dynamic programming algorithm.

Soft priors. A popular application of clustering is image segmentation. Unfortunately, simple K-means-based algorithms do not take into account the two-dimensional structure and do not lead to meaningful partitions. In this chapter, we consider adding a Laplacian-based penalty term that takes into account the spatial structure, for which the similarity measure can be learned efficiently.

Summary. We make the following contributions:

- We propose an efficient algorithm, based on the large-margin framework for structured prediction of Tsochantaridis et al. [2005] that learns a similarity measure for some commonly used partitioning problem. To that extent, we assume to be given several labeled datasets sharing the same metric. Our algorithm chooses automatically the number of clusters at test time, and can deal with hard or soft structural priors about the partitions. We first focus on a structural *sequential* prior, that is closely related to the standard mean-based change-point detection problem. Then, we consider a structural spatial prior that we apply to an image segmentation problem. Experiments in Section 4.5 show that our algorithm leads to a metric that significantly improves the performance over the Euclidean one. We support our claim by experiments as well on synthetic examples as well as on real-world ones.
- When imposing that subsets of the partition should have some sequential contiguity (change-point detection), we propose a dynamic programming algorithm that can solve the loss-augmented decoding problem in polynomial time (Algorithm 7).
- We try to leverage our approach to a much broader setting by going beyond full supervision. We propose an extension of our algorithm that can learn a similarity measure from *partially labeled* datasets (Section 4.4.2).

– We show in Section 4.4.2 how our approach can help in detecting changes in the full distribution of univariate time series, rather than only in the mean. We propose an application to the segmentation of genomics data in bioinformatics.

4.2 Partitioning through matrix factorization

In this section, we consider T multi-dimensional observations $x_1, \dots, x_T \in \mathbb{R}^p$, which may be represented in a design matrix $X \in \mathbb{R}^{T \times p}$. Partitioning the T observations into K *clusters* or subsets is equivalent to finding an *assignment matrix*

$$Y \in \{0,1\}^{T \times K},$$

such that $Y_{ij} = 1$ if the i-th observation is assigned to cluster j and $Y_{i,j} = 0$ otherwise. For general partitioning problems, no additional constraints are used, but for change-point detection problems, it is assumed that the subsets are contiguous and with increasing labels. That is, the matrix Y is block-diagonal with each block equals to $\mathbf{1}_{T_j}$, where $\mathbf{1}_{T_j} \in \mathbb{R}^{T_j}$ is the T_j-dimensional vector with constant components equal to one, and T_j is the number of elements in cluster j. For any partition, we may re-order the data points so that the assignment matrix has the same form; this is useful for the understanding of partitioning problems. For instance, for a change-point detection task, an authorized assignment matrix for a partition in $K = 3$ contiguous subsets can be:

$$Y = \begin{pmatrix} \mathbf{1}_{T_1} & 0 & 0 \\ \vdots & \mathbf{1}_{T_2} & \vdots \\ 0 & 0 & \mathbf{1}_{T_3} \end{pmatrix},$$

4.2.1 Distortion measure

In this chapter, we consider partitioning problems where each datapoint in cluster j is modeled by a vector (often called a centroid or a mean) $c_j \in \mathbb{R}^p$, the overall goal being to find a partition and a set of means so that the distortion measure

$$\sum_{i=1}^{T} \sum_{j=1}^{K} Y_{ij} \|x_i - c_j\|^2 = \sum_{i=1}^{T} \sum_{j=1}^{K} Y_{ij} \|X_{i,\cdot} - c_j\|^2$$

is as small as possible. [5]. By considering the Frobenius norm defined for $A \in \mathbb{R}^{T \times p}$ through $\|A\|_F^2 = \mathrm{Tr}(AA^\top) = \sum_{i=1}^{T} \sum_{j=1}^{p} A_{ij}^2$, this is equivalent to minimizing

$$\|X - YC\|_F^2 \tag{4.1}$$

5. Recall that here $\|\cdot\|$ is the Euclidean norm of $\mathbb{R}^p$.

with respect to assignment matrices Y and centroid matrix $C \in \mathbb{R}^{K\times p}$. This is a matrix factorization problem where the goal is to represent X more compactly.

4.2.2 Representing partitions

Following Bach and Jordan [2006], the quadratic minimization problem in C can be solved in closed form, with solution [6]

$$C = (Y^\top Y)^{-1} Y^\top X.$$

So, the partitioning problem that aims to minimize Eq. (4.1) when the number K of clusters is set by advance, is equivalent to:

$$\underset{Y\in\{0,1\}^{T\times K},\ Y\mathbf{1}_K=\mathbf{1}_p}{\text{minimize}} \|X - Y(Y^\top Y)^{-1}Y^\top X\|_F^2 \ . \tag{4.2}$$

Thus, the problem is naturally parameterized by the $T \times T$-matrix $M = Y(Y^\top Y)^{-1}Y^\top$. The output of any algorithm aiming at solving this problem can thus be represented as such a matrix. This, which we refer to as a *rescaled equivalence matrix*, has a specific structure. The matrix $Y^\top Y$ is diagonal, with i-th diagonal element equal to the number of elements in the i-th cluster. Thus, $M_{ij} = 0$ if i and j are in different clusters, otherwise $M_{i,j}$ equals to $1/T_j$ where T_j is the number of elements in the cluster containing the i-th data point. If the points are re-ordered so that the K subsets have a contiguity structure [7], then M is block-diagonal with blocks $\frac{1}{T_j}\mathbf{1}\mathbf{1}^\top$, $j = 1, \dots, K$. For instance M can have the following form

$$M = \begin{pmatrix} \frac{1}{T_1}\mathbf{1}\mathbf{1}^\top & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \frac{1}{T_K}\mathbf{1}\mathbf{1}^\top \end{pmatrix}. \tag{4.3}$$

In this chapter, we represent partitions through this rescaled equivalence matrix. Note its difference with alternative representations such as the equivalence matrix $YY^\top$ which has the same non-zero entries but whose values are in $\{0, 1\}$. This is a commonly used representation for image segmentation [Joulin et al., 2010].

We denote by $\mathcal{M}_K$ the set of rescaled equivalence matrices with K clusters, i.e., matrices $M = Y(Y^\top Y)^{-1}Y^\top \in \mathbb{R}^{T\times T}$ for some assignment matrix $Y \in \mathbb{R}^{T\times K}$. For

6. This can be found by computing the gradient with respect to C of the distortion measure of Eq. (4.1) and setting it to zero. This gradient is indeed $g(C) = 2Y^\top YC - 2Y^\top X$.

7. Formally, for a given partition in K subsets, let us call $c : \{1, \dots, T\} \to \{1, K\}$ the mapping that assigns each index i of a datapoint x_i to the index of the subset of the partition to which it is assigned. Thus, reordering means that we find a permutation π of $\{1, \dots, T\}$ such that

$$\forall k \in \{1, \dots, K\}, \{t,\ c(\pi(t)) = k\}$$

is a contiguous subset of $\{1, \dots, T\}$.

situations where the number of clusters is unspecified, we denote by $\mathcal{M}$ the union of all $\mathcal{M}_K$ for $K \in \{1, \dots, T\}$.

Note that the number of clusters may be obtained from M, since for every $M \in \mathcal{M}_K$

$$\operatorname{Tr}(M) = \operatorname{Tr}\big(Y(Y^\top Y)^{-1}Y^\top\big) = \operatorname{Tr}\big((Y^\top Y)^{-1}Y^\top Y\big) = K.$$

This can also be seen by noticing that $M^2 = Y(Y^\top Y)^{-1}Y^\top Y(Y^\top Y)^{-1}Y^\top = M$, i.e., M is a projection matrix, with eigenvalues in $\{0, 1\}$, and the number of eigenvalues equal to one is exactly the number of clusters. Thus,

$$\mathcal{M}_K = \big\{M \in \mathcal{M},\ \operatorname{Tr} M = K\big\}.$$

Learning the number of clusters K. Given the number of clusters K, we have seen from Eq. (4.2) that the partitioning problem consists in

$$\underset{M \in \mathcal{M}_K}{\text{minimize}} \|X - MX\|_F^2, . \tag{4.4}$$

which is equivalent to:

$$\underset{M \in \mathcal{M}_K}{\text{minimize}} \operatorname{Tr}\big[XX^\top(I - M)\big].$$

Note that in change-point detection problems, an extra constraint of contiguity of subsets is added (see Section 4.2.3).

In the common situation where the number of clusters K is unknown, it may be estimated directly from data by penalizing the distortion measure with a term proportional to the number of clusters. This is an usual approach in the change-point detection literature [Lavielle, 2005], and can be traced back to the AIC criterion [Akaike, 1974]. Given that the number of clusters for a rescaled equivalence matrix M is $\operatorname{Tr} M$, this leads to the following penalized formulation:

$$\underset{M \in \mathcal{M}}{\text{minimize}} \left\{ \operatorname{Tr}\big[XX^\top(I - M)\big] + \lambda \operatorname{Tr} M \ \right\}. \tag{4.5}$$

The extra-parameter λ has to be finely tuned. Some approaches do it a priori [Lebarbier, 2005], without using any supplementary data. The similarity measure learning algorithm of Section 4.3 learns this extra parameter λ, but makes use of annotated data.

Thus, the two types of partitioning problems (with fixed or unknown number of clusters) can be cast as the problem of maximizing a linear function of the form $\operatorname{Tr}(AM)$ with respect to $M \in \mathcal{M}$, with the potential constraint that $\operatorname{Tr} M = K$. In general, such optimization problems can not be solved in polynomial time. In Section 4.2.3, we present a polynomial-time dynamic programming approach that can solve the problem with additional hard contiguity constraint. For general situations, the K-means algorithm, although not exact, can be used to get a good partitioning in polynomial time when A is positive semidefinite. In Section 4.2.4, we provide a spectral relaxation, which can be used when adding a soft constraint.

We now consider two variants of the Euclidean distortion model for partitioning that we have presented so far by including in it some knowledge directly coming from the precise partitioning task at hand. We also introduce two algorithmic ways to deal with the problem of recovering the partition that minimizes the distortion. This task is called in what follows the *decoding* of the partitioning algorithm.

4.2.3 Hard prior : change-point detection by dynamic programming

The change-point detection problem is a restriction of the general partitioning problem where the subsets are composed of contiguous elements. In that case we often talk about *segments*. We denote by $\mathcal{M}^{\text{seq}}$ the set of all rescaled equivalence matrices satisfying the contiguity structure of the change-point detection problem, and $\mathcal{M}_K^{\text{seq}}$ its restriction to partitions with K segments.

The problem is thus solving Eq. (4.4) (known number of clusters) or Eq. (4.5) (unknown number of clusters) with the extra constraint that $M \in \mathcal{M}^{\text{seq}}$. This may be cast as maximizing $\text{Tr}(AM)$ with respect to $\mathcal{M}_K^{\text{seq}}$ or $\mathcal{M}_K$, for a certain matrix A. For the case where $M \in \mathcal{M}_K$, and when A is positive definite with a known square root we have seen in Alg. 5 of Chapter 3 that it was possible to solve the problem *exactly* using a polynomial-time algorithm based on dynamic programming (see, e.g., Rigaill [2010], Killick et al. [2012]). However, both for Eq. (4.5) and more generally for all loss-augmented decoding problems in Section 4.3, we need to maximize $\text{Tr}(AM)$ where A can be any matrix.

Algorithm 7 solves $\max_{M \in \mathcal{M}^{\text{seq}}} \text{Tr}(AM)$ for any matrix A, potentially with negative eigenvalues. This algorithm is a dynamic programming one for an analoguous reason as Alg. 5 of chapter 3. It has complexity $O(T^2)$.

Algorithm 7 only requires some preprocessing of the input matrix A, namely computing its summed area table I (or image integral [Crow, 1984]), defined to have the same size as A and such that $I_{ij} = \sum_{i' \leq i,\ j' \leq j} A_{i'j'}$ (i.e., the sum of the elements of A which are above i and to the left of j). A similar algorithm can be derived in the case where $M \in \mathcal{M}_K^{\text{seq}}$, with complexity $O(KT^2)$. It has been described in details in Alg. 5 in Chapter 3.

4.2.4 Soft priors: Laplacian based penalty

We recall that the so-called K-means problem aims at finding a partition of T datapoints in K subsets or clusters in which the intra-class variance is minimal. Thus this is the problem of minimizing the distortion of Eq. (4.4).

K-means distortion with structural spatial prior. Many real-life partitioning tasks come with a strong prior about the shape of the partition that should be outputed.

Algorithm 7 Dynamic programming for maximizing $\mathrm{Tr}(AM)$ such that $M \in \mathcal{M}^{\mathrm{seq}}$

Input: Cost matrix $A \in \mathbb{R}^{T\times T}$ and its image integral I
$(I_{k,j} = \sum_{p_1\leq k, p_2\leq j} A_{p_1,p_2})$
$\forall i \in [[1 \dots T]]$, initialize $C(1,i) = I(i,i)/i$.
for $t = 1$ to $T-1$ **do**
  $m = \max_{i\leq t} C(i,t)$
  for $u = t+1$ **to** T **do**
    $C(t+1,u) = \frac{I(t,t)+I(u,u)-I(u,t)+I(t,u)}{u-t} + m$
  end for
end for
Backtracking step:
$j = 1$, $T_c(1) = T$
repeat
  $j = j+1$
  $T_c(j+1) = \mathrm{argmax}_{i<T_c(j)} C(i, T_c(j))$
until $T_c(j) > 1$
Output: Time of changes T_c

In some cases, the prior can be encoded through affinities of a graph [8]. This spatial prior can be encoded in a soft manner very easily as a penalization of the K-means objective. Namely, this corresponds to optimizing the following penalized model, where L is the Laplacian matrix of a certain graph:

$$\underset{M\in\mathcal{M}_K}{\text{minimize}} \left\{ \|X - MX\|_F^2 + \mu\,\mathrm{Tr}(LM) \right\} \Leftrightarrow \underset{M\in\mathcal{M}_K}{\text{maximize}}\ \mathrm{Tr}\big((XX^\top - \mu L)M\big). \quad (4.6)$$

Since the problem (4.6) is NP-hard [Aloise et al., 2009], we need to approximately solve it. A first idea could be to use the popular K-means algorithm. However, K-means can only be cast as solving linear programs of the form

$$\underset{M\in\mathcal{M}_K}{\text{maximize}}\, \mathrm{Tr}(AM)$$

where A is positive semidefinite. In our case, due to the Laplacian penalty, we need to solve such LPs with A an arbitrary symmetric matrix. Thus, instead of using K-means, we propose to relax the K-means distortion. Following Shi and Malik [1997] and Ng et al. [2002], we now present a spectral relaxation of this problem. This is done by relaxing the set $\mathcal{M}$ to the set of matrices that satisfy $M^2 = M$ (i.e., removing the constraint that M takes a finite number of distinct values). When the number of clusters is known, this leads to the classical spectral relaxation. We have indeed that:

$$\max_{M\in\mathcal{M},\ \mathrm{Tr}\, M=K} \mathrm{Tr}(AM) \leqslant \max_{M^2=M,\ \mathrm{Tr}\, M=K} \mathrm{Tr}(AM),$$

8. In computer vision, the graph of pixels is a very common manner to set priors over segmentations and that way to enforce spatial contiguity [Shi and Malik, 1997].

which is equal to the sum of the K largest eigenvalues of A. The optimal matrix M of the spectral relaxation is the orthogonal projector on the span of the eigenvectors of A with K largest eigenvalues.

When the number of clusters is unknown, we can penalize the distortion cost in Eq. (4.6) by the same term as in Eq. (4.5) and consider the whole set $\mathcal{M}$. Then we have

$$\max_{M \in \mathcal{M}} \operatorname{Tr}(AM) \leqslant \max_{M^2 = M} \operatorname{Tr}(AM) = \operatorname{Tr}(A)_+,$$

where $\operatorname{Tr}(A)_+$ is the sum of positive eigenvalues of A. The optimal matrix M of the spectral relaxation is the orthogonal projector on the span of the eigenvectors of A with positive eigenvalues. Note that in the formulation from Eq. (4.5), this corresponds to thresholding all eigenvalues of $XX^\top$ which are less than λ.

We denote by

$$\mathcal{M}^{\text{spec}} = \{M \in \mathbb{R}^{T \times T},\ M^2 = M\},$$

and

$$\mathcal{M}_K^{\text{spec}} = \{M \in \mathbb{R}^{T \times T},\ M^2 = M,\ \operatorname{Tr} M = K\},$$

the relaxed sets of rescaled equivalence matrices.

Spectral decoding. From the relaxed solution $M \in \mathcal{M}_K^{\text{spec}}$, it can sometimes be of interest to get $M' \in \mathcal{M}_K$. This problem is closely related to spectral clustering [Ng et al., 2002], and one way to obtain an hard assignment is to run K-means over the K leading eigenvectors of the spectral solution.

4.2.5 Similarity measure learning

In this chapter, we consider the learning of a *Mahalanobis quasimetric*[9], which may be parameterized by a positive definite matrix $B \in \mathbb{R}^{p \times p}$. Using a similarity measure given by B is equivalent to replacing the dot-products $x_i^\top x_j$ by $x_i^\top B x_j$, and $XX^\top$ by $XBX^\top$. In the general case, this corresponds to replacing Eq. (4.4) by:

$$\underset{M \in \mathcal{M}_K}{\text{minimize}} \operatorname{Tr}\big(XBX^\top (I - M)\big)\ . \tag{4.7}$$

Thus, in the case of the sequential hardcoded prior of Section 4.2.3, this corresponds to:

$$\underset{M \in \mathcal{M}^{\text{seq}}}{\text{minimize}} \operatorname{Tr}\big(XBX^\top (I - M)\big)\ . \tag{4.8}$$

When the number of segments is unknown, we penalize by adding $\lambda \operatorname{Tr} M$:

$$\underset{M \in \mathcal{M}_K^{\text{seq}}}{\text{minimize}} \operatorname{Tr}\big(XBX^\top (I - M)\big) + \lambda \operatorname{Tr}(M).$$

9. For more details about these, please refer to chapter 3.

In this case, note that replacing B by λB and dividing the equation by λ, we can use an equivalent formulation with $\lambda = 1$, that is:

$$\underset{M \in \mathcal{M}^{\text{seq}}}{\text{minimize}} \operatorname{Tr}\big(XBX^\top(I - M)\big) + \operatorname{Tr}(M). \tag{4.9}$$

For the soft prior of Section 4.2.4, the corresponding minimization problem becomes

$$\underset{M \in \mathcal{M}_K^{\text{spec}}}{\text{minimize}} \operatorname{Tr}\big(XBX^\top(I - M) + \mu LM\big), \tag{4.10}$$

when the number of clusters is known and

$$\underset{M \in \mathcal{M}^{\text{spec}}}{\text{minimize}} \operatorname{Tr}\big(XBX^\top(I - M) + \mu LM\big) + \operatorname{Tr}(M), \tag{4.11}$$

when it is not. The key aspect of the partitioning problem is that it is formulated as optimizing with respect to M a function *linearly* parametrized by B. The linear parametrization in M will also be useful when defining proper losses and efficient loss-augmented decoding in Section 4.3.

Note that we may allow B to be just positive semi-definite. In that case, the zero-eigenvalues of the quasi-metric correspond to irrelevant directions. This means in particular that we have performed dimensionality reduction on the input data. We propose a simple way to encourage this desirable property in Section 4.3.3. The associated similarity measure is not a metric anymore but a *quasimetric*.

4.3 Structured prediction for similarity measure learning

We want to learn a matrix B to maximize the performance of the structured output algorithm that solves the minimization problems of Eq. (4.8), Eq. (4.9), Eq. (4.10) or Eq. (4.11). All these partitioning problems can be cast as

$$\underset{M \in \mathcal{N}}{\text{maximize}} \ \langle W, \varphi(X, M) \rangle,$$

where $\langle A, B \rangle = \operatorname{Tr}(A^\top B)$ is the Frobenius dot product for any A and B with compatible dimensions and $\mathcal{N}$ is a set among $\mathcal{M}$, $\mathcal{M}^{\text{spec}}$, $\mathcal{M}^{\text{seq}}$, $\mathcal{M}_K$, $\mathcal{M}_K^{\text{spec}}$, $\mathcal{M}_K^{\text{seq}}$

For the general case ($\mathcal{N} = \mathcal{M}_K^{\text{seq}}$) and the one corresponding to change-point detection ($\mathcal{N} = \mathcal{M}_K^{\text{seq}}$), when the number of segments is known, then Eq. (4.7) and Eq. (4.8) corresponds to

$$\begin{cases} \varphi(X, M) = X^\top M X \\ W = B \end{cases}.$$

If the number of segments is unknown (case $\mathcal{N} = \mathcal{M}$ or $\mathcal{N} = \mathcal{M}^{\text{seq}}$) then Eq. (4.9)

as well as the general case can be dealt with:

$$\begin{cases} \varphi(X, M) = \mathrm{Diag}(X^\top M X, M) \\ W = \mathrm{Diag}(B, -I) \end{cases}.$$

For the model corresponding to the spatial prior for partitioning (Eq. (4.10)), when the number of clusters is known ($\mathcal{N} = \mathcal{M}_K^{\mathrm{spec}}$), we get

$$\begin{cases} W = \mathrm{Diag}(B, -I) \\ \varphi(X, M) = \mathrm{Diag}(X^\top M X, \mu L) \end{cases},$$

and if the number is unknown (Eq. (4.11)), ($\mathcal{N} = \mathcal{M}^{\mathrm{spec}}$)

$$\begin{cases} W = \mathrm{Diag}(B, -I) \\ \varphi(X, M) = \mathrm{Diag}(X^\top M X, M + \mu L) \end{cases}.$$

4.3.1 Large-margin structured output learning

Let $\mathcal{F}$ denote the set where the above-defined W lies. Our goal is to estimate $W \in \mathcal{F}$ from N pairs of observations $(X^i, M^i) \in \mathcal{X} \times \mathcal{M}$. This is exactly what large-margin structured prediction [Tsochantaridis et al., 2005] does. In the margin-rescaling framework of Tsochantaridis et al. [2005], using a loss

$$\ell : \mathcal{N} \times \mathcal{N} \to \mathbb{R}_+$$

between elements of $\mathcal{N}$ (here partitions), the learning is performed by minimizing with respect to $W \in \mathcal{F}$,

$$\frac{1}{N} \sum_{i=1}^{N} \ell\Big(\operatorname*{argmax}_{M \in \mathcal{N}} \big\langle W, \varphi(X^i, M) \big\rangle, M^i\Big) + \Omega(W),$$

where Ω is any (convex) regularizer. This framework is standard in machine learning in general and similarity measure learning in particular (see, e.g, Jain et al. [2012]). The loss function

$$W \mapsto \ell\Big(\operatorname*{argmax}_{M \in \mathcal{N}} \big\langle W, \varphi(X^i, M) \big\rangle, M^i\Big)$$

is not convex in W, and can be replaced by the convex surrogate

$$L_i(W) = \max_{M \in \mathcal{N}} \Big\{ \ell(M, M^i) + \langle W, \varphi(X^i, M) - \varphi(X^i, M^i) \rangle \Big\},$$

leading to an estimator $\widehat{W}$ minimizing

$$\frac{1}{N}\sum_{i=1}^{N} L_i(W) + \Omega(w). \tag{4.12}$$

In order to apply this framework, several elements are needed:

1. a regularizer Ω,
2. a loss function ℓ,
3. the associated efficient algorithm for computing L_i, i.e., solving the *loss-augmented decoding* problem

$$\underset{M\in\mathcal{N}}{\text{maximize}}\Big\{\ell(M, M^i) + \big\langle W, \varphi(X^i, M) - \varphi(X^i, M^i)\big\rangle\Big\}.$$

Optimization in W. Given that the objective function is not smooth, we have used projected subgradient descent (stochastic and non-stochastic), with convergence rates of $O(1/t)$ after t iterations [Shalev-Shwartz et al., 2007]. However, this method relies on the fine tuning of the decaying stepsize at each iteration. The last implementation we have done and which we used for the experiments of Sec. 4.6.1 makes use of the stochastic block-coordinate Frank-Wolfe algorithm for structured SVM [Lacoste-Julien et al., 2013]. This allows us to get rid of the problem of tuning the subgradient descent parameter that plagues stochastic subgradient techniques.

Now, we need to find a suitable loss for our large-margin objective.

4.3.2 Loss between partitions

The Rand index. When comparing two partitions [Hubert and Arabie., 1985], a standard way to measure how different they are is to use the Rand index [Rand, 1971]. Let us consider two partitions P and Q of the same set of T elements $\{x_1, \ldots, x_T\}$. These partitions are collections of subsets $P_1, \ldots, P_K$ and $Q_1, \ldots, Q_L$. The *Rand index* is the agreement score between the two partitions. Let us define the *concordant pairs* as the number of pairs of elements which either (a) both belong to the same subset in P and Q or (b) are in different subsets both in P and Q. The Rand Index is the ratio of the number of concordant pairs over the total number of pairs. In matrix terms, if we represent the partition P by an assignment matrix [10] Y_P and the partition Q by Y_Q, the Rand index is

$$\text{Rand}(P, Q) = 1 - \frac{1}{T(T-1)}\left\|Y_P Y_P^\top - Y_Q Y_Q^\top\right\|_2^2.$$

10. For completeness, we recall that the assignment matrix is a $T \times K$ binary matrix such that $Y_{t,k} = 1$ if, and only if, element x_t is in cluster j of the partition.

From this index, it is easy to build a loss

$$d_{\text{Rand}}(P,Q) = 1 - \text{Rand}(P,Q) = \frac{1}{T(T-1)} \left\| Y_P Y_P^\top - Y_Q Y_Q^\top \right\|_2^2.$$

The Rand loss is proportional to the Frobenius square norm between the equivalence matrices $Y_P Y_P^\top$ and $Y_Q Y_Q^\top$. An equivalence matrix is a $T \times T$ matrix whose (i,j) coefficient is 1 if i and j are in the same subset of the partitions and 0 otherwise. In comparison with the rescaled equivalence matrix of Eq. (4.3), up to a permutation of rows and columns, it has the same form but with diagonal blocks non rescaled. Namely it is a matrix of the form:

$$\begin{pmatrix} \mathbf{1}_{T_1}\mathbf{1}_{T_1}^\top & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \mathbf{1}_{T_K}\mathbf{1}_{T_K}^\top \end{pmatrix}.$$

Frobenius loss between partitions. In this chapter, we make an extensive use of the following loss, based on rescaled equivalence matrices:

$$\ell_F(M,N) = \|M-N\|_F^2 = \text{Tr}(M) + \text{Tr}(N) - 2\,\text{Tr}(MN). \tag{4.13}$$

This, that we refer to as the *Frobenius loss* is practical (it is a bilinear function of M and N) and corresponds to a well-known loss between partitions [Hubert and Arabie., 1985, Bach and Jordan, 2006]. If the partitions encoded by M and N are collections of clusters $P_1, \dots, P_K$ and $Q_1, \dots, Q_L$, then

$$\ell_F(M,N) = K + L - 2\sum_{k,l} \frac{|P_k \cap Q_l|^2}{|P_k|\,|Q_l|}.$$

This loss is equal to zero if and only if the partitions are equal, always larger than $|K-L|$ and smaller than $K+L$.

Note that the Frobenius loss is to rescaled equivalence matrices, up to a scaling, what the Rand loss is to equivalence matrices [Hubert and Arabie., 1985, Finley and Joachims, 2005]. Note however that the Rand index/loss is not necessarily well suited to our problem, since intuitively it does not treat similarly large and small clusters. Our concern is to optimize intra-class variance which is a rescaled indicator and for which each cluster (independently on its size) should have the same importance. [11]

11. This is a similar intuition than the one behind the difference between cuts and normalized cuts in graph, that are widely used in computer vision [Shi and Malik, 1997].

4.3.3 Regularizers and diagonal variant.

With a slight abuse of notation, we refer there to B as the upper $p \times p$ corner of the parameter W to learn. Since what we really want to regularize is B and not W, in this section we note W_B instead of W. Let us now consider several regularizers Ω for B. The most popular choice for Ω is the Frobenius norm (see, e.g, Tsochantaridis et al. [2005]).

Low-rank quasimetric. A desirable property for the matrix is to be interpretable. Ideally, we would like to have the quasimetric associated to the matrix W learned with a small rank. That is W selects only few but meaningful variables. The classical relaxation of the rank is the sum of the singular values, that is, $\Omega(W_B) = \mathrm{Tr}(B)$ since B is symmetric positive semi-definite.

Diagonal quasimetric. Considering only diagonal matrices $B = \mathrm{Diag}(b)$ for some $b \in \mathbb{R}^p$ with $b \geqslant 0$ limits the number of parameters to learn, and reduces the similarity measure learning problem to reweighing the coordinates of the data. Then, the two regularizers seen so far can be written $\|b\|_2^2$ for the Frobenius one and $\|b\|_1 = \mathbf{1}_p^\top b$ for the low-rank, the latter leading to variable selection.

Now, let us more precisely see how we can address the loss-augmented decoding problem. In the next section, we focus on the change-point detection case since it is possible to solve exactly the problem.

4.4 Learning a metric for change-point detection tasks

From now on, the loss used is set to be $\ell = \ell_F$ introduced in the previous section. To fix ideas, we recall the large-margin objective based on Eq. (4.12) in the case of change-point detection [12]

$$\frac{1}{N}\sum_{i=1}^{N} \max_{M \in \mathcal{N}} \Big\{ \mathrm{Tr}\big(M^i + M - 2M^i M\big) + \big\langle W, \phi(X^i, M) - \phi(X^i, M^i) \big\rangle \Big\} + \Omega(W). \quad (4.14)$$

Efficient minimization with respect to $W \in \mathcal{F}$ is key to the applicability of large-margin structured prediction. The bottleneck of the method lies in the loss-augmented decoding that is the evaluation of the quantity

$$\max_{M \in \mathcal{N}} \Big\{ \mathrm{Tr}\big(M^i + M - 2M^i M\big) + \big\langle W, \phi(X^i, M) - \phi(X^i, M^i) \big\rangle \Big\}.$$

12. Note that the sets $\mathcal{N}$ here can be either $\mathcal{M}^{\mathrm{seq}}$ or $\mathcal{M}_K^{\mathrm{seq}}$.

4.4.1 Loss-augmented decoding problem for change-point detection

In the case of change-point detection, the cardinality of $\mathcal{N}$ is exponential in T, a brute force approach is thus unrealistic in our case. However, our choice for the loss ℓ_F leads to considering a loss-augmented decoding problem of the form:

$$\underset{M \in \mathcal{N}}{\text{maximize}}\, \text{Tr}(A^i M),$$

where

$$A^i = (X^i B X^{i\top} - 2M^i + \text{Id}),$$

if the number of clusters is known, and

$$A^i = (X^i B X^{i\top} - 2M^i),$$

otherwise. Thus, the loss-augmented problem can be performed exactly for the change-point problems.

Thus, the problem of minimizing the loss-augmented decoding within the sum of Eq. (4.14) can be solved in polynomial time thanks to Alg. 7.

We now present two variants of the approach presented so far which make our similarity measure learning slightly more practical.

4.4.2 Variants

Partial labellings. The large-margin convex optimization framework we use relies on fully labeled datasets $(X^i, M^i)_{i=1,\dots,N}$ where X^i is a time series and M^i the corresponding rescaled equivalence matrix. In many situations however, only partial information is available about the partition associated to each X^i. We might only know parts of the segments. For instance we may know that timestamp i is in the same segment than $j > i$ (and thus that all the timestamps in between are in the same segment) and that timestamps $i' > j$ and $j' > i'$ are in the same segment, but with no precise idea of what is happening between timestamps j and i'. To deal with these cases, we propose the following heuristic:

1. we start from the PCA Mahalanobis quasimetric [13],
2. we label all datasets by doing a decoding step using the current quasimetric and respecting the constraints imposed by the partial labels, that is we complete the annotations by optimizing over the subsets of $\mathcal{M}^{\text{seq}}$ that correspond to segmentations respecting the partial annotation,
3. we learn a metric using our fully-supervised approach,
4. we alternate between the two last steps.

We propose a simulated example in Section 4.5.3.

13. Or any other "good guess". The choice of initialization is crucial since the method is not convex anymore. The PCA quasimetric is obtained by doing a PCA of the data and keeping only some eigenvectors.

Detecting changes in the distribution of temporal signals. The approach to change-point detection presented in Section 4.4.1 can only detect changes in the mean of the distribution of the x_j. Indeed, it only makes use of the distortion measure of Section 4.2.1, which is the sum of intra-class variances of each segment. Nevertheless, in the literature, change-point detection refers to the more general problem of finding changes in the whole distribution of the x_j [Basseville and Nikiforov, 1993]. One can be interested in detecting change-points in other features of the distribution like the variance or the kurtosis. In order to tackle this problem when the observation matrix is a 1-dimensional time series $X \in \mathbb{R}^T$, we propose to apply our approach by considering the multiple time series $(f_i(x_j))_{i=1\dots r} \in \mathbb{R}^r$, $j = 1 \dots T$, where the f_i are well-chosen functions so that changes in the distribution of x_j appear through changes in the mean of $f_i(x_j)$. For instance, in order to detect changes in the first moments of the distribution, a naive choice is $f_i(x) = x^i$, but the x_j^i explode when i grows [14]. A way to prevent this explosion is to use the robust Hermite moments [Welling, 2005], that is, to take

$$f_i(x) = H_i(x) = e^{\frac{x^2}{2}} \frac{d^i}{dx^i}\left(e^{\frac{-x^2}{2}}\right),$$

the i-th Hermite polynomial. See an application in Section 4.5.5. We recall there that the first five Hermite polynomials are:

$$\begin{aligned} H_0(x) &= 1, \\ H_1(x) &= x, \\ H_2(x) &= x^2 - 1, \\ H_3(x) &= x^3 - 3x, \\ H_4(x) &= x^4 - 6x^2 + 3. \end{aligned}$$

These polynomials are an orthonormal basis of the space of polynomials, and thus there is no lack of information when using these Hermite moments rather than the classical ones. In Fig. 4.1, we represent the Hermite polynomial of order 4 as an example. We see that these polynomials have a better behaviour than simple monomials since they send large values to infinity less quickly than the plain monomials, and in the neighbourhood of 0, values are not stuck to 0.

4.5 Experiments for change-point detection

We have conducted a series of experiments to demonstrate the interest of our approach for learning a similarity measure for change-point detection problems

14. Little values are stuck to 0 and large ones sent to infinity and this leads to computational issues quickly.

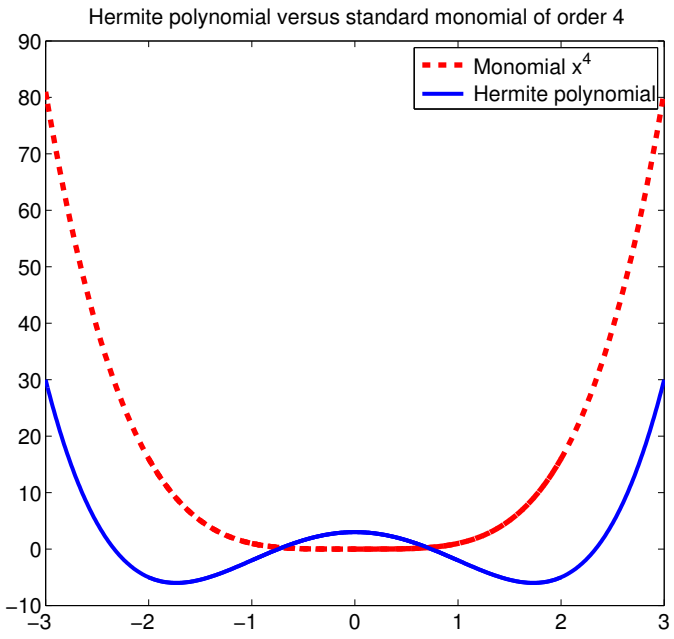


Figure 4.1: On this figure, we depict the Hermite polynomial of order 4, $x \to x^4 - 6x^2 + 3$ with the function $x \to x^4$. Note that the Hermite polynomial goes less fast to infinity and around 0 does not push values to 0.

4.5.1 Toy example: similarity measure learning quality

In this section, we show a case in which our similarity measure approach succeed in learning an interpretable Mahalanobis qusimetric and where usual methods based on side-information in the form of must-link or cannot-link constraints fail. In this section, we consider to assess the quality of the Mahalanobis quasimetric by looking at its coefficients in small dimension. So implicitely, we are looking at a feature selection task.

In this experiment, we consider $N = 10$ time series of length 350 in dimension three. In this experiments, the groundtruth segmentation has two change-points at timestamps 150 and 225. Only the first coordinate has relevant change-points (that is the groundtruth is obtained by performing a change-point detection algorithm using the first coordinate only), the second one is pure white noise, and the third one has a change-point that is *irrelevant*. See Fig. 4.2 for an illustration. Note also that the first coordinate has first and third segments with the same mean.

We ran three standard metric learning algorithms on these data: our approach where we assumed the number of segments to be known in advance and with $\lambda = 0$ (no regularization for this experiment), the RCA approach with groundtruth segments as chunklets and the approach of Xing et al. [2002] with must-link and cannot link constraints corresponding to the groundtruth segmentation. In this experiment, we consider the shape of the resulting matrix. A good similarity measure learning algorithm for this task is expected to provide a matrix that is close

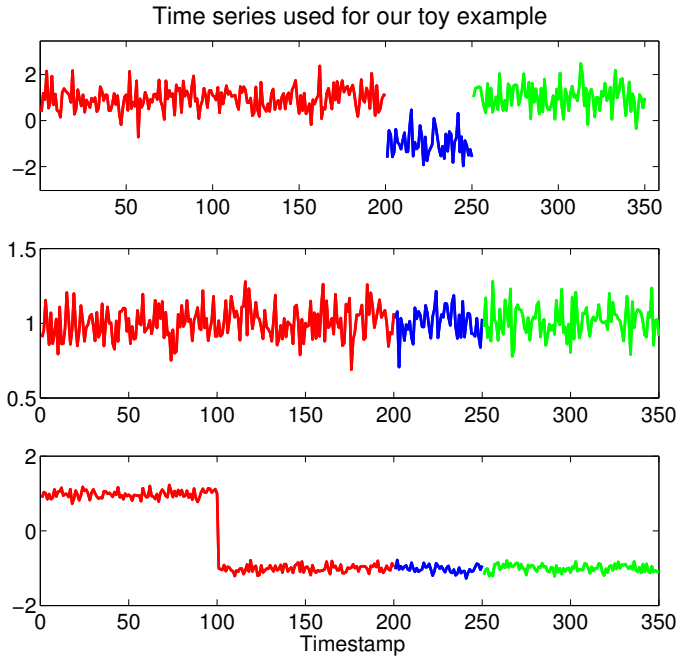


Figure 4.2: An example of the data we used in the experiment of Sec. 4.5.1. The time series is three dimensional, the relevant change-points (segmentation in colour) are only on the first coordinate, the two others are only noise or worse, with a change-point we do not want to detect.

to:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

That is, the matrix that we have in output should have a meaningful interpretation. The matrix given by our method is the following [15]:

$$\begin{pmatrix} 0.9421 & 0.2269 & -0.044 \\ 0.2269 & 0.1108 & 0.0389 \\ -0.0144 & 0.0389 & 0.0617 \end{pmatrix}.$$

The one given by RCA is:

$$\begin{pmatrix} 0.0348 & 0.0087 & 0.007 \\ 0.0087 & 0.9999 & 0.0026 \\ 0.0007 & 0.0026 & 0.010 \end{pmatrix}.$$

The matrix learned by RCA puts a lot of weight on a coordinate that is not relevant for the task and just pure noise. Note however that the RCA quasimetric puts a little weight on the coordinate that has a change we do not want to detect.

15. All matrices are rescaled to have unit Frobenius norm.

The Mahalanobis matrix corresponding to the quasimetric learned by Xing et al. [2002] is:

$$\begin{pmatrix} 0.2504 & 0.4328 & 0.007 \\ 0.4328 & 0.7483 & 0.0320 \\ 0.0185 & 0.0320 & 0.014 \end{pmatrix}.$$

Since this matrix is less easy to interpret than the two previous ones, let us have a look at its eigendecomposition. The first eigenvector, corresponding to more than 99% of eigenvalues is

$$\begin{pmatrix} 0.5004 \\ 0.8650 \\ 0.0370 \end{pmatrix}.$$

Thus the matrix associated to the metric learned by the method of Xing et al. [2002] puts also a high weight on the irrelevant second coordinate which is noise and can only disturb the change-point process.

4.5.2 Synthetic example

We consider synthetic time series of dimension $p = 50$ and length $T = 600$ with $K = 4$ relevant changes in the mean of a few coordinates for which we know the groundtruth optimal segmentation [16]. The non-relevant coordinates are ~ 45 noisy random series, either with no changes in their global mean (pure noise like in the second coordinate of the time series depicted in Fig. 4.2) or changes which are not aligned with the ones we aim at detecting (irrelevant changes like in the third coordinate of the time series depicted in Fig. 4.2). By learning a metric, we hope to obtain high weights on the relevant coordinates and small weights on the others.

Given $N = 30$ instances of such time series sharing the same relevant coordinates, we compare the performance of our algorithm to

1. the Euclidean metric that is, the penalized formulation of Eq. (4.5) with $B = \alpha I$ and $\alpha > 0$ learned on a validation set,
2. the PCA metric (obtained by performing a PCA, keeping the four leading eigenvectors),
3. the RCA approach of Bar-Hillel et al. [2005], for which we stack all datasets into a single one with the corresponding supervision, note that this approach only makes use of must-link constraints,
4. the learning algorithm of [Xing et al., 2002] which makes use of both must-link and cannot-link constraints. Note that all algorithms except ours are given the exact true number of change-points $K^\star$.

All Mahalanobis quasimetrics are learned on the same data. Performance measure reported there is assessed through the Frobenius loss ℓ_F. For our similarity

16. We call a change relevant if it is consistent with the groundtruth. On Fig. 4.2, the first coordinate has this property.

measure algorithm, we validate parameters on a validation set of 50 validation instances that shares the same relevant coordinate than the training examples. We test the performances by plugging the learned similarity measure in the change-point detection algorithm. For our approach and for the Euclidean metric, since we considered the penalized model of Eq. (4.5), we used Alg. 7 for decoding. For the others baselines, since they cannot learn the extra-penalization parameter, we used Alg. 5 of Chapter 3. Results are shown on Figure 4.3 (points at the extreme right of the graph), illustrating the interest of learning a metric (Euclidean is bad and PCA only slightly better), and showing our approach does better than RCA. The results are measured using the loss ℓ_F on a test set of 50 time series sharing the same metric as the one of training instances. Note that RCA or the method of Xing et al. [2002] are not specifically designed for change-point detection and have a wider applicability domain than our method.

4.5.3 Robustness to partial labeling

Using the same training/validation and testing data as for the above experiment, we tried the approach of Section 4.4.2 to deal with the problem of partial annotations. Indeed, it is really uncommon to have fully supervised and segmented temporal series. Results are presented on Figure 4.3. The x-axis represents the fraction of the labels $M^1, \ldots, M^N$ available to the two existing semi-supervised clustering methods as well to our algorithm. When the fraction of annotated data is large enough, our algorithm performs better since it takes into account for the underlying structure of the segmentation task at hand. But, when the fraction of supervised data is small (less than 10%) [17] our approach performed poorly compared to the other ones.

4.5.4 Video segmentation

We applied our method to data coming from old TV shows where some speaking passages alternate with singing ones. The videos are from 60 up to 90 minutes long so the temporal horizon T is of order of thousands, around 5400 in mean, with 60 to 120 change-points in each of these. We aim at recovering the segmentation induced by the speaking parts and the musical ones. Following the approach of Arlot et al. [2012], we use GIST features for the video part and MFCC features for the audio (13 leading coefficients). We rescaled the videos frames to small size (64 by 64) before computing these GIST with 4 prefilters, 4 different scales (with 8 orientations at each scale and thus 32 filters in total). In the end each image was represented by a vector of length 512. The features were aggregated every second leading to temporal horizons of time series which are still computationally tractable using Algorithm 7. Using $N = 4$ shows for training, 3 for validation, 3 for test, we report below the test errors for each test show with the loss ℓ_F (smaller is better).

17. Note that the first point of Fig. 4.3 is not for 0 supervision, but for a small ϵ, otherwise we could not have ran the baselines as well as our method.

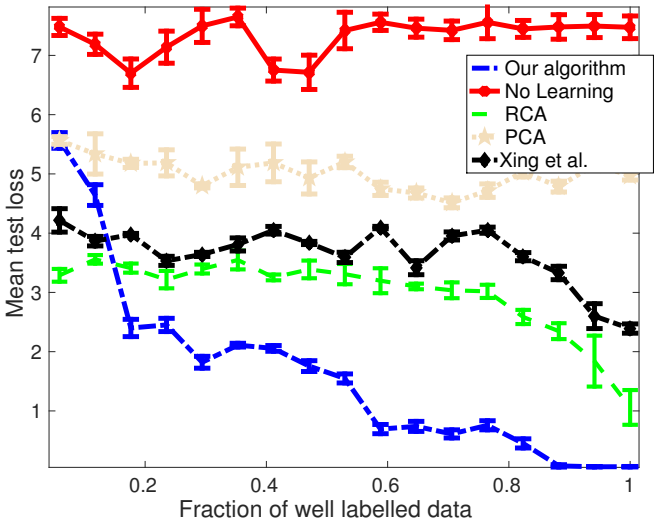


Figure 4.3: Performances on synthetic data vs. the quantity of information available in the time series, measured in terms of the loss ℓ_F defined by Eq. (4.13). Note the small error bars (90% quantiles). We compare ourselves against the Euclidean metric ('No learning'), a metric learned by RCA (with 3 components), PCA and Xing et al. [2002].

Method	Audio			Video			Both		
No similarity measure learning	29	48	33	59	55	47	40	48	36
PCA	23	41	34	40	55	25	29	53	37
Our algorithm	**6.1**	**9.3**	**7**	**10**	**14**	**11**	**8.7**	**9.6**	**7.8**

We consider three different settings: using only the image stream, only the audio stream or both. In these three cases, we consider using the Euclidean metric (no learning), PCA, or our approach. In all settings, similarity measure learning improves performance. Note that the performance is best with the audio stream only; our similarity measure learning, given both streams, manages to do almost as well as with only the audio stream, thus illustrating the robustness of using similarity measure learning in this context where the video stream is not useful.

4.5.5 Bioinformatics application

Detection of change-points in DNA sequences for cancer prognosis provides a natural testbed for the approach to detect changes in distribution of Section 4.4.2. Indeed, researchers from this field face data which are linked to the number of copies of each gene along the DNA (a-CGH data as used by Hocking et al. [2013]). The presence of such changes is generally related to the development of certain types of cancers. On data coming from the Neuroblastoma dataset [Hocking et al., 2013], $N = 30$ caryotypes with changes of distribution were manually annotated. These time series are of temporal horizon T around some hundreds. We consider

the approach of Section 4.4.2 and compare the Euclidean and a learned metric on the five first Hermite moments of the data. For this experiment, we evaluate using the loss suggested by Hocking et al. [2013]. The dataset have been annotated by medical experts. For each time series, these experts have defined pretty large contiguous sets of timestamps in which they are sure that a breakpoint occur, and others in which it does not. The loss proposed by Hocking et al. [2013] counts the number of missed change-points (that is, the number of regions that are annotated in the dataset as having a change-point and for which we did not recover the change-points) as well as the falsely detected ones (change-points located in regions that are not supposed to have one).

Without any similarity measure learning, just by adjusting the AIC criterion by trial and error, we reach a global error rate in change-point identification of 12%. By learning a diagonal metric over Hermite features, we reach a rate of 6.9%, thus improving the performance. Note that, in this application it is *irrelevant* to run standard similarity measure learning like RCA, since they are unable to do model selection and learn the parameter λ of penalization of Eq. (4.5). Note that λ can be seen as the expected level of noise in the series, see Lebarbier [2005].

After having reviewed how our approach has allowed us to tackle the problem of change-point detection, let us now consider what we can do with a similar approach for the case of general partitioning problems.

4.6 Extension: general similarity measure learning for partitioning algorithms based on Euclidean distortions

In this section, we consider the general penalized partitioning problem of Eq. (4.6).

4.6.1 Similarity measure learning algorithm

To keep the presentation clear, and since we have not conducted experiments in the general case, we restrict our presentation to the case where the number K of clusters in the partition is known. Assuming that we are provided N pairs of observation from (X^i, M^i), our approach aims at minimizing the following large-margin objective:

$$\frac{1}{N}\sum_{i=1}^{N}\max_{M\in\mathcal{M}_K}\left\{\mathrm{Tr}\left(M^i - M - 2M^iM\right) + \left\langle W, \phi(X^i, M) - \phi(X^i, M^i)\right\rangle\right\} + \Omega(W). \tag{4.15}$$

As in the previous section, the bottleneck of the method lies in the evaluation of the loss-augmented decoding.

Relaxation of the structured SVM objective for similarity measure learning. Unfortunately, contrary to the case of change-point detection, the inner sum maxi-

mization, in Eq. (4.15), which is an LP over $\mathcal{M}_K$, is NP-hard [Aloise et al., 2009]. One first approach could be to minimize (4.15) using K-means to approximately solve the loss-augmented decoding. This is the approach of Finley and Joachims [2008]. However, this technique does not have any computational guarantee and all the optimization certificates that could be used as a stopping criterion in convex optimization cannot apply with this approach. This is why, we used another one. We have seen in Sec. 4.2.4 that, if we replace $\mathcal{M}_K$ by the set of orthogonal projection matrices of rank K, minimizing an LP over $\mathcal{M}_K^{\text{spec}}$ is affordable. That is why we propose to replace our large-margin objective by another one, that is tractable in polynomial time:

$$\frac{1}{N}\sum_{i=1}^{N} \max_{M\in\mathcal{M}_K^{\text{spec}}} \left\{ \text{Tr}\left[M^i - M - 2M^iM\right] + \langle W, \phi(X^i, M) - \phi(X^i, M^i)\rangle \right\}. \tag{4.16}$$

We just relax the set on which we perform decoding. Evaluation of the loss-augmented decoding indeed reduces to the computation of the K leading eigenvalues as seen in Sec. 4.2.4, which can be efficiently done using for instance the power method [Von Mises and Pollaczek-Geiringer, 1929].

4.6.2 Application to image segmentation

In this section, we present a way to apply our method to the image segmentation problem. We can see foreground/background image segmentation as the task of finding a partition of the pixels in two subsets.

Most of popular image segmentation algorithms are not explicitly based on minimizing Euclidean distortions [18]. Indeed, such methods suffer from the fact that they do not push pixels which are close to belong to the same cluster which is known to be a crucial aspect for accurate image segmentation since works of Gestalt psychologists [Wertheimer, 1923].

On the opposite, approaches based on the minimization of some energy defined on a graph are able to enforce spatial consistency [Rother et al., 2004]. The normalized cuts framework [Shi and Malik, 1997], which is popular for segmenting images, has some relationship with our framework, since it can be cast as

$$\underset{M\in\mathcal{M}_K^{\text{spec}}}{\text{maximize}} \Big[\text{Tr}\big(\Delta M\big) \Big]$$

where Δ is some variant of the normalized Laplacian [Bach and Jordan, 2006, Von Luxburg, 2007] of the graph of pixels.

Inspired by this connection, we propose a simple foreground/background segmentation model which consists in adding a prior term to K-means distortion, as proposed in Eq. (4.6). Namely we cast image segmentation as a decoding of the

18. Even if sometimes the use of simple K-means, for instance over colors features, can lead to very accurate segmentations, see, e.g, Forsyth and Ponce [2002].

Table 4.1: Test performance on the Horses dataset according to our loss. Lower is better.

Loss used	Constrained similarity measure learning	Unconstrained similarity measure learning	Ncuts
ℓ_F	1.47	1.65	1.81
Rand loss	0.35	0.40	0.48

Table 4.2: Test performance on the Flowers dataset according to our loss.

Loss used	Constrained similarity measure learning	Unconstrained similarity measure learning	Ncuts
ℓ_F	1.2	1.4	1.59
Rand loss	0.29	0.35	0.44

type of the one of Eq. (4.10)

$$\underset{M \in \mathcal{M}_{K=2}}{\text{maximize}} \operatorname{Tr}(XBX^\top M) - \operatorname{Tr}(LM)$$

where L is the unnormalized Laplacian of the graph underlying the image. [19]. In our experiments the use of the different versions of the Laplacian did not lead to significantly different performances. This second term permits to give to spatially contiguous clusters the preference over non contiguous ones. Note that in our experiments, we simply consider the graph associated to the 4-connected grid.

4.6.3 Image segmentation experiments

We consider the task of segmenting images of the Weizmann horses dataset [Borenstein and Ullman, 2004], using $N = 60$ training images with colour and dense SIFT features. Results are presented in Table 4.1, where we used both normalized cuts and an our approach for similarity measure learning with no spatial prior [20] as baselines. In Table 4.2, we present analogous results for the Oxford flowers [Nilsback and Zisserman, 2006] dataset, for which the training set size is bigger: 100 images. Note that the parameter λ of the structured SVM is adjusted using a validation set. We report results using the Frobenius loss between partition that is the performance measure that we explicitly optimize. We also gives some qualitative results in Fig. 4.4 and 4.5.

19. We recall that, for a graph with non-negative weights and of adjacency matrix A, the Laplacian is defined by $\operatorname{Diag}(A\mathbf{1}) - A$, the normalized version are respectively $\mathrm{Id} - D^{-1}A$ for the asymmetric version and $\mathrm{Id} - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ for the symmetric one.

20. That corresponds to setting $\mu = 0$ in Eq. (4.6).

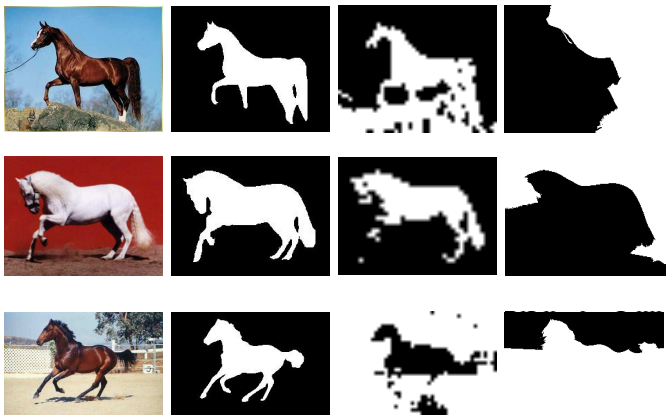


Figure 4.4: From left to right: original image, groundtruth segmentation, image segmented with our learned metric, Ncuts with tuned parameters for colors and position features.

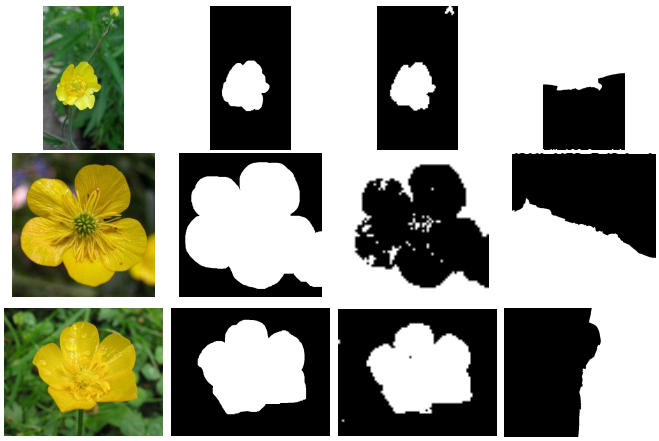


Figure 4.5: The images are in the same order as in Fig.4.4

In these image segmentation experiments, the approach based on unconstrained similarity measure learning leads to inferior performance while our approach based on appropriately constrained similarity measure learning leads to improvements.

4.7 Conclusion

In this chapter we have addressed the problem of learning a metric in a supervised way for improving the performances of unsupervised partitioning algorithms. We have focused on the practically important case in which a prior over the resulting partition is available. More precisely we have demonstrated that, for the temporal sequential prior, such a metric can be learned in an efficient way using a structured SVM. Our formulation allowed us to solve exactly all the classical

bottlenecks of the structured prediction approach. For the general case, we provide an efficient relaxation of the structured SVM objective that leads to a tractable convex optimization learning problem that we solve using off-the-shelf solvers. We have explored several applications, in particular the detection of change-points in video streams or DNA sequences and the problem of image segmentation, with a significant improvement in partitioning performance. Then, driven by the case in which the prior is given through a graph Laplacian, we have proposed a soft model based on Euclidean distortion that we plugged into a structured SVM. We have demonstrated that our approach is well founded with experiments on image segmentation datasets.

For future works, following recent trends in image segmentation (see, e.g., Joulin et al. [2010]), it would be interesting to extend our change-point framework so that it allows unsupervised co-segmentation of several videos: each segment could then be automatically labeled so that segments from different videos but with the same label correspond to the same action. Another extension would be to generalize our algorithm to kernel learning. Indeed, some recent work [Jain et al., 2012] proved links between similarity measure learning and kernel learning, allowing to kernelize any Mahalanobis distance learning problem.

The two major drawbacks of our approach are (i) the fact that we require fully timestamped annotations which are in practice hard to obtain and (ii) the computational slowness due to the non-smoothness of the objective of structured support vector machines. In Sec. 4.4.2, we propose an heuristic to leverage our algorithm and require full supervision. However, this is not fully satisfactory, and we might be able to reuse in our setting some ideas coming from recent works about weak supervision when temporal structure is at stake, like the ones presented on chapter 6.

Chapter 5

Similarity Measure Learning for Warping Temporal Sequences

Abstract

In this chapter, we propose to learn a Mahalanobis quasimetric to perform warping of multivariate time series. The learning examples for this task are time series for which the true warping is known. We cast the warping problem as a structured prediction task. We propose a realistic loss between warpings and a proper convex optimization procedure built on the ideas of large-margin structured prediction [Tsochantaridis et al., 2005]. We provide experiments on real data for the audio-to-audio task, where we show that the learning of a similarity measure leads to improvements in the warping performance. We also propose to use this metric learning framework to perform feature selection and, from basic audio features, build a combination of these that yields to better warping performance.

The material of this chapter is based on our article:
R.Lajugie*, D.Garreau*, S.Arlot, F.Bach, Metric Learning for Temporal Sequences warping, In *Proc. NIPS, 2014*.
*equal contributions

5.1 Introduction

The problem of aligning temporal sequences or more generally to warp them is ubiquitous in applications ranging from bioinformatics [Thompson et al., 1999, Aach and Church, 2001, Cuturi et al., 2007] to audio processing [Dixon and Widmer, 2005, Cont et al., 2007]. Warping is also used in the community of speech recognition since the pioneering work of Sakoe and Chiba [1978]; for instance it can be used for the task of alignment of phonemes to some template. The goal is to warp two similar time series that share the same global structure, but with local temporal differences. Most warping algorithms rely on local similarity measures.

Having a good one is crucial, especially in the high-dimensional setting where some features of the signals can be irrelevant to the warping task. The goal of this chapter is to show how to learn this similarity measure from annotated examples in order to improve the relevance of the warping task on new data.

For example, in the context of music information retrieval, warping is used in two different cases: (1) audio-to-audio warping and (2) audio-to-score alignment. In the first case, the goal is to match two audio interpretations of the same piece that are potentially different in rhythm, whereas audio-to-score alignment focuses on matching an audio signal to a symbolic representation of the score or the partition [Dannenberg, 1984, Vercoe, 1985, Cont, 2010] [1]. In the second case, there has been some attempts to learn from annotated data a similarity measure for performing the alignment. Joder et al. [2013] propose to fit a random field model to perform the task, and Keshet et al. [2007] learn this measure in a discriminative setting.

Similarly to Keshet et al. [2007], we use a discriminative loss to learn the similarity measure, but our work focuses on audio-to-audio warping. In that context, the set of authorized warpings is much larger. Moreover, contrary to Keshet et al. [2007] we explicitly cast the problem as a structured prediction task, that we solve using off-the-shelf stochastic optimization techniques [Lacoste-Julien et al., 2013] but with proper and significant adjustments, in particular in terms of losses.

The need for similarity measure learning goes far beyond warping problems. Weinberger and Saul [2009] proposed a large-margin framework for learning a quasimetric [2] for nearest-neighbour algorithms based on sets of must-link/cannot link constraints. Lajugie et al. [2014a] [3] proposed the same large-margin similarity measure learning as ours to learn a Mahalanobis quasimetric for partitioning problems based the minimization of an Euclidean distortion.

Note also that structured support vector machines (structured SVM) can be applied to tackle various problems, beyond similarity measure learning. Since their proposition by Taskar et al. [2003] and Tsochantaridis et al. [2005], structured SVM have successfully been used to solve many learning problems, for instance to learn weights for graph matching [Caetano et al., 2009] or a metric for ranking tasks [Mcfee and Lanckriet, 2010]. They have also been used to learn graph structures using graph cuts [Szummer and Hoiem, 2008].

In this chapter, we make the following five contributions:

1. We cast the learning of a Mahalanobis quasimetric for warping temporal sequences as a structured prediction problem.
2. We show that, on real musical datasets, this similarity measure improves the performance of warping algorithms that uses high-level handcrafted features.
3. We propose to use the similarity measure learning framework to learn combinations of basic audio features and get good warping performances.

1. For a deeper explanation of the differences between warpings and alignment, please have a look at Sec. 3.3.1 of Chapter 3.

2. We recall that, in this manuscript, we call a quasimetric a function satisfying all axioms of a distance except the separation one.

3. See also Chapter 4.

4. We show experimentally that the standard Hamming loss, although tractable computationally, does not let us learn a relevant similarity measure in some real world settings.
5. We propose a new loss, closer to the evaluation loss used by practitioners to compare warpings, leading to a tractable learning task. We derive an efficient Frank-Wolfe-based algorithm to deal with this new loss. That loss solves some issues that plague the Hamming loss.

5.2 Matrix formulation of warping problems

In this chapter, we consider the problem of warping two multivariate time series sharing the same number of dimensions p, but possibly of different temporal horizons T_A and T_B. Namely $A \in \mathbb{R}^{T_A \times p}$ and $B \in \mathbb{R}^{T_B \times p}$. We refer to the rows of A as $a_1, \ldots, a_{T_A} \in \mathbb{R}^p$ and those of B as $b_1, \ldots, b_{T_B} \in \mathbb{R}^p$. From now on, we denote by X the *pair* of signals (A, B).

5.2.1 Warping using pairwise affinities

Let $C(X) \in \mathbb{R}^{T_A \times T_B}$ be an arbitrary pairwise *affinity matrix* associated to the pair X, that is, $C(X)_{i,j}$ encodes the affinity or similarity between a_i and b_j.[4] The goal of the warping task is to find two non-decreasing sequences of timestamps α and β of common length $u \geq \max(T_A, T_B)$ such that

$$\sum_{i=1}^{u} C(X)_{\alpha(i),\beta(i)} \tag{5.1}$$

is maximal. Moreover the sequence α of timestamps of A and β, the sequence of timestamps of B satisfy:

$$\begin{cases} \alpha(1) = \beta(1) = 1 & \text{(warp beginnings)} \\ \alpha(u) = T_A,\ \beta(u) = T_B & \text{(warp endings)} \\ \forall i,\ \big(\alpha(i+1), \beta(i+1)\big) - \big(\alpha(i), \beta(i)\big) \in \big\{(1,0), (0,1), (1,1)\big\} & \text{(three moves)} \end{cases}$$

Any pair of sequences (α, β) satisfying the three aforementioned properties is a *warping*, or *warp* between time series. The warping task or dynamic time warping problem consists is finding the optimal warping according to the quantity of Eq. (5.1).

Parametrization of warpings. For a given warping (α, β), we can define the binary matrix $Y \in \{0,1\}^{T_A \times T_B}$ such that $Y_{\alpha(i),\beta(i)} = 1$ for every $i \in \{1, \ldots, u\}$ and 0 otherwise. We denote by $\mathcal{Y}(X)$ the set of such matrices, which is uniquely determined by T_A and T_B. An example is given in Fig. 5.1. A vertical move in the Y

4. Note that our framework can be extended to the case where A and B are multivariate signals of different dimensions, as long as $C(X)$ is well-defined.

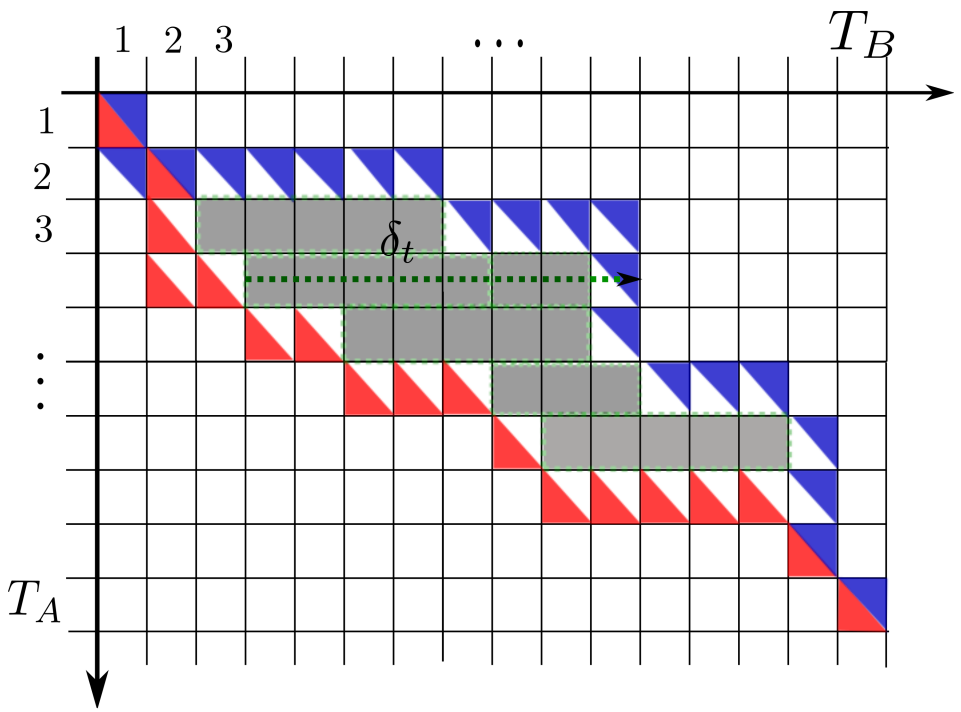


Figure 5.1: Example of two valid warpings encoded by matrices Y^1 and Y^2. Red upper triangles show the (i, j) such that $Y^1_{i,j} = 1$, and the blue lower ones show the (i, j) such that $Y^2_{i,j} = 1$. The grey zone corresponds to the sum of the δ_t, that we have defined as being proportional to $\delta_{\text{abs}}(Y^1, Y^2)$.

matrix means that the signal B is waiting for A, whereas an horizontal one means that A is waiting for B, and a diagonal move means that they move together. In this sense the time reference is "warped".

When $C(X)$ is known, the warping task can be cast as the following linear program (LP) over the set $\mathcal{Y}(X)$:

$$\underset{Y \in \mathcal{Y}(X)}{\text{maximize}} \operatorname{Tr}\big(C(X)^\top Y\big). \tag{5.2}$$

Our goal is to learn a way to build the affinity matrix. Our approach do it directly using data for which the optimal warping is known. Once $C(X)$ is fixed, the warping is obtained from Eq. (5.2). The optimization problem in Eq. (5.2) will be referred to as the *decoding* of our model.

5.2.2 Dynamic time warping

Given the affinity matrix $C(X)$ associated with the pair of signals $X = (A, B)$, finding the warping that solves the LP of Eq. (5.2) can be done efficiently in $O(T_A T_B)$ using a dynamic programming algorithm. It is often referred to as the dynamic time warping problem [Cuturi et al., 2007, Müller, 2007]. This algorithm is de-

scribed in Alg. 8. Various additional constraints may be used in the dynamic time warping algorithm [Müller, 2007]. The most widespread ones are the Sakoe-Chiba band [Sakoe and Chiba, 1978] and the Itakura parallelogram [Itakura, 1975] which both reduce the set of authorized warpings by forbidding some moves. These refinements can be easily added to Alg. 8, but we do not make use of these in practice.

The cardinality of the set $\mathcal{Y}(X)$ is huge and is in fact exponential in the length of time series. It corresponds to the number of paths on a rectangular grid similar to the one of Fig. 5.1 from the northhwest $(1,1)$ to the southeast corner (T_A, T_B) with vertical, horizontal and diagonal moves allowed. This is the definition of the Delannoy numbers [Delannoy, Banderier and Schwer, 2005]. As noted in Torres et al. [2003], when $t = T_A = T_B$ goes to infinity, and one can show that the cardinality of $\mathcal{Y}(X)$ satisfies

$$|\mathcal{Y}(X)| \sim \frac{(3+2\sqrt{2})^t}{\sqrt{\pi t}\sqrt{3\sqrt{2}-4}}.$$

5.2.3 Mahalanobis quasimetric

In many applications (see, e.g., [Dixon and Widmer, 2005]), for a pair $X = (A, B)$, the affinity matrix is defined by

$$C(X)_{i,j} = C(A,B)_{i,j} = -\|a_i - b_j\|_2^2.$$

In this chapter we propose to learn a quasimetric to compare a_i and b_j instead of using the plain Euclidean metric. That is, we consider that $C(X)$ is parametrized by a matrix $W \in \mathcal{W}$, where $\mathcal{W} \subset \mathbb{R}^{p\times p}$ can be, for instance ,the set of semi-definite positive matrices of size $p \times p$, and we use the corresponding Mahalanobis quasimetric to compute the pairwise affinity between a_i and b_j:

$$C(X;W)_{i,j} = -(a_i - b_j)^\top W (a_i - b_j).$$

Note that the decoding of Eq. (5.2) is the maximization of a linear function in the parameter W:

$$\underset{Y\in\mathcal{Y}(X)}{\text{maximize}}\, \text{Tr}\big(C(X;W)^\top Y\big), \tag{5.3}$$

which is equivalent to

$$\underset{Y\in\mathcal{Y}(X)}{\text{maximize}}\, \text{Tr}\big(W^\top \phi(X,Y)\big).$$

If we define the joint feature map ϕ by

$$\phi(X,Y) = -\sum_{i=1}^{T_A}\sum_{j=1}^{T_B} Y_{i,j}(a_i - b_j)(a_i - b_j)^\top \in \mathbb{R}^{p\times p}. \tag{5.4}$$

Algorithm 8 Dynamic programming algorithm for warping temporal sequences. It aims at maximizing $\mathrm{Tr}(CM)$ where C is an arbitrary affinity matrix in $T_B \times T_A$.

Input: Affinity matrix $C \in \mathbb{R}^{T_B \times T_A}$
Computing the cumulative affinity matrix D:
$T_A, T_B \leftarrow \mathrm{size}(C)$
$D \leftarrow \mathrm{zeros}(T_A + 1, T_B + 1)$
Indices of D goes from 0 to T_A and 0 to T_B
for $i = 1$ **to** T_A **do**
  $D(i, 0) \leftarrow -\infty$
end for
for $j = 1$ **to** T_B **do**
  $D(0, j) \leftarrow -\infty$
end for
for $i = 1$ **to** T_A **do**
  for $j = 1$ **to** T_B **do**
    $D(i, j) \leftarrow C(i, j) + \max\{D(i-1, j), D(i, j-1), D(i-1, j-1)\}$
  end for
end for
Backtracking:
$Y \leftarrow \mathrm{zeros}(T_A, T_B)$
$i \leftarrow T_A$
$j \leftarrow T_B$
while $i > 1$ **or** $j > 1$ **do**
  $Y(i, j) \leftarrow 1$
  if $i = 1$ **then**
    $j \leftarrow j - 1$
  else if $j = 1$ **then**
    $i \leftarrow i - 1$
  else
    $m \leftarrow \max\{D(i-1, j), D(i, j-1), D(i-1, j-1)\}$
    if $D(i-1, j) = m$ **then**
      $i \leftarrow i - 1$
    else if $D(i, j-1) = m$ **then**
      $j \leftarrow j - 1$
    else
      $i \leftarrow i - 1$
      $j \leftarrow j - 1$
    end if
  end if
end while
$Y(1, 1) \leftarrow 1$
Output: Y

5.3 Learning the quasimetric

From now on, we assume that we are given N pairs of training instances [5]

$$(X^i, Y^i) = \big((A^i, B^i), Y^i\big) \in \mathbb{R}^{T_A^i \times p} \times \mathbb{R}^{T_B^i \times p} \times \{0,1\}^{T_A^i \times T_B^i},$$

for $i = 1, \ldots, N$. Our goal is to find a matrix W such that the predicted warpings are close to the groundtruth on unseen examples. To that extent, we need to be able to measure how far two warpings are. We first define a *loss* between warpings.

5.3.1 Losses between warpings

In our work, the warpings are encoded by matrices in $\mathcal{Y}(X)$, thus we are interested in functions

$$\ell : \mathcal{Y}(X) \times \mathcal{Y}(X) \to \mathbb{R}_+.$$

Hamming loss. A simple loss between matrices is the squared Frobenius norm of their difference [6]. This turns out to be the unnormalized Hamming loss [Hamming, 1950] for binary-valued matrices [7]. For two matrices $Y_1, Y_2 \in \mathcal{Y}(X)$, it is defined as:

$$\begin{aligned} \ell_H(Y_1, Y_2) &= \|Y_1 - Y_2\|_F^2 = \operatorname{Tr}(Y_1^\top Y_1) + \operatorname{Tr}(Y_2^\top Y_2) - 2\operatorname{Tr}(Y_1^\top Y_2) \\ &= \operatorname{Tr}(Y_1 \mathbf{1}_{T_B} \mathbf{1}_{T_A}^\top) + \operatorname{Tr}(Y_2 \mathbf{1}_{T_B} \mathbf{1}_{T_A}^\top) - 2\operatorname{Tr}(Y_1^\top Y_2), \end{aligned} \tag{5.5}$$

where $\mathbf{1}_T$ is the vector of $\mathbb{R}^T$ with all coordinates equal to 1. The last line of Eq. (5.5) comes from the fact that the Y_i are 0/1-valued; that makes the Hamming loss affine in Y_1 and Y_2. This loss is often used in other structured prediction tasks [Lacoste-Julien et al., 2013]; in the audio-to-score setting, Keshet et al. [2007] uses a modified version of this loss, which is the average number of times the difference between the two warpings is greater than a fixed threshold.

This loss leads to a computationally tractable learning procedure because it is linear in our parametrization of the alignment problem. However, it is not a good loss for audio-to-audio warping. Indeed, a major drawback of the Hamming loss is that, for warpings of fixed length, it depends only on the number of "crossings" between warping paths: one can easily find Y_1, Y_2, Y_{gt} such that

5. We will see that it is necessary to have fully labeled instances, which means that for each pair X^i we need an *exact* warping Y^i between A^i and B^i. Partial warping might be dealt with by alternating between similarity measure learning and constrained warping following the approach of Sec. 4.4.2 of Chapter 4.

6. We recall that the Frobenius norm of a matrix $M \in R^{I \times J}$ is defined by $\|M\|_F^2 = \sum_{i \in I, j \in J} M_{i,j}^2$.

7. The Hamming loss between two binary matrices M and M' is defined as the sum of disagreements, that is $\sum_{i \in I, j \in J} |M_{i,j} - M'_{i,j}|$. Due to the fact that M and M' are binary this quantity is equal to

$$\sum_{i \in I, j \in J} |M_{i,j} - M'_{i,j}|^2,$$

that is exactly the definition of the Frobenius squared distance.

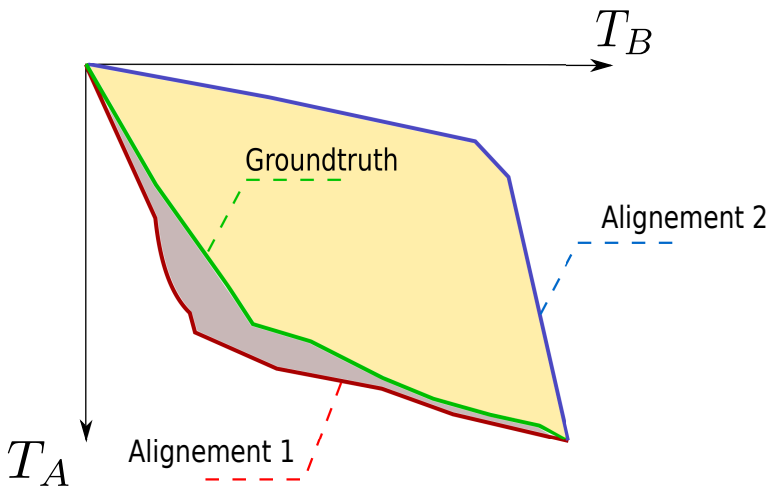


Figure 5.2: This figure depicts three different warpings: the groundtruth Y_{gt} and two warpings Y_1 and Y_2. Y_1 and Y_2 both cross the groundtruth only twice (at the beginning and at the end). Thus we have $\ell(Y_1, Y_{gt}) = \ell(Y_2, Y_{gt})$. However in an intuitive view, Y_1 seems much closer to Y_{gt} than Y_2 because the local deformations of time are more similar.

$\ell_H(Y_{\text{gt}}, Y_2) = \ell_H(Y_2, Y_{\text{gt}})$ but Y_2 seems intuitively much closer to Y_{gt} than Y_2 (see Fig. 5.2). It is important to notice this is often the case when the length of the signals grows.

Area loss. A more natural loss seems to be the "mean" distance between the paths depicted by two matrices $Y^1, Y^2 \in \mathcal{Y}(X)$. This corresponds to the area between the paths of two matrices Y, as represented by the grey zone on Fig. 5.1 or the coloured area in Fig. 5.2.

Formally, as in Fig. 5.1, for each $t \in \{1, \dots, T_B\}$ we set δ_t as

$$|\max\{k, Y^1_{k,t} = 1\} - \max\{k, Y^2_{k,t} = 1\}|.$$

Then the area between the warpings is the sum of the δ_t over $t \in \{1, \dots, T_B\}$. In the audio literature [Kirchhoff and Lerch, 2011], this loss is generally rescaled by dividing by the length of T_B and is called the "mean absolute deviation" loss and is noted $\delta_{\text{abs}}(Y^1, Y^2)$.

Unfortunately, for the general warping problem, δ_{abs} cannot be expressed linearly in the matrices Y. However it is in the case of alignment of a time series on a reference signal or template, that is, when the index sequence α defined in Eq. (5.2.1) is increasing[8]. This situation includes some important practical cases such as the audio-to-partition alignment problem [Joder et al., 2013]. This corresponds to a warping situation in which vertical moves are forbidden. In that case the loss δ_{abs} turns out to be linear in each of its arguments.

8. For a more detailed description of alignments as objects, please have a look at Sec. 3.3.1 of Chapter 3.

More precisely, if we introduce the matrix $L_{T_A} \in \mathbb{R}^{T_A \times T_A}$ which is lower triangular with ones (including on the diagonal), we can define a loss as

$$\begin{aligned} \ell_O(Y^1, Y^2) &= \|L_{T_A}(Y^1 - Y^2)\|_F^2 \\ &= \operatorname{Tr}(L_{T_A}^\top L_{T_A} Y_1 \mathbf{1}_{T_B} \mathbf{1}_{T_A}^\top) + \operatorname{Tr}(L_{T_A}^\top L_{T_A} Y^2 \mathbf{1}_{T_B} \mathbf{1}_{T_A}^\top) - 2\operatorname{Tr}(L_{T_A} Y^1 Y^{2\top} L_{T_A}^\top). \end{aligned} \tag{5.6}$$

We now prove that this loss corresponds exactly to the δ_{abs} up to a multiplicative factor in this special case.

Proof. Let Y be an alignment, then it is easy to see that

$$(L_{T_A} Y)_{i,j} = \sum_k (L_{T_A})_{i,k} Y_{k,j} = \sum_{k=1}^{i} Y_{k,j}.$$

Since Y does not have vertical moves, i.e., for each j there is an unique k_j such that $Y_{k_j,j} = 1$, we have that $(L_{T_A} Y)_{i,j} = 1$ if $i \geq k_j$ and 0 otherwise. Thus since L_{T_A} and L_{T_B} are binary matrices,

$$\|L_{T_A}(Y_1 - Y_2)\|_F^2 = \sum_{i,j} |(L_{T_A} Y)_{i,j} - (L_{T_B} Y)_{i,j}|.$$

Thus, $\|L_{T_A}(Y_1 - Y_2)\|_F^2$ is the sum of the δ_l on Fig. 5.3, this turns out to be exactly the sum of δ_t which is equal to $T_B \delta_{\text{abs}}(Y^1, Y^2)$. □

In all our experiments, we use δ_{abs} for evaluation but not for training.

Approximation of the area loss: the symmetric area loss. In many real world applications [Kirchhoff and Lerch, 2011], a meaningful loss to assess the quality of a warping is the area loss. As shown by our experiments, if the Hamming loss is sufficient in some simple situations and allows to learn a Mahalanobis quasimetric that leads to good warping performance in terms of area loss, on more challenging datasets it does not work at all (see Sec. 5.5). This is due to the fact that two warpings that are very close in terms of the loss δ_{abs} (or equivalently in terms of area) can suffer a big Hamming loss (cf. Fig. 5.2). However, directly using δ_{abs} for learning would lead to an intractable problem. However, this problem can be make tractable if we use the loss ℓ_O of Eq. (5.6). In the previous paragraph, we have seen that this loss was exactly proportional to δ_{abs} for alignments. Thus, we propose to use formulation of Eq. (5.6) to matrices in $\mathcal{Y}(X)$ as an approximation for δ_{abs} in the general case. In the general case, the formulation of Eq. (5.6) penalizes vertical moves too much. To limit this effect, we first make the formulation of Eq. (5.6) symmetric so that vertical and horizontal moves are surpenalized in the same way.

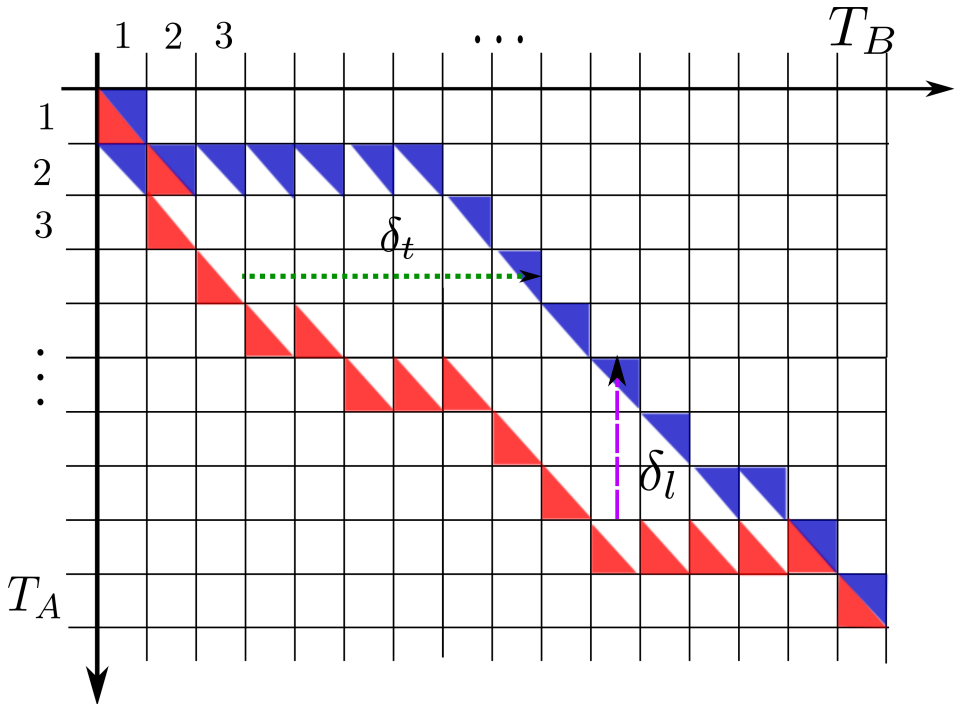


Figure 5.3: Illustration of the proof of the equivalence between the δ_{abs} and our ℓ_O. In the case of alignments, ℓ_O sums the quantities δ_l whereas δ_{abs} is proportional to the sum of δ_t.

We define, for any couple of binary matrices (Y_1, Y_2),

$$\begin{aligned}\ell_S(Y_1, Y_2) &= \frac{1}{2}\left(\|L_{T_A}(Y_1 - Y_2)\|_F^2 + \|(Y_1 - Y_2)L_{T_B})\|_F^2\right) \\ &= \frac{1}{2}\Big[\operatorname{Tr}(Y_1^\top L_{T_A}^\top L_{T_A} Y_1) + \operatorname{Tr}(Y_2^\top L_{T_A}^\top L_{T_A} Y_2) - 2\operatorname{Tr}(Y_2^\top L_{T_A}^\top L_{T_A} Y_1) \\ &+ \operatorname{Tr}(Y_1 L_{T_B} L_{T_B}^\top Y_1^\top) + \operatorname{Tr}(Y_2 L_{T_B} L_{T_B}^\top Y_2^\top) - 2\operatorname{Tr}(Y_2 L_{T_B} L_{T_B}^\top Y_1^\top)\Big].\end{aligned} \tag{5.7}$$

We call this loss the symmetric area loss (SAL).

Other expressions for the symmetric area loss. For now, the symmetric area loss is only defined on the binary matrices of the set $\mathcal{Y}(X)$. However, this set is parametrized through elements of a matrix space. It is natural to wonder how to extend this loss between points of the discrete set $\mathcal{Y}(X)$ to the whole matrix space. The expression of Eq. (5.7) is well defined over the whole space.

Since matrices $Y \in \mathcal{Y}(X) \in \{0,1\}^{T_A \times T_B}$ are binary, for any diagonal matrix $D_{T_A} \in \mathbb{R}^{T_A}$, we have that

$$\operatorname{Tr}(Y^\top D_{T_A} Y) = \operatorname{Tr}(Y^\top D_{T_A} \mathbf{1}_{T_A}^\top \mathbf{1}_{T_A}).$$

Let D_{T_A} be a diagonal matrix of size $T_A \times T_A$ and D_{T_B} be a diagonal matrix of size $T_B \times T_B$, then for any matrix Y of the set $\mathcal{Y}(X)$, we have the following formulation

for the loss ℓ_S :

$$\begin{aligned}\ell_S(Y_1, Y_2) = \frac{1}{2}\Big[&\operatorname{Tr}(Y_1^\top (L_{T_A}^\top L_{T_A} - D_{T_A}) Y_1) + \operatorname{Tr}(D_{T_A} Y_1 \mathbf{1}_{T_B} \mathbf{1}_{T_A}^\top) \\ &+ \operatorname{Tr}(Y_2^\top (L_{T_A}^\top L_{T_A} - D_{T_A}) Y_2) + \operatorname{Tr}(D_{T_A} Y_2 \mathbf{1}_{T_B} \mathbf{1}_{T_A}^\top) - 2\operatorname{Tr}(Y_2^\top (L_{T_A}^\top L_{T_A} - D_{T_A}) Y_1) \\ &+ \operatorname{Tr}(Y_1 (L_{T_B} L_{T_B}^\top - D_{T_B}) Y_1^\top) + \operatorname{Tr}(Y_1 D_{T_B} \mathbf{1}_{T_B} \mathbf{1}_{T_A}^\top) + \operatorname{Tr}(Y_2 (L_{T_B} L_{T_B}^\top - D_{T_B}) Y_2^\top) \\ &+ \operatorname{Tr}(Y_2 D_{T_B} \mathbf{1}_{T_B} \mathbf{1}_{T_A}^\top) - 2\operatorname{Tr}(Y_2 L_{T_B} L_{T_B}^\top Y_1^\top)\Big].\end{aligned} \tag{5.8}$$

Due to this fact, it is possible to extend the loss ℓ_S in infinitely many ways. Let us choose the more convenient for us: for computational tractability, we need indeed this loss to be concave. This is the crucial step of our approach.

A concave extension of the symmetric area loss. Equation (5.8) is the sum of a linear part in the first argument Y_1 and of a quadratic part of Y_1. For computational tractability, we need to have a formulation which is concave in Y_1. By a fine selection of the matrices D_{T_A} and D_{T_B}, the expression (5.8) allows us to make ℓ_S *concave* over the convex hull of $\mathcal{Y}(X)$ that we denote from now on $\overline{\mathcal{Y}(X)}$ For $T = T_A$ and $T = T_B$, a simple choice for the diagonal matrices

$$D_T = \lambda_{\max}(L_T^\top L_T)\mathrm{Id}_T$$

with $\lambda_{\max}(U)$ the largest eigenvalue of U. This is the matrix we used the most in our experiments. Note that any diagonal matrix $D_T \succeq L_T^\top L_T$ would have been a suitable choice. Thus, for completeness, in our experiments, we also try to set the matrices D_T to the minimal trace matrix that dominates $L_T^\top L_T$ in the SDP sense by solving a semidefinite program (SDP). We report the associated result in Fig 5.7. Note also that other matrices could have been chosen. In particular, since our matrices L_T are pointwise positive, the matrix

$$\operatorname{Diag}(L_T^\top L_T) - L_T^\top L_T$$

makes ℓ_S concave. With these choices for D_{T_A} and D_{T_B}, we can extend ℓ_S to a concave function over the whole space and thus in the convex hull $\overline{\mathcal{Y}}(X)$.

5.3.2 Empirical loss minimization

Recall that we are given N examples $(X^i, Y^i)_{1 \le i \le N}$ of one pair of time series and the corresponding warping. For the loss ℓ_S, we consider learning the Mahalanobis quasimetric by solving the following minimization problem in W:

$$\underset{W \in \mathcal{W}}{\text{minimize}} \Big\{ \frac{1}{N} \sum_{i=1}^{N} \ell_S\big(Y^i, \underset{Y \in \mathcal{Y}(X^i)}{\operatorname{argmax}} \operatorname{Tr}(C(X^i; W)^\top Y)\big) + \lambda \Omega(W) \Big\}, \tag{5.9}$$

where $\Omega = \frac{\lambda}{2}\|W\|_F^2$ is a convex regularizer preventing from overfitting, with $\lambda \geq 0$.[9]

5.4 Large margin approach

In this section we describe a large margin approach to deal with the problem of Eq. (5.9), which is intractable. As shown in Eq. (5.3), the decoding task is the maximization of a linear function in the parameter W and aims at predicting an output over a large and discrete space (the space of potential warpings with respect to the constraints in Eq. (5.2.1)). Learning W thus falls into the structured prediction framework [Tsochantaridis et al., 2005, Taskar et al., 2003]. Thus, we can replace the loss minimization problem by a convex version of it by introducing a proper surrogate to the function

$$W \to \ell_S\Big(Y^i, \operatorname*{argmax}_{Y \in \mathcal{Y}(X)} \operatorname{Tr}\big(C(X^i;W)^\top Y\big)\Big).$$

5.4.1 Convex surrogate to the empirical loss

We define the hinge loss, a convex surrogate to the empirical loss function, as

$$L(X,Y;W) = \max_{Y' \in \mathcal{Y}(X)} \left\{ \ell_S(Y,Y') - \operatorname{Tr}\Big(W^\top\big[\phi(X,Y) - \phi(X,Y')\big]\Big)\right\}. \tag{5.10}$$

The evaluation of L is usually referred to as *loss-augmented decoding*, see [Tsochantaridis et al., 2005]. We now aim at solving the following problem, sometimes called the *margin-rescaled problem*:

$$\operatorname*{minimize}_{W \in \mathcal{W}} \frac{\lambda}{2}\|W\|_F^2 + \frac{1}{N}\sum_{i=1}^{N} \max_{Y' \in \mathcal{Y}(X)} \left\{ \ell_S(Y',Y^i) - \operatorname{Tr}\Big(W^\top\big[\phi(X^i,Y^i) - \phi(X^i,Y')\big]\Big)\right\}. \tag{5.11}$$

In order to use the large-margin structured prediction framework, we need to be able to solve the associated loss-augmented decoding. We address this issue in the following subsections, for the two losses we have introduced so far.

5.4.2 Hamming loss case

Among the aforementioned losses, the Hamming loss ℓ_H is the only one leading directly to a tractable loss-augmented decoding problem and thus that falls directly into the structured prediction framework. Indeed, plugging Eq. (5.5) into (5.10) leads to a loss-augmented decoding that is an LP over the set $\mathcal{Y}(X)$. If we define $\widehat{Y}^i$ as the argmax in Eq. (5.10) when $(X,Y) = (X^i,Y^i)$, then elementary computations

9. Note that others regularizers could be used to enforce some sparsity patterns.

show that

$$\widehat{Y}^i = \operatorname*{argmax}_{Y \in \mathcal{Y}(X)} \operatorname{Tr}\Big(\big(-U^\top + 2Y^{i\top} + C(X^i; W)^\top\big) Y \Big),$$

where $U = \mathbf{1}_{T_A} \mathbf{1}_{T_B}^\top \in \mathbb{R}^{T_A \times T_B}$. Thus, the loss-augmented decoding is the maximization of a linear function over the spaces $\mathcal{Y}(X)$ that we can solve efficiently using the dynamic programming algorithm 8.

That way, plugging the Hamming loss (Eq. (5.5)) in the objective function of Eq. (5.11) leads to a structured prediction problem for which we know how to solve the loss-augmented decoding. This problem can be solved using standard techniques such as cutting plane methods [Joachims et al., 2009], stochastic gradient descent [Shalev-Shwartz et al., 2011], or block-coordinate Frank-Wolfe in the dual [Lacoste-Julien et al., 2013]. [10]

5.4.3 Symmetric area loss case

For the symmetric area loss, we do not know how to perform the loss-augmented decoding efficiently. Thus, the standard off-the-shelf solvers for structured prediction cannot be used directly. In this paragraph, we show, how, by relaxing the set of matrices $\mathcal{Y}(X)$ into its convex hull $\overline{\mathcal{Y}}(X)$, we manage to get a tractable optimization problem.
Instead of the problem of Eq. (5.11), we consider the following relaxed one:

$$\operatorname*{minimize}_{W \in \mathcal{W}} \frac{\lambda}{2}\|W\|_F^2 + \frac{1}{N}\sum_{i=1}^{N} \max_{Y \in \overline{\mathcal{Y}}(X)} \left\{ \ell_S(Y, Y^i) - \operatorname{Tr}\Big(W^\top \big[\phi(X^i, Y^i) - \phi(X^i, Y)\big] \Big) \right\}. \tag{5.12}$$

We just have relaxed the optimization sets, and are thus performing the loss-augmented decoding on a much bigger set (which is not even finite anymore). Since, by Eq. (5.8) we have extended the symmetric area loss in a concave function in its first argument over $\overline{\mathcal{Y}}(X)$, the problem of Eq. (5.12) is in min/max form. We can derive a dual using the same argument as in Sec. 1.4.4 of Chapter 1.

Let us briefly recall the major steps. For simplicity we consider the case where $N = 1$, that is we have a single training pair (X^1, Y^1). Starting from the problem of Eq. (5.12), we are in a situation in which we can apply a saddle-point there like the one of Prop. 5.5.7 of [Bertsekas, 2015]. Thus we can switch the min and the max to get a dual problem that is

$$\operatorname*{maximize}_{Y \in \mathcal{Y}(X)} \left\{ \min_{W \in \mathcal{W}} \frac{\lambda}{2}\|W\|_F^2 + \left\{ \ell_S(Y, Y^1) - \operatorname{Tr}\Big(W^\top \big[\phi(X^1, Y^1) - \phi(X^1, Y)\big] \Big) \right\} \right\}.$$

By setting the matrix gradient with respect to W to 0, we get a relation between Y

10. Please note that these techniques are designed for learning parameters W that lie in the whole space $\mathbb{R}^{p \times p}$. Thus, we have adapted the standard unconstrained optimization methods to our setting, where $W \succeq 0$.

and W:

$$W = \frac{1}{\lambda}\phi(X^1, Y^1) - \phi(X^i, Y).$$

Plugging back this expression and using the expression of Eq. (5.4) yields to the following dual program

$$\underset{Z \in \overline{\mathcal{Y}(X)}}{\text{minimize}} \frac{1}{2\lambda} \left\| \sum_{j,k} (Y^1 - Z)_{j,k} (a_j - b_k)(a_j - b_k)^\top \right\|_F^2 - \ell_S(Y^1, Z),$$

More generally, when considering N training instances (X^i, Y^i), if we call $\overline{\mathcal{Y}}$ the Cartesian product of the $\overline{Y}(X^i)$, the dual has the following form:

$$\underset{(Z^1,\dots,Z^N) \in \overline{\mathcal{Y}}}{\text{minimize}} \Bigg\{ \frac{1}{2\lambda N^2} \quad \left\| \sum_{i=1}^n \sum_{j,k} (Y_i - Z^i)_{j,k} (a_j - b_k)(a_j - b_k)^\top \right\|_F^2 \tag{5.13}$$

$$- \frac{1}{N} \sum_{i=1}^n \ell_S(Z, Z^i) \Bigg\}, \tag{5.14}$$

where we denote by $\overline{\mathcal{Y}}(X^i)$ the convex hull of the sets $\mathcal{Y}(X^i)$, and by $\overline{\mathcal{Y}}$ the cartesian product over all the training examples i of such sets. Note that we recover a similar result as [Lacoste-Julien et al., 2013]. Since the SAL loss is concave, the aforementioned problem is convex. Note also that, for any N-uple of dual variables $(Z^1, \dots, Z^N)$, we can get the corresponding primal variable by the relation

$$W = \frac{1}{\lambda N} \sum_{i=1}^N \Big(\phi(X^i, Y^i) - \phi(X^i, Y) \Big).$$

Optimization procedure. The problem of Eq. (5.13) is a quadratic program over the compact set $\overline{\mathcal{Y}}$. However, this convex set is hard to describe in terms of inequalities and even if it is possible, the number of these would prohibit the use of standard quadratic programming solvers. However, on the set $\mathcal{Y}$, we are able to solve any linear program thanks to the dynamic programming algorithm 8. Since $\overline{\mathcal{Y}}$ is the convex hull of $\mathcal{Y}$, being able to solve an LP over $\mathcal{Y}$ yields us to a solution of the LP over $\overline{\mathcal{Y}}$ [11]. Thus we can use a Frank-Wolfe [Frank and Wolfe, 1956] algorithm to solve problem of Eq. (5.13). This algorithm minimizes a convex program by iteratively

1. minimizing the linear approximation of the objective function given by its gradient at the current point,
2. solving an LP over the set, yielding to a new point,
3. making a convex combination between the current point and the new point [12].

11. See [Bertsekas, 1999].

12. Note that in the case of quadratic functions, the combination parameter that makes the objective decrease the most can be computed in closed form. For general objective functions, the universal stepsize $\frac{2}{k+2}$, where k is the iteration counter, guarantees the convergence of the procedure [Jaggi, 2013].

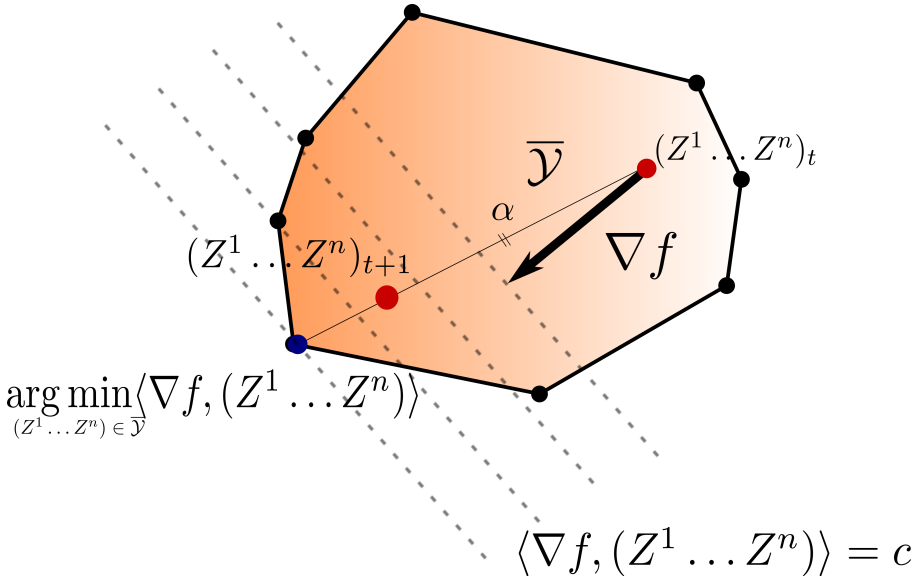


Figure 5.4: We depict the typical behaviour of a Frank-Wolfe algorithm on the polytope $\overline{\mathcal{Y}}$. At the current point, we compute a gradient, we minimize the linear form associated to the linearization of objective function, this yields to an extremal point of the polytope, we do a convex combination between the current point and this extremal point.

This algorithm is described more precisely in Sec. 1.6 of Chapter 1. An illustration is provided in Fig. 5.4.

5.5 Experiments

First, we conduct an experiment on a synthetic example to see if the quasimetric learning is able to recover relevant dimensions among noisy or irrelevant ones.

5.5.1 Synthetic data

In order to demonstrate the validity of our approach, we first consider $N = 100$ synthetic examples $(A^i, B^i, Y^i)_{1 \leq i \leq N}$. The A^i, B^i are time series in dimension $p = 11$ and of respective lengths $T_A = 500$ and $T_B = 600$. For each dimension the two time series are piecewise affine functions whose slope values are 1, 2, and 3. The location of the changes of slopes vary across the dimensions. Moreover, we add some Gaussian noise of variance $\sigma^2 = 0.01$ at each timestamp and independently on each coordinate. The warping matrix Y^i is obtained by matching A^i and B^i according to the quasimetric induced by the matrix depicted on the left of Fig. 5.5 [13].

We use these examples as training instances to learn a quasimetric W using the optimization problem (5.13). For this experiment, we only looked at the resulting

13. The affinity matrix $C(A^i, B^i)$ from which we compute the warping using Alg. 8 is thus computed as $C(A^i, B^i)_{t_1,t_2} = a^i_{t_2 \top} W a^i_{t_1}$, where W is the aforementioned matrix.

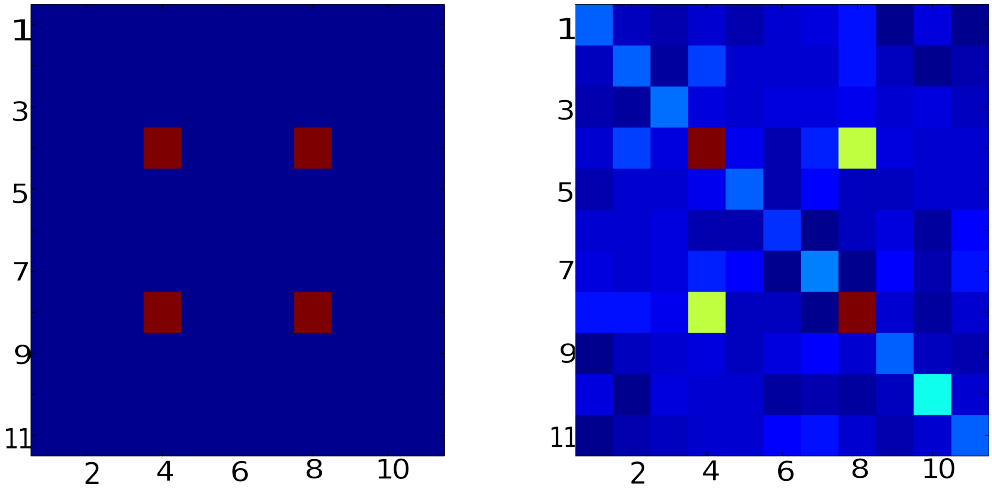


Figure 5.5: W matrix after optimization (Right) compared to the matrix associated to the quasimetric that have generated the groundtruth warping (Left). Best seen in colour.

matrix associated to the Mahalanobis similarity measure learned. On the left of Fig. 5.5, the groundtruth matrix, the one that have been used for building the cost matrix for warping the temporal series. On the right, we present the matrix that we recover in the end of our optimization procedure. It is interesting to notes that only the relevant dimensions are selected by our method. This assess the validity of our approach.

5.5.2 Dataset of Kirchhoff and Lerch [2011]

We applied our method to the task of learning a good similarity measure for aligning audio signals. In this field researchers have spent a lot of efforts in designing well-suited and meaningful features [Joder et al., 2013, Cont et al., 2007]. But the problem of combining these features for aligning temporal sequences is still challenging. For simplicity, we use a diagonal variant of our learning algorithm. We restrict the optimization set $\mathcal{W}$ in Eq. (5.11) to the set of diagonal matrices for our experiments. This reduces to learning a vector rather than a full matrix.

Dataset description. First, we applied our method on the dataset of Kirchhoff and Lerch [2011]. In this dataset, pairs of aligned examples (A^i, B^i) are artificially created by stretching an original audio signal. That way, the groundtruth warping Y^i is known and thus the data falls into our setting. A more precise description of the dataset can be found in [Kirchhoff and Lerch, 2011].

The $N = 60$ pairs are stretched along two different tempo curves. Each signal is made of 30s of music divided in half-overlapping frames of 46ms, thus with a hop of 23ms between frames, thus leading to a typical length of the signals of $T \approx 1300$ in our setting. We keep $p = 11$ features that are simple to implement and known to perform well for warping tasks [Kirchhoff and Lerch, 2011]. Those

were: five mel frequency cepstral coefficients (MFCC) [Gold et al., 2011] (labeled $M1, \ldots, M5$ in Fig. 5.6), the spectral flatness (SF), the spectral centroid (SC), the spectral spread (SS), the maximum of the envelope (Max), and the power level of each frame (Pow), see [Kirchhoff and Lerch, 2011] for more details about the computation of the features. We normalize each feature by subtracting the median value and dividing by the standard deviation to the median. Note that we use the median since audio data are subject to outliers.

Experiments. We have conducted the following experiment: for each individual feature, we perform warping using dynamic time warping algorithm and evaluate the performance of this single feature in terms of the loss δ_{abs} that is typically used to assess performance in this setting [Kirchhoff and Lerch, 2011]. In Fig. 5.6, we report the results of these experiments.

Then, we apply our approach on these data, using the Hamming loss to learn a linear non-negative diagonal combination of these features. The result is reported in Fig. 5.6. Thus, combining these features on this dataset yields to better performances than only considering a single feature.

For completeness, we also conducted the experiments using the standard 13 first MFCCs coefficients and their first and second order derivatives as features. These results competed with the best learned combination of the handcrafted features. Namely, in terms of the δ_{abs} loss, they perform at 0.046 seconds. Note that these results are slightly worse than the best single handcrafted feature, but better than the best single MFCC coefficient used as a feature.

As a baseline, we also compared ourselves against the uniform combination of handcrafted features (the metric being the identity matrix). The results are off the charts on Fig. 5.6 with δ_{abs} at 4.1 seconds (individual values ranging from 1.4 seconds to 7.4 seconds).

5.5.3 Chorales dataset

Dataset description. The Bach 10 dataset [14] consists in ten J. S. Bach's Chorales [15]. For each monophonic voice, we are provided a MIDI reference file corresponding to the "score", or a representation of the partition. We have converted these MIDI files into audio following a standard musical approach (see e.g, [Hu et al., 2003]) [16]. That way, we fall into the audio-to-audio framework in which we can apply our technique. Each piece of music is approximately 25s long, leading to time series of temporal horizon $T \approx 1300$.

Experiments. We use the same features as in Sec. 5.5.2. As depicted in Fig. 5.7, the optimization with Hamming loss performs poorly on this dataset. In fact, the best individual feature performance is far better than the performance of the

14. `http://music.cs.northwestern.edu/data/Bach10.html`.

15. These are small quadriphonic pieces. They are the harmonic superposition of four monophonic voices.

16. Thus we know exactly the groundtruth warpings.

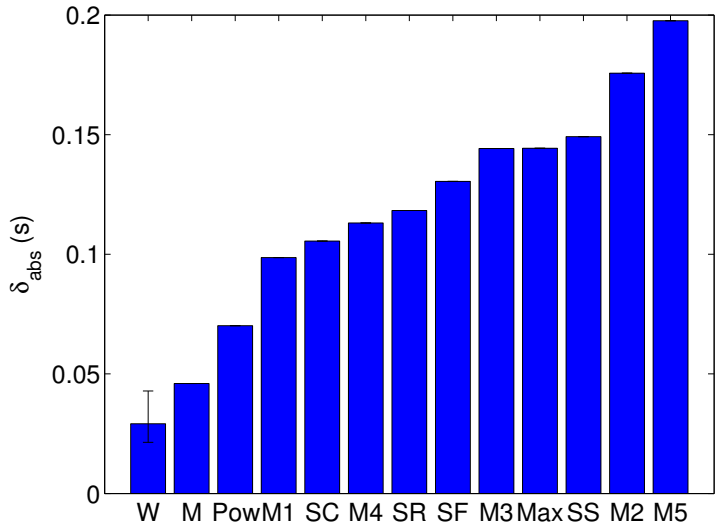


Figure 5.6: Comparison of performance between individual features and the learned quasimetric. The error bar for the performance of the learned quasimetric are determined by the best and the worst performance on 5 different experiments. W denotes the learned combination using our method, and M the best MFCC combination.

learned W. Thus similarity measure learning with the computationally practical Hamming loss performs much worse than the best single feature.

Then, we conduct the same learning experiment with the symmetric area loss ℓ_S. The resulting learned parameter is far better than the one learned using the Hamming loss. We get a performance that is similar to the one of the best feature. Note that these features were handcrafted and reaching their performance on this hard task with only a few training instances is already challenging.

5.5.4 Feature building by combination of low-level features

Last, we conduct feature selection experiments over the same dataset. Starting from low level features, namely the 13 leading MFCCs coefficients and their first two derivatives, we learn a linear combination of these using our approach. Results corresponds to (3), (4), and (5) on Fig. 5.7. This combination achieves good warping performance in terms of the area loss. Note that very little musical prior knowledge is put into these features. In the case of the Chorales dataset, we do not improve on the best handcrafted feature but our learned combination of MFCC performs similarly. [17].

17. However, on the dataset of [Kirchhoff and Lerch, 2011] our combination of low level MFCCs outperforms the best single handcrafted feature.

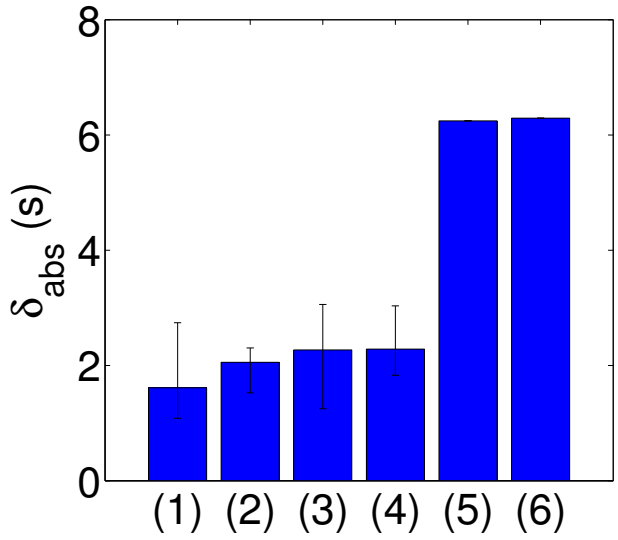


Figure 5.7: Performance of our algorithms on the Chorales dataset. From left to right: (1) Best single feature, (2) Best learned combination of features using the symmetric area loss ℓ_S, (3) Best combination of MFCC using SAL and D_T obtained via SDP (see footnote in section 5.3) (4) Best combination of MFCC and derivatives learned with ℓ_S, (5) Best combination of MFCCs and derivatives learned with Hamming loss, (6) Best combination of features of [Kirchhoff and Lerch, 2011] using Hamming loss.

5.6 Conclusion

In this chapter, we have presented a structured prediction framework for learning the similarity measure for temporal warping problems. We are able to combine hand-crafted features, as well as building automatically new state-of-the-art features from basic low-level information with little expert knowledge. Technically, this is made possible by considering a loss beyond the Hamming loss which is typically used because it is "practical" within a structured prediction framework (linear in the output representation).

Our technical approach can be extended to many structured prediction problems. As soon as elements of a structured set $\mathcal{Y}$ can be represented using some extremal points of the d-dimensional hypercube and if the loss can naturally be expressed as function twice differentiable whose Hessian has bounded eigenvalues over the whole d-dimensional space, then it is possible to make it concave and thus to derive the same optimization problem than the one we used, and solve it efficiently using a Frank-Wolfe algorithm.

This work may be extended in several ways, the main one being to consider cases where only partial information about the warpings is available. This is often the case in music [Cont et al., 2007] or bioinformatics applications. Note that, similarly to Lajugie et al. [2014a] and to Sec. 4.4.2 of Chapter 4 a simple alternating optimization between similarity measure learning and constrained warping provides a simple first solution, which could probably be improved upon. This is the goal of the next chapter.

Chapter 6

A Weakly-Supervised Framework for Structured Prediction With Sequential Structure

Abstract

In this chapter, we present a framework to deal with weakly-supervised sequential data for the specific task of classifying each timestamp of a time series into classes. We consider in particular the problem of audio-to-score that is to classify each timestamp of an audio recording in some vocabulary of events $\mathcal{D}$ that may be notes or combination of these. For this task, getting fully timestamped audio recordings, or equivalently a large collection of individual samples to train classifiers is time-consuming. On the opposite, it is easy to get audio recordings with the corresponding score that has been played. However, the score is not a timestamped annotation and only gives the order of notes that are played and an idea about their relative duration. Due to the freedom let to the interpret, many local or global distortions can occur. However the temporal order of the audio recording and of the score are the same. We propose an abstract model to deal with such data, that are pairs of one temporal signal X and of a list of events that we call a template. In that case we show that the classification task can be seen as an *alignment* problem. We cast the learning of classifiers as a structured prediction task using a cost function based on the Hamming loss. As suggested in Chapter 1, we use the order of events as an uncertainty about actual groundtruth timestamps labels. We also show how it is possible to include further information given by the score in the case of audio-to-score. We consider the score as giving an expected shape about the alignment and cast this information as penalties that we include directly in our cost function. We eventually demonstrate our approach on a comprehensive study with data coming from the Finnish folk song dataset.

This work is a joint work with Piotr Bojanowski, Philippe Cuvilier, Sylvain Arlot and Francis Bach. It is under submission to a peer-reviewed conference. It is also linked to the publications:

P. Bojanowski, R. Lajugie, F. Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, Josef Sivic, Weakly Supervised Action Labeling in Videos Under Temporal Ordering Constraints, In *Proc. ECCV*, 2014.

P. Bojanowski, R. Lajugie, E. Grave, F. Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, Weakly-Supervised Alignment of Video With Text, *arXiv Preprint*, 2015.

The results of these papers will be extensively detailed in the PhD dissertation of P. Bojanowski.

6.1 Introduction

General motivation. As we have seen in chapters 1, 4 and 5, data are often scarce. In the case of sequential structure we have considered so far, this is due to the difficulty to have fully timestamped annotated data. For instance, in Chapter 5, we were provided with pairs of time series that have been manually warped by some human expert. In Chapter 4, we made use of exactly located change-points and delimitation of segments which are in practice hard or unrealistic to get. For example, in bioinformatics, "aCGH" arrays [Hocking et al., 2013] often come with an approximate segmentation. This takes the form of zones that have been manually selected as containing a change-point. In image segmentation also, gathering pixelwise information is generally hard and time-consuming; thus most of state-of-the-art techniques for image segmentation have to make use of very few information or very coarse annotations that may take the form of bounding boxes [Kuettel et al., 2012].

A first way to deal with this kind of partial annotations without drastically changing the approaches proposed in Chapters 4 and 5 is to use the non-convex extension of the fully supervised setting we have proposed in Sec. 4.4.2 of Chapter 4. This is an alternating optimization scheme that iterates between predicting labels and learning a Mahalanobis metric. More precisely, starting from an initial similarity measure (for instance the one induced by the identity matrix), we complete the partial annotation so that we have fully timestamped annotated data. Then we iterate between learning a similarity measure using these data and completing the partial annotations using this measure.

In this chapter, we propose to overcome the issue of partial supervision in the case of sequential structure in an other way and on an alignment task. Using the idea of weak supervision that we have quickly presented in Sec. 6.2 of Chapter 1, we propose to use partial annotations as a weak information directly into the learning procedure.

Classification of a time series into events. Time series are often recordings of natural or human phenomena. Such signals have generally some high-level in-

terpretation, e.g., a video can show someone opening a door, an audio recording can be the succession of notes EEFGGFEDC in a piano recording. They are the succession of meaningful *events* that belongs to a finite *dictionary* $\mathcal{D}$. Facing such a recording of temporal horizon T, it is natural to look for classifying each timestamp to an element of the dictionary $\mathcal{D}$. This is a multiclass classification task and can be cast as the search for an *assignment mapping*:

$$n : \{1, \dots, T\} \to \{1, \dots, |\mathcal{D}|\}.$$

Alignment tasks. When sequential structure is at stake, we generally have some high-level knowledge about the shape of the assignment mapping n. A signal X of T timestamps often comes with a *template* of E events. The template is a symbolic and ordered representation of the signal. Formally it is a list of E elements of $\mathcal{D}$ ordered by time, that is, event 1 occurs before event 2, and so on. A template can be the score [1] of a music piece, the script for a video, the text of a speech recording and so on. With this additional information, we have some knowledge about the mapping n: we know that it will respect the order of events of the template. But we do not know necessarily about the temporal extent of each of these events [2]. In this chapter, we call an *alignment task* the task of classifying a time series when we are provided a template. The goal of such a task is to provide an *alignment mapping* m from X to its template. As seen in Chapter 3, this is a non-decreasing mapping m from $\{1, \dots, T\}$ to $\{1, \dots E\}$ such that:

1. $m(1) = 1$,
2. $m(T) = E$,
3. $m(t+1) = m(t) + 1$ or $m(t+1) = m(t)$.

Note that, from the alignment mapping m, it is straightforward to get the assignment n since the template gives the information necessary: a template can be seen as a mapping $\tau : \{1, \dots, E\} \to \mathcal{D}$, so that $n = \tau \circ m$. To make the connection with Chapter 3, let us introduce a matrix C of size $T \times E$. The alignment tasks we perform there will look for an alignment mapping m such that the cost:

$$\sum_{t=1}^{T} C_{t,m(t)}$$

is maximal. In our case, $C_{t,e}$ should be thought as the score of a classifier associated to the element of dictionary corresponding to event e used at timestamp t.

Alignment tasks arise in various domains of signal processing. Let us describe more formally two examples: the first one has been tackled in Bojanowski, Lajugie, Bach, Laptev, Ponce, Schmid, and Sivic [2014], and the second one will be described in this chapter.

1. Note that the score is much more than a simple ordered list of notes or chords. But for now we do not consider the information about the temporal extent of notes.

2. For instance, by reading the score we know the order in which notes (A,B,C) will be played in an audio recording but this do not gives the exact temporal extent of each note.

6.1.1 Examples

Example 1: localization of actions in videos. The first example comes from computer vision and we have deeply detailed it in Bojanowski, Lajugie, Bach, Laptev, Ponce, Schmid, and Sivic [2014]. The goal is to perform action recognition of K actions belonging to a dictionary $\mathcal{D}$. Note that, in this application, for simplicity, the possibility that several actions occur simultaneously is not considered.

In previous works, action recognition is generally cast as a classification problem for which training instances of *one* action are provided in the form of short video samples during training [Laptev et al., 2008, Liu et al., 2011, Wang et al., 2011]. In Bojanowski et al. [2014], we assume to be given a video stream with an ordered list of *several* actions actually occuring in the video in the given order. For example, this can be a video of someone going to the restaurant, sitting at a table, eating and then leaving for which we have a list of the form "sit down", "eat", "stand up". These data can come from the shooting script of the movie. They are a pretty accurate description of what happens in the video.

However, the temporal localization as well as the temporal extent of actions are missing because scripts do not include any information about the localization of what is described. Bojanowski, Lajugie, Bach, Laptev, Ponce, Schmid, and Sivic [2014] cast this problem as a weakly-supervised localization task using a square loss cost. This turns out to be a special case of the approach described in this chapter.

Example 2: audio-to-score problem. In this chapter, we focus on another example: the audio-to-score task. The goal is to align an audio interpretation to the score (a.k.a known as *partition* or *music sheet*) that is supposed to be played. This is also called the score-following task [Orio et al., 2003, Cont et al., 2007] in a real-time context. This has an interest in itself for the tracking of live performances for instance. But it is also a front-end task for some important applications in music such as the query-by-humming [Ghias et al., 1995], which corresponds to recognize a song approximately hummed by someone or for audio editing [Dannenberg, 2007].

We focus on the offline setting, namely when the whole temporal signal has been observed and that the task is performed offline. Note however, that alignment need often to be performed online or in real-time. The approach presented there is offline for the training stage, but can be used at testing time in the online setting thanks to some algorithmic advances designed to bridge the gap between online and offline audio-to-score [Dixon and Widmer, 2005].

6.1.2 Contributions of this chapter

In this chapter, we make the following contributions:

1. we propose a weakly-supervised structured prediction model based on regularized empirical loss minimization for the problem of alignment of a time series to a predefined template. Our approach uses two information: (i) an

exact list of events, (ii) an *approximate* information about duration about durations of events,

2. we define a proper loss for this task,
3. we propose a convex relaxation of this loss that lead to a tractable convex optimization problem that we solve using a Frank-Wolfe algorithm,
4. as a minor contribution, we relate the model for aligning videos and actions of Bojanowski, Lajugie, Bach, Laptev, Ponce, Schmid, and Sivic [2014] to structured prediction,
5. in the specific case of the audio-to-score task, we propose an easy way to use the prior knowledge about the durations of events given by the score and include this directly in the convexified loss minimization problem.

6.1.3 Notations

For now, let us describe an abstract model that allows us to deal with these two applicative cases.

Let us consider a temporal signal X of temporal horizon T. Each timestamp is assumed to be the observation of p features, that is, $X \in \mathbb{R}^{T \times p}$. We recall that our aim is to align X with a template that may consist in a list of actions or a list of successive musical events played. Let us see how we encode such a template.

Representation of templates. A template is a list of E symbols called *events*. An event is an element of a dictionary $\mathcal{D}$. We suppose that each event is a combination of *classes* of a certain set $\mathcal{A}$ of cardinal K. The dictionary is the set of all subsets of $\mathcal{A}$ that we denote $2^{\mathcal{A}}$ and which has cardinal $2^{|\mathcal{A}|}$. Thus each event can be represented by a binary vector $\Phi^e \in \{0,1\}^K, e = \{1, \dots, E\}$ such that $\Phi^e_i = 1$ if class i is present in the event. A template is an ordered list of E successive events. It can thus be identified to as a sequence of E binary vectors, that we stack in a binary matrix $\Phi \in \{0,1\}^{E \times K}$. Note that two settings can be distinguished.
First, the *exclusive* one where only one element per row of Φ can be non-zero. In the case of music, we can think about the *monophonic* setting where one note is played at a time. Second, the *inclusive* one where several elements per row of Φ can be non-zero. In the inclusive setting, one event is a superposition of several atomic elements of the vocabulary. In music, this is the case of chords and this is referred to the *polyphonic* setting. This is a *multi-label* classification problem we have seen in Chapter 2 with K possible labels. The classes used to make the elements of the dictionary $\mathcal{D}$ can take the form of actions labels in the context of videos or of notes in the context of music.

Parametrization of alignments. Recall that the goal of the alignment procedure between a signal X and a template of length E is to find a non-decreasing mapping m satisfying time contiguity constraints. For more details about this object, we refer to Sec. 3.3.1 of Chapter 3. Such an alignment can be represented through

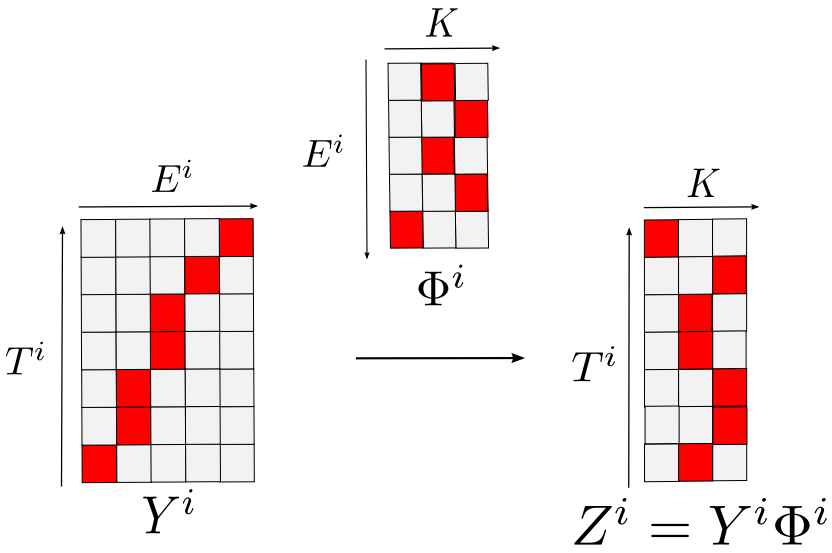


Figure 6.1: Illustration of the relation between the assignment matrix Z^i and the corresponding alignment one Y^i.

an *alignment matrix* Y. Figure 6.1 shows that Y can be thought as the graphical representation of the function m. The matrix Y is binary of dimension $T \times E$ and $Y_{t,e}$ is equal to 1 if $m(t) = e$ and 0 otherwise. The set of all alignments matrices corresponding to an alignment of a template Φ of E events with a temporal signal X of length T is denoted by $\mathcal{Y}(\Phi, X)$. As we did in previous chapters, to avoid cumbersome notation, we omit this dependence in the data Φ and X when it is clear from the context.

From alignments to assignments. Because a template is a sequence of subsets of $\{1, \dots, K\}$, alignments can also be represented using an *assignment* matrix Z which is the binary indicator membership of timestamps. Roughly speaking, if we see Y as the graphical representation of the alignment mapping m, Z is the representation of the corresponding assignment mapping $n : \{1, \dots, T\} \to \mathcal{D}$. The matrix Z is binary of size $T \times K$ and such that $Z_{i,j} = 1$ if and only class j of the template is present at timestamp i. The set of assignment matrices Z corresponding to assignments associated to elements of $\mathcal{Y}(\Phi, X)$ is denoted by $\mathcal{Z}(\Phi, X)$. We have

$$Z = Y\Phi.$$

Note that the mapping $Y \to Y\Phi$ is bijective from $\mathcal{Y}(\Phi, X)$ to $\mathcal{Z}(\Phi, X)$. The relation between Z and Y is depicted by Fig. 6.1. In the next sections, depending on our needs, we will switch between these two representations.

6.2 Alignment as a weakly-supervised structured prediction task

In this section we show how the task of finding an alignment between a signal X and its template Φ can be seen as a weakly supervised version of classification and we relate our approach to structured prediction

6.2.1 Fully-supervised setting

First, let us recall how we can cast multiclass classification using the notations of structured prediction.

Classification task. Let $X \in \mathbb{R}^{T \times p}$ be a time series. Let us assume that we want to learn one *affine* discriminative function $d_k : \mathbb{R}^p \to \mathbb{R}$ for each class $k \in \{1, \dots, K\}$. The goal is then to classify each timestamp $t \in \{1, \dots, T\}$ using the associated prediction function $f_k(x) = \text{sign}(d_k(x))$. That is, we aim at learning for each $k \in \{1, \dots, K\}$ a mapping

$$d_k : \mathbb{R}^p \to \mathbb{R}$$

such that there exists $(w_k, b_k) \in \mathbb{R}^p \times \mathbb{R}$ such that

$$\forall x \in \mathbb{R}^p, d_k(x) = w_k^\top x + b_k.$$

Note that b_k is usually not regularized during the learning process for square loss based discriminative models [Bach and Harchaoui, 2008] as well for the standard support vector machine formulation [Cortes and Vapnik, 1995]. We need to stress that, to get more compact notations and as is usually done in practice [Hastie et al., 2009], we deal with the constant term by adding a constant column of value α to the matrix X and learn a *linear* mapping f_k on the p relevant features augmented with this additional one [3]. Thus, f_k is of the form

$$\forall x \in \mathbb{R}^p \qquad f_k(x) = w_k^\top x.$$

We stack the p classifiers in a matrix $W \in \mathbb{R}^{p \times K}$. Using these, our goal is to predict a classification mapping n that we represent through its assignment matrix Z.

Following the approach described in Chapter 1, we consider a prediction model

3. Since our learning algorithms are based on the minimization of a data fitting term plus a regularizer that is the sum of the squared norms of each discriminative function $\|w_k\|_2^2$, this trick makes the bias regularized. So typically we set $\alpha = 10000$ so that regularization does not have influence on the learning of the constant term. Note also that we cannot get rid of the constant term in practice since the constant term allows us to adjust the threshold of the linear part of $\widetilde{f}_k$.

based on the following *discriminative function*

$$\begin{aligned} F(X, Z; W) &= \operatorname{Tr}(Z^\top X W) \\ &= \sum_{t=1}^{T} \sum_{k=1}^{K} z_{t,k} w_k^\top X_{t,.}. \end{aligned}$$

For now, we do not make use of any template and consider prediction to be done using by finding $f(X, \Phi; W)$ such that

$$f(X, \Phi; W) \in \operatorname*{argmax}_{Z \in \{0,1\}^{T \times K}} \operatorname{Tr}(Z^\top X W), \tag{6.1}$$

or equivalently

$$f(X, \Phi; W) \in \operatorname*{argmax}_{z_{t,k} \in \{0,1\}} \sum_{t=1}^{T} \sum_{k=1}^{K} z_{t,k} w_k^\top X_{t,.}.$$

Note that this corresponds to considering one multi-label classification problem at each timestamp. Since the $z_{t,k}$ are binary, this quantity is maximal for $z_{t,k} = \mathbb{1}_{w_k^\top X_{t,.} \geq 0}$. Thus, for each timestamp t, predictions are done independently and the predictions are also independent across labels.

Alignment task. We consider the alignment problem, in which predictions are not independent anymore. This means that the set where Z lies is constrained. Let us assume to be given a pair (X, Φ). Finding an alignment is similar to what we have done in Eq. (6.1) but we perform the argmax on the subset $\mathcal{Y}(\Phi, X)$ of binary indicator matrices authorized by the template Φ. The prediction function f is thus defined by:

$$f(X, \Phi; W) \in \operatorname*{argmax}_{Y \in \mathcal{Y}(\Phi, X)} \operatorname{Tr}(\Phi^\top Y^\top X W),$$

or equivalently using the assignment matrices $Z = \Phi Y$:

$$f(X, \Phi; W) \in \operatorname*{argmax}_{Z \in \mathcal{Z}(\Phi, X)} \operatorname{Tr}(Z^\top X W). \tag{6.2}$$

Let us now see how to learn the parameter W of this function.

Full-supervision. Let us consider that we are given N training triplets of data

$$(X^i, \Phi^i, Y^i),$$

or

$$(X^i, \Phi^i, Z^i),$$

where Y^i corresponds to the groundtruth alignment mapping m^i. The goal is to learn W that is used in Eq. (6.2) directly from data. To learn W, we adopt the standard regularized empirical loss minimization setting that minimizes the dis-

agreement between predictions and training examples plus a regularization term. Assuming that we are given a loss ℓ between assignment matrices, we ideally solve the following optimization problem:

$$\underset{W\in\mathbb{R}^{p\times K}}{\text{minimize}}\Big\{\frac{1}{N}\sum_{i=1}^{N}\ell\big(f(X^i,\Phi^i;W),Z^i\big)\Big\}+\frac{\lambda}{2}\|W\|_2^2. \tag{6.3}$$

Input and output. There are two ways of looking at the problem depending on what we call input and output among the N data triplets

$$(X^i,\Phi^i,m^i)_{1\leq i\leq N}.$$

If X^i is the input and the pair (Φ^i,m^i) the output, we are addressing a multi-label classification problem. For music, this means that we see the audio recording as our only input and that the output is a classification mapping that assigns to each timestamp one event. If we consider the input data to be the pair (X^i,Φ^i) and the output to be the alignment mapping[4], then we are considering the problem as a structured prediction one whose goal is to find a correspondence between two input signals. These two views are different and give rise to several possible interpretations of the task we are performing.

At test time, we consider to be provided as input a pair (X^i,Φ^i), thus we adopt the point of view of prediction of an alignment mapping. However, at training time we adopt the classification point of view. Namely our inputs are time series X^i and our outputs are (Φ^i,Z^i). We consider we have only a partial information about our outputs.

Our major concern comes from the fact that in real applications, the triplets (X^i,Φ^i,m^i) are very rarely available. We generally only have access to pairs (X^i,Φ^i). In the following section, we propose to handle this lack of supervision starting from the fully supervised case.

6.2.2 Weakly-supervised setting

Starting from Eq. (6.3), we need to derive a counterpart to the fully-supervised empirical loss minimization problem when the groundtruth alignment Y^i is not present anymore. However we are still provided the Φ^i of the output pair (Φ^i,Y^i). That is, we only have access to a partial supervision, that we call in the following *weak supervision*. The template Φ^i still gives us a strong knowledge about what the classification mapping should look like. The idea is thus to minimize the empirical loss not only over the parameter of the discriminative function but also with respect to the alignments, and to solve a minimization program of the form:

$$\underset{W\in\mathbb{R}^{p\times K}}{\text{minimize}}\ \frac{1}{N}\sum_{i=1}^{N}\min_{Z^i\in\mathcal{Z}^i}\ell\big(f(X^i,\Phi^i,W),Z^i\big)+\frac{\lambda}{2}\|W\|_2^2, \tag{6.4}$$

4. This corresponds to having the same kind of inputs than the one we had in Chapter 5. That is, we have two input signals that we want to align.

where we denote the sets $\mathcal{Z}^i = \mathcal{Z}(\Phi^i, X^i)$.[5] An intuition about this objective is given in Sec. 1.5.2 of Chapter 1 and in Fig. 1.4. The goal is to minimize the loss between prediction and the set of outputs authorized by the weak supervision. In other words, if we denote the set of all subsets of $\mathcal{Z}$ by $2^{\mathcal{Z}}$, we consider the empirical loss minimization problem with the loss

$$\widetilde{\ell} : \mathcal{Z} \times 2^{\mathcal{Z}} \to \mathbb{R}$$

defined by

$$\forall Z \in \mathcal{Z}, \forall \widetilde{\mathcal{Z}} \in 2^{\mathcal{Z}} \qquad \ell(Z, \widetilde{\mathcal{Z}}) = \inf_{Z' \in \widetilde{\mathcal{Z}}} \ell(Z, Z').$$

Unsupervised view. The approach of Eq. (6.4) can also be interpreted in an other manner if we see our training data as pairs of one time series and a template (X^i, Φ^i). In that case, we see Eq. (6.4) as a (non-convex) discriminative clustering objective [Guo and Schuurmans, 2007, Bach and Harchaoui, 2008]. Imagine that we are given pairs of audio recordings X^i with a template Φ^i which is the order of notes. Equation (6.4) seeks to jointly finding the optimal correspondence Z^i between signals and templates while learning a classifier for each note. If $\lambda = 0$ and if the loss function has non-negative value and such that $\forall Z,\ \ell(Z, Z) = 0$, this cost is minimal and equals to 0 in a situation where it is possible to find $Z^i \in \mathcal{Z}^i$ such that there exists classifiers that allow us to recover exactly this alignment Z^i. Thus, seeing things that way, what we called a weakly-supervised objective is an unsupervised objective making use of data pairs (X^i, Φ^i).

6.2.3 Losses over alignments

Now, let us discuss the possible choices for the loss between the elements of $\mathcal{Z}$.

Area loss of Lajugie et al. [2014b] and Chapter 5. In Chapter 5, we have seen that a very appealing loss between warpings is the area between them. On Fig. 6.2, this loss is represented as the area between the two depicted alignments. This loss is also very attractive for alignments. Let Y^1 and Y^2 be two alignment matrices. The area as defined in Chapter 5 and in Lajugie et al. [2014b] corresponds to the sum over each event of the difference of the ending timestamps of events in the two alignment. That is, for each event e corresponding to a row e of Y^1 and Y^2, we define

$$\delta_l^e = |\max\{t, Y^1_{t,e} = 1\} - \max\{t, Y^2_{t,e} = 1\}|,$$

and thus the area is

$$\ell_{\text{end}}(Y^1, Y^2) = \sum_{e=1}^{E} \delta_l^e.$$

On Fig. 6.2, we give an illustration of the quantity δ_l^e.

5. Similarly for alignment matrices, we denote the set by $\mathcal{Y}^i = \mathcal{Y}(\Phi^i, X^i)$.

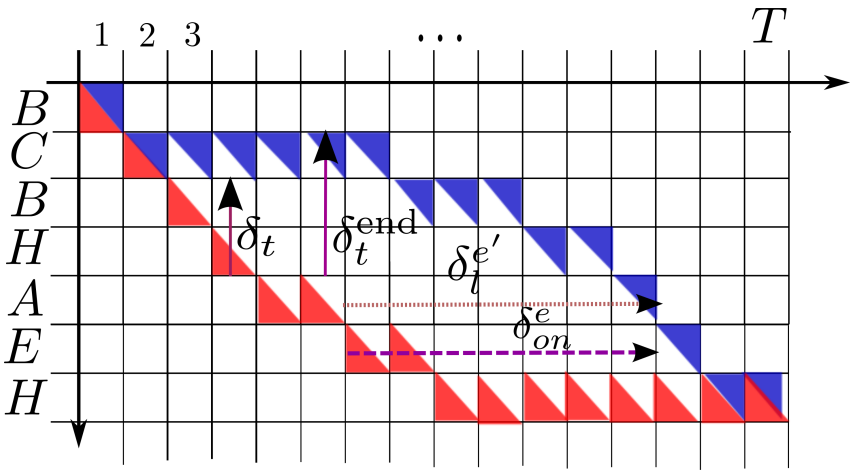


Figure 6.2: An illustration of the onset loss and the delay between onsets of events. The events there are music notes. The formula of Eq. (6.5) corresponds to summing the δ_t whereas the delay between onsets is the sum of the δ_{on}^e. These two quantities are the same for alignments.

Onset loss. However, in several applications and especially in music, one is more interested in the delay between the *onsets* that are the beginnings of events. If we introduce

$$\delta_{on}^e = |\min\{t, Y_{t,e}^1 = 1\} - \min\{k, Y_{t,e}^2 = 1\}|,$$

the sum of the delays between onsets is:

$$\ell_{on}(Y^1, Y^2) = \sum_{e=1}^{E} \delta_{on}^e.$$

This quantity is exactly the loss that is used for assessing the quality of audio-to-score alignment [Cont et al., 2007] and is depicted in Fig. 6.2. More formally, the *onset loss* between two alignments depicted by their alignment matrices Y^1 and Y^2 can be written as:

$$\ell_{on}(Y^1, Y^2) = \|Y^1 L_E - Y^2 L_E\|_2^2, \tag{6.5}$$

where L_E is the lower triangular matrix of dimension E made of ones including on the diagonal[6]. On the figure, the difference between onsets is the sum over all rows of the quantities δ_{on}, and the loss defined by Eq. (6.5) corresponds to the sum over all columns $t \in \{1, \dots, T\}$ of the quantities δ_t. In the case of alignments, these two quantities are equal.

Note also that this loss is closed to the symmetric area loss defined in Chapter 5.

Link between area loss and onset loss. In the case of alignments, for the same reason that the onset loss can be expressed easily with matrices the area loss of La-

6. This quantity is exactly the sum of the δ_t in Fig. 6.2 which turns out to be also the sum of δ_{on}^e, the sum of differences between onsets.

jugie et al. [2014b] and Chapter 5 can be expressed as:

$$\ell_{\text{end}}(Y^1, Y^2) = \sum_{e=1}^{E} \delta_l^e = \|Y^1 \widetilde{L}_E - Y^2 \widetilde{L}_E\|_F^2,$$

where $\widetilde{L}_E$ is the E-dimensional lower triangular matrix filled with ones and with zeros on the diagonal. Namely, $\widetilde{L}_E = L_E - \text{Id}_E$. This equality can be understood as the equality between the sum over all columns $t' \in \{1, \dots, T\}$ of the quantities $\delta_{t'}^{\text{end}}$ and the sum over all rows of the $\delta_l^{e'}$ in Fig. 6.2. Note that, for t fixed, we have that $\delta_t = \delta_t^{\text{end}}$, thus the area loss and the onset loss are the same quantities for alignments.

The Hamming loss. In Chapter 5, we have seen that the Hamming loss is not a good way to measure the difference between the alignment matrices of $\mathcal{Y}$. However, in our case, it is reasonable to consider the Hamming loss over the *assignment* matrices. Recall that our fundamental problem is a multiclass classification one. The matrices Z can be thus considered as binary indicator matrices of class membership at each timestamp. For the multiclass setting, the Hamming loss has attracted a lot of attention because of its computational tractability and simplicity to analyze [Allwein et al., 2001, Hastie et al., 2009]. We recall that the Hamming loss is defined for two assignments Z and Z' by:

$$\ell_H(Z, Z') = \sum_{t=1}^{T} \sum_{e=1}^{E} \mathbb{1}_{Z_{t,e} \neq Z'_{t,e}}. \tag{6.6}$$

Since Z and Z' are binary matrices, we have the following expression for the Hamming loss:

$$\ell_H(Z, Z') = \|Z - Z'\|_F^2.$$

Equivalently, for the alignment matrices coming with corresponding templates, we define:

$$\ell_H(Y, \Phi, Y', \Phi') = \|\Phi Y - \Phi' Y'\|_F^2. \tag{6.7}$$

In this work, due to its computational simplicity, we use the Hamming loss. From now on, $\ell = \ell_H$. However, note that the onset loss will be used in Sec. 6.4.1.

6.3 Convexification using the square surrogate

As seen in Chapter 1, the problem of Eq. (6.4) is NP-hard in the general case. A common way to address this issue in the fully-supervised setting is to use a convex approximation of the functions involved in the empirical loss minimization problem (6.4), that are defined for each instance $i \in \{1, \dots, N\}$, by

$$g_i : W \to \ell\big(f(X^i, \Phi^i; W), Z^i\big).$$

We propose to use a similar approach. Due to the joint minimization over the labels Z and the classifiers W, the logistic surrogate or the SVM would lead to a non-convex formulation. Indeed, replacing ℓ by the margin rescaling surrogate used for structured SVMs [Tsochantaridis et al., 2005] would lead to:

$$\underset{W\in\mathbb{R}^{p\times K}}{\text{minimize}} \sum_{i=1}^{N} \min_{Z^i\in\mathcal{Z}^i} \max_{Z\in\mathcal{Z}} \Big\{ \|Z^i - Z\|_F^2 - \operatorname{Tr}\big((Z - Z^i)WX\big) \Big\} + \lambda\|W\|_2^2. \tag{6.8}$$

This is in general the minimization of a non-convex function and there is no obvious way to make it convex. An approach to deal with this cost could be to iterate between optimizing in $(Z^1, \dots, Z^N)$ and in W.[7]

However, we can easily find a square surrogate to the objective of the problem of Eq. (6.8), we will see that in this case, it is possible to get a convex relaxation. Minimization of a square loss based criterion have been studied for a long time [Legendre, 1805, Gauss, 1809]. When such a square loss criterion is penalized using a square regularizer (originally for dealing with numerical issues), we talk about ridge regression [Hoerl, 1959, 1962] or ridge classification for multiclass classification.

6.3.1 Convexification of the empirical loss minimization objective

Recall that we consider N training examples made of one time series

$$X^i \in \mathbb{R}^{T^i\times p}$$

with the associated template Φ^i. We use the structured decoding model of Eq. (6.1). We use the Hamming loss to measure the discrepancy between elements of our structured set of assignments:

$$\ell_H\big(Z, f(X, \Phi; W)\big) = \sum_{t=1}^{T} \mathbb{1}_{Z_t \neq f(X,\Phi;W)_t}.$$

We extend the convexification strategy using the square surrogate to the $0-1$ loss we have seen in Chapter 1. Namely, we replace each $\mathbb{1}_{Z_t\neq f(X,\Phi;W)_t}$ by $(Z_t - (WX)_t)^2$. This yields replacing ℓ_H by:

$$\sum_{t=1}^{T} \big(Z_t - (WX)_t\big)_2^2 = \|Z - WX\|_F^2.$$

We consider now a convexified learning problem of the following form:

7. This is the approach of Sec. 4.4.2 of Chapter 4.

$$\underset{W\in\mathbb{R}^{K\times p}}{\text{minimize}}\Big\{\frac{1}{N}\sum_{i=1}^{N}\min_{Z^i\in\mathcal{Z}^i}\|Z^i - X^iW\|_2^2\Big\} + \frac{\lambda}{2}\|W\|_2^2. \tag{6.9}$$

Note that if we define the set $\mathcal{Z}$ as the Cartesian product of all the individual $\mathcal{Z}^i$, and if we stack all the Z^i into a single matrix Z and all the time series into a single one, we can consider the problem as if there were only one training example [8]:

$$\underset{W\in\mathbb{R}^{K\times p}}{\text{minimize}}\ \min_{Z\in\mathcal{Z}}\Big\{\frac{1}{N}\|Z - XW\|_2^2 + \frac{\lambda}{2}\|W\|_2^2\Big\}. \tag{6.10}$$

With this convention our input data can be seen as a single example (X, Φ).

Since the optimization in W is unconstrained, we can set the gradient with respect to this variable to 0 and get W implicitly from Z. This yields the following relation:

$$\Big(\frac{1}{N}X^\top X + \lambda \mathrm{Id}_p\Big)W - \frac{1}{N}X^\top Z = 0. \tag{6.11}$$

We then use this relation to get

$$W = \Big(X^\top X + N\lambda \mathrm{Id}_p\Big)^{-1}X^\top Z,$$

and plug back this expression in the objective function of problem (6.10). We get the following optimization problem:

$$\underset{Z\in\mathcal{Z}}{\text{minimize}}\ \mathrm{Tr}(Z^\top BZ). \tag{6.12}$$

where

$$B = \frac{1}{N}\Big(\mathrm{Id}_T - X(X^\top X + N\lambda \mathrm{Id}_p)^{-1}X^\top\Big),$$

or equivalently, using the matrix inversion lemma:

$$B = \frac{1}{N}\Big(\mathrm{Id}_T - XX^\top(XX^\top + N\lambda \mathrm{Id}_T)^{-1}\Big),$$

which can in turn be simplified in:

$$B = \lambda(XX^\top + N\lambda I_T)^{-1}$$

Note that we recover the fact that the matrix B is positive semidefinite since the eigenvalues of $XX^\top(XX^\top + N\lambda \mathrm{Id}_T)^{-1}$ are smaller than 1. The objective function of Eq. (6.12) is convex in Z but the overall optimization problem is non-convex since the set $\mathcal{Z}$ is discrete.

8. We also introduce the analogous of Z that is Y as the stacking of all the Y^i and the corresponding set $\mathcal{Y}$.

Convex relaxation strategies. To make the problem convex we can relax the optimization problem on the convex hull $\overline{\mathcal{Z}}$ of $\mathcal{Z}$, that is

$$\underset{Z \in \overline{\mathcal{Z}}}{\text{minimize}} \operatorname{Tr}(Z^\top B Z). \tag{6.13}$$

This relaxation suffers from two different effects that make it attracted by some *degenerate* solutions.

The first one occurs if the convex optimization set $\mathcal{Z}$ is "too symmetric". Indeed, as noted by Guo and Schuurmans [2007], Bach and Harchaoui [2008], Joulin et al. [2010], if $\mathcal{Z}$ is symmetric by permutation of columns, the matrix Z with constant entries is an optimum of the cost in problem of Eq. (6.13). In our case, thanks to the sequential structure of the set $\mathcal{Z}$, we are not subject to this effect.

Second, other degenerate solutions are the columnwise constant assignment matrices. Indeed, let us consider the matrix Z that collapses all classes into one, for instance such a degenerate $Z \in \mathbb{R}^{5\times 3}$ could be

$$Z = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

Z is such that $\operatorname{Tr}(Z^\top B Z)$ is close to 0 for the objective function of Eq. (6.13) [9]. This columnwise constant solution is not in $\mathcal{Z}$ due to the alignment constraint, but if T is large, elements of $\mathcal{Z}$ similar to Z_0 are in $\mathcal{Z}$. To prevent these solution to trap our learning process, we need to go away from these. In general, in the weakly supervised setting, and especially when Z has some sequential structure and when its temporal horizon is short enough, these solutions are far away from the optimization set [Bojanowski et al., 2013, 2014, Joulin et al., 2014]. However, in our experiments (Sec. 5.5) we show that the bigger our optimization polytope becomes, the closer to degenerate solution we go.

6.3.2 Optimization

We are now optimizing a convex function over a compact and convex set. However, the convex set is defined as a convex hull of a finite but huge [10] number of points. The convex set $\overline{\mathcal{Z}}$ is thus a polytope with many faces. It is not clear if it has a description using a polynomial number of constraints. However, the dynamic

9. The fact that this Z is a degenerate solution is linked to the important fact that the classifier matrix W corresponds to *affine* classifiers. Indeed, we add to our feature matrix X a constant column of value α with α large, thus our p-th feature is constant equal to α for all timestamps t. Thus the matrix $B = \lambda(XX^T + N\lambda I_T)^{-1}$ has an eigenvalue almost equal to 0 associated to a vector very close to $\mathbf{1}_T$.

10. They are $\binom{T}{E}$ if the template has length E.

programming algorithm 6 presented in Chapter 3 let us

$$\underset{Z \in \mathcal{Z}}{\text{maximize}} \operatorname{Tr}(A^\top Z), \tag{6.14}$$

given any affinity matrix $A \in \mathbb{R}^{T \times K}$. So, we are able to solve linear programs over the set $\mathcal{Z}$. But, due to the fundamental theorem of linear programming [11], this yields to a solution of any linear program over the set $\overline{\mathcal{Z}}$.

In other words, we know how to maximize linear forms over our optimization set. Moreover, the gradient of the objective function of Eq. (6.13) is easy to compute and is:

$$g(Z) = BZ.$$

Thus, we are in a situation where we can use the Frank-Wolfe algorithm we have described in Sec. 1.6 of Chapter 1.

This optimization procedure yields to a solution $Z^* \in \overline{\mathcal{Z}}$. We can get back the corresponding classifiers W^* using the relation of Eq. (6.11).

For classification purpose, getting W^* is enough. However, one can be interested to get an assignment matrix $Z \in \mathcal{Z}$.

Need for rounding solutions. Since the mapping from timestamps to elements of the template is not given in data (X, Φ), we want to recover it as well as the classifiers. So, from the solution Z^*, we need to find a solution Z which lies in the set $\mathcal{Z}$. Several strategies exist, let us describe two we use in this chapter and that have been used by Bojanowski, Lajugie, Bach, Laptev, Ponce, Schmid, and Sivic [2014]:

1. The *rounding in Z* consists in rounding the solution Z^* to the closest point of $\mathcal{Z}$ in the sense of L_2 distance, that amounts to minimizing the following quantity over $\mathcal{Z}$:
$$\|Z - Z^*\|_2^2.$$
In the very important case where Z has all rows summing to one, a short calculation shows that $\operatorname{Tr}(ZZ^\top)$ is constant over $Z \in \mathcal{Z}$ and thus this rounding procedure amounts to finding a minimizer of
$$\operatorname{Tr}(Z^\top Z^*),$$
over $\mathcal{Z}$.

2. The *rounding using the decoding model of Eq.* (6.1). does not use directly the solution Z^*. It consists in using the classifier to solve the decoding problem of Eq. (6.1), that is to find a maximizer over $Z \in \mathcal{Z}$ of:
$$\operatorname{Tr}(Z^\top X W^*).$$

11. This theorem guarantees that when solving a linear program over a polytope, one optimal solution will be on a vertex of the polytope. See e.g Prop B.5.1 of [Bertsekas, 1999].

Now let us briefly describe a first application of this approach: the temporal localization of actions in video using a list of action given by the shooting script of a movie. We have considered this application in Bojanowski, Lajugie, Bach, Laptev, Ponce, Schmid, and Sivic [2014] and Bojanowski, Lajugie, Grave, Bach, Laptev, Ponce, and Schmid [2015]. Note that we did not make the link with structured prediction that we have done in this chapter.

6.3.3 Link with Bojanowski et al. [2014]

The optimization objective of Eq. (6.10) relaxed in Eq. (6.13), is exactly the optimization problem we solved in Bojanowski, Lajugie, Bach, Laptev, Ponce, Schmid, and Sivic [2014] for aligning a sequence of actions with a video recording. We considered $\sim$ 900 video clips stacked in a matrix $X \in \mathbb{R}^{T \times p}$ where $T \sim 70000$ and $p \sim 1000$. These clips are provided with a list of actions occuring in the video clips. Each clip is associated with around 6 to 12 events belonging to a set of 17 classes. We solve Eq. (6.13) using the Frank-Wolfe algorithm and demonstrate on a challenging video dataset that the use of the simple temporal order of events was able to produce classifiers as good at action localization and recognition than classifiers trained on fully supervised timestamped video.

In Bojanowski et al. [2014], we saw our approach as an instance of discriminative clustering methods [Guo and Schuurmans, 2007, Bach and Harchaoui, 2008] using a square loss cost. The previous sections show the deep links between our approach and structured prediction.

In this chapter, we propose to consider an other applicative case: the audio-to-score problem.

6.4 Weakly supervised approach for the audio-to-score problem

In this section, we focus on the audio-to-score problem. This is the task of aligning an audio recording $X \in \mathbb{R}^{T \times p}$ of temporal horizon T, on a template corresponding to the score. As for movie scripts, it is possible to see a score as an ordered list of E events that are played in the audio recording. Each event is one or a combination of notes among a set of K. This template is represented through a matrix $\Phi \in \{0,1\}^{E \times K}$. Thus, as Bojanowski et al. [2014] did for videos, we could just solve of Eq. (6.13), and test the performance of the resulting classifiers and the relevance of the alignment.

However, as shown in the experimental section 6.5, this approach is not satisfactory as soon as the number of events occurring in the recording is too large.

Using additional information. A score is much more than an ordered list of events. It gives information about the relative duration between them. Indeed, we know that a quarter note would last around twice less than a half one even if the interpreter of the score is always free in its interpretation. So, we have a strong

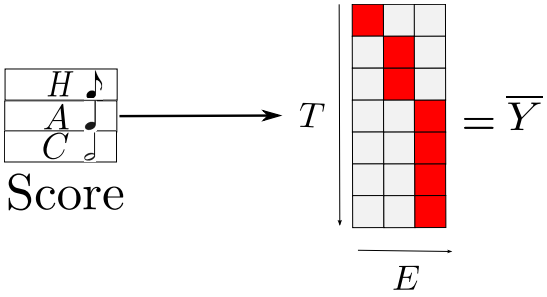


Figure 6.3: Illustration of the kind of information available for music. We are given a list of event with an a priori about their duration and we can build an expected alignment $\overline{Y}$.

prior about what the output alignment should look like, even if the score is never played strictly [12]. The standard way to use this knowledge in the musical literature is to build a hidden semi Markovian model (HSMM) [Raphael, 1999, Orio et al., 2003]. Our approach to cope with this knowledge is different. We build on the discriminative approach developed in Sec. 6.2, and propose to use the relative duration information as a penalty that we include in Eq. (6.13).

From a score, we propose to build an expected alignment matrix $\overline{Y}$ (or the corresponding assignment matrix $\overline{Z}$). The alignment $\overline{Y}$ is the one that would be obtained if the score were played strictly or if the recording were produced by a synthetizer. From now on, we assume to be given N training instances $(X^i, \Phi^i, \overline{Y}^i)$ where X^i is a temporal signal, Φ^i, the template and $\overline{Y}^i$ is the expected alignment. Note however that the expected alignment is a much weaker information than a manually fully timestamped information. To simplify notations, we do in the subsequent sections as if there were only one training example (X, Φ) [13]. In Fig. 6.3, we represent an example of such an expected alignment.

6.4.1 Duration priors

Global prior. The idea is to penalize the dissimilarity between the alignments Y outputed by the weakly-supervised optimization procedure and the expected alignment corresponding to the information we get. We call $\overline{Y}$ this expected alignment. As described in the previous section, a good way to measure the distance between an alignment $Y \in \mathcal{Y}$ and $\overline{Y}$ is to considering the sum of absolute differences between onsets in the two alignments. As noticed in Sec. 6.2.3, this can be written, with L the lower triangular matrix of size $E \times E$ with ones, as

$$\|YL_E - \overline{Y}L_E\|_F^2.$$

12. In top of the variability within an interpretation due to the interpreter, the composer often let some freedom. For instance, a musical phrase can be marked as having a "moderate" tempo and an other one as a "fast" tempo.

13. Imagine that we have stacked all the X^i, $\overline{Y}^i$ and Φ^i in three matrices, respectively X of size $T = \sum_{i=1}^N T^i$ by p, Φ of size $E = \sum_{i=1}^N E^i$ by K and $\overline{Y}^i$ of size T by E.

We call this term the *global prior*, it is a global penalization on the shape of the alignments.

Local prior. Another way to use the prior knowledge given by a score is to consider the mean alignment but not to penalize by the global shape of it. Instead we penalize individually the discrepancy between the duration of each event in the outputed alignment and the expected alignment. The idea is thus to penalize by:

$$\frac{1}{T}\sum_{e=1}^{E}(\mathbf{1}_T^\top Y_{\cdot,e} - \mathbf{1}_T^\top \overline{Y}_{\cdot,e})^2.$$

This can be written more compactly as:

$$\frac{1}{T}\|\mathbf{1}_T^\top Y - \mathbf{1}_T^\top \overline{Y}\|_2^2. \tag{6.15}$$

Musical interpretation of local and global prior. The global prior is expected to be more robust when the expected alignment is globally correct but when lots of local distortions occurs, that is when the global shape of the alignment is close to the expected one. In music this corresponds to the *rubato* setting [Kennedy, 1994]. On the contrary, the local prior is more suited for cases where the expected alignment is not globally correct but locally correct. For instance, in music this corresponds to *fermata* setting [Kennedy, 1994]; namely the relative durations of notes is globally correct except for some notes that correspond to end of musical phrases and where the interpret can wait as long as he or she wants.

6.4.2 The complete cost function

Adding the priors, which are only functions of Y, into the cost function of Eq. (6.10) yields to the following minimization problem over the set of binary matrices $\mathcal{Y}$:

$$\underset{W\in\mathbb{R}^{p\times K}, Y\in\mathcal{Y}}{\text{minimize}} \frac{1}{N}\Big(\|Y\Phi - XW\|_2^2 + \mu\|YL_E - \overline{Y}L_E\| + \nu\|\mathbf{1}_T^\top Y - \mathbf{1}_T^\top \overline{Y}\|_2^2\Big) + \frac{\lambda}{2}\|W\|_2^2, \tag{6.16}$$

where μ, ν and λ are parameters that need to be tuned by proper validation. Equation (6.16) is exactly the same as Eq. (6.10) with ΦY instead of Z and two priors term added. Like in the previous section, since W is unconstrained, we can set to 0 the gradient with respect to W, this yields to the following normal equation that links the optimization variables Y and W:

$$\frac{1}{N}(Y\Phi - XW) = (\lambda \mathrm{Id}_p + X^\top X)W. \tag{6.17}$$

As in the previous section we can thus write our optimization program as a quadratic one in the variables Y:

$$\underset{Y \in \mathcal{Y}}{\text{minimize}} \Big\{ \text{Tr}(\Phi^\top Y^\top B Y \Phi) + \mu \|Y L_E - \overline{Y} L_E\| + \nu \|\mathbf{1}_T Y - \mathbf{1}_T \overline{Y}\|_2^2) \Big\}, \tag{6.18}$$

where

$$B = \frac{1}{N}\big(\text{Id}_T - X(X^\top X + N\lambda \text{Id}_p)^{-1} X^\top\big).$$

6.4.3 Convex relaxation

Due to the discrete nature of the set $\mathcal{Y}$, Eq. (6.18) is a hard discrete combinatorial problem. As we did in in Sec. 6.3.1, we choose to relax $\mathcal{Y}$ into its convex hull $\overline{\mathcal{Y}}$. This leads to the following convex optimization problem:

$$\underset{Y \in \overline{\mathcal{Y}}}{\text{minimize}} \Big\{ \text{Tr}(\Phi^\top Y^\top B Y \Phi) + \mu \|Y L_E - \overline{Y} L_E\| + \nu \|\mathbf{1}_T Y - \mathbf{1}_T \overline{Y}\|_2^2) \Big\}, \tag{6.19}$$

On the set $\overline{\mathcal{Y}}$, as underlined in Chapter 3, we are able to minimize or maximize any linear form. Indeed, for any matrix $A \in \mathbb{R}^{E \times T}$ we can

$$\underset{Y \in \mathcal{Y}}{\text{maximize}} \, \text{Tr}(AY)$$

using the dynamic programming algorithm 6 of Chapter 3. As in the previous section, due to the so-called fundamental theorem of linear programming, minimizing a linear program over $\mathcal{Y}$ provides a solution of the LP over $\overline{\mathcal{Y}}$. We are thus in a situation where we can use the Frank-Wolfe algorithm that we have described in Sec. 1.6.2 of Chapter 1.

6.4.4 Rounding procedures

Once the problem (6.19) is solved over the convex set $\overline{\mathcal{Y}}$, we get a solution Y^* that is not a valid alignment (which is represented through a binary matrix). Note that, from Y^*, it is always possible to get classifiers W^*.

However, one can also be interested in the mapping obtained on the train set, since the exact alignment matrix was not part of the data. For that purpose it is necessary to round the solution Y^*. We have explored several types of rounding, let us present them now.

Rounding using the decoding model of Eq. (6.1). The first way to round, which is the simplest and that gives in our experiments the more accurate results, is to look at training data as if we were at testing time. That is, we use the optimal classifiers W^* associated to the optimum of Eq. (6.19) to solve the structured classification model of Eq. (6.1) using the template Φ corresponding to the training

instances. We thus solve the following linear program:

$$\underset{Y \in \mathcal{Y}}{\text{maximize}} \ \operatorname{Tr}(\Phi^\top Y^\top X W),$$

that can be done using the dynamic programming algorithm 6.

Rounding with the classifiers W. The rounding using the classifiers W, consists in using the W^* corresponding to the solution Y^* of Eq. (6.19) and to predict from it a valid assignment by solving the following least-squares criterion:

$$\underset{Y \in \mathcal{Y}}{\text{minimize}} \ \|\Phi Y - X W^*\|_2^2. \tag{6.20}$$

This is similar to the rounding in Z, but by replacing ΦY^* by its square relaxation XW.

Expanding the squares and noticing that, in the monophonic case $\operatorname{Tr}(Y^\top \Phi^\top \Phi Y)$ is constant for all matrices $Y \in \mathcal{Y}$, reduces the minimization problem of Eq. (6.20) to an LP over $\mathcal{Y}$ that we can solve using dynamic programming. It is purely equivalent in this case to the rounding using the decoding model.

In the polyphonic setting, that is when ΦY can have several non-zero entries per row, $\operatorname{Tr}(Y^\top \Phi^\top \Phi Y)$ is not constant anymore and expanding the squares yields to consider the following optimization program:

$$\underset{Y \in \mathcal{Y}}{\text{minimize}} \operatorname{Tr}(Y^\top Y \Phi \Phi^\top) - 2 \operatorname{Tr}(Y^\top \Phi^\top X W). \tag{6.21}$$

We can notice that $Y^\top Y = \operatorname{Diag}(Y^\top \mathbf{1})$, thus

$$\operatorname{Tr}(Y^\top Y \Phi \Phi^\top) = \operatorname{Tr}\big(\operatorname{Diag}(Y^\top \mathbf{1}) \Phi \Phi^\top\big),$$

and a simple calculation shows that:

$$\operatorname{Tr}\big(\operatorname{Diag}(Y^\top \mathbf{1}) \Phi \Phi^\top\big) = \operatorname{Tr}(Y^\top \mathbf{1} \operatorname{Diag}(\Phi \Phi^\top).$$

Thus the polyphonic setting can be dealt as the monophonic one by considering a linear programming problem over the set $\mathcal{Y}$.

Rounding in Z. As Bojanowski et al. [2014] did for videos and as suggested in Sec. 6.3.2, a natural idea is to round with respect to the assignment matrices. Namely, the idea is to round $Z^* = \Phi Y^*$ to the closest assignment matrix $Z \in \mathcal{Z}$ in terms of the Euclidean squared norm, that is:

$$\underset{Y \in \mathcal{Y}}{\text{minimize}} \ \|Y^* \Phi - Y \Phi\|_2^2. \tag{6.22}$$

By expanding the squares, we notice that this simply reduces to an LP, in the monophonic as well as in the polyphonic setting for the same reason than the one presented in the previous paragraph.

Rounding with the priors. Given the objective function of Eq. (6.18), one can be interested, from W^* to get a Y that minimizes Eq. (6.18):

$$\frac{1}{N}\left(\|Y\Phi - XW^*\|_2^2 + \mu\|YL_E - \overline{Y}L_E\|^2 + \nu\|\mathbf{1}_T^\top Y - \mathbf{1}_T^\top\overline{Y}\|_2^2\right) + \frac{\lambda}{2}\|W\|_2^2.$$

After expanding the squares, and using the same trick as before, namely that $Y^\top Y = \mathrm{Diag}(Y^\top \mathbf{1}_T)$ one gets that this is equivalent to minimizing:

$$\mathrm{Tr}(YY^\top\Delta) + \mathrm{Tr}(Y^\top\Gamma),$$

with

$$\Delta = \nu\mathbf{1}_T\mathbf{1}_T^\top,$$

and

$$\Gamma = \mathbf{1}_T\,\mathrm{Diag}(\Phi\Phi^\top + \mu LL^\top)^\top - 2XW\Phi^\top - 2\mu L\overline{Y} - 2\nu\mathbf{1}_T^\top\mathbf{1}_T\overline{Y}.$$

Such a problem is not a linear program over $\mathcal{Y}$. However, this can be solved using a dynamic programming algorithm. Let us detail how it can be cast in the framework presented in Sec. 3.1.1 of Chapter 3. Let E be the number of events associated to $\mathcal{Y}$. We define $\mathcal{A}_1, \ldots, \mathcal{A}_E$ as the set of authorized localizations for the beginnings of event e. We also add a last set $\mathcal{A}_{E+1}$, which is a dummy beginning, to enforce the E-th event to stop at T. These decision sets can thus be identified to the sets of timestamps $\{e, \ldots, T-(E-e)\}$ (the e-th action cannot start before the e-th timestamp and a symmetric condition holds concerning the end of the segment). The payoffs function are the following:

$$\forall e < E, \forall (t,t') \in \mathcal{A}_e \times \mathcal{A}_{e+1}, \quad V_e(t,t') = \sum_{i=t}^{t'} \Gamma_{i,e} + \sum_{i,j=t}^{t'} \Delta_{i,j} - I_{t\leq t'},$$

and for $e = E+1$,

$$V_{E+1}(t) = I_T(t),$$

where I_T is the convex indicator of timestamp T.

This allows us to derive a dynamic programming algorithm that is detailed in algorithm 9 [14].

6.5 Experiments: Monophonic case

For now, we have just conducted experiments in the monophonic setting. That is, the assignment matrix Z has exactly one nonzero value per row equals to one.

14. This algorithm is very similar to the one presented in Alg. 5. That could be expected since this algorithm also optimizes a quadratic function of the matrix Y as seen in Chapter 4.

Algorithm 9 Dynamic programming algorithm to minimize a quadratic program over $\mathcal{Y}$ of the form $\mathrm{Tr}(Y Y^\top \Delta) + \mathrm{Tr}(Y^\top \Gamma)$

Input: $X \in \mathbb{R}^{T \times p}$, $E \in \{2, \dots, T\}$, $\Delta \in \mathbb{R}^{T \times T}$, $\Gamma \in \mathbb{R}^{T \times E}$.
Set $C \in \mathbb{R}^{E \times T}$, $\mathrm{Ind} \in \mathbb{R}^{E \times T}$
Initialize $\forall t, C_{1,t} = \sum_{k=1}^{t} \Gamma_{k,1} + \sum_{k=1,k'=1}^{t} \Delta_{k,k'}$, $\mathrm{Ind}(1,t) = 1$
for $e = 2, \dots, K$ **do**
  for $t = e, \dots T$ **do**
    for $j = e, \dots t$ **do**
      Set $\alpha(j) = C(e-1, j-1) + \sum_{k=1}^{j} \Gamma_{k,1} + \sum_{k=j,k'=j}^{t} \Delta_{k,k'}$
    end for
    $C(e,t) = \min(\alpha)$
    $\mathrm{Ind}(e,t) = \mathrm{argmin}(\alpha)$
  end for
end for
Backtracking step: Set $\mu(E) = T$
for $i = E, \dots, 2$ **do**
  $\mu(i-1) = \mathrm{Ind}(i-1, \mu(i))$
end for
Output: μ the sequence of the end of events from which we compute Y.

6.5.1 Experimental setting

Dataset and features. Our experiments are conducted on the Finnish folk song dataset [Eerola and Toiviainen, 2004]. These songs have been collected across all Finland during the end of the 19-th century. This dataset is made of approximately 48 hours of music available in MIDI format (corresponding to 8614 songs).

We choose to model each of the 44 individual notes that appear in the dataset as a separate class. We model silences by adding a 45-th "silence" class. Thus we have $K = 45$.Note that, whereas the background class we used in Bojanowski, Lajugie, Bach, Laptev, Ponce, Schmid, and Sivic [2014] for videos had a fuzzy interpretation and can be seen as a model for noise, a silence has a musical meaning and is very different from simple noise.

We synthesize the musical interpretation from the available MIDI files by adding some white noise and modifying the tempo. The MIDI can be seen as a representation of the score and thus provides us with $\overline{Y}$. Since we know how tempo has been stretched, we have the exact groundtruth for our alignment. Due to the stretching of tempo, note that the groundtruth alignment Y_{gt} is different from the expected alignment $\overline{Y}$. Because of that, we are able to provide experiments in a strictly controlled setup.

In our experiments, we consider four different kind of stretches of the partition:

1. Non-stretched data: the audio recording is the exact interpretation of the MIDI score.
2. Rubato performance: we generate tempo curves that alternatively speed up and slow down the tempo.

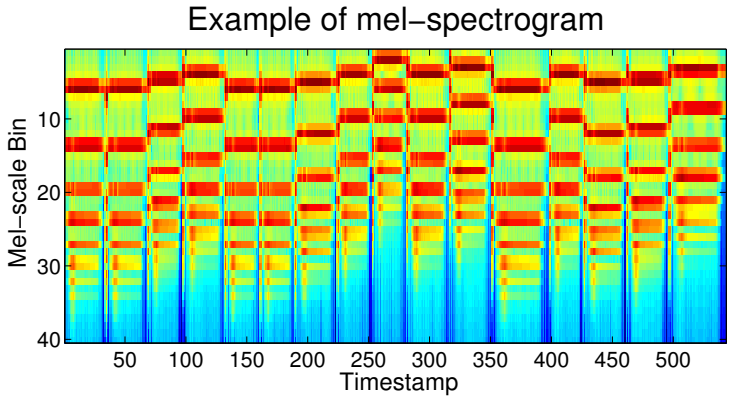


Figure 6.4: A typical audio signal representation that we use in this work. From an audio recording sampled at a rate of 44100Hz, we consider at each timestamp spaced of 0.08s a window centered around the timestamp of 0.16s on which we compute a discrete Fourier transform of the signal. This gives us a spectrogram. Then, we bin frequencies using the mel scale.

3. Fermata performance: the notes are in general played for the duration indicated on the MIDI score except for a few which have an increased duration.
4. A combination of settings (2) and (3).

We compute the spectrogram of the audio signal using overlapping windows of length 160ms and a stride of 80ms. For every window, the resulting spectrogram is a vector of dimension 1200. Additionally we compute a 40 dimensional transformation of the spectrogram by binning the frequencies over the mel-scale. This is known to be an approximate model of human auditory system (see, e.g., chapter 22 of [Gold and Morgan, 2000]). We use these as our final features. We also tried our method on a complete spectrogram; the results are similar to the ones presented in the rest of this section. An illustration of the kind of input signals we have is depicted in Fig. 6.4.

Performance measure. In this chapter, we evaluate our method by computing the mean delay between onsets of the predicted alignment and the groundtruth. This corresponds to the mean absolute temporal difference between the beginnings of notes in the two alignments. This is a common way to report quantitative results in music information retrieval [Cont et al., 2007]. As stated in Sec. 6.2.3, this quantity can be computed using alignment matrices and is proportional to

$$\ell_{\text{on}}(Y, Y_{\text{gt}}) = \|YL_E - Y_{\text{gt}}L_E\|_2^2.$$

This difference is divided by the number of events in the template and is in seconds. We average it over all music pieces we used for evaluation.

Train, validation and test splits. We split the available songs into a train, validation and test sets. Depending on the experiments, we use between 100 and 300 for

training and between 200 and 500 for both validation and testing. The validation and test sets are not involved in the optimization of the weakly-supervised model. The parameters of our method are always adjusting by a grid search: for various μ and ν, we compute the performance on validation data and select the best μ and ν. We obtain an alignment $Y^* \in \mathcal{Y}$ on validation and testing data by solving the decoding of the structured model in Eq. (6.2):

$$\underset{Y \in \mathcal{Y}}{\text{maximize}}\, \text{Tr}(Y^\top X W \Phi^\top).$$

Two kind of results. For our weakly-supervised approach, two kind of results can be looked at:

1. First ones are the performance of classifiers. That is, after solving the problem of Eq. (6.19), we get classifiers W^* and we can evaluate their performance by performing the decoding of Eq. (6.2) on test data that where not involved in the optimization stage.
2. Second ones are similar to the results we report in [Bojanowski et al., 2014, 2015] and that are the *training* error : after solving Eq. (6.19), we get $Y^* \in \overline{\mathcal{Y}}$ that we round using a rounding procedure of Sec. 6.4.4 to get $\widetilde{Y} \in \mathcal{Y}$. We compare this to the groundtruth.

Note that in all our experiments, we set $\lambda = 0$ since we have never seen a significant effect of this parameter in practice, at least using the mel-spectrogram features. This is probably due to the fact that our data are 40 dimensional and that we have a lot of examples for each notes (tenth of events occur in each musical piece).

6.5.2 Experiment: the need for priors.

In this experiment, we want to exhibit the fact that the proposed priors are indeed needed. To this end, we conduct experiments with no duration prior and no tempo stretching. This corresponds to considering the plain optimization problem of Eq. (6.13), similar to what was proposed in Bojanowski et al. [2014]. We use 300 full songs that we split into shorter songs of a fixed length k. We want to observe the variation of performance when the length k changes. Results for various lengths are presented in Table 6.1 for $\mu = \nu = 0$.

We see that our method works best when the number of maximum events is the smallest and fails for series with a large number of events. Indeed, as we add more and more events, the set from the optimization problem formulated in Eq. (6.13) has more and more symmetries. Constant degenerate solutions are not allowed due to the alignment constraint, but we get a solution that is constant on almost all the musical piece when we length of time series increase. An example of such a degenerate solution is depicted in Fig. 6.5.

k	10	15	20	30	50
mean length	64	93	124	174	224
delay (s)	0.42	0.59	0.68	0.93	1.09

Table 6.1: Effects of the maximal number of events E on the test performance when $\mu = \nu = 0$. Note that when the time series are short with few events occuring, the performance are the best. As the length of time series increases, the performances collapse. This is because, the longer the time series are, the more "almost piecewise constant" matrices like the one in the middle of Fig. 6.5 becomes attractive.

Rubato performance. In this experiment, we compare the separate effect of the global and the local prior in Eq. (6.18) in the Rubato setting. For this experiment, we considered 100 Finnish songs for training, 200 for testing and the same amount for validation purpose. The dashed curves (train MS, val MS and test MS) correspond to the performance of a prediction done using the actual score as a predictor.

In Fig. 6.6 (left), for μ large enough, the rounding in Z (which is the the closest assignment matrix in the L_2 sense) has the same performance as the baseline. On the contrary, in Fig. 6.6 (right), when ν is large, our method gets above the dashed baseline. This can be explained by the fact that when μ is large enough, the problem of Eq. (6.18) is the minimization of a well-conditionned quadratic form. The optimum of this function is very close to $\overline{Y}$, which explains why the black and dashed red curves meet for large μ. On the opposite, for large ν, the problem is ill-conditioned (quadratic form of rank one) and therefore has several minima.

In Fig. 6.6 (left), for large μ, for the reasons mentioned above, the optimal Y^* is almost equal to the score $\overline{Y}$. In that case, using the classifier rounding corresponds to solving the decoding problem from Eq. (6.2) with a model W learned on $\overline{Y}$. In the rubato setting, this solution gives the best results. We recall that a rubato setting corresponds to a fluctuation of the tempo around its average. The good performance can be partially explained by the fact that the average delay between the score and the recording is low in the rubato setting. Therefore, learning a model on a slightly faulty alignment provides a good classifier in the end. As expected, the local prior is not robust to rubato at all, whereas the global prior is.

Fermata performance. In the fermata setting, the performance of our approach is depicted in Fig. 6.7. This figure shows the same baseline as Fig. 6.6. Note that we also conducted experiments with respectively $\mu = 0$ and $\nu = 0$. The goal is to see if we observe a tradeoff between these terms and the discriminative one. Note that, for the ν parameter a clear tradeoff appears. On the opposite, the global prior does not seem to help the discriminative term, it just leads the solution to the one corresponding to the expected alignment $\overline{Y}$. As soon as μ is too big, the predicted Y sticks to the expected alignment $\overline{Y}$. Our method with properly adjusted local prior outperforms all baselines.

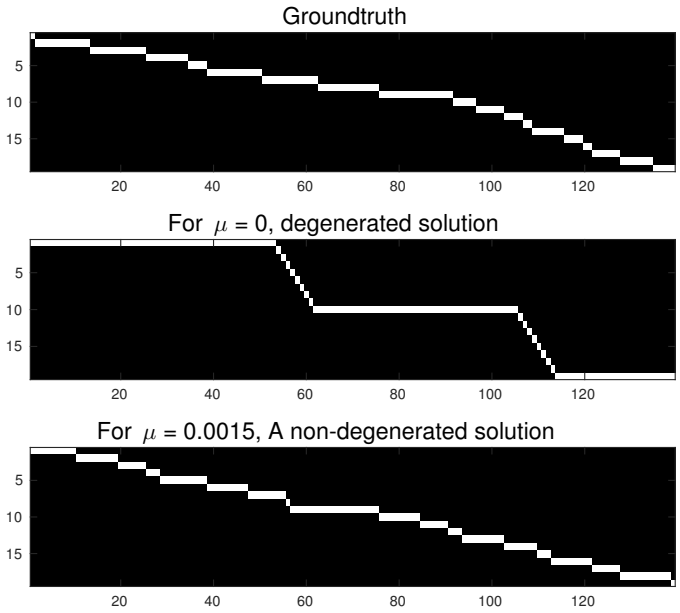


Figure 6.5: In the top, the groundtruth alignment. In the middle, example of a typical degenerated solution. In the bottom, when the prior is set to 0.015, we see that the algorithm converges to a non-degenerated solution.

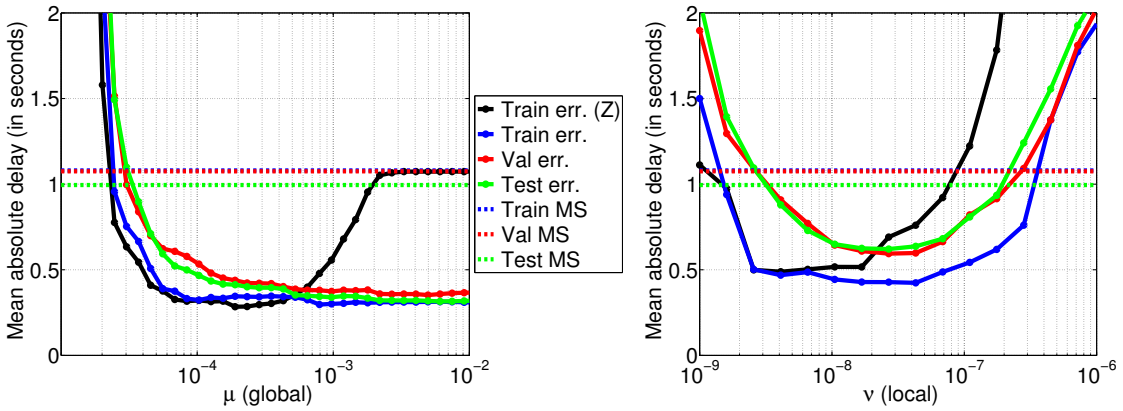


Figure 6.6: Individual effects of the local and global prior on data stretched in the rubato setting. Train MS, Val MS and Test MS is the mean delay between the score and the groundtruth, that is proportional to $\|Y_{\text{gt}} - \overline{Y}\|_2^2$.

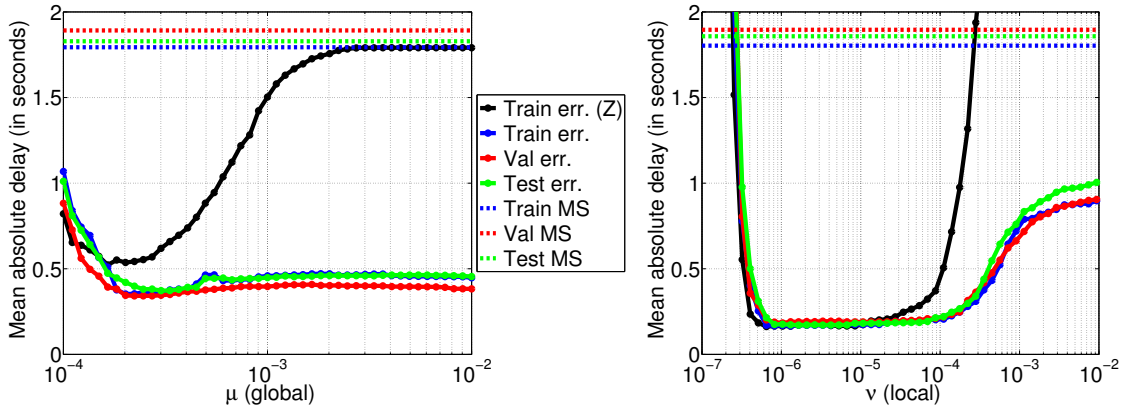
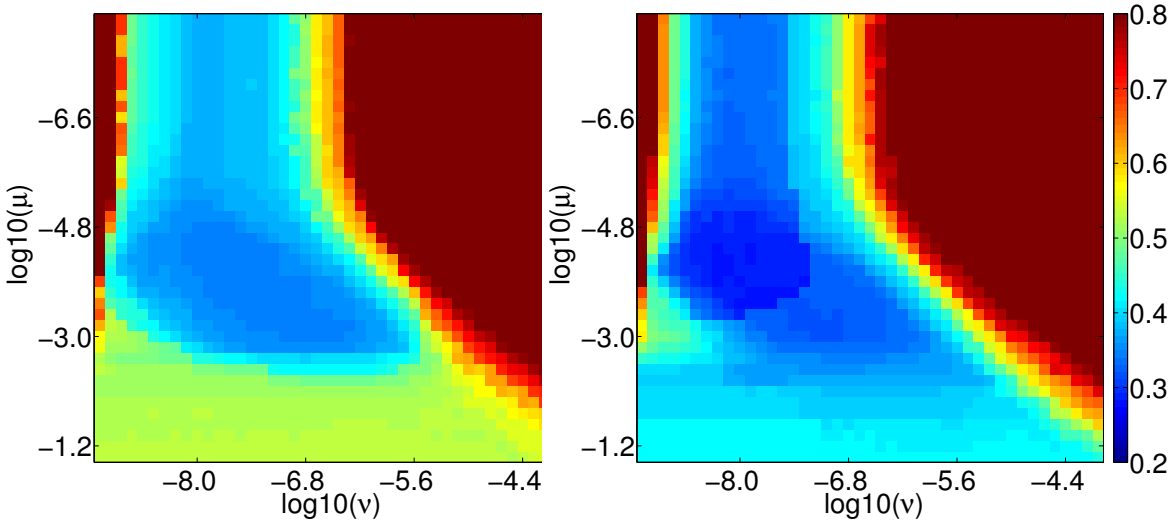


Figure 6.7: Individual effects of the local and global priors on data stretched by random fermata.

Combined performance. In the most general setting (with Fermata and Rubato present in the dataset), we observe the tradeoff of Fig. 6.8(a-b). In this experiment λ is fixed to 0 since in our previous experiments it does not seem to have an high impact. We observe that jointly penalizing using the global and the local prior is thus useful to improve the performance of the learned classifiers. Also, the achieved performance outperforms the baselines.



(a) Validation error (b) Test error

Figure 6.8: Performance of the proposed approach on the combined rubato and fermata performance. **(a-b)** Performance as a function of μ and ν on the validation and testing sets.

Robustness to noise. In this experiment, we consider data that have been stretched both with Fermata and Rubato, and we consider the robustness of our approach against a white noise. Namely, we add at each timestamp to all our synthesized signals white noise whose intensity is controlled through the ratio between the

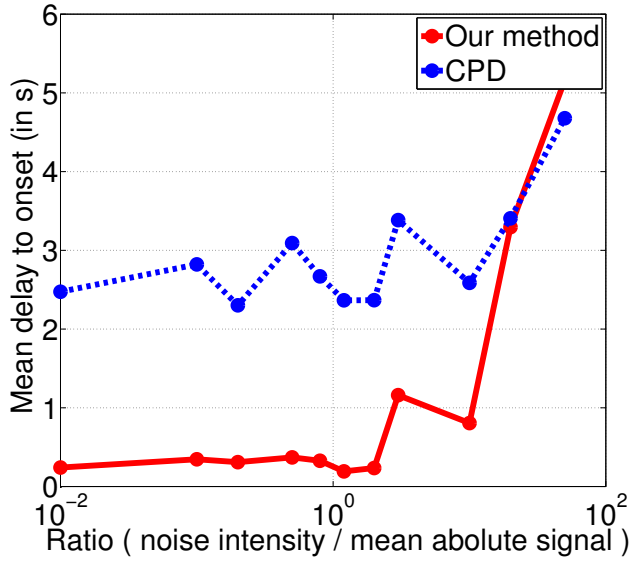


Figure 6.9: Robustness to noise and comparision to a change-point detection baseline (CPD).

standard deviation of the noise σ and the mean absolute value of the signal. We use our approach on 300 training songs, validate the parameters μ and ν on a validation set of size 250 and evaluate it on a testing set of the same size. We compare in Fig. 6.9 our method on testing data to a change-point detection algorithm that knows the number of notes. This baseline looks for the best segmentation using the 40-dimensional mel-spectrogram, and does not use learning at all [Bello et al., 2005].

6.6 Conclusion

In this chapter, we have proposed a way to deal with partial knowledge in the structured prediction framework for the alignment task. We have defined proper losses and convex relaxation of the associated empirical loss minimization problem. We have shown how this approach can be linked to some weakly-supervised methods based on discriminative clustering. In the case of audio-to-score alignment, we have also proposed specific and task dependent priors that can be included directly in the empirical loss minimization procedure.

Further, we would like to extend our approach to the mono-instrumental polyphonic setting, for instance a piano recording where chords are played. All the tools to deal with this setting were presented in that chapter. One step further would be to consider the polyphonic poly-instrumental setting. In this setting, we would learn for each of the different instruments $j \in \{1, \dots, J\}$, some classifiers W_j where W_j stacks all individual classifiers for possible notes played by instrument j. We would also consider J different alignment matrices Y_j and corresponding sets $\mathcal{Y}_j$. A first approach would be to consider the alignment of the score for each instrument independently. We would replace the minimization problem of Eq. (6.18)

by an objective of the form:

$$\underset{W_j \in \mathbb{R}^{p \times K}, Y_j \in \mathcal{Y}_j}{\text{minimize}} \frac{1}{N} \sum_{j=1}^{J} \left(\|Y_j \Phi_j - X W_j\|_2^2 + \mu \|Y_j L_E - \overline{Y}_j L_E\| + \nu \|\mathbf{1}_T^\top Y_j - \mathbf{1}_T^\top \overline{Y}_j\|_2^2 \right) + \frac{\lambda}{2} \|W_j\|_2^2,$$

that is the minimization of the cost of Eq. (6.18) for all instruments taken separately.

Bibliography

J. Aach and G. Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001.

H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Proc. ISIT*, 1973.

H. Akaike. A new look at the statistical model identification. *IEEE Trans. on Automatic Control*, 19(6):716 – 723, 1974.

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research*, 1:113–141, 2001.

D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

S. Arlot and M. Lerasle. Why V= 5 is enough in V-fold cross-validation. *arXiv preprint arXiv:1210.5830*, 2012.

S. Arlot, A. Celisse, and Z. Harchaoui. Kernel change-point detection. *arXiv preprint arXiv:1202.3878*, 2012.

Y. Artan, M. A. Haider, D. L. Langer, T. H. van der Kwast, A. J. Evans, Y. Yang, M. N. Wernick, J. Trachtenberg, and I. S. Yetik. Prostate cancer localization with multispectral mri using cost-sensitive support vector machines and conditional random fields. *IEEE Trans. on Image Processing*, 19(9):2444–2455, 2010.

A. Aspremont and S. Boyd. *Relaxations and randomized methods for nonconvex QCQPs*. 2003.

F. Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends in ML*, 2013.

F. Bach and M. Jordan. Learning spectral clustering. In *Adv. NIPS*, 2003.

F. Bach and Z. Harchaoui. DIFFRAC: a discriminative and flexible framework for clustering. In *Adv. NIPS*, 2008.

F. Bach and M. Jordan. Learning spectral clustering, with application to speech separation. *The Journal of Machine Learning Research*, 7:1963–2001, 2006.

F. Bach, D. Heckerman, and E. Horvitz. Considering cost asymmetry in learning classifiers. *The Journal of Machine Learning Research*, 7:1713–1741, 2006.

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012.

R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval*. ACM press New York, 1999.

J. Bai and P. Perron. Estimating and testing linear models with multiple structural changes. *Econometrica*, 1:47–78, 1998.

C. Banderier and S. Schwer. Why Delannoy numbers? *Journal of statistical planning and inference*, 135(1):40–54, 2005.

A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a Mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6:937–965, 2005.

P. L. Bartlett, M. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

M. Basseville and I. Nikiforov. *Detection of abrupt changes: theory and application*. Prentice hall, 1993.

A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

R. Bellman. Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences of the United States of America*, 42(10):767, 1956.

J. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler. A tutorial on onset detection in music signals. *IEEE Trans. on SAP*, 13(5):1035–1047, 2005.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on pattern analysis and machine intelligence*, 24(4):509–522, 2002.

P. Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. 2006.

D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *Proc. KDD*, 1994.

D. Bertsekas. *Nonlinear programming*. Athena Scientific, 1999.

D. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

D. Bertsekas. *Convex optimization algorithms*. Athena scientific, 2015.

W. Bi and J. Kwok. Efficient multi-label classification with many labels. In *Proc. ICML*, 2013.

W. Bi and J. T. Kwok. Multi-label classification on tree-and dag-structured hierarchies. In *Proc. ICML*, 2011.

C. M. Bishop et al. *Pattern recognition and machine learning*. Springer, 2006.

P. Bojanowski, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Finding actors and actions in movies. In *Proc. ICCV*, 2013.

P. Bojanowski, R. Lajugie, E. Grave, F. Bach, I. Laptev, J. Ponce, and C. Schmid. Weakly-supervised alignment of video with text. *arXiv preprint arXiv:1505.06027*, 2015.

P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Weakly supervised action labeling in videos under ordering constraints. In *Proc. ECCV*. 2014.

R. J. Bolton and D. J. Hand. Statistical fraud detection: A review. *Statistical science*, pages 235–249, 2002.

E. Borenstein and S. Ullman. Learning to segment. In *Proc. ECCV*, 2004.

L. Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

O. Bousquet and L. Bottou. The tradeoffs of large scale learning. In *Adv. NIPS*, 2008.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge Univ.Press, 2004.

Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *PAMI*, 2004.

P. S. Bradley, O. L. Mangasarian, and W. N. Street. Clustering via concave minimization. *Adv. NIPS*, 1997.

B. Brodsky and B. Darkhovsky. *Nonparametric methods in change-point problems*. Kluwer Academic Publishers Group, Dordrecht, 1993.

S. Bubeck. Theory of convex optimization for machine learning. *arXiv preprint arXiv:1405.4980*, 2014.

T. S. Caetano, J. J. McAuley, L. Cheng, Q. Le, and A. J. Smola. Learning graph matching. 31(6):1048–1058, 2009.

G. Carlier. Programmation dynamique, 1997.

A. Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comptes rendus des séances de l'Académie des sciences de Paris*, 25:536–538, 1847.

A. Chambolle and J. Darbon. On total variation minimization and surface evolution using parametric maximum flows. *International journal of computer vision*, 84 (3):288–307, 2009.

J. Chen and A. Gupta. *Parametric Statistical Change Point Analysis*. Birkhäuser, 2011.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM journal on scientific computing*, 20(1):33–61, 1998.

Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

T. Cohn and P. Blunsom. Semantic role labelling with tree conditional random fields. In *Proc. CCNNL*, 2005.

A. Cont. A coupled duration-focused architecture for real-time music-to-score alignment. *IEEE Trans. on Pattern analysis and machine intelligence*, 32(6):974–987, 2010.

A. Cont, D. Schwarz, N. Schnell, C. Raphael, et al. Evaluation of real-time audio-to-score alignment. In *Proc. ISMIR*, 2007.

T. H. Cormen, C. E. Leiserson, et al. *Introduction to algorithms*, volume 2. 2001.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. on information theory,*, 13(1):21–27, 1967.

F. C. Crow. Summed-area tables for texture mapping. *Proc. SIGGRAPH*, 18(3): 207–212, 1984.

M. Cuturi, J. Vert, O. Birkenes, and T. Matsui. A kernel for time series based on global alignments. In *Proc. ICASSP*, 2007.

R. B. Dannenberg. *An on-line algorithm for real-time accompaniment*. Ann Arbor, MI: MPublishing, University of Michigan Library, 1984.

R. B. Dannenberg. An intelligent multi-track audio editor. In *Proc. ICMC*, 2007.

J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proc. ICML*, 2007.

F. De la Torre and T. Kanade. Discriminative cluster analysis. In *Proc. ICML*, pages 241–248. ACM, 2006.

A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Adv. NIPS*, 2014.

H. Delannoy. Sur une question de probabilité traitée par d'alembert. *Buletin de la Société mathématique de France*, 23:262–265.

K. Dembczynski, A. Jachnik, W. Kotlowski, W. Waegeman, and E. Huellermeier. Optimizing the F-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *Proc. ICML*, 2013.

J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs. In *Proc. ECCV*, 2014.

F. Desobry, M. Davy, and C. Doncarli. An online kernel change detection algorithm. *IEEE Trans. on signal processing*, 53(8):2961–2974, 2005.

S. Dixon and G. Widmer. Match: A music alignment tool chest. In *Proc. ISMIR*, 2005.

J. S. Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

S. du Manoir, E. Schröck, M. Bentz, M. R. Speicher, S. Joos, T. Ried, P. Lichter, and T. Cremer. Quantitative analysis of comparative genomic hybridization. *Cytometry*, 19(1):27–41, 1995.

J. Edmonds and R. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

T. Eerola and P. Toiviainen. Suomen kansan esavelmat. Finnish folk song database. *http://esavelmat.jyu.fi/*, 2004.

A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *Adv. NIPS*, 2001.

M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88 (2):303–338, 2010.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

W. Fenchel. On conjugate convex functions. *Canadian Journal of Mathematics*, 1 (73-77), 1949.

T. Finley and T. Joachims. Supervised clustering with support vector machines. In *Proc. ICML*, 2005.

T. Finley and T. Joachims. Supervised k-means clustering, 2008.

R. A. Fisher. Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika*, pages 507–521, 1915.

E. Fix and J. L. Hodges Jr. Discriminatory analysis. nonparametric discrimination: consistency properties. Technical report, 1951.

L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

G. Forsythe and G. Golub. On the stationary values of a second-degree polynomial on the unit sphere. *Journal of the Society for Industrial & Applied Mathematics*, 13 (4):1050–1068, 1965.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

S. Fujishige. *Submodular functions and optimization*. Elsevier, 2005.

W. Gander, G. H. Golub, and U. von Matt. A constrained eigenvalue problem. *Linear Algebra and its applications*, 114:815–839, 1989.

C. Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. sumtibus Frid. Perthes et IH Besser, 1809.

S. Geisser. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70(350):320–328, 1975.

A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. Query by humming: musical information retrieval in an audio database. In *Proc. ICM*, 1995.

O. Gillet, S. Essid, and G. Richard. On the correlation of automatic audio and visual segmentations of music videos. *IEEE Trans. Circ. Syst. Vid. Tech.*, 17(3):347–355, 2007.

M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

B. Gold and N. Morgan. *Speech and audio signal processing*. John Wiley, 2000.

B. Gold, N. Morgan, and D. Ellis. *Speech and audio signal processing: processing and perception of speech and music*. John Wiley & Sons, 2011.

J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood component analysis. In *Adv. NIPS*, 2004.

J. C. Gower and G. J. S. Ross. Minimum spanning trees and single linkage cluster analysis. *Applied statistics*, 18:54–64, 1969.

D. Greig, B. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 271–279, 1989.

Y. Guo and D. Schuurmans. Convex relaxations of latent variable training. In *Adv. NIPS*, 2007.

S. B. Guthery. Partition regression. *Journal of the American Statistical Association*, 69 (348):945–947, 1974.

R. Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.

Z. Harchaoui, E. Moulines, and F. Bach. Kernel change-point analysis. In *Adv. NIPS*, 2009.

B. Hariharan, L. Zelnik-Manor, S. V. N. Vishwanathan, and M. Varma. Large scale max-margin multi-label classification with priors. In *Proc. ICML*, 2010.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer, 2009.

T. Hocking, G. Schleiermacher, I. Janoueix-Lerosey, V. Boeva, J. Cappo, O. Delattre, F. Bach, and J. Vert. Learning smoothing models of copy number profiles using breakpoint annotations. *BMC Bioinformatics*, 14(164), 2013.

A. Hoerl. Optimum solution of many variables equation. *Chemical Engineering Progress*, 55:67–78, 1959.

A. Hoerl and R. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

A. E. Hoerl. Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58:55–67, 1962.

D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *NIPS*, volume 22, pages 772–780, 2009.

N. Hu, R. B. Dannenberg, and G. Tzanetakis. Polyphonic audio matching and alignment for music retrieval. *Computer Science Department*, page 521, 2003.

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 23(1):67–72, 1975.

M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proc. ICML*, 2013.

A. Jain and R. Dubes. *Algorithms for clustering data*, volume 6. Prentice hall.

P. Jain, B. Kulis, J. Davis, and I. Dhillon. Metric and kernel learning using a linear transformation. *Journal of Machine Learning Research*, 13:519–547, 2012.

R. Jenatton, A. Gramfort, V. Michel, G. Obozinski, E. Eger, F. Bach, and B. Thirion. Multiscale mining of fmri data with hierarchical structured sparsity. *SIAM Journal on Imaging Sciences*, 5(3):835–856, 2012.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proc. ECML*, 1998.

T. Joachims. A support vector method for multivariate performance measures. In *Proc. ICML*, 2005.

T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

C. Joder, S. Essid, and G. Richard. Learning optimal features for polyphonic audio-to-score alignment. *IEEE Trans. on Audio, Speech, and Language Processing*, 21(10): 2118–2128, 2013.

D. Johnson. The NP-completeness column: An ongoing guide. *Journal of Algorithms*, 3(2):182–195, 1982.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Adv. NIPS*, 2013.

A. Joulin, F. Bach, and J. Ponce. Discriminative clustering for image co-segmentation. In *Proc. CVPR*. IEEE, 2010.

A. Joulin, K. Tang, and L. Fei-Fei. Efficient image and video co-localization with Frank-Wolfe algorithm. In *Proc. ECCV*, pages 253–268. 2014.

M. Journée, F. Bach, P. Absil, and R. Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010.

A. Kallioniemi, O.-P. Kallioniemi, D. Sudar, D. Rutovitz, J. W. Gray, F. Waldman, and D. Pinkel. Comparative genomic hybridization for molecular cytogenetic analysis of solid tumors. *Science*, 258(5083):818–821, 1992.

I. Katakis, G. Tsoumakas, and I. Vlahavas. Multilabel text classification for automated tag suggestion. In *Proc. ECML*, 2008.

M. Kennedy. *The Oxford dictionary of music*. Oxford University Press, 1994.

J. Keshet, S. Shalev-Shwartz, Y. Singer, and D. Chazan. A large margin algorithm for speech-to-phoneme and music-to-score alignment. *IEEE Trans. on Audio, Speech, and Language Processing*, 15(8):2373–2382, 2007.

S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for max-cut and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1): 319–357, 2007.

R. Killick, P. Fearnhead, and I. A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500): 1590–1598, 2012.

S. Kim, S. Nowozin, P. Kohli, and C. Yoo. Task-specific image partitioning. *IEEE Trans. on Image Processing*, 22(1-2):488–500, 2013.

H. Kirchhoff and A. Lerch. Evaluation of features for audio-to-audio alignment. *Journal of New Music Research*, 40(1):27–41, 2011.

A. Kolesnikov, M. Guillaumin, V. Ferrari, and C. H. Lampert. Closed-form approximate CRF training for scalable image segmentation. In *Proc. ECCV*. 2014.

D. Kuettel, M. Guillaumin, and V. Ferrari. Segmentation propagation in imagenet. In *Proc. ECCV*. Springer, 2012.

B. Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5 (4):287–364, 2012.

S. Kumar and M. Hebert. Discriminative random fields. *International Journal of Computer Vision*, 68(2):179–201, 2006.

S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate frank-wolfe optimization for structural SVMs. In *Proc. ICML*, 2013.

J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

J. Lagrange. Manière plus simple et plus générale de faire usage de la formule de l'équilibre. pages 77–112, 1811.

R. Lajugie, F. Bach, and S. Arlot. Large-margin metric learning for constrained partitioning problems. In *Proc. ICML*, 2014a.

R. Lajugie, D. Garreau, F. Bach, and S. Arlot. Metric learning for temporal sequence alignment. In *Adv. NIPS*, 2014b.

C. Lampert. Maximum margin multi-label structured prediction. In *Adv. NIPS*, 2011.

I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. CVPR*, 2008.

M. Lavielle. Using penalized contrasts for the change-point problem. *Signal Proces.*, 85(8):1501–1510, 2005.

E. Lebarbier. *Quelques approches pour la détection de ruptures à horizon fini*. PhD thesis, 2002.

E. Lebarbier. Detecting multiple change-points in the mean of a gaussian process by model selection. *Signal Processing*, 85:717–736, 2005.

A.-M. Legendre. Appendice sur la méthodes des moindres carrés. *Nouvelles méthodes pour la détermination de l'orbite des comètes*, pages 72–80, 1805.

C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. In *Proc. SIGIR*. ACM, 2003.

J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. In *Proc. CVPR*, 2011.

T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. ICCV*, 1999.

H. Masnadi-Shirazi and N. Vasconcelos. Risk minimization, probability elicitation, and cost-sensitive SVMs. In *Proc. ICML*, 2010.

P. Massart. *Concentration inequalities and model selection*. Springer, 2007.

B. Mcfee and G. Lanckriet. Metric learning to rank. In *Proc. ICML*, 2010.

B. McFee, L. Barrington, and G. Lanckriet. Learning content similarity for music recommendation. *IEEE Trans. on audio, speech, and language processing,*, 20(8): 2207–2218, 2012.

K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. on pattern analysis and machine intelligence*, 27(10):1615–1630, 2005.

M. Müller. *Information retrieval for music and motion*. Springer, 2007.

C. Myers and L. Rabiner. A comparative study of several dynamic time-warping algorithms for connected-word recognition. *Bell System Technical Journal*, 60(7): 1389–1409, 1981.

A. Nemirovskii. *Problem complexity and method efficiency in optimization.*

Y. Nesterov. *Introductory lectures on convex optimization*. Springer Science & Business Media, 2004.

Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

Y. Nesterov, A. Nemirovskii, and Y. Ye. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Adv. NIPS*, 2002.

A. Ng, Y. Weiss, and M. Jordan. On spectral clustering: Analysis and an algorithm. In *Adv. NIPS*, 2002.

M. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Proc. CVPR*, 2006.

H. Noma. Dynamic time-alignment kernel in support vector machine. *Adv. NIPS*, 2002.

S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365, 2011.

H. Nyquist. Certain topics in telegraph transmission theory. *American Institute of Electrical Engineers, Transactions of the*, 47(2):617–644, 1928.

N. Orio, S. Lemouton, and D. Schwarz. Score following: State of the art and new developments. In *Proc. NIME*, 2003.

M. Overton and H. Wolkowicz. Semidefinite programming. *Mathematical Programming*, 77(1):105–109, 1997.

C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 229–234. ACM, 1988.

J. P. Pestian, C. Brew, P. Matykiewicz, D. J. Hovermale, N. Johnson, K. B. Cohen, and W. Duch. A shared task involving multi-label classification of clinical free text. In *Proc. Workshop BioNLP 2007 of ACL*, 2007.

J. Petterson and T. Caetano. Submodular multi-label learning. In *Adv. NIPS*, 2011.

D. Picard. Testing and estimating change-points in time series. *Adv. Applied Probability*, pages 841–867, 1985.

F. Picard, S. Robin, M. Lavielle, C. Vaisse, and J.-J. Daudin. A statistical approach for array cgh data analysis. *BMC bioinformatics*, 6(1):27, 2005.

W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):pp. 846–850, 1971.

F. Rapaport, A. Zinovyev, M. Dutreix, E. Barillot, and J.-P. Vert. Classification of microarray data using gene networks. *BMC bioinformatics*, 8(1):35, 2007.

C. Raphael. Automatic segmentation of acoustic musical signals using hidden markov models. *IEEE Trans. on pattern analysis and machine intelligence*, 21(4): 360–370, 1999.

G. Rigaill. Pruned dynamic programming for optimal multiple change-point detection. Technical Report 1004.0887, arXiv, 2010.

H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

R. Rockafellar. *Convex Analysis, volume 28 of Princeton Mathematics Series*. Princeton university press, 1970.

C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004.

J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *JMLR*, 2006.

N. L. Roux, M. Schmidt, and F. R. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Adv. NIPS*, 2012.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *To appear in International Journal of Computer Vision*, 2015.

H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 26(1): 43–49, 1978.

G. Schleiermacher, I. Janoueix-Lerosey, A. Ribeiro, J. Klijanienko, J. Couturier, G. Pierron, V. Mosseri, A. Valent, N. Auger, D. Plantaz, et al. Accumulation of segmental alterations determines progression in neuroblastoma. *Journal of Clinical Oncology*, 28(19):3122–3130, 2010.

M. Schmidt, B. Babanezhad, M. Ahmed, A. Defazio, A. Clifton, and A. Sarkar. Non-uniform stochastic average gradient method for training conditional random fields. In *Proc. AISTATS*, 2015.

B. Schölkopf and A. J. Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2): 461–464, 1978.

B. Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 104–107. Association for Computational Linguistics, 2004.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proc. NAACL*, 2003.

S. P. Shah, W. L. Lam, R. T. Ng, and K. P. Murphy. Modeling recurrent DNA copy number alterations in array CGH data. *Bioinformatics*, 23(13):i450–i458, 2007.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proc. ICML*, 2007.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1):567–599, 2013.

S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for SVM. *Mathematical programming*, 127(1):3–30, 2011.

C. E. Shannon. A mathematical theory of communication. *Bell Technical journal*, 27: 379–423,623–656, 1948.

J. Shawe-Taylor and N. Cristianini. Support vector machines. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, pages 93–112, 2000.

N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. In *ECCV 2002*, pages 776–790. Springer, 2002.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on pattern analysis and machine intelligence*, 22:888–905, 1997.

Y. Shi, A. Bellet, and F. Sha. Sparse compositional metric learning. In *Proc. AAAI*, 2014.

P. Simard, Y. LeCun, and J. S. Denker. Efficient pattern recognition using a new transformation distance. In *Adv. NIPS*, 1993.

E. Spjøtvoll. A note on a theorem of Forsythe and Golub. *SIAM Journal on Applied Mathematics*, 1972.

H. Steinhaus. Sur la division des corps matériels en parties. *Bull. Acad. Pol. Sci., Cl. III*, 4:801–804, 1957.

I. Steinwart and A. Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

S. M. Stigler et al. The epic story of maximum likelihood. *Statistical Science*, 22(4): 598–620, 2007.

M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 111–147, 1974.

P. Szummer, M.and Kohli and D. Hoiem. Learning CRFs using graph cuts. In *ECCV*. 2008.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. *Adv. NIPS*, 2003.

B. Taskar, S. Lacoste-Julien, and M. Jordan. Structured prediction, dual extragradient and bregman projections. *The Journal of Machine Learning Research*, 7:1627–1653, 2006.

J. D. Thompson, F. Plewniak, and O. Poch. Balibase: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15 (1):87–88, 1999.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

F. Tisseur and K. Meerbergen. The quadratic eigenvalue problem. *SIAM review*, 2001.

A. Torres, A. Cabada, and J. J. Nieto. An exact formula for the number of alignments between two dna sequences. *Mitochondrial DNA*, 14(6):427–430, 2003.

I. Tsochantaridis, T. Joachims, T., Y. Altun, and Y. Singer. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 2007.

V. Vapnik. *Statistical learning theory*. Wiley New York, 1998.

B. Vercoe. Synthetic rehearsal: Training the synthetic performer. In *Proc. ICMC*, 1985.

P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. *Adv. NIPS*, 2001.

S. Vishwanathan, N. Schraudolph, M. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *Proc. ICML*, 2006.

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. on information theory*, 13(2):260–269, 1967.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007.

R. Von Mises and H. Pollaczek-Geiringer. Praktische verfahren der gleichungsauflösung. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 9(1):58–77, 1929.

M. Wainwright and M. Jordan. *Graphical models, exponential families, and variational inference*, volume 1. Now Publishers Inc., 2008.

H. Wallach. *Efficient training of conditional random fields*. PhD thesis, 2002.

H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *Proc. CVPR*, 2011.

K. Weinberger and L. Saul. Fast solvers and efficient implementations for distance metric learning. In *Proc. ICML*, pages 1160–1167, 2008.

K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009.

K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Adv. NIPS*, 2006.

M. Welling. Robust higher order statistics. *Proc. AISTATS*, 2005.

M. Wertheimer. Laws of organization in perceptual forms. *A Source Book of Gestalt Psychology*, 1923.

J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*. IEEE, 2010.

E. Xing, M. Jordan, S. Russell, and A. Ng. Distance metric learning with application to clustering with side-information. In *Adv. NIPS*, 2002.

D.-Y. Yeung and H. Chang. Extending the relevant component analysis algorithm for metric learning using both positive and negative equivalence constraints. *Pattern Recognition*, 39(5):1007–1010, 2006.

C. Yu and T. Joachims. Learning structural SVMs with latent variables. In *Proc. ICML*. ACM, 2009.

D.-C. Zhan, M. Li, Y.-F. Li, and Z.-H. Zhou. Learning instance specific distances using metric propagation. In *Proc. ICML*, 2009.

M. Zhang and Z. Zhou. A review on multi-label learning algorithms. *TKDE*, 2013.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, pages 56–85, 2004.

P. Zhao and B. Yu. On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7:2541–2563, 2006.

J. Zhu and E. P. Xing. Conditional topic random fields. In *Proc. ICML*, 2010.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Prédiction structurée pour l'analyse de données séquentielles

RÉSUMÉ : Dans cette thèse nous nous intéressons à des problèmes d'apprentissage automatique dans le cadre de sorties structurées avec une structure séquentielle. D'une part, nous considérons le problème de l'apprentissage de mesure de similarité pour deux tâches: (i) la détection de rupture dans des signaux multivariés et (ii) le problème de déformation temporelle entre paires de signaux. Les méthodes généralement utilisées pour résoudre ces deux problèmes dépendent fortement d'une mesure de similarité. Nous apprenons une mesure de similarité à partir de données totalement étiquettées. Nous présentons des algorithmes usuels de prédiction structuré, efficaces pour effectuer l'apprentissage. Nous validons notre approche sur des données réelles venant de divers domaines.

D'autre part, nous nous intéressons au problème de la faible supervision pour la tâche d'alignement d'un enregistrement audio sur la partition jouée. Nous considérons la partition comme une représentation symbolique donnant (i) une information complète sur l'ordre des symboles et (ii) une information approximative sur la forme de l'alignement attendu. Nous apprenons un classifieur pour chaque symbole avec ces informations. Nous développons une méthode d'apprentissage fondée sur l'optimisation d'une fonction convexe. Nous démontrons la validité de l'approche sur des données musicales.

MOTS-CLEFS : apprentissage, faible supervision, prédiction structurée, apprentissage de métrique, alignement musique sur partition, déformation temporelle

---

Structured prediction for sequential data

ABSTRACT: In this manuscript, we consider structured machine learning problems and consider more precisely the ones involving sequential structure.

In a first part, we consider the problem of similarity measure learning for two tasks where sequential structure is at stake: (i) the multivariate change-point detection and (ii) the time warping of pairs of time series. The methods generally used to solve these tasks rely on a similarity measure to compare timestamps. We propose to learn a similarity measure from fully labelled data, i.e., signals already segmented or pairs of signals for which the optimal time warping is known. Using standard structured prediction methods, we present algorithmically efficient ways for learning. We propose to use loss functions specifically designed for the tasks. We validate our approach on real-world data.

In a second part, we focus on the problem of weak supervision, in which sequential data are not totally labeled. We focus on the problem of aligning an audio recording with its score. We consider the score as a symbolic representation giving: (i) a complete information about the order of events or notes played and (ii) an approximate idea about the expected shape of the alignment. We propose to learn a classifier for each note using this information. Our learning problem is based on the optimization of a convex function that takes advantage of the weak supervision and of the sequential structure of data. Our approach is validated through experiments on the task of audio-to-score on real musical data.

KEYWORDS: machine learning, weak supervision, structured prediction, metric learning, music to partition alignment, time warping