UNIVERSITÉ DE GRENOBLE

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique et Mathématiques appliquées**

Arrêté ministériel : 7 août 2006

Présentée par
Cao Tri DO

Thèse dirigée par **Ahlame DOUZAL-CHOUAKRIA** et codirigée par **Michèle ROMBAUT**et co-encadré par **Sylvain Marié**

préparée au sein du
Laboratoire d'Informatique de Grenoble (LIG)
dans **l'école doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique (MSTII)**

Metric Learning for Time Series Analysis

Thèse soutenue publiquement le **date de soutenance**, devant le jury composé de:

Prénom NOM
Labo de bidule, Rapporteur, Présidente du jury
Prénom NOM
Labo de bidule, Rapporteur
Prénom NOM
Labo de bidule, Examinateur
Prénom NOM
Labo de bidule, Examinateur
Ahlame DOUZAL-CHOUAKRIA
LIG, Directeur de thèse
Michèle ROMBAUT
GIPSA-Lab, Co-Directeur de thèse

UNIVERSITÉ DE GRENOBLE

ÉCOLE DOCTORALE MSTII

Description de complète de l'école doctorale

T H È S E

pour obtenir le titre de

docteur en sciences

de l'Université de Grenoble-Alpes

Mention : Informatique et Mathématiques appliquées

Présentée et soutenue par

Cao Tri DO

Metric Learning for Time Series Analysis

Thèse dirigée par Ahlame DOUZAL-CHOUAKRIA

préparée au Laboratoire d'Informatique de Grenoble (LIG)

soutenue le date de soutenance

Jury :

Rapporteurs :	Prénom NOM	-	Labo de bidule
	Prénom NOM	-	Labo de bidule
Directeur :	Ahlame DOUZAL-CHOUAKRIA	-	LIG
Co-Directeur :	Michèle ROMBAUT	-	GIPSA-Lab
Encadrant :	Sylvain Marié	-	Schneider Electric
Président :	Prénom NOM	-	Labo de bidule
Examinateur :	Prénom NOM	-	Labo de bidule
	Prénom NOM	-	Labo de bidule

Todo list

Comment [CTD1]: Initial in [CAO] then your comment in the bracket 2

Comment [CTD2]: Initial in [CAO] then your comment in the bracket 2

Figure: Testing a long text string 2

qqch ne va pas, ça ne peut pas être $\frac{T}{f_e}$. Les indices ne correspondent pas 7

biblio application time series + applis 8

[biblio time series static learning] 8

[biblio supervisé] 9

[biblio semi-supervisé] 9

[biblio clustering] 9

Generalize for kNN 10

[biblio regression kNN] 10

[biblio kNN pondéré] 10

[biblio fuzzy kNN] 10

bilbio 11

Michèle: chiffre ou lettre pour la numérotation? 12

reformuler pour le gamma 18

refaire la figure 18

réécrire 19

Biblio 21

Partie non encore rédigée. A faire à la fin. 23

Acknowledgements

I would like to thanks:

- my directors
- my GIPSA collegues
- my AMA collegues
- my Schneider collegues
- my parents

Contents

Table des sigles et acronymes **xiii**

Introduction **1**

I Work positionning **5**

1 Machine Learning: state of the art **7**

1.1 Definition of a time series 7

1.2 Machine learning algorithms 9

1.3 Learning protocol 23

1.4 Limits of classical technics 26

2 Time series basic metrics **27**

2.1 Properties of a distance measure 28

2.2 Basic metrics for time series 28

2.3 Kernels for time series 29

2.4 Time series alignment 29

2.5 Multi-scale aspect 30

3 Time series advanced metrics **31**

3.1 Combined metrics for time series 31

3.2 Metric Learning: state of the art 32

II Metric learning for time series from multiple modalities and multiple scales **35**

4 Projection in the pairwise space **37**

4.1 Pairwise embedding . . . 37
4.2 Interpretation of the pairwise space . . . 37
4.3 Pros & Cons . . . 38

5 M^2TML: formalization 39

5.1 LP optimization problem . . . 39
5.2 QP optimization problem . . . 39
5.3 SVM approximation . . . 39

6 M^2TML: implementation 41

6.1 Projection in the pairwise space . . . 41
6.2 M-NN M-diff strategy . . . 41
6.3 Radius normalization . . . 42
6.4 Solving the SVM problem . . . 42
6.5 Definition of the dissimilarity measure . . . 42
6.6 Extension to regression problem . . . 42
6.7 Extension to multivariate problem . . . 42

7 M^2TML: Experiments 43

7.1 Dataset presentation . . . 43
7.2 Experimental protocol . . . 43
7.3 Results . . . 43
7.4 Discussion . . . 43

Conclusion 47

A Detailed presentation of the datasets 49

B Solver library 51

C SVM librairy **53**

Bibliographie **55**

List of Figures

1.1 The Beveridge wheat price index is the average in nearly 50 places in various countries measured in successive years from 1500 to 1869. [1] 8

1.2 Example of k-NN classification. The test sample (green circle) is classified either to the first class (blue squares) or to the second class (red triangles). If $k = 3$ (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If $k = 5$ (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle). 10

1.3 Example of linear classifiers in a 2-dimensional plot. For a set of points of classes +1 and -1 that are linearly separable, there exists an infinite number of separating hyperplans corresponding to $\mathbf{w}.\mathbf{x} + b = 0$. 12

1.4 The argument inside the decision function of a classifier is $\mathbf{w}.\mathbf{x} + b$. The separating hyperplane corresponding to $\mathbf{w}.\mathbf{x} + b = 0$ is shown as a line in this 2-dimensional plot. This hyperplane separates the two classes of data with points on one side labelled $y_i = +1$ ($\mathbf{w}.\mathbf{x} + b \geq 0$) and points on the other side labelled $y_i = -1$ ($\mathbf{w}.\mathbf{x} + b < 0$). Support vectors are circled in blue and lies on the hyperplans $\mathbf{w}.\mathbf{x} + b = +1$ and $\mathbf{w}.\mathbf{x} + b = -1$ 12

1.5 Obtained hyperplan after a dual resolution (full blue line). The 2 canonical hyperplans (dash blue line) contains the support vectors whose $\alpha_i > 0$. Other points have their $\alpha_i = 0$ and the equation of the hyperplan is only affected by the support vectors. 16

1.6 Left: in two dimensions these two classes of data are mixed together, and it is not possible to separate them by a line: the data is not linearly separable. Right: using a Gaussian kernel, these two classes of data (cross and circle) become separable by a hyperplane in feature space, which maps to the nonlinear boundary shown, back in input space. 17

1.7 Illustration of the Gaussian Kernel in the input space. 18

1.8 Example of several SVMs and how to interpret the weight vector $\mathbf{w}$ 20

1.9 21

1.10 Illustration of SVM regression (left), showing the regression curve with the ϵ-insensitive "tube" (right). Samples $\mathbf{x}_i$ above the ϵ-tube have $\xi_1 > 0$ and $\xi_1 = 0$, points below the ϵ-tube have $\xi_2 = 0$ and $\xi_2 > 0$, and points inside the ϵ-tube have $\xi = 0$. 22

1.11 General framework for building a supervised (classification/regression) model. . 24

1.12 An example of overfitting for a classification problem. The objective is to separate blue points from red points. Black line shows a separator f_1 with low complexity where as green line illustrates a model f_2 with high compelexity. On training examples (blue and red points), the model f_2 separates all the classes perfectly but may lead to poor generalization on new unseen examples. 24

1.13 Example of a 2 dimensional grid search for parameters C and γ. It defines a grid where each cell of the grid contains a combination (C, γ). Each combination is used to learn the model and is evaluated on the validation set. 25

1.14 v-fold Cross-validation for one combination of parameters. For each of v experiments, use $v-1$ folds for training and a different fold for Testing, then the training error for this combination of parameter is the mean of all testing errors. This procedure is illustrated for $v = 4$. 26

List of Tables

Table of Acronyms

LIG	*Laboratoire d'Informatique de Grenoble*
AMA	*Apprentissage, Méthode et Algorithme*
GIPSA-Lab	*Grenoble Images Parole Signal Automatique Laboratoire*
AGPiG	*Architecture, Géométrie, Perception, Images, Gestes*
A4S	*Analytic for Solutions*
k**-NN**	k*-nearest neighbors*
SVM	*Support Vector Machines*
SVR	*Support Vector Regression*
d_E	*Euclidean distance*
corr	*Pearson correlation*
cort	*Temporal correlation*
dtw	*Dynamic Time Warping*
IoT	*Internet of Things*

Introduction

Motivation

- Qu'est-ce qu'une série temporelle ? (réponse d'un système dynamique complexe (= pas de modèle du système)
- Motiver l'intérêt des séries temporelles dans les applications aujourd'hui: données de plus en plus présentes dans de nombreux domaines divers et variés
- Les séries temporelles sont impliquées dans des problèmes de classification, régression et clustering
- Pourquoi sont-elles challenging ? (délais, dynamique)
- On fait face à la fois, à un problème de small et big data

Problem statement (with words)

- Dans de nombreux algorithmes de classification ou de régression (kNN, SVM), la comparaison des individus (séries temporelles) reposent sur une notion de distance entre individus (séries temporelles).
- Contrairement aux données statiques, les données temporelles peuvent être comparés sur la base de plusieurs modalités (valeurs, forme, distance entre spectre, etc.) et à différentes échelles. La « métrique idéale », càd, celle qui permettra de résoudre au mieux le problème de classification/régression peut donc impliquer plusieurs modalités.
- Objectif de notre travail : Apprendre une métrique adéquate tenant compte de plusieurs modalités et de plusieurs échelles en vue d'une classification/régression kNN

PhD contributions

- Définition d'un nouvel espace de représentation: la représentation par paires
- Apprentissage d'une métrique multimodale et multi-échelle en vue d'une classification kNN à vaste marge de séries temporelles monovariées.
- Extension/Transposition du problème d'apprentissage de métrique (Metric Learning) dans l'espace des paires

- Comparaison de la méthode proposée avec des métriques classiques sur un vaste jeu de données (30 bases) de la littérature dans le cadre de la classification univariée de séries temporelles
- Extension du framework d'apprentissage de métrique au problème de régression de séries temporelles univariés
- Extension du framework d'apprentissage de métrique au problème de classification/régression de séries temporelles multivariés.
- Donner une solution interprétable.
- Donner un algorithme à la fois pour les small et big data.

Organisation du manuscrit

Présenter les différents chapitres

Note pour Ahlame, Michèle et Sylvain: Pour ajouter des commentaires dans le fichier .TEX, merci de les ajouter sous cette forme:

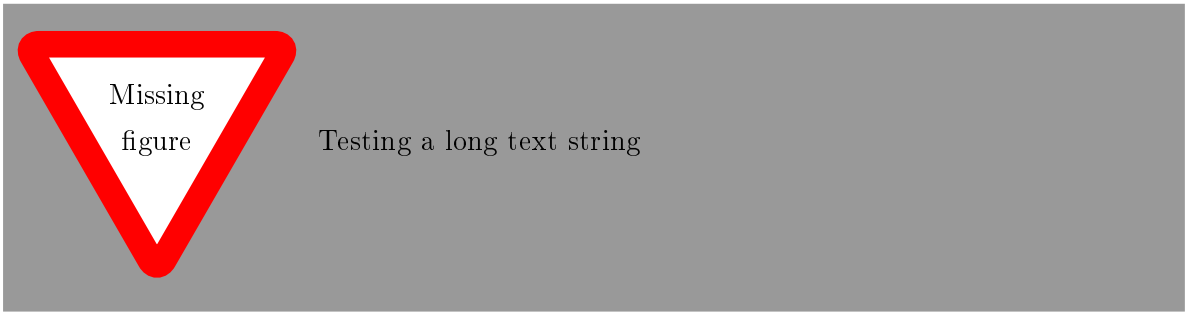
- dans le texte :

  Comment [CTD1]: Initial in [CAO] then your comment in the bracket

- dans la marge :

Comment [CTD2]: Initial in [CAO] then your comment in the bracket

If you think that they are missing figures, you can add them with a description with this command line :



Notations

$\mathbf{x}_i$	a time series
y_i	a label (discrete or continous)
$\mathbf{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$	a set of $n \in \mathbb{N}$ labeled time series
d_E	Euclidean distance
L_q	Minkovski q-norm
$\|\|\mathbf{x}\|\|_q$	q-norm of the vector $\mathbf{x}$
d_A	Value-based distance
$corr$	Pearson correlation
$cort$	Temporal correlation
d_F	Euclidean distance between the Fourier spectrum
D	Distance
$\mathbf{x}_{ij}$	a pair of time series $\mathbf{x}_i$ and $\mathbf{x}_j$
y_{ij}	the pairwise label of $\mathbf{x}_{ij}$
t	time stamp/index with $t = 1, ..., T$
T	length of the time series (supposed fixed)
f	frequential index
F	length of the Fourier transform
ξ	Relaxation term
p	number of metric measure considered in the metric learning process
r	order of the temporal correlation
k	number of nearest neighbors
$K(\mathbf{x}_i, \mathbf{x}_j)$	Kernel function between $\mathbf{x}_i$ and $\mathbf{x}_j$
$\phi(\mathbf{x}_i)$	embedding function from the original space to the Hilbert space
C	Hyper-parameter of the SVM (trade-off)
α	
λ	

Part I

Work positionning

The first part of the manuscript aims to position the work context. Our objective is the comparison and the classification or regression of time series. The first chapter considers time series as static vector data and presents classic machine learning algorithms used to classify them. We note that most of these methods relies on the comparison of objects (time series in our case) through a distance measure. In the second chapter, to cope with the characteristics of time series (amplitude, behavior, frequential spectrum, etc.), we recall some basic metrics used to compare time series. We show that time series may be compared by several modalities and at different granularities. We finally cast that learning an adequate distance based on several modalities and several granularities is a key challenge nowadays to well classify time series using classic machine learning algorithms.

Chapter 1

Machine Learning: state of the art

Sommaire

1.1 Definition of a time series 7
1.2 Machine learning algorithms 9
1.2.1 Classification, regression 9
1.2.2 k-Nearest Neighbors (k-NN) 9
1.2.3 Support Vector Machine (SVM) 11
1.2.4 Other classification algorithms 23
1.3 Learning protocol 23
1.3.1 Supervised learning framework 23
1.3.2 Model evaluation 25
1.3.3 Data pre-processing 26
1.4 Limits of classical technics 26

In this chapter, we first present what a time series is. Then, we vectorize time series, consider them as static data and apply classic machine learning algorithms used for classification and regression. We detail some of these learning algorithms. Finally, we review the protocol used to learn the best fitting of the hyper-parameters of these algorithms and how we evaluate and compare the different algorithms performances.

1.1 Definition of a time series

Time series and more generally temporal data are data objects that frequently occur in physical sciences (meterology, marine science, geophysics), marketing or process control [Cha04]. For physical systems, a time series of length T can be seen as a signal, sampled at a frequency f_e, in a window $[0; \frac{T}{f_e}]$. From a mathematical perspective, a time series is a collection of a finite number of realized observations made sequentially at discrete time instants $t = 1, ..., T$.

qqch ne va pas, ça ne peut pas être $\frac{T}{f_e}$. Les indices ne correspondent pas

Let $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{iT})$ be a univariate time series of length T. Each observation x_{it} is bounded (i.e., the infinity is not a valid value: $x_{it} \neq \pm\infty$). The time series $\mathbf{x}_i$ is said to be univariate if the collection of observations x_{it} comes from the observations of one variable (i.e., it has been measured by one sensor, the temperature for example). When the observations

are made at the same time from Q variables (several sensors such as the temperature, the pressure, etc.), the time series is said multivariate and is denoted $\mathbf{x}_i = (\mathbf{x}_{i,1}, ..., \mathbf{x}_{i,Q}) = (x_{i1,1}, ..., x_{iT,1}, x_{i1,2}, ..., x_{iT,2}, ..., x_{i1,Q}, ..., x_{iT,Q})$. For simplification purpose, we consider in the following univariate time series.

Time series can be found in various emerging applications such as sensor networks, smart buildings, social media networks or Internet of Things (IoT) [Naj+12]; [Ngu+12]; [YG08]. They are involved in many learning problems such as recognizing a human movement in a video, detect a particular operating mode, etc. . In clustering problems, one would like to organize similar time series together into homogenous groups. In classification problems, the aim is to assign time series to one of several predefined categories (different types of defaults in a machine). In regression problems, the objective is to predict a continuous value from observed time series (e.g. forecasting the measurement of a power meter from pressure or temperature sensors). Our work focus on classification and regression problems. However, due to their temporal and structured nature, time series constitute complex data to be analyzed by classic machine learning approaches.

biblio application time series + applis

To overcome this complexity, some authors propose to extract representative features from time series. Fig. 1.1 illustrates a model for time series proposed by Chatfield in [Cha04]. It states that a time series can be decomposed into 3 components: a trend, a cycle (periodic component) and a residual (irregular variations).

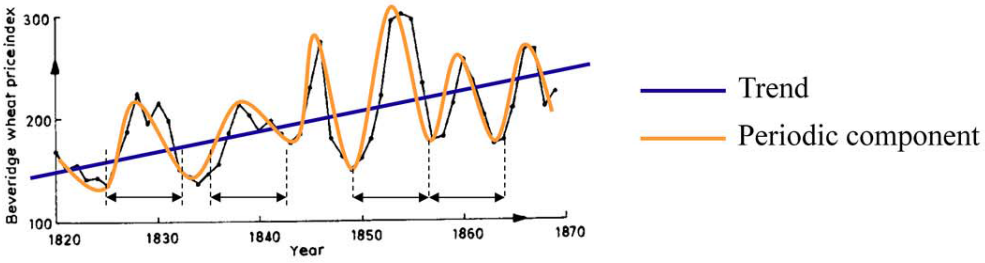


Figure 1.1: The Beveridge wheat price index is the average in nearly 50 places in various countries measured in successive years from 1500 to 1869. [1]

According to Chatfield, most time series exhibit a variation at a fixed period of time (seasonality) such as for example the seasonal variation of temperature. Beyond this cycle, there exists either or both a long term change in the mean (trend) that can be linear, quadratic, and a periodic (cyclic) component. By definition, a signal is periodic of period R if $x_{t+R} = x_t$ for all time instants t. In practice, this condition is not valid for real time series. In our work, we focus on the raw time series and do not try to extract global features from the time series.

Due to their complexity, other authors made the hypothesis of independency between the observations x_{it}. They consider time series as a static vector data (samples) and use classic machine learning algorithms .

[biblio time series static learning]

[1] This time series can be downloaded from `http://www.york.ac.uk/depts/maths/data/ts/ts04.dat`

1.2 Machine learning algorithms

We are going to detail now a few number of machine learning algorithms used classically to solve classification or regression problems. Other algorithms are popular nowadays such as Deep neural network, Decision tree or Relevance vector machine. We focus on k-Nearest Neighbors (k-NN) and Support Vector Machine (SVM) because these algorithms are based on the comparison of samples (time series in our case) through a distance measure, notion detailed in the next chapters.

1.2.1 Classification, regression

The idea of Machine learning (also refer as Pattern learning or Pattern recognition) is to imitate with algorithms executed on computers, the ability of living beings to learn from examples. For example, to teach a child how to read letters, we show him during the training phase labeled examples of letters ('A', 'B', 'C', etc.) written in different styles and fonts. We don't give him a complete and analytic description of the topology of the characters but labeled examples. After the training phase (testing phase), we want the child to be able to recognize and to label correctly the letters that have been seen during the training, and also to generalize to new instances [G. 06].

Let $\mathbf{X} = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$ be a training set of n samples $\mathbf{x}_i$ (time series in our case) and y_i their corresponding labels. The aim of Machine learning is to learn a relation (model) f between the samples $\mathbf{x}_i$ and their labels y_i based on examples. This relationship can include static relationships, correlations, dynamic relationship, etc. After the training phase based on labeled examples $(\mathbf{x}_i, y_i)$, the model f has to be able to generalize on the testing phase, i.e., to give a correct prediction y_j for new instances $\mathbf{x}_j$ that haven't been seen during the training.

When y_i are class labels (i.e., class 'A', 'B', 'C'), learning the model f is a classification problem; when y_i is a continuous value (i.e., the energy consumption in a building), learning f is a regression problem. Both problems corresponds to supervised learning as $\mathbf{x}_i$ and y_i are known during the training phase . For both problems, when a part of the labels y_i are known and an other part of y_i is unknown during training, learning f is a semi-supervised problem . Note that when the labels y_i are unknown, learning f refers to a clustering problem (unsupervised learning) . Our work focus on supervised learning (classification, regression).

[biblio supervisé]

[biblio semi-supervisé]

[biblio clustering]

1.2.2 k-Nearest Neighbors (k-NN)

A simple approach to classify samples is to consider that "close" samples have a great probability to belong to the same class. Given a test sample $\mathbf{x}_j$, one can decide that $\mathbf{x}_j$ belong to the same class of its nearest neighbor in the training set. More generally, we can consider the k nearest neighbors of $\mathbf{x}_j$. The class y_j of the test sample $\mathbf{x}_j$ is assigned with a voting scheme among them, i.e., using the majority of the class of nearest neighbors. This algorithm is refer

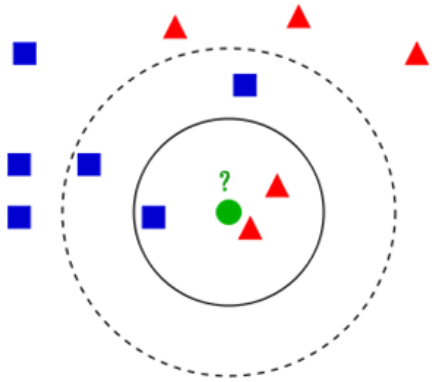


Figure 1.2: Example of k-NN classification. The test sample (green circle) is classified either to the first class (blue squares) or to the second class (red triangles). If $k = 3$ (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If $k = 5$ (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).

as the k-nearest neighbors algorithm (k-NN) [OE73]; [G. 06]. Fig. 1.2 illustrates the concept for a neighborhood of $k = 3$ and $k = 5$.

A key point in the k-NN algorithm is that the notion of "closeness" between samples is based on the computation of a metric [2]. Let D be a metric. Usally, for static data, standard metrics are the Euclidean distance, the Minkowski distance or the Mahalanobis distance[3]. Solving the 1-NN classification problem is equivalent to solve the optimization problem:
For a new sample $\mathbf{x}_j$, $\forall i \in \{1...n\}$,

$$y_j = y_{i^*} \tag{1.1}$$

Generalize for kNN

where $i^* = \underset{i \in \{1...n\}}{\operatorname{argmin}} D(\mathbf{x}_i, \mathbf{x}_j)$.

The k-NN algorithm can be extended to estimate continous labels (regression problems). The procedure is similar. The label y_j is defined as :

$$y_j = \sum_{i=1}^{k} y_i \tag{1.2}$$

[biblio regression kNN]

where i corresponds to the index of the k-nearest neighbors. There exists other variants of the k-NN algorithms: in a weighed k-NN, the approach consists in weighting each neighbors labels y_i by a factor equal to the inverse of the distance $\frac{1}{D(\mathbf{x}_i, \mathbf{x}_j)}$; in a fuzzy k-NN, the idea is to assign memberships of samples $\mathbf{x}_j$ to classes. The class membership is a function of the sample's distance $D(\mathbf{x}_i, \mathbf{x}_j)$ from its k-NN training samples..

[biblio kNN pondéré]

[biblio fuzzy kNN]

Despite its simplicity, the k-NN algorithm presents many advantages and have been shown

[2] A clarification of the terms metric, distance, dissimilarity, etc. will be given in Chapter 2. For now, we refer all of them as metrics.

[3] A recall of these metrics will be in Chapter 2.

to be successful on time series classification problems [Xi+06]; [Din+08]. One main advantage is that a 1-NN classifier can be used to evaluate and compare the efficacy of different metrics [Din+08]. First, the underlying metric is critical in the performance of the 1-NN classifier [TSK05]. Thus, the accuracy of the 1-NN classifier directly reflects the effectiveness of the metric. Second, 1-NN classifier is easy to implement and doesn't need to learn any hyper-parameters, which make it straightforward for anyone to reproduce results. All of this advantages allows one who want to evaluate a benchmark of metrics. Other methods exists such as clustering with small data sets which are not statistically significant, or compare the compactness of the metric . The 1-NN algorithm will be used in our experiments to compare the performances different metrics used for time series. bilbio

On the other hand, the k-NN algorithm presents some disavantages, mainly due to its computational complexity, both in space (storage of the training samples $\mathbf{x}_i$) and time (search) [OE73]. Suppose we have n labelled training samples in T dimensions, and find the closest neighbors to a test sample $\mathbf{x}_j$ ($k = 1$). In the most simple approach, we look at each stored samples $\mathbf{x}_i$ ($i = 1...n$) one by one, calculate its metric to $\mathbf{x}_i$ (D($\mathbf{x}_i$,$\mathbf{x}_j$)) and retain the index of the current closest one. For the standard Euclidean distance, each metric computation is $O(T)$ and thus the search is $O(Tn^2)$. Moreover, using standard metrics (such as the Euclidean distance) uses all the T dimensions in its computation and thus assumes that all dimensions have the same effect on the metric. This assumption may be wrong and can impact the classification performances. Wrong classification due to presence of many irrelevant dimensions is refered as the curse of dimensionality. The importance of defining adapted metrics for time series will be discussed in Chapter 2.

1.2.3 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a state-of-the-art classication method introduced in 1992 by Boser, Guyon, and Vapnik [BGV92]; [CV95]. The SVM classifier have demonstrate high accuracy, ability to deal with high-dimensional data, good generalization properties and interpretation for various applications from recognizing handwritten digits, to face identification, text categorisation, bioinformatics and database marketing [CY11]. SVMs belong to the category of kernel methods, algorithms that depends on the data only through dot-products [SS13].

As SVMs will be used in Chapter 5, we first present an intuition of maximum margin concept. We give the primal formulation of the SVM optimization problem. Then, by transforming the latter formulation into its dual form, the kernel trick can be applied to learn non-linear classifiers. Finally, we detail how we can interpret the obtained coefficients and how SVMs can be extended for regression problems.

Note that this section doesn't aim to give an detailed comprehension of SVMs. It aims to give an overview of the mathematical key points interpretation and comprehension of the method and an interpretation and comprehension. For more informations, the reader can consult [SS13]; [CY11]; [CV95].

1.2.3.1 Intuition

Michèle: chiffre ou lettre pour la numéro-tation?

Let consider a classification problem with 2 classes ($y_i = \pm 1$). The objective is to learn a hyperplan, whose equations are $\mathbf{w}.\mathbf{x} + b = 0$, that can separate samples of class $+1$ from the ones of class -1. When the problem is linearly separable such as in Fig. 1.3, there exists an infinite number of hyperplans.

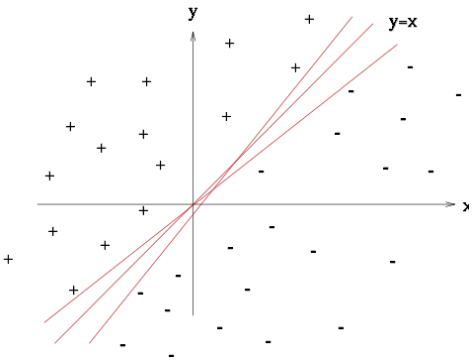


Figure 1.3: Example of linear classifiers in a 2-dimensional plot. For a set of points of classes $+1$ and -1 that are linearly separable, there exists an infinite number of separating hyperplans corresponding to $\mathbf{w}.\mathbf{x} + b = 0$.

Vapnik & al. [CV95] propose to choose the separating hyperplane that maximizes the margin, e.g. the hyperplane that leaves as much distance as possible between the hyperplane and the closest examples of each class, called the support vectors. This distance is equal to $\frac{1}{||\mathbf{w}||_2}$. The hyperplanes passing through the support vectors of each class are refered as the canonical hyperplanes, and the region between the canonical hyperplanes is called the margin band (Fig. 1.4).

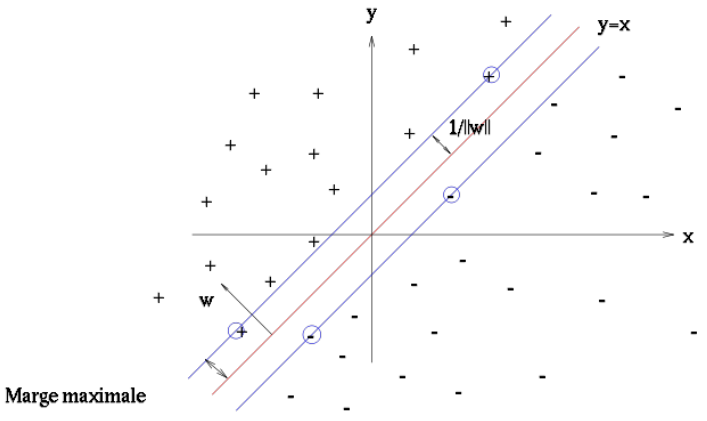


Figure 1.4: The argument inside the decision function of a classifier is $\mathbf{w}.\mathbf{x}+b$. The separating hyperplane corresponding to $\mathbf{w}.\mathbf{x} + b = 0$ is shown as a line in this 2-dimensional plot. This hyperplane separates the two classes of data with points on one side labelled $y_i = +1$ ($\mathbf{w}.\mathbf{x}+b \geq 0$) and points on the other side labelled $y_i = -1$ ($\mathbf{w}.\mathbf{x} + b < 0$). Support vectors are circled in blue and lies on the hyperplans $\mathbf{w}.\mathbf{x} + b = +1$ and $\mathbf{w}.\mathbf{x} + b = -1$

1.2.3.2 Primal formulation

Finding $\mathbf{w}$ and b by maximizing the margin $\frac{1}{||\mathbf{w}||_2}$ is equivalent to minimizing the norm of $\mathbf{w}$ such that all samples from the training set are correctly classified:

$$\underset{\mathbf{w},b}{\operatorname{argmin}} ||\mathbf{w}||_2^2 \tag{1.3}$$

$$\textbf{s.t. } y_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1 \tag{1.4}$$

This is a constrained optimization problem in which we minimize an objective function (Eq. 1.3) subject to constraints (Eq. 1.4). This formulation is refered as the primal hard margin problem. Many real life datasets are subjected to noise and SVM can lead to poor generalization if it tries to fit to this noise, represented by the constraints in Eq. 1.4. The effects of noise can be reduced by introducing slack variables $\xi_i \geq 0$ to relax the optimization problem:

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \left(\overbrace{\frac{1}{2}||\mathbf{w}||_2^2}^{Regularization} + C \overbrace{\sum_{i=1}^{n} \xi_i(\mathbf{w}; b; x_i; y_i)}^{Loss} \right) \tag{1.5}$$

$$\textbf{s.t. } y_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1 - \xi_i \tag{1.6}$$

$$\xi_i \geq 0 \tag{1.7}$$

where $C > 0$ is a penalty hyper-parameter.

This formulation is refered as the primal soft margin problem. It is quadratic programming optimization problem subjected to constraints. Thus, it is a convex problem: any local solutions is a global solution. The objective function in Eq. 1.5 is made of two terms. The first one, the regularization term, penalize the complexity of the model and thus, controls the ability of the algorithm to generalize on new samples. The second one, the loss term, is an adaptation term to the data. The hyper-parameter C is a trade-off between the regularization and the loss term. When C tends to $+\infty$, the problem is equivalent to the primal hard margin problem. The hyper-parameter C is learnt during the training phase.

For SVM, the two common loss functions ξ_i are $\max(1\text{-}y_i\mathbf{w}.\mathbf{x}_i, 0)$ and $\max(1\text{-}y_i\mathbf{w}.\mathbf{x}_i, 0)^2$. The former is referred to as L1-Loss and the latter is L2-Loss function. L2-loss function will penalize more slack variables ξ_i during training. Theorically, it should lead to less error in training and poorer generalization in most of the case.

An other thing to specify is the type of regularizer. For SVM, the two common regularizer are $||\mathbf{w}||_2$ and $||\mathbf{w}||_2^2$. The former is referred to as L1-Regularizer while the latter is L2-Regularizer. L1-Regularizer is used to obtain sparser models than L2-Regularizer. Thus, it can be used for variable selection. L2-Regularizer allows to transform the primal formulation into a dual form.

From this, for a binary classification problem, to classify a new sample $\mathbf{x}_j$, the decision function

is:

$$f(\mathbf{x}_j) = sign(\mathbf{w}.\mathbf{x}_j + b) \tag{1.8}$$

1.2.3.3 Dual formulation

From the primal formulation, using a L2-Regularizer, it is possible to have an equivalent dual form. This latter formulation allows samples $\mathbf{x}_i$ to appear in the optimization problem through dot-products only. Thanks to that, a Kernel trick can be applied to extend the methods to learn non-linear classifiers.

First, to simplify the calculation development, let consider the hard margin formulation in Eq. 1.5, 1.6 and 1.7 with a L1-Loss function. As a constrained optimization problem, the formulation is equivalent to the minimization of a Lagrange function, consisting of the sum of the objective function and the n constraints multiplied by their respective Lagrange multipliers:

$$\underset{\alpha}{\operatorname{argmax}} \left(L(\mathbf{w}, b) = \frac{1}{2}(\mathbf{w}.\mathbf{w}) - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}.\mathbf{x}_i + b) - 1) \right) \tag{1.9}$$

$$\textbf{s.t. KKT } \forall i = 1...n : \tag{1.10}$$

$$\alpha_i \geq 0 \tag{1.11}$$

$$y_i(\mathbf{w}.\mathbf{x}_i + b) - 1 \geq 0 \tag{1.12}$$

$$\alpha_i (y_i(\mathbf{w}.\mathbf{x}_i + b) - 1) = 0 \tag{1.13}$$

where $\alpha_i \geq 0$ are the Lagrange multipliers. Eq. 1.11, 1.12 and 1.13 are called in optimization theory the Karush-Kuhn-Tucker (KKT) conditions. It corresponds to the set of conditions which must be satisfied at the optimum of a constrained optimization problem. The KKT conditions will play an important role in the interpretation of SVM in Section 1.2.3.5.

At the minimum, we can take the derivatives with respect to b and $\mathbf{w}$ and set them to zero:

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0$$

that leads to:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{1.14}$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{1.15}$$

By substituing $\mathbf{w}$ into $L(\mathbf{w}, b)$ in Eq. 1.9, we obtain the dual formulation (*Wolfe dual*):

$$\underset{\alpha}{\text{argmax}} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i . \mathbf{x}_j) \right) \tag{1.16}$$

$$\textbf{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{1.17}$$

$$\alpha_i \geq 0 \tag{1.18}$$

where α is the vector of corresponding α_i: $\alpha = [\alpha_1, ..., \alpha_n]'$. The dual objective in 1.16 is quadratic in the parameters α_i. Adding the constraints in Eq. 1.17 and 1.18, it is a constrained quadratic programming optimization problem (QP). Note that while the primal formulation is minimization, the equivalent dual formulation is maximization. It can be shown that the objective functions of both formulations reach the same value when the solution is found [CY11]. In the same spirit, considering the soft margin primal problem, it can be shown that it leads to the following formulation [CY11]:

$$\underset{\alpha}{\text{argmax}} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i . \mathbf{x}_j) \right) \tag{1.19}$$

$$\textbf{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{1.20}$$

$$0 \leq \alpha_i \leq C \tag{1.21}$$

The only difference between the hard margin and soft margin formulation in its dual form is that the Lagrange multipliers α_i are upper bounded by the trade-off C in the soft margin formulation. The constraints in Eq. 1.21 are called the Box constraints [CY11]. From the optimal value of α_i, denoted α_i^*, it is possible to compute the weight vector $\mathbf{w}^*$ and the bias b^* at the optimality:

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \tag{1.22}$$

$$b^* = \sum_{i=1}^{n} (\mathbf{w} . \mathbf{x}_i - y_i) \tag{1.23}$$

At the optimality point, only a few number of datapoints have $\alpha_i^* > 0$ as shown as in Fig. 1.5. These samples are the vector supports. All other datapoints have $\alpha_i^* = 0$, and the decision function is independent of them. Thus, the representation is sparse.

From this, to classify a new sample $\mathbf{x}_j$, the decision function for a binary classification problem is:

$$f(\mathbf{x}_j) = sign(\sum_{i=1}^{n} \alpha_i^* y_i (\mathbf{x}_i . \mathbf{x}_j) + b^*) \tag{1.24}$$

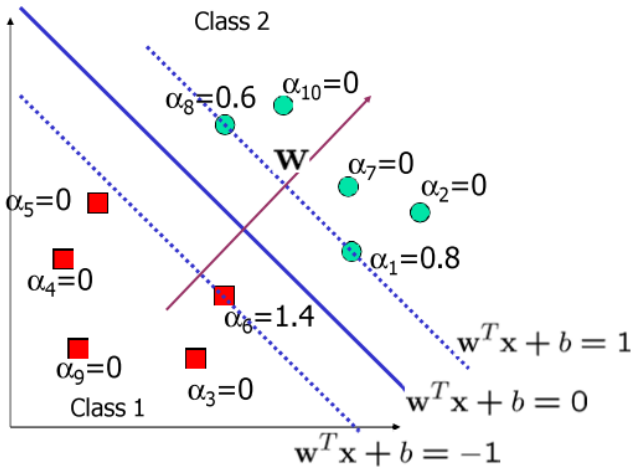


Figure 1.5: Obtained hyperplan after a dual resolution (full blue line). The 2 canonical hyperplans (dash blue line) contains the support vectors whose $\alpha_i > 0$. Other points have their $\alpha_i = 0$ and the equation of the hyperplan is only affected by the support vectors.

1.2.3.4 Kernel trick

The concept of Kernels was introduced by Aizerman & Al in 1964 to design potential functions in the context of pattern recognition [ABR64]. The idea was re-introduced in 1992 by Boser & al. for Support Vector Machine (SVM) and has been received a great number of improvments and extensions to symbolic objects such as text or graphs [BGV92].

One theorical interesting property of SVM if that it has been shown that the generalization error bound does not depend on the dimensionality T of the space [SS13]. From the dual objective in Eq. 1.16, we note that the samples $\mathbf{x}_i$ are only involves in a dot-product. Therefore, we can map these samples $\mathbf{x}_i$ into a higher dimensional hyperspace, called the feature space, through the replacement:

$$(\mathbf{x}_i.\mathbf{x}_j) \rightarrow \Phi(\mathbf{x}_i).\Phi(\mathbf{x}_j) \tag{1.25}$$

where Φ is the mapping function. The intuition behind is that in many datasets, it is not possible to find a hyperplan that can separate the two classes in the input space. The problem is not linearly separable. However, by applying a transformation Φ, data might become linearly separable in a higher dimensional space (feature space). Fig. 1.6 illustrates the idea: in the original 2-dimensional space (left), the two classes can't be separated by a line. However, with a third dimension such that the $+1$ labelled points are moved forward and the -1 labelled moved back the two classes become separable.

In most of the case, the mapping function Φ does not need to be known since it will be defined by the choice of a Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i).\Phi(\mathbf{x}_j)$. We call Gram matrix G, the

matrix containing all $K(\mathbf{x}_i, \mathbf{x}_j)$:

$$G = (K(\mathbf{x}_i, \mathbf{x}_j))_{1 \leq i,j \leq n} = \begin{pmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & ... & K(\mathbf{x}_1, \mathbf{x}_n) \\ ... & & ... \\ K(\mathbf{x}_n, \mathbf{x}_1) & ... & K(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

Defining a kernel has to follow rules. One of them is that it has to define a proper inner product in the feature space. Mathematically, the Gram matrix has to be semi-definite positive (Mercer's theorem) [SS13]. These restricted feature spaces, containing an inner product are called Hilbert space.

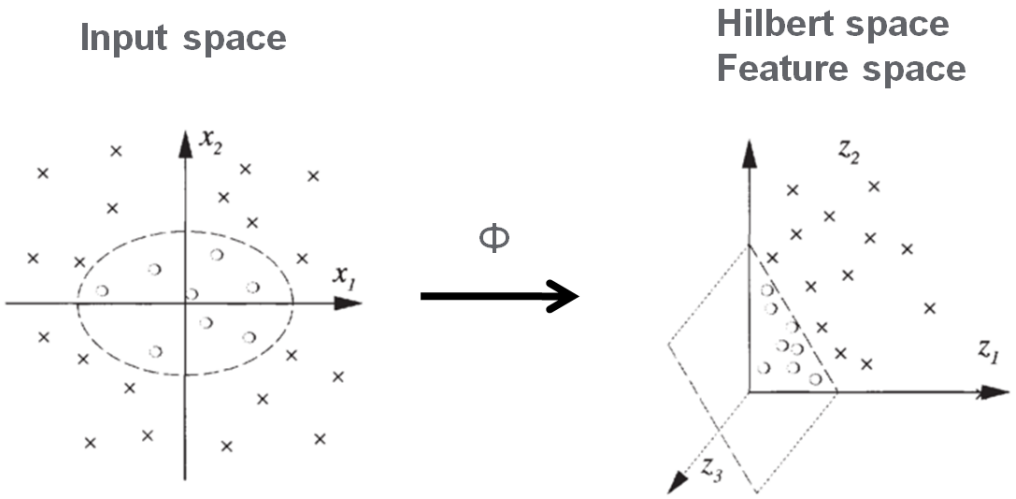


Figure 1.6: Left: in two dimensions these two classes of data are mixed together, and it is not possible to separate them by a line: the data is not linearly separable. Right: using a Gaussian kernel, these two classes of data (cross and circle) become separable by a hyperplane in feature space, which maps to the nonlinear boundary shown, back in input space.

Many kernels have been proposed in the litterature such as the polynomial, sigmoid, exponential or wavelett Kernels [SS13]. The most popular ones that we will use in our work are respectively the Linear and the Gaussian (or Radial Basis Function (RBF)) Kernels:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i.\mathbf{x}_j \tag{1.26}$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{||\mathbf{x}_j - \mathbf{x}_i||_2^2}{2\sigma^2}) = \exp(-\gamma.||\mathbf{x}_j - \mathbf{x}_i||_2^2) \tag{1.27}$$

where $\gamma = \frac{1}{2\sigma^2}$ is the parameter of the Gaussian Kernel and $||\mathbf{x}_j - \mathbf{x}_i||_2$ is the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. Note that the Linear kernel is an identity transformation. In practice, for large scale problem (when T is high), using a Linear Kernel is sufficient **Fan2008**

The Gaussian Kernel computed between a sample $\mathbf{x}_j$ with a support vector $\mathbf{x}_i$ is an exponentially decaying function in the input feature space. The maximum value of the Kernel

($K(\mathbf{x}_i, \mathbf{x}_j)$=1) is attained at the support vector (when $\mathbf{x}_i = \mathbf{x}_j$). Then, the value of the Kernel decreases uniformly in all directions around the support vector, with distance and ranges between zero and one. It can thus be interpreted as a similarity measure. Geometrically speaking, it leads to hyper-spherical contours of the kernel function as shown in Fig. 1.7 [4]. The parameter γ controls the decreasing speed of the sphere. . In pratice, this parameter is learnt during the training phase.

reformuler pour le gamma

refaire la figure

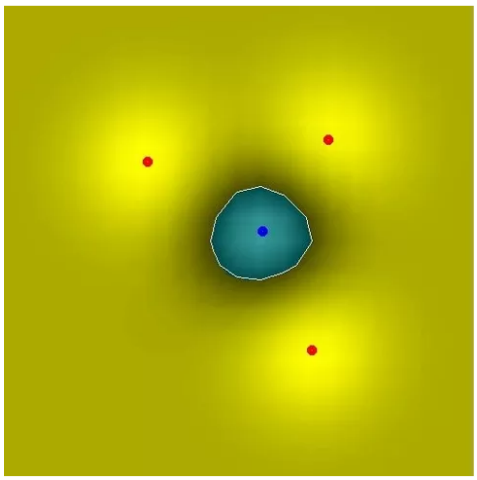


Figure 1.7: Illustration of the Gaussian Kernel in the input space.

By applying the Kernel trick to the soft margin formulation in Eq. 1.19, 1.20 and 1.21, the following optimization problem allows to learn non-linear classifiers:

$$\underset{\alpha}{\text{argmax}} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i.\mathbf{x}_j) \right) \tag{1.28}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{1.29}$$

$$0 \leq \alpha_i \leq C \tag{1.30}$$

and the decision function f becomes:

$$f(\mathbf{x}_j) = sign(\sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}_i.\mathbf{x}_j) + b^*) \tag{1.31}$$

Note that in this case, we can't recover the weight vector $\mathbf{w}^*$. Let n_{SV} be the number of

[4] https://www.quora.com/Support-Vector-Machines/What-is-the-intuition-behind-Gaussian-kernel-in-SVM

support vectors ($n_{SV} \leq n$). To recover b^*, we recall that for support vectors $\mathbf{x}_i$:

$$y_j \left(\sum_{i=1}^{n_{SV}} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}_j) + b^* \right) = 1 \tag{1.32}$$

From this, we can solve b^* using an arbitrarily chosen support vector $\mathbf{x}_i$:

$$b^* = \frac{1}{y_j} - \sum_{i=1}^{n_{SV}} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}_j) \tag{1.33}$$

1.2.3.5 Complexity and interpretation

Complexity

réécrire

As the objective is to provide an algorithm for both small and large datasets, let us examine the complexity of SVMs and in the computation of the Gram matrix G **Bottou2007**

In the dual, suppose that we know which samples are not support vectors ($\alpha_i = 0$) and which sample are bounded support vectors ($\alpha_i = C$). The R remaining support vectors are determined by a system of R linear equations. They represent the derivatives of the objective function and requires a number of operations proportional to R^3. Verifying that a vector α is a solution of the SVM problem involves computing the gradient of the dual and checking the optimality conditions. With n samples and n_{SV} support vectors, the number of operations required is equal to $n.n_{SV}$. When C gets large, few support vectors reach the upper bound, the cost is then $R^3 \approx n_{SV}^3$. The term $n.n_{SV}$ is usually larger. The final number of support vectors n_{SV} therefore is the critical component of the computational cost of solving the dual problem. Since the asymptotical number of support vectors n_{SV} grows linearly with the number of samples n, the computational cost of solving the SVM problem has both a quadratic and a cubic component. It grows at least like n^2 when C is small and n^3 when C gets large.

Computing the n^2 components of the kernel matrix $G = \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i=1}^n$ is a quadratic matter. Note that it can lead to technical issue in pratice such as the kernel matrix does not fit in memory.

Interpretation in the primal

In the primal, the weight vector $\mathbf{w} = [w_1, \ldots, w_T]'$ contains as many elements as there are variables in the dataset, i.e., $\mathbf{w} \in \mathbb{R}^T$. The magnitude of each element in $\mathbf{w}$ denotes the importance of the corresponding variable for the classification problem. If the element of $\mathbf{w}$ for some variable is 0, these variables are not important for the classification problem.

In order to visualize the above interpretation of the weight vector $\mathbf{w}$, let us examine several hyperplanes $\mathbf{w}.\mathbf{x} + b = 0$ shown in Fig. 1.8. Figure (a) shows a hyperplane where elements

of $\mathbf{w}$ are the same for both variables $\mathbf{x}_1$ and $\mathbf{x}_2$. The interpretation is that both variables contribute equally for classification of objects into positive and negative. Figure (b) shows a hyperplane where the element of w for $\mathbf{x}_1$ is 1, while that for $\mathbf{x}_2$ is 0. This is interpreted as that $\mathbf{x}_1$ is important but $\mathbf{x}_2$ is not. An opposite example is shown in figure (c) where $\mathbf{x}_2$ is considered to be important but $\mathbf{x}_1$ is not. Finally, figure (d) provides a 3-dimensional example where an element of w for $\mathbf{x}_3$ is 0 and all other elements are equal to 1. The interpretation is that $\mathbf{x}_1$ and $\mathbf{x}_2$ are important but $\mathbf{x}_3$ is not.

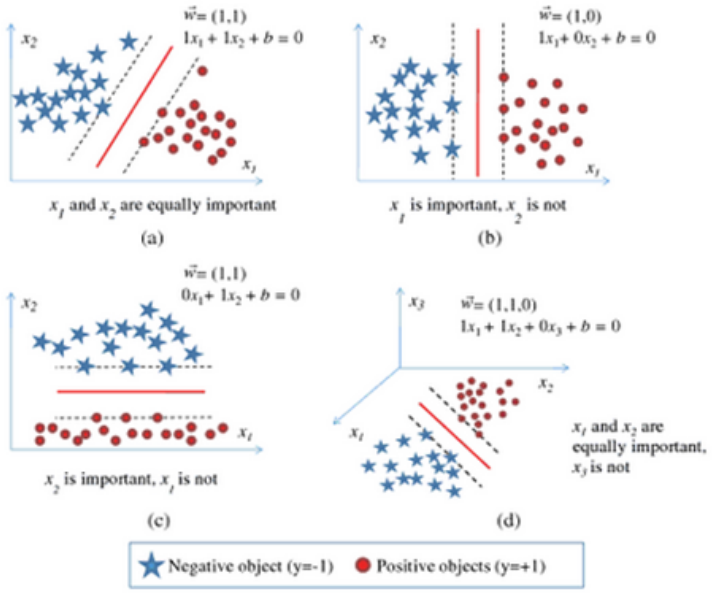


Figure 1.8: Example of several SVMs and how to interpret the weight vector $\mathbf{w}$

Another way to interpret how much a variable contributes in the vector $\mathbf{w}$ is to express the contribution in percentage. To do that, if the variables $\mathbf{X}_j$ are normalized before learning the SVM model, they evolves in the same range. Thus, the ratio $\frac{||w_j||_2}{||\mathbf{w}||_2} * 100$ defines the percentage of contribution for each variable $\mathbf{X}_j$ in the SVM model.

Geometrically, the vector $\mathbf{w}$ represents the direction of the hyperplan (Fig. 1.9). The bias b is equal to the distance of the hyperplan to the origin point $\mathbf{x} = \mathbf{0}$[5]. The orthogonale projection of a sample $\mathbf{x}_i$ on the direction $\mathbf{w}$ is $P_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w}.\mathbf{x}_i$. In the soft margin problem, the slack variables ξ_i of the samples $\mathbf{x}_i$ that lies within the two canonical hyperplan are equal to zero. Outside of these canonical hyperplans, the slack variables $\xi_i > 0$ are equal to the distance to the hyperplan.

Interpretation in the dual

As a constrained optimization, the dual form satisfies the Karush-Kuhn-Tucker (KKT) conditions (Eq. 1.11, 1.12 and 1.13). We recall Eq. 1.13)

$$\alpha_i(y_i(\mathbf{w}.\mathbf{x}_i + b) - 1) = 0$$

[5] $\mathbf{0}$ stands for the null vector: $\mathbf{0} = [0, \ldots, 0]^T$

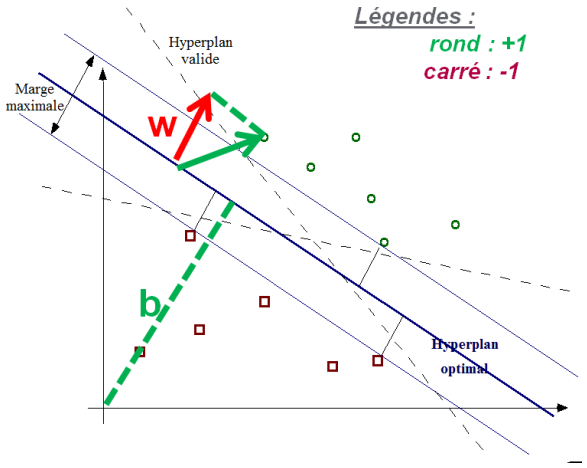


Figure 1.9

From this, for every datapoint, either $\alpha_i^* = 0$ or $y_i(\mathbf{w}.\mathbf{x}_i + b) = 1$. Any datapoint with $\alpha_i^* = 0$ do not appear in the sum of the decision function f in Eq. 1.24 or 1.31. Hence, they play no role in the prediction of new samples $\mathbf{x}_j$. The others ($\alpha_i^* > 0$) corresponds to the support vector. Looking at the distribution of α_i^* allows also to have either a better understanding of the datasets, or either to detect outliers. The higher is the coefficient α_i^* for a sample $\mathbf{x}_i$, the more the sample $\mathbf{x}_i$ impacts on the decision function f. However, unusual high value of α_i^* among the samples can lead to two interpretations: either this point is a critical point to the decision, either this point is an outlier. By constraining α_i^* to be inferior to C (Box constraints) in the soft margin problem, the effect of outliers can be reduced and controlled.

1.2.3.6 Extensions of SVM

SVM has received many interests in recent years. Many extensions has been developped such as ν-SVM, asymetric soft margin SVM or multiclass SVM . One interesting extension is Biblio the extension of Support Vector Machine to regression problems, also called Support Vector Regression (SVR). The objective is to find a linear regression model $f(\mathbf{x}) = \mathbf{w}.\mathbf{x} + b$. To preserve the property of sparsiness, the idea is to consider an ϵ-insensitive error function. It gives zero error if the absolute difference between the prediction $f(\mathbf{x}_i)$ and the target y_i is less than ϵ where $\epsilon > 0$ penalize samples that are outside of a ϵ-tube as shown as in Fig. 1.10.

An example of ϵ-insensitive error function E_ϵ is given by,

$$E_\epsilon(f(\mathbf{x}_i) - y_i) = \begin{cases} 0 & \text{if} \quad |f(\mathbf{x}_i) - y_i| < \epsilon \\ |f(\mathbf{x}_i) - y_i| - \epsilon & \text{otherwise} \end{cases} \tag{1.34}$$

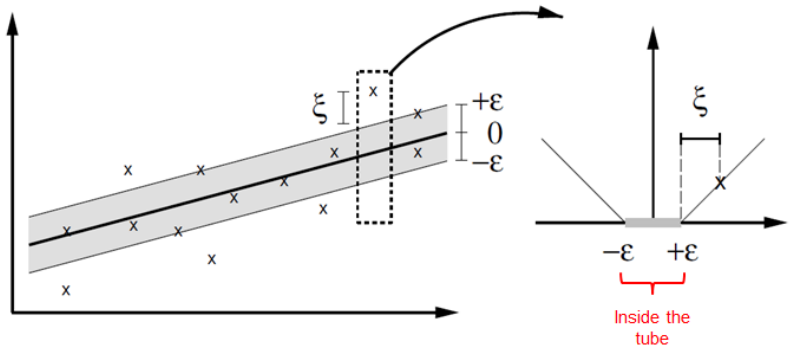


Figure 1.10: Illustration of SVM regression (left), showing the regression curve with the ϵ-insensitive "tube" (right). Samples $\mathbf{x}_i$ above the ϵ-tube have $\xi_1 > 0$ and $\xi_1 = 0$, points below the ϵ-tube have $\xi_2 = 0$ and $\xi_2 > 0$, and points inside the ϵ-tube have $\xi = 0$.

The soft margin optimization problem in its primal form is formalized as:

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \left(\overbrace{\frac{1}{2}||\mathbf{w}||_2^2}^{Regularization} + C \overbrace{\sum_{i=1}^{n}(\xi_{i_1} + \xi_{i_2})}^{Loss} \right) \tag{1.35}$$

$$\textbf{s.t. } \forall i = 1 \dots n : \tag{1.36}$$

$$y_i - (\mathbf{w}.\mathbf{x}_i + b) \geq \epsilon - \xi_{i_1} \tag{1.37}$$

$$(\mathbf{w}.\mathbf{x}_i + b) - y_i \geq \epsilon - \xi_{i_2} \tag{1.38}$$

$$\xi_{i_1} \geq 0 \tag{1.39}$$

$$\xi_{i_2} \geq 0 \tag{1.40}$$

$$\tag{1.41}$$

The slacks variables are divided into 2 slacks variables, one for samples above the decision function f (ξ_{i_1}), and one for samples under the decision function f (ξ_{i_2}). As for SVM, it is possible to have a dual formulation:

$$\underset{\alpha}{\operatorname{argmax}} \left(\sum_{i=1}^{n} y_i(\alpha_{i_1} - \alpha_{i_2}) - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_{i_1} - \alpha_{i_2})(\alpha_{j_1} - \alpha_{j_2})(\mathbf{x}_i.\mathbf{x}_j) \right) \tag{1.42}$$

$$\textbf{s.t. } \forall i = 1 \dots n : \tag{1.43}$$

$$\sum_{i=1}^{n} \alpha_{i_1} = \sum_{i=1}^{n} \alpha_{i_2} \tag{1.44}$$

$$0 \leq \alpha_{i_1} \leq C \tag{1.45}$$

$$0 \leq \alpha_{i_2} \leq C \tag{1.46}$$

As in SVM, we obtain three possible decision functions for a new sample $\mathbf{x}_j$, respectively in its primal, dual, and non-linear form:

$$f(\mathbf{x}_j) = \mathbf{w}.\mathbf{x}_j + b \tag{1.47}$$

$$f(\mathbf{x}_j) = \sum_{i=1}^{n}(\alpha_{i_1}^* - \alpha_{i_2}^*)(\mathbf{x}_i.\mathbf{x}_j) + b \tag{1.48}$$

$$f(\mathbf{x}_j) = \sum_{i=1}^{n}(\alpha_{i_1}^* - \alpha_{i_2}^*)K(\mathbf{x}_i.\mathbf{x}_j) + b \tag{1.49}$$

More informations about the calculation develpment can be found in **Bishop2006**

1.2.4 Other classification algorithms

Partie non encore rédigée. A faire à la fin.

- S'intéresser au Deep neural network (à la mode en ce moment)
- RVM, Decision Tree,
- Ne pas trop développer
- Dans notre cas, on ne s'intéressera pas à ce type d'algorithmes (type deep learning) car il ne repose pas sur une notion de distance et les features qui sont trouvés ne sont pas interprétables

1.3 Learning protocol

1.3.1 Supervised learning framework

A key objective of learning algorithms is to build models f with good generalization capability, i.e., models f that correctly predict the class labels y_j of new unknown samples $\mathbf{x}_j$. Fig. 1.11 shows a general approach for solving machine learning problems. First, a training set $X = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ consisting of m samples $\mathbf{x}_i$ whose labels y_i are known is provided. The training set is used to build the supervised model f. Then, the model is applied to the test set $X_T = \{(\mathbf{x}_j, y_j)\}_{j=1}^{m}$, which consists of samples $\mathbf{x}_j$ with unknown labels y_j. The test is used to evaluate the performance of the learnt model.

The errors committed by a classification or regression model are divided into two types: training errors and generalization errors (testing error). Training error is the number of misclassification errors in classification (Root Mean Square Error or other error measures in regression) committed on training samples $\mathbf{x}_i$, whereas generalization error is the expected

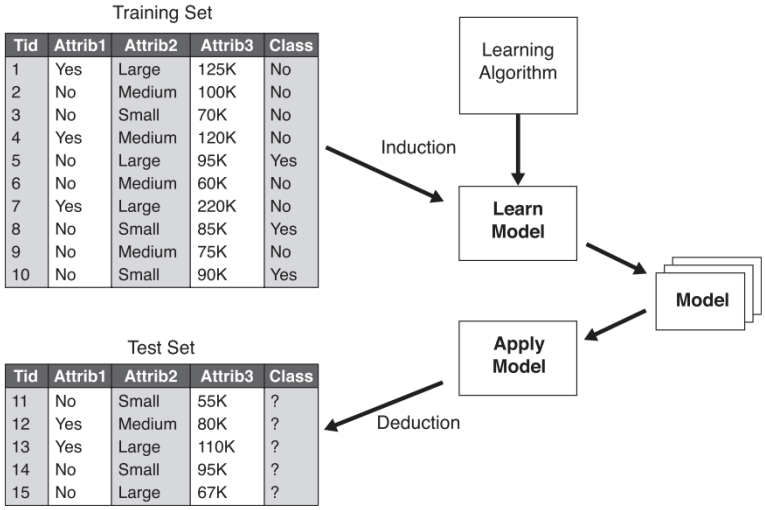


Figure 1.11: General framework for building a supervised (classification/regression) model.

error of the model f on unseen samples $\mathbf{x}_j$. A good supervised model f must not only fit the training data X well, it must also accurately classify records it has never seen before X_T. In other words, a good model f must have low training error as well as low generalization error. This is important because a model that fits the training data too well can have a poorer generalization error than a model with a higher training error. Such a situation is known as model overfitting (Fig. 1.12).

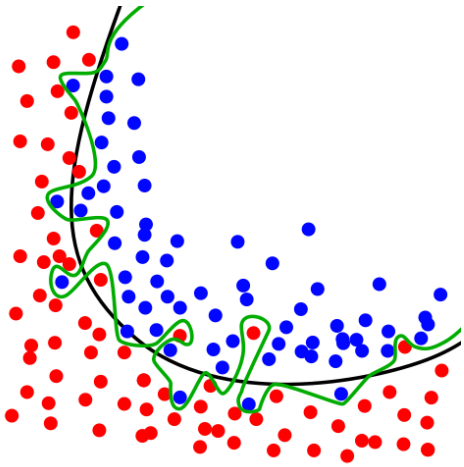


Figure 1.12: An example of overfitting for a classification problem. The objective is to separate blue points from red points. Black line shows a separator f_1 with low complexity where as green line illustrates a model f_2 with high compelexity. On training examples (blue and red points), the model f_2 separates all the classes perfectly but may lead to poor generalization on new unseen examples.

In most cases, learning algorithms requires to tune some hyper-parameters. For that, the training set can be divided into 2 sets: a learning and a validation set. Suppose we have

two hyper-parameters to tune: C and γ. We make a grid search for each combination of the hyper-parameters, that is in this case a 2-dimensional grid (Fig. 1.13). For each combination (a cell of the grid), the model is learnt on the learning set and evaluated on the validation set. At the end, the model with the lowest error on the validation set is retained. This process is refered as the model selection.

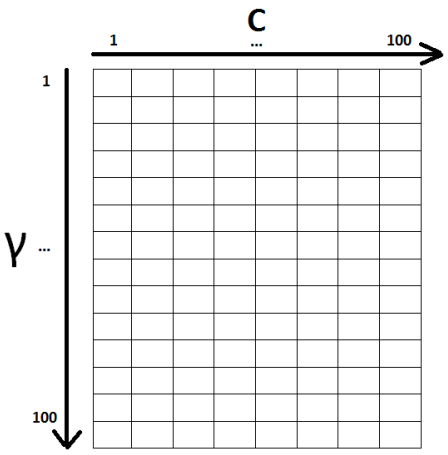


Figure 1.13: Example of a 2 dimensional grid search for parameters C and γ. It defines a grid where each cell of the grid contains a combination (C, γ). Each combination is used to learn the model and is evaluated on the validation set.

An alternative is cross-validation with v folds, illustrated in Fig. 1.14. In this approach, we partition the training data into v equal-sized subsets. The objective is to evaluate the error for each combination of hyper-parameters. For each run, one fold is chosen for validation, while the $v-1$ rest folds are used as the learning set. We repeat the process for each fold, thus v times. Each fold gives one validation error and thus we obtain v errors. The total error for the current combination of hyper-parameters is obtained by summing up the errors for all v folds. When $v = n$, the size of training set, this approach is called leave-one-out. Each test set contains only one sample. The advantage is much data are used as possible for training. Moreover, the test sets are exclusive and they cover the entire data set. The drawback is that it is computationally expensive to repeat the procedure n times. Furthermore, since each test set contains only one record, the variance of the estimated performance metric is usally high. This procedure is often used when n, the size training set, is small.

There exists other methods such as subsampling or bootstraps [OE73]; [G. 06]. We use cross-validation in our experiments.

1.3.2 Model evaluation

- Classification Error et test de significativité
- Regression Error

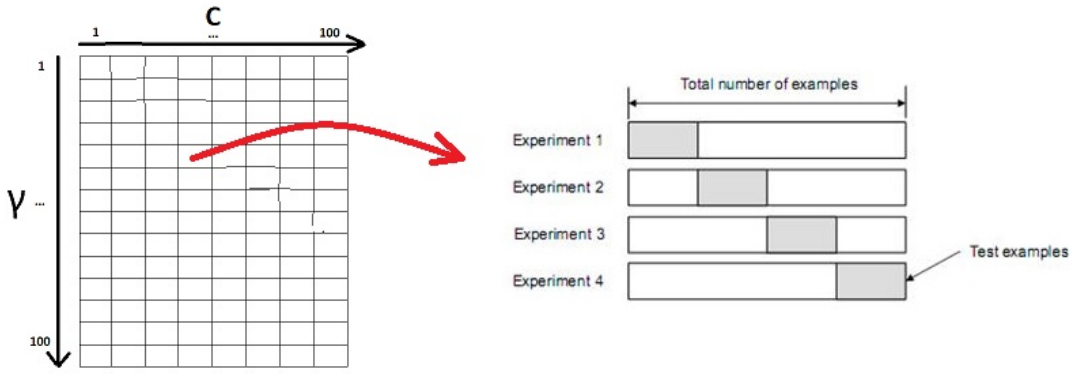


Figure 1.14: v-fold Cross-validation for one combination of parameters. For each of v experiments, use $v-1$ folds for training and a different fold for Testing, then the training error for this combination of parameter is the mean of all testing errors. This procedure is illustrated for $v = 4$.

1.3.2.1 Classification

1.3.2.2 Regression

1.3.3 Data pre-processing

- Normalization
- PCA

1.4 Limits of classical technics

- The concept of order of temporal data in not take into account (dynamic, frequence, etc.)
- Introduction to next chapter: all of these algorithms (kNN, SVM, etc.) are based on a notion of distance or similarity between objects to compare/classify. Let consider now the object time series and let recall the concept of distance between time series

Chapter 2

Time series basic metrics

Sommaire

2.1 Properties of a distance measure 28
2.2 Basic metrics for time series 28
2.2.1 Value-based metrics 28
2.2.2 Behavior-based metrics 28
2.2.3 Frequential-based metrics 29
2.2.4 Other metrics 29
2.3 Kernels for time series 29
2.4 Time series alignment 29
2.5 Multi-scale aspect 30

Chapeau introductif

- Rappel : notion de similaire : dans le cadre de la classification, on a un comportement « similaire » pour une même classe. La notion de « similaire » est lié à une notion de distance ou (dis)similarité.
- Donner les hypothèses de travail :
  - Considérons la série temporelle comme étant un objet ordonné.
  - les séries temporelles sont de même taille
  - les séries temporelles ont la même période d'échantillonnage
  - les séries temporelles peuvent être comparés sur l'ensemble des valeurs, sur une partie des valeurs, sur un ensemble de fenêtre (fréquences, etc.)
- On va définir dans la suite la notion de métrique, d'alignement, de localité pour des séries temporelles.
- Mettre un graphique (dit GRAPHIQUE GENERAL) qui prend 5 séries temporelles que l'on va utiliser pour la suite : proche en valeur, proche en forme, proche en fréquence, proche en valeur avec un délai, proche en forme avec un délai

2.1 Properties of a distance measure

- Rappeler les propriétés d'une mesure de distance (positivité, symétrique, distinguabilité, inégalité triangulaire)
- Donner les différences entre métriques, distance, dissimilarités, similarités, pseudo-métrique, etc.
- Dans la suite du travail, on va tout assimiler au mot métrique pour une meilleure simplicité

2.2 Basic metrics for time series

2.2.1 Value-based metrics

- Distance classiquement utilisée dans la littérature
- Distance de Minkowski (norm Lp)
- Distance de Mahalanobis (norm pondéré)
- D_E qui est une forme particulière de Minkowski
- Prendre le GRAPHIQUE GENERAL et faire le calcul des distances entre les courbes et montrer que pour 2 courbes qui ont des "amplitudes proches", on obtient une valeur de distance faible.

2.2.2 Behavior-based metrics

- Intuition : expliquer ce que signifie "2 séries temporelles sont proches en forme".
- Dans la littérature classique, on trouve la corrélation de Pearson
- Récemment, Douzal & al. propose une généralisation: cort
- Transformer la cort en mesure de dissimilarité
- Prendre le GRAPHIQUE GENERAL et faire le calcul des distances entre les courbes et montrer que pour 2 courbes qui ont des "formes proches", on obtient une valeur de distance faible.

2.2.3 Frequential-based metrics

- Dans le cadre du traitement de signal, les gens utilisent des représentations fréquentielles (Fourier, etc.)
- Rappeler la transformée de Fourier (TF) + spectre (module de la TF)
- On peut définir une distance dans la représentation de Fourier.
- Prendre le GRAPHIQUE GENERAL et faire le calcul des distances entre les courbes et montrer que pour 2 courbes qui ont des "spectres proches", on obtient une valeur de distance faible.

2.2.4 Other metrics

- Il existe dans la littérature de nombreuses autres métriques pour les séries temporelles (laisser la porte ouverte).
- Certaines métriques sont utilisées dans le domaine temporelle
- D'autres métriques sont utilisés dans d'autres représentations (Wavelet, etc.)
- Certaines combinent la représentation temporelles et fréquentielles (Représentation spectrogramme en temps-fréquence)
- Se baser sur l'article "TSclust : An R Package for Time Series Clustering".
- Fermer le cadre : dans la suite de notre travail, on ne va pas les utiliser mais elles pourront être intégrées dans le framework qui suivra au chapitre suivant

2.3 Kernels for time series

(à compléter)

2.4 Time series alignment

- Les données réelles peuvent présenter des délais, des changements de dynamique de l'échelle de temps : extension, compression (dans la limite du raisonnable).
- Il existe des techniques qui permettent de ré-aligner les séries temporelles comme la DTW
- Définir la notion d'alignement

- Présenter la DTW (+ algorithme)
- Présenter les variantes de la DTW
- Dans la suite du travail, on suppose que les séries temporelles sont ré-alignées.
- Prendre le GRAPHIQUE GENERAL et faire le calcul des distances entre les courbes et montrer que pour 2 courbes qui ont des "valeurs proches" mais décalés, on obtient une valeur de distance faible. (prendre DTW standard avec une fonction de coût D_E par exemple)

2.5 Multi-scale aspect

- Dans le cadre de la classification, on peut avoir des données où l'information qui permet de discriminer une classe d'une autre n'est pas globale mais est localisé sur une partie du signal
- Limite des métriques basiques présentées précédemment (valeur, forme, fréquence) considère la comparaison sur l'intégralité du signal
- On propose la définition de métriques locales. Pour cela, on va découper notre signal. Il existe plusieurs manières de réaliser ce découpage. On va utiliser la dichotomie proposée par Douzal & al.

Chapter 3

Time series advanced metrics

Sommaire

3.1 Combined metrics for time series 31
3.2 Metric Learning: state of the art 32

Chapeau introductif

- Objectif : Trouver une distance, combinaison des distances basiques qui donne une bonne classification k-NN sur une base de données.
- Pourquoi une distance combinée? Dans le cadre de données réelles, plusieurs modalités peuvent être impliquées (forme, valeur, fréquence), de manière globale ou locale.
- Dans le cadre des données réelles, plusieurs composantes/modalités peuvent être impliqués (forme, valeur, fréquence). = attribut (feature) en traitement du signal. Hypothèse : valeur sur une série complète, sur un intervalle ou sur une fenêtre (dans le cadre des métriques à base fréquentielle).

3.1 Combined metrics for time series

- Certains travaux dans la littérature propose des combinaisons : linéaire, exponentielle, sigmoïde.
- Limites:
  - Implique que 2 modalités et au niveau global. Pour intégrer d'autres modalités et à d'autres échelles, il faut changer la formule et ajouter de nouveaux hyperparamètres à optimiser → l'apprentissage de ces paramètres est plus long.
  - La combinaison est définie a priori
  - La combinaison est indépendante de la tâche d'analyse.
  - Pour répondre à ces problèmes, certains auteurs proposent d'apprendre une métrique en vue de la tâche d'analyse considérée (classification, régression, clustering).

3.2 Metric Learning: state of the art

- Placer le contexte : travaux réalisés dans le cadre de la classification de données statiques.
- Présenter l'intuition du Metric Learning sur la base des travaux de Weinberger.
- Donner la terminologie (target, imposter, push, pull)
- Objectif : push des imposters et pull des targets
- Formalisation du problème (optimisation)
- Limites:
  - On apprend les poids d'une distance de Mahalanobis
  - L'apprentissage ne prend pas en compte l'aspect multi-modal dans les données

Conclusion of Part I

This first part of the manuscrit present a state of the art of classic machine learning framework and algorithms to make the classification or the regression of time series. We note that the considered algorithms (k-NN, SVM) are based on the comparison of time series through distance measure. Considering time series as static data lead to compare time series only the comparison of their value, on the same instant between the two considered time series. To take into account over characteristics of time series, other metrics (cort, d_F, etc.) and other methods (DTW, dichomotie) have been proposed to cope with these characteristics. Learning an adequate metrics is a key challenge to well classify time series. Inspired by the metric learning work, we propose in the following a framework to learn a multi-modal and multi-scale metric for a nearest neighbors classification.

Part II

Metric learning for time series from multiple modalities and multiple scales

The first part has enlightened the importance of combining several modalities and several scales to compare time series in order to make a better classification. We propose, in this part, a framework to learn this metric. For that, the idea is to define a new space representation, the pairwise space, where a vector is a pair of time series which is described by the basic metrics. Then, we formalize the problem of learning the metric as an optimization problem and show its equivalence by solving an adequate SVM problem. In the first chapter, we present this pairwise representation and gives interpretation in this new space. In the second chapter, we formalize the optimization problem and its adapted SVM equivalence. In the third chapter, we present the details of the algorithm. In the final chapter, we experiment our proposed approach on datasets, discuss and interpret the results.

Chapter 4

Projection in the pairwise space

Sommaire

4.1 Pairwise embedding 37
4.2 Interpretation of the pairwise space 37
4.3 Pros & Cons 38

Chapeau introductif

- Le calcul d'une métrique implique toujours 2 individus. On va proposer un changement d'espace, un nouvel espace : la représentation par paire.
- Le cadre : on suppose que l'on a p métriques.

4.1 Pairwise embedding

- Changement de l'espace
- Normalisation de l'espace des paires
- Label des pairwise

4.2 Interpretation of the pairwise space

- Proximity to the origin (les individus sont identiques)
- Proximity of 2 pairwise points in the pairwise space
- Norm in the pairwise space
- Representation of combined metric in the pairwise space

4.3 Pros & Cons

- perte de la classe initiale des individus. L'information qui nous reste est : les 2 individus sont de la même classe ou sont de classes différentes.

Chapter 5

M^2TML: formalization

Sommaire

5.1 LP optimization problem 39
5.2 QP optimization problem 39
5.3 SVM approximation 39

Chapeau introductif

- Rappeler : quel problème on résout : pull des targets et push des individus de classes différentes
- Formaliser le problème général avec D

5.1 LP optimization problem

- Formaliser le problème sous forme d'un problème d'optimisation sous contraintes

5.2 QP optimization problem

- Passer de la forme LP (forme primale) et par transformation, arriver à la forme duale
- Montrer les similitudes avec la résolution SVM
- Montrer que l'on peut kerneliser la méthode

5.3 SVM approximation

- Faire remarquer que le problème LP ressemble à un problème SVM
- Faire la démonstration de l'équivalence (ou mettre la démonstration en annexe).

- Expliquer les différences entre la résolution LP/QP et la résolution SVM. (ajout de sur-contraintes dans le problème SVM)
- Expliquer pourquoi on va préférer le cadre SVM. Expliquer mathématiquement et avec des interprétations géométriques.
- Cadre connu
- Utilisation de librairie standard de Machine Learning
- Extension directe à l'apprentissage de métrique non linéaire grâce au kernel trick

Chapter 6

M^2TML: implementation

Sommaire

6.1 Projection in the pairwise space 41
6.2 M-NN M-diff strategy 41
6.3 Radius normalization 42
6.4 Solving the SVM problem 42
6.5 Definition of the dissimilarity measure 42
6.6 Extension to regression problem 42
6.7 Extension to multivariate problem 42

Chapeau introductif :

- Quel problème on résout?
- Donner les étapes principales de résolution (sous forme de puces). Cela doit rester général, clair et concis.
- Développer dans chaque section les puces énumérés précédemment.

6.1 Projection in the pairwise space

- Projection
- Log normalization

6.2 M-NN M-diff strategy

- Expliquer les différentes stratégies (k-NN VS All / M-NN VS M-diff / k-NN VS Imposters)
- Expliquer pourquoi on va choisir une stratégie M-NN VS M-diff

6.3 Radius normalization

- Expliquer le problème de la non-homogénéité des radius.
- Expliquer comment on résout ce problème par une normalisation des radius de chaque voisinage.

6.4 Solving the SVM problem

- Expliquer l'apprentissage avec le SVM.
- Utilisation de la version L1 du SVM pour avoir une solution sparse.

6.5 Definition of the dissimilarity measure

- Produit scalaire
- Papier PR : norme pondérée x fonction exponentielle
- Version Sylvain : norme x fonction exponentielle?

6.6 Extension to regression problem

(To do)

6.7 Extension to multivariate problem

(To do)

Chapter 7

M^2TML: Experiments

Sommaire

7.1 Dataset presentation 43
7.2 Experimental protocol 43
7.3 Results 43
7.4 Discussion 43

Chapeau introductif

- Application sur des bases de séries temporelles univariés de la littérature (Keogh)
- Données Schneider? ou Expliquer les problématiques de Schneider

7.1 Dataset presentation

7.2 Experimental protocol

7.3 Results

7.4 Discussion

Conclusion of Part II

Conclusion and perspectives

- Bilan des apports de la thèse
- Perspectives
  - Multi-pass learning
  - Kernel pour la résolution du problème QP
  - Utilisation de la distance apprise dans d'autres algorithmes de machine learning (Arbre de décision) pour obtenir une interprétabilité?
  - Utilisation d'autres distances (wavelets, etc.)
  - Apprentissage locale de la métrique

Appendix A

Detailed presentation of the datasets

Appendix B

Solver library

Appendix C

SVM librairy

Bibliography

[ABR64] M. Aizerman, E. Braverman, and L. Rozonoer. "Theoretical foundations of the potential function method in pattern recognition learning." In: *Automation and Remote Control* 25 (1964), pp. 821–837 (cit. on p. 16).

[BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers." In: *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. 1992, pp. 144–152 (cit. on pp. 11, 16).

[Cha04] Christopher Chatfield. *The analysis of time series : an introduction*. 2004, xiii, 333 p. (Cit. on pp. 7, 8).

[CV95] Corinna Cortes and Vladimir Vapnik. "Support-vector networks." In: *Machine Learning* 20.3 (1995), pp. 273–297. arXiv: `arXiv:1011.1669v3` (cit. on pp. 11, 12).

[CY11] Colin Campbell and Yiming Ying. *Learning with Support Vector Machines*. Vol. 5. 1. Feb. 2011, pp. 1–95 (cit. on pp. 11, 15).

[Din+08] Hui Ding et al. "Querying and Mining of Time Series Data : Experimental Comparison of Representations and Distance Measures." In: *Proceedings of the VLDB Endowment* 1.2 (2008), pp. 1542–1552. arXiv: `1012.2789v1` (cit. on p. 11).

[Naj+12] H. Najmeddine et al. "Mesures de similarité pour l'aide à l'analyse des données énergétiques de bâtiments." In: *RFIA*. 2012 (cit. on p. 8).

[Ngu+12] L. Nguyen et al. "Predicting collective sentiment dynamics from time-series social media." In: *WISDOM*. 2012 (cit. on p. 8).

[OE73] Richard O Duda and Peter E Hart. *Pattern Classification and Scene Analysis*. Vol. 7. 1973, p. 482 (cit. on pp. 10, 11, 25).

[SS13] Bernhard Schlkopf and Alexander J. Smola. *Learning with Kernels*. Vol. 53. 2013, pp. 1689–1699. arXiv: `arXiv:1011.1669v3` (cit. on pp. 11, 16, 17).

[TSK05] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. 2005, p. 500 (cit. on p. 11).

[Xi+06] Xiaopeng Xi et al. "Fast time series classification using numerosity reduction." In: *Proceedings of the 23rd international conference on Machine learning (ICML)*. 2006, pp. 1033–1040 (cit. on p. 11).

[YG08] J. Yin and M. Gaber. "Clustering distributed time series in sensor networks." In: *ICDM*. 2008 (cit. on p. 8).

[G. 06] S. Thiria G. Dreyfus, J.-M. Martinez, M. Samuelides M. B. Gordon, F. Badran. *Apprentissage Apprentissage statistique*. Eyrolles. 2006, p. 471 (cit. on pp. 9, 10, 25).

Résumé — Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula. Pellentesque rhoncus nunc et augue. Integer id felis.

Mots clés : Série temporelle, Apprentissage de métrique, k-NN, SVM, classification, régression.

Abstract — Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula. Pellentesque rhoncus nunc et augue. Integer id felis.

Keywords: Time series, Metric Learning, k-NN, SVM, classification, regression.

Schneider Electric
Université Grenoble Alpes, LIG
Université Grenoble Alpes, GIPSA-Lab