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**,	
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:

Patrick GALLINARI
Laboratoire LIP6, Président du jury
Stéphane CANU
Laboratoire LITIS, Rapporteur
Marc SEBBAN
Laboratoire LAHC, Rapporteur
Gustavo CAMPS-VALLS
Laboratoire IPL, Examinateur
Prénom NOM
Labo de bidule, Examinateur
Ahlame DOUZAL-CHOUAKRIA
Laboratoire LIG, Directeur de thèse
Michèle ROMBAUT
Laboratoire 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 :	Stéphane CANU	-	Laboratoire LITIS
	Marc SEBBAN	-	Laboratoire LAHC
Directeur :	Ahlame DOUZAL-CHOUAKRIA	-	Laboratoire LIG
Co-Directeur :	Michèle ROMBAUT	-	Laboratoire GIPSA-Lab
Encadrant :	Sylvain MARIÉ	-	Schneider Electric
Président :	Patrick GALLINARI	-	Laboratoire LIP6
Examinateur :	Gustavo CAMPS-VALLS	-	Laboratoire IPL
	Prénom NOM	-	Labo de bidule

Todo list

Acknowledgements

I would like to thanks:

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

Contents

List of Figures

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*
Acc	*Classification accuracy*
Err	*Classification error rate*
MAE	*Mean Absolute Error*
RMSE	*Root Mean Square Error*
FAQ	Frequently Asked Questions / Foire Aux Questions

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

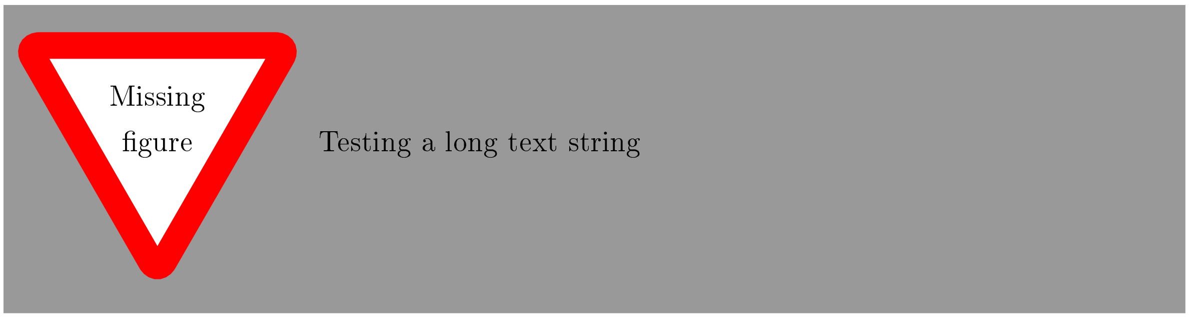
Présenter les différents chapitres

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
$\lvert\lvert\mathbf{x}\rvert\rvert_q$	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 positioning

The first part of the manuscript aims at positioning the work context. Our objective is the classification and regression of time series. The first chapter presents classic machine learning technics for static data. In particular, we focus on k-Nearest Neighbors classification and Support Vector Machine approach. In the second chapter, we recall metrics used in the literature to compare time series and present the concept of metric learning.

CHAPTER 1

Related work

Sommaire

In this chapter, we recall some concepts of machine learning. First, we review the principle, the learning framework and the evaluation protocol in supervised learning. Then, we present the algorithms used in our work: k-Nearest Neighbors (k-NN) and Support Vector Machine (SVM).

1.1 Classification, Regression

In this section, we review some terminology in machine learning. First, we recall the principle of machine learning. Then, we detail how to design a framework for supervised learning. After that, we present model evaluation. Finally, we review data normalization.

1.1.1 Machine learning principle

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 instance, to teach a child how to read letters, we show him during a 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. Then, during a 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 $X = \{\mathbf{x}_i, y_i\}_{i=1}^n$ be a training set of n samples $\mathbf{x}_i \in \mathbb{R}^p$ 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.

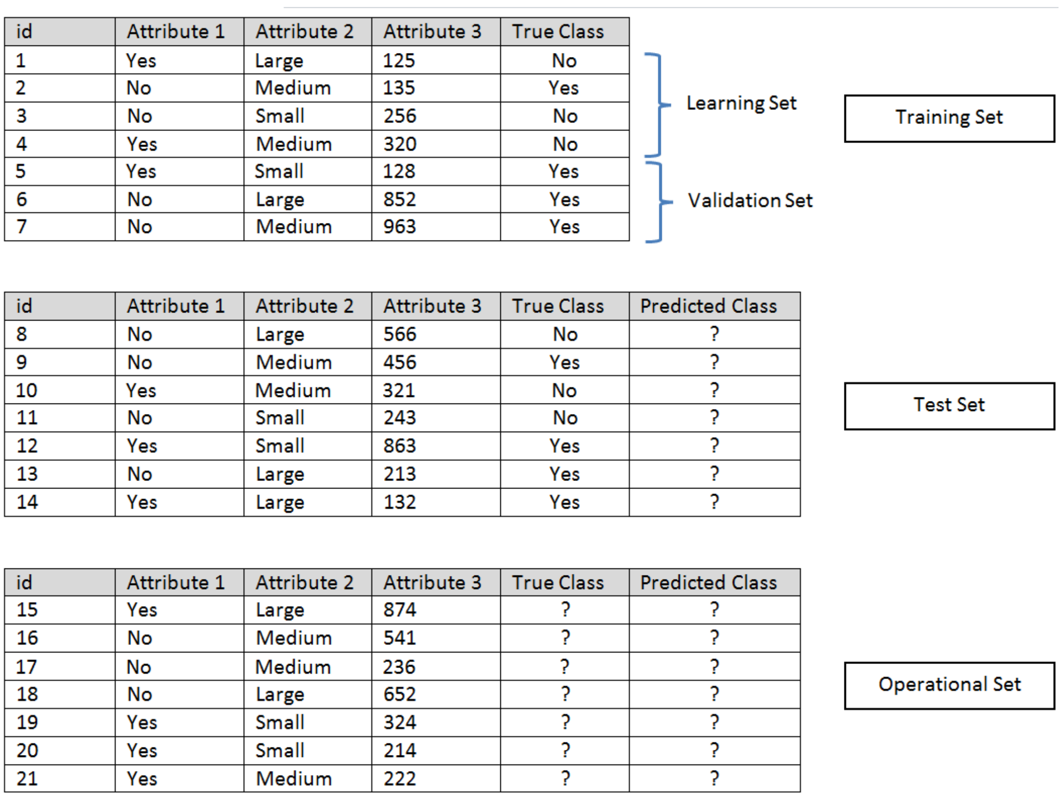
When y_i are class labels (e.g., class 'A', 'B', 'C' in the case of child's reading), learning the model f is a classification problem; when y_i is a continuous value (e.g., the energy consumption in a building), learning f is a regression problem. Both problems corresponds to supervised learning as $\mathbf{x}_i$ and y_i are known during the training phase [Bis06]; [G. 06]; [OE73]. 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 totally unknown, learning f refers to a clustering problem (unsupervised learning) [JMF99]; [CHY96], out of the scope of this work.

[biblio semi-supervisé]

1.1.2 Model selection

A key objective of learning algorithms is to build models f with good generalization abilities, i.e., models f that correctly predict the class labels y_j of new unknown samples $\mathbf{x}_j$. Fig. 1.2 shows a general approach for solving machine learning problems. In general, a dataset can be divided into 3 sub-datasets (illustrated in Fig. 1.1):

- A **training set** $X = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ consisting of n samples $\mathbf{x}_i$ whose labels y_i are known. The training set is used to build the supervised model f. When the learning algorithm needs to tune hyper-parameters, the training set X is divided into two subsets :
  - A **learning set** which is used to build the supervised model f for each value of the hyper-parameter.
  - A **validation set** which is used to evaluate the supervised model f for each value of the hyper-parameter. The model f with the lowest error on the validation set is kept.
- A **test set** $X_{Test} = \{(\mathbf{x}_j, y_j)\}_{j=1}^m$, which consists of m samples $\mathbf{x}_j$ whose labels y_j are also known but the model f is applied to predict the label $\hat{y_j}$ of samples $\mathbf{x}_j$. The test is used to evaluate the performance of the learnt model between $\hat{y_j}$ and y_j.
- An **operational set** $X_{op} = \{(\mathbf{x}_l, y_l)\}_{l=1}^L$, which consists of L samples $\mathbf{x}_l$ whose labels y_l are totally unknown. The operational set is in general a new dataset on which the learnt algorithm is applied.

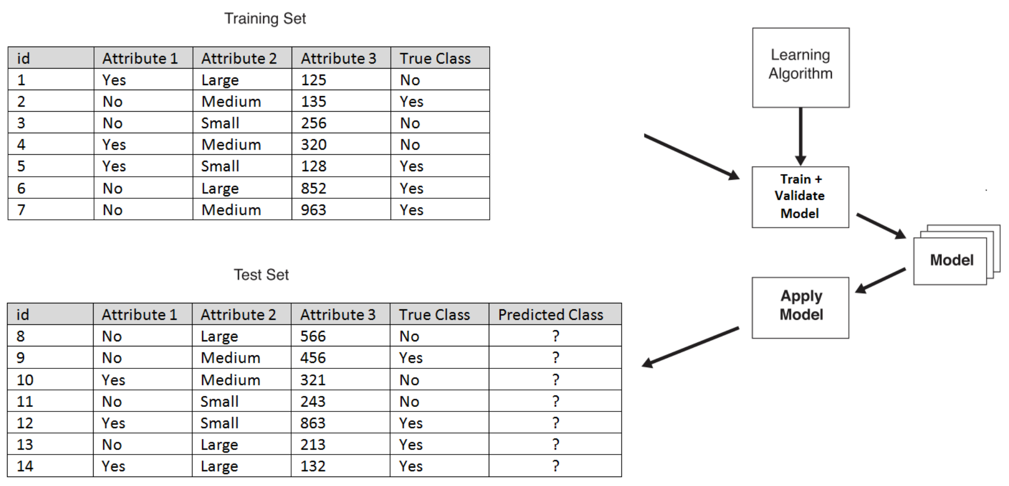


id	Attribute 1	Attribute 2	Attribute 3	True Class
1	Yes	Large	125	No
2	No	Medium	135	Yes
3	No	Small	256	No
4	Yes	Medium	320	No
5	Yes	Small	128	Yes
6	No	Large	852	Yes
7	No	Medium	963	Yes

id	Attribute 1	Attribute 2	Attribute 3	True Class	Predicted Class
8	No	Large	566	No	?
9	No	Medium	456	Yes	?
10	Yes	Medium	321	No	?
11	No	Small	243	No	?
12	Yes	Small	863	Yes	?
13	No	Large	213	Yes	?
14	Yes	Large	132	Yes	?

id	Attribute 1	Attribute 2	Attribute 3	True Class	Predicted Class
15	Yes	Large	874	?	?
16	No	Medium	541	?	?
17	No	Medium	236	?	?
18	No	Large	652	?	?
19	Yes	Small	324	?	?
20	Yes	Small	214	?	?
21	Yes	Medium	222	?	?

Figure 1.1: Division of a dataset into 3 datasets: training, test and operational.



id	Attribute 1	Attribute 2	Attribute 3	True Class
1	Yes	Large	125	No
2	No	Medium	135	Yes
3	No	Small	256	No
4	Yes	Medium	320	No
5	Yes	Small	128	Yes
6	No	Large	852	Yes
7	No	Medium	963	Yes

id	Attribute 1	Attribute 2	Attribute 3	True Class	Predicted Class
8	No	Large	566	No	?
9	No	Medium	456	Yes	?
10	Yes	Medium	321	No	?
11	No	Small	243	No	?
12	Yes	Small	863	Yes	?
13	No	Large	213	Yes	?
14	Yes	Large	132	Yes	?

Figure 1.2: General framework for building a supervised (classification/regression) model. Example with 3 features and 2 classes ('Yes' and 'No').

There exists two types of errors committed by a classification or regression model f: training error and generalization error. **Training error** is the error on the training set and **generalization error** is the error on the testing set. 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 (test set X_{Test}). 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 much can have a poorer generalization error than a model with a higher training error. Such a situation is known as model overfitting (Fig. 1.3).

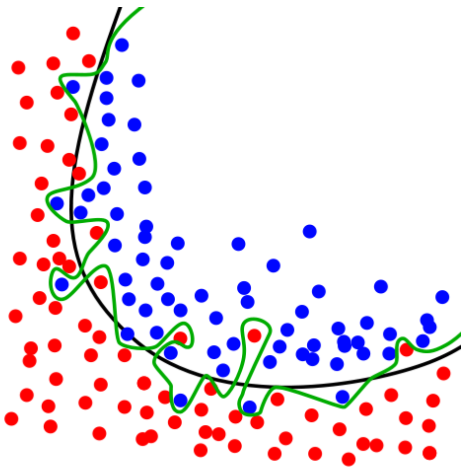


Figure 1.3: An example of overfitting in the case of classification. The objective is to separate blue points from red points. Black line shows a classifier f_1 with low complexity where as green line illustrates a classifier f_2 with high complexity. On training examples (blue and red points), the model f_2 separates all the classes perfectly but may lead to poor generalization on new unseen examples. Model f_1 is often preferred.

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 (C, γ) of the hyper-parameters, that is in this case a 2-dimensional grid (Fig. 1.4). 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.

An alternative is cross-validation with v folds, illustrated in Fig. 1.5. 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 $v-1$ remaining 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 or Jackknife. Each test set contains only one sample. The advantage is much data are used as possible for training. Moreover, the validation 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 validation set contains only one record, the variance of the estimated performance metric is usually high. This procedure is often used when n, the size training set,

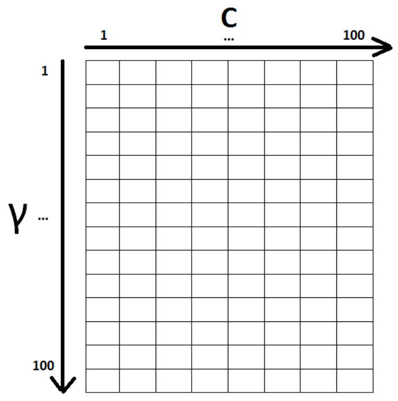


Figure 1.4: Example of a 2 dimensional grid search for parameters C and γ. It defines a grid where each cell of the grid contains a combination (C, γ). Each combination is used to learn the model and is evaluated on the validation set.

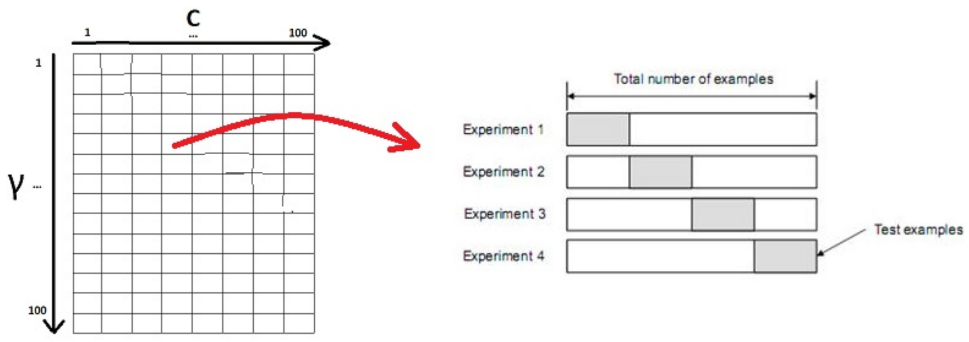


Figure 1.5: v-fold Cross-validation for one combination of parameters. For each of v experiments, use $v-1$ folds for training and a different fold for Testing, then the training error for this combination of parameter is the mean of all testing errors. This procedure is illustrated for $v = 4$.

is small. There exists other methods such as sub-sampling or bootstraps [OE73]; [G. 06]. We only use cross-validation in our experiments.

Comment [AD3]: ne pas dire ici. Michèle dit oui

1.1.3 Model evaluation

1.1.3.a Classification evaluation

The performance of a classification model is based on the counts of test samples $\mathbf{x}_j$ correctly and incorrectly predicted by the model f. These counts are tabulated in a table called the confusion matrix. Table 1.1 illustrates the concept for a binary classification problem. Each cell f_{ij} the table stands for the number of samples from class i predicted to be of class j.

Based on this matrix, the number of correct predictions made by the model is $\sum_{i=1}^{C} f_{ii}$, where C is the number of classes. Equivalently, the ratio of incorrect predictions is $1 - \sum_{i=1}^{C} f_{ii}$.

		Predicted class	
		Class = 1	Class = 0
Actual Class	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

Table 1.1: Confusion matrix for a 2-class problem.

To summarize the information, it generally more convenient to use performance metrics such as the classification accuracy (Acc) or error rate (Err). This allows to compare several models with a single number. Note that $Err = 1 - Acc$.

$$Acc = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{\sum\limits_{i=1}^{C} f_{ii}}{\sum\limits_{i,j=1}^{C} f_{ij}} \tag{1.1}$$

$$Err = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} = \frac{\sum\limits_{i,j=1,i\neq j}^{C} f_{ij}}{\sum\limits_{i,j=1}^{C} f_{ij}} \tag{1.2}$$

Using these performance metrics allows to compare the performance of different classifiers f. It allows to determine in particular whether one learning algorithm outperforms another on a particular learning task on a given test dataset X_{Test}. However, depending on the size of the test dataset, the difference in error rate Err between two classifiers may not be statistically significant. Snedecor & Cochran proposed in 1989 a statistical test based on measuring the difference between two learning algorithms [Coc77]. It has been used by many researchers [Die97]; [DHB95].

Let consider 2 classifiers f_A and f_B. We test these classifiers on the test set X_{Test} and denote p_A and p_B their respective error rates. The intuition of this statistical test is that when algorithm A classifies an example $\mathbf{x}_j$ from the test set X_{Test}, the probability of misclassification is p_A. Thus, the number of misclassification of m test examples is a binomial random variable with mean mp_A and variance $p_A(1-p_A)m$. The binomial distribution can be approximated by a normal distribution when m has a reasonable value (Law of large numbers). The difference between two independent normally distributed random variables is also normally distributed. Thus, the quantity $p_A - p_B$ is a normally distributed random variable. Under the null hypothesis (the two algorithm should have the same error rate), this will have a mean of zero and a standard error se of:

$$se = \sqrt{\frac{2p(1-p)}{m}} \tag{1.3}$$

where $p = \frac{p_A + p_B}{2}$ is the average of the two error probabilities. From this analysis, we obtain the statistic:

$$z = \frac{p_A - p_B}{\sqrt{2p(1-p)/m}} \tag{1.4}$$

which has (approximatively) a standard normal distribution. We can reject the null hypothesis if $|z| > Z_{0.975} = 1.96$ (for a 2-sided test with probability of incorrectly rejecting the null hypothesis of 0.05).

1.1.3.b Regression evaluation

As the concept of classes is restricted to classification problems, the performance of a regression model f is based on metrics that measure the difference between the predicted label $\hat{y}_j$ and the known label y_j. The Mean Absolute Error function (MAE) computes the mean absolute error, a risk metric corresponding to the expected value of the absolute error loss or L1-norm loss.

$$MAE(\hat{y}, y) = \frac{1}{m}\sum_{j=1}^{m} |\hat{y}_j - y_j| \tag{1.5}$$

A commonly used performance metrics is the Root Mean Squared Error function (RMSE) that computes the root of the mean square error, a risk metric corresponding to the expected value of the squared (quadratic) error loss.

$$RMSE(\hat{y}, y) = \sqrt{\frac{1}{m}\sum_{j=1}^{m} (\hat{y}_j - y_j)^2} \tag{1.6}$$

Many works relies on the R^2 function, the coefficient of determination. It provides a measure of how well future samples are likely to be predicted by the model.

$$R^2(\hat{y}, y) = 1 - \frac{\sum_{j=1}^{m} (\hat{y}_j - y_j)^2}{\sum_{j=1}^{m} (\bar{y} - y_j)^2} \tag{1.7}$$

where $\bar{y} = \sum_{j=1}^{m} y_j$ is the mean over the known labels y_j.

1.1.4 Data normalization

Real dataset are often subjected to noise or data scaling. Before applying any learning protocol, it is often necessary to pre-process the data: data scaling, data filtering (e.g., de-noising), etc. We focus on data normalization (data scaling) in our work.

Part 2 of Sarle's Neural Networks FAQ (1997) [1] explains the importance of data normalization for neural network but they can be applied to any learning algorithms. The main advantage of normalization is to avoid attributes in greater numeric ranges to dominate those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. For example, in the case of Support Vector Machine (SVM), because kernel values usually depend on the inner products of feature vectors, i.e. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems [HCL08].

In most cases, it is recommended to scale each attribute to the range [-1; +1] or [0; 1]. Many normalization methods have been proposed such as Min/Max normalization, Z-normalization or normalization of the log normalization . Let $X = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$ be a training set, $\mathbf{x}_i$ being a sample described by T features $\mathbf{X}_1, \dots, \mathbf{X}_T$. We define μ_j and σ_j as the mean and the standard deviation of a variable $\mathbf{X}_j$, applying the Z-normalized variable $\mathbf{X}_j^{norm}$ is given by:

références normal-ization

$$\mathbf{X}_j^{norm} = \frac{\mathbf{X}_j - \mu_j}{\sigma_j} \tag{1.8}$$

Note that the underlying assumption supposes that the variable $\mathbf{X}_j$ is normally distributed: data evolves between $[-\infty; +\infty]$ and are coming from a Gaussian process. In some cases, the data are skewed such as monetary amounts, incomes or distance measures. These data are often log-normally distributed, e.g., the log of the data is normally distributed (Fig. 1.6). The underlying idea is to take the log of the data ($\mathbf{X}_j^{log}$) to restore the symmetry, and then, to apply a Z-normalization of this transformation:

$$\mathbf{X}_j^{log} = \ln(\mathbf{X}_j); \tag{1.9}$$

$$\mathbf{X}_j^{log,norm} = \frac{\mathbf{X}_j^{log} - \mu_j^{log}}{\sigma_j^{log}} \tag{1.10}$$

$$\mathbf{X}_j^{norm} = \exp(\mathbf{X}_j^{log,norm}) \tag{1.11}$$

where ln denotes the Natural Logarithm function, μ_j^{log} and σ_j^{log} the mean and the standard deviation of a variable $\mathbf{X}_j^{log}$.

Finally, we recall some precautions to the practitioner in the learning protocol, experimented by Hsu & al. in the context of SVM [HCL08]. First, training and testing data must be scaled using the same method. Second, training and testing data must not be scaled separately. Third, the whole dataset must not be scaled together at the same time. These often leads to poorer results. A proper way to do normalization is to scale the training data, store the parameters of the normalization (i.e. μ_i and σ_i for Z-normalization), then apply the same normalization to the testing data.

[1] `http://www.faqs.org/faqs/ai-faq/neural-nets/`

[2] source: `http://www.r-statistics.com/2013/05/log-transformations-for-skewed-and-wide-distributions-from-practical-data-science-with-r/`

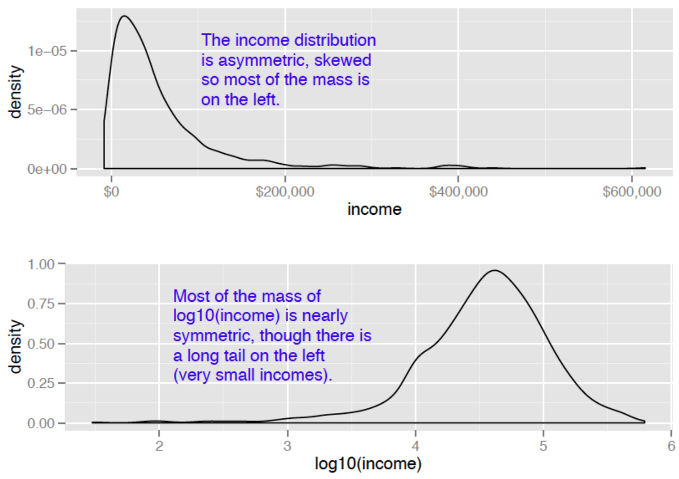


Figure 1.6: A nearly log-normal distribution, and its log transform [2]

1.2 Machine learning algorithms

Many algorithms have been proposed in the context of supervised learning, such as the Deep Neural Network, the Decision Tree or the Relevance Vector Machine (RVM). Our proposition uses Support Vector Machine (SVM) in the context of k-Nearest Neighbors (k-NN) classification. We limit the section to present these two algorithms.

Comment [AD4]: dit d'enlever 1ère phrase mais Michèle dit de garder

1.2.1 k-Nearest Neighbors (k-NN) classifier

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 class y_i of its nearest neighbor $\mathbf{x}_i$ in the training set.

Comment [AR5]: Ahlam trouve que ce n'est pas clair. A refaire

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 as the k-Nearest Neighbors algorithm (k-NN) [SJ89]; [CH67]. Fig. 1.7 illustrates the concept for a neighborhood of $k = 3$ and $k = 5$.

In the k-NN algorithm, the notion of "closeness" between samples $\mathbf{x}_i$ is based on the computation of a metric [3] D. For static data, usually used metrics are the Euclidean distance, the Minkowski distance or the Mahalanobis distance. Considering a training set X of n samples, solving the 1-NN classification problem is equivalent to solve the optimization problem:
For a new sample $\mathbf{x}_j$, $\forall i \in \{1...n\}$,

$$y_j = y_{i^*} \tag{1.12}$$

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

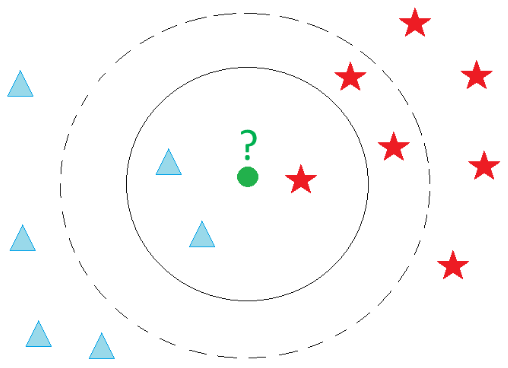


Figure 1.7: Example of k-NN classification. The test sample (green circle) is classified either to the first class (red stars) or to the second class (blue triangles). If $k = 3$ (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 star inside the inner circle. If $k = 5$ (dashed line circle) it is assigned to the first class (3 stars vs. 2 triangles inside the outer circle).

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

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

$$y_j = \frac{1}{k} \sum_{i=1}^{k} y_i \tag{1.13}$$

where i corresponds to the index of the k-nearest neighbors [Alt92]. There exists other variants of the k-NN algorithms. In a weighed k-NN, the approach consists in weighting the k-NN decision by assigning to each neighbor $\mathbf{x}_i$ from an unknown sample $\mathbf{x}_j$, a weight w_i defined as a function of the distance $D(\mathbf{x}_i, \mathbf{x}_j)$ [Dud76]. To cope with uncertainty or imprecision in the labeling of the training data $\mathbf{x}_i$, other authors propose in a fuzzy k-NN to determine the membership degree in each class of an unseen sample $\mathbf{x}_j$ by combining the memberships of its neighbors [KGG85]. Denoeux propose a framework based on Dempster-Shafer theory where the k-NN rule takes into account the non-representativity of training data, the weighting rule and uncertainty in the labeling [Den95].

Comment [AD6]: Expliquer d'avantage

Despite its simplicity, the k-NN algorithm has been shown to be successful on time series classification problems [BMP02]; [Xi+06]; [Din+08]. However, the k-NN algorithm presents some disadvantages, mainly due to its computational complexity, both in space (storage of the training samples $\mathbf{x}_i$) and time (search of the neighbors) [OE73]. Suppose we have n labeled training samples in p 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 $O(p)$ and thus the search is $O(pn)$. Moreover, using standard metrics (such as the Euclidean distance) uses all the p 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.

1.2.2 Support Vector Machine (SVM) algorithm

Support Vector Machine (SVM) is a classification method introduced in 1992 by Boser, Guyon, and Vapnik [BGV92]; [CV95] to solve at first linearly separable problems. 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 categorization, bioinformatics and database marketing [Wan02]; [YL99]; [HHP01]; [SSB03]; [CY11]. SVMs belong to the category of kernel methods, algorithms that depends on the data only through dot-products [SS13]. It allows thus to solve non-linear problem. This section gives a brief overview of the mathematical key points and interpretation of the method. For more informations, the reader can consult [SS13]; [CY11]; [CV95].

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.

1.2.2.a Intuition

Let $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$ be a set of n samples $\mathbf{x}_i \in \mathbb{R}^p$ and their labels $y_i = \pm 1$ (2 class-problem). The objective is to learn a hyperplane, whose equations are $\mathbf{w}^T\mathbf{x} + b = 0$, that can separate samples of class $+1$ from the ones of class -1. When the problem is linearly separable such as in Fig. 1.8, there exists an infinite number of hyperplanes.

Comment [AD7]: Mettre dans les figures des + et - pour les classes

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 samples $\mathbf{x}_i$ of each class, called the support vectors. This distance is equal to $\frac{1}{||\mathbf{w}||_2}$. We denote $||\mathbf{w}||_2$, the L2-norm of the vector $\mathbf{w}$ and $||\mathbf{w}||_1$ the L1-norm of $\mathbf{w}$:

$$||\mathbf{w}||_2 = \sqrt{\mathbf{w}^T\mathbf{w}} = \sqrt{\sum_{h=1}^{p} w_h^2} \tag{1.14}$$

$$||\mathbf{w}||_1 = \sum_{h=1}^{p} |w_h| \tag{1.15}$$

where $\mathbf{w} = [w_1, \ldots, w_p]$ denotes the weight vector.
The hyperplanes passing through the support vectors of each class are referred as the canonical hyperplanes, and the region between the canonical hyperplanes is called the margin band (Fig. 1.9).

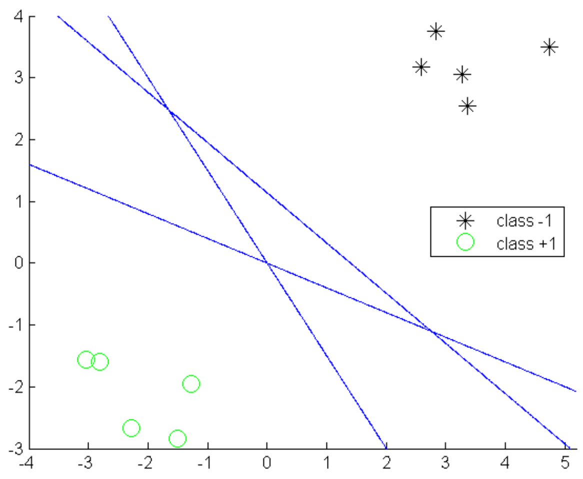


Figure 1.8: Example of linear classifiers in a 2-dimensional plot. For a set of points of classes +1 and -1 that are linearly separable, there exists an infinite number of separating hyperplanes corresponding to $\mathbf{w}^T\mathbf{x} + b = 0$.

1.2.2.b 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:

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \frac{1}{2}||\mathbf{w}||_2^2 \tag{1.16}$$

$$\textbf{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \tag{1.17}$$

This is a constrained optimization problem in which we minimize an objective function (Eq. 1.16) subject to constraints (Eq. 1.17). This formulation is referred as the primal hard margin problem. When the problem is not linearly separable, slack variables $\xi_i \geq 0$ are introduced to relax the optimization problem:

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \left(\overbrace{\frac{1}{2}||\mathbf{w}||_2^2}^{\textit{Regularization}} + C \overbrace{\sum_{i=1}^{n} \xi_i(\mathbf{w}; b; x_i; y_i)}^{\textit{Loss}} \right) \tag{1.18}$$

$$\textbf{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \tag{1.19}$$

$$\xi_i \geq 0 \tag{1.20}$$

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

This formulation is referred as the primal soft margin problem. It is a quadratic programming optimization problem subjected to constraints. Thus, it is a convex problem: any local

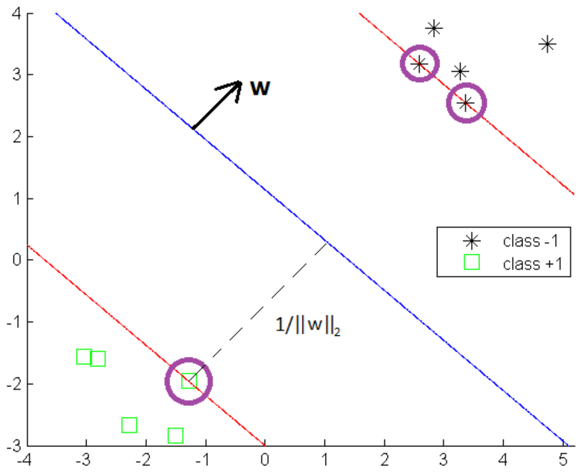


Figure 1.9: The argument inside the decision function of a classifier is $\mathbf{w}^T\mathbf{x}+b$. The separating hyperplane corresponding to $\mathbf{w}^T\mathbf{x}+b=0$ is shown as a line in this 2-dimensional plot. This hyperplane separates the two classes of data with points on one side labeled $y_i=+1$ ($\mathbf{w}^T\mathbf{x}+b\geq 0$) and points on the other side labeled $y_i=-1$ ($\mathbf{w}^T\mathbf{x}+b<0$). Support vectors are circled in purple and lies on the hyperplanes $\mathbf{w}^T\mathbf{x}+b=+1$ and $\mathbf{w}^T\mathbf{x}+b=-1$

solutions is a global solution. The objective function in Eq. 1.18 is made of two terms. The first one, the regularization term, penalizes 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 $\max(1-y_i\mathbf{w}^T\mathbf{x}_i,0)$ and $[\max(1-y_i\mathbf{w}^T\mathbf{x}_i,0)]^2$. 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.

Two common regularizer are $||\mathbf{w}||_1$ and $||\mathbf{w}||_2$. 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.

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

$$f(\mathbf{x}_j)=sign(\mathbf{w}^T\mathbf{x}_j+b) \tag{1.21}$$

1.2.2.c 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. The 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. 1.18, 1.19 and 1.20 with a L1-Loss function. As a constrained optimization problem, the formulation is equivalent to the minimization of a Lagrange function $L(\mathbf{w}, b)$, consisting of the sum of the objective function and the n constraints multiplied by their respective Lagrange multipliers $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^T$:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \left(L(\mathbf{w}, b) = \frac{1}{2}(\mathbf{w}^T\mathbf{w}) - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) \right) \tag{1.22}$$

s.t. $\forall i = 1...n$:

$$\alpha_i \geq 0 \tag{1.23}$$

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0 \tag{1.24}$$

$$\alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) = 0 \tag{1.25}$$

where $\alpha_i \geq 0$ are the Lagrange multipliers. In optimization theory, Eq. 1.23, 1.24 and 1.25 are called 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 1.2.2.e.

At the minimum value of $L(\mathbf{w}, b)$, we assume the derivatives with respect to b and $\mathbf{w}$ are set 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:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{1.26}$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{1.27}$$

By substituting $\mathbf{w}$ into $L(\mathbf{w}, b)$ in Eq. 1.22, we obtain the dual formulation (*Wolfe dual*):

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

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

$$\alpha_i \geq 0 \tag{1.30}$$

The dual objective in Eq. 1.28 is quadratic in the parameters α_i. Adding the constraints in Eq. 1.29 and 1.30, 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 same formulation [CY11] (Eqs. 1.28 and 1.29), except that the Lagrange multipliers α_i are upper bounded by the trade-off C in the soft margin formulation:

$$0 \leq \alpha_i \leq C \tag{1.31}$$

The constraints in Eq. 1.31 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:

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \tag{1.32}$$

$$b^* = \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i) \tag{1.33}$$

At the optimality point, only a few number of datapoints have $\alpha_i^* > 0$ as shown as in Fig. 1.10. 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:

$$f(\mathbf{x}_j) = sign(\sum_{i=1}^{n} \alpha_i^* y_i (\mathbf{x}_i . \mathbf{x}_j) + b^*) \tag{1.34}$$

1.2.2.d 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 &

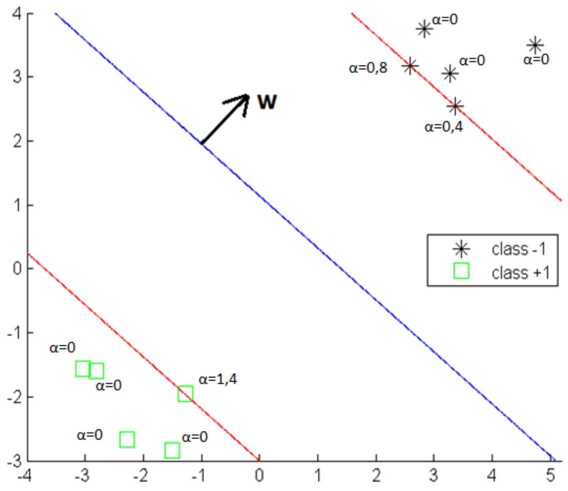


Figure 1.10: Obtained hyperplane after a dual resolution (full blue line). The 2 canonical hyperplanes (dash blue line) contains the support vectors whose $\alpha_i > 0$. Other points have their $\alpha_i = 0$ and the equation of the hyperplane is only affected by the support vectors.

al. for Support Vector Machine (SVM) and has been received a great number of improvements and extensions to symbolic objects such as text or graphs [BGV92].

From the dual objective in Eq. 1.28, 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:

$$(\mathbf{x}_i.\mathbf{x}_j) \rightarrow \Phi(\mathbf{x}_i).\Phi(\mathbf{x}_j) \tag{1.35}$$

where Φ is the mapping function. The intuition behind is that for many datasets, it is not possible to find a hyperplan that can separate the two classes in the input space if 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. 1.11 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 labeled points are moved forward and the −1 labeled 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 $K(\mathbf{x}_i, \mathbf{x}_j)$:

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

Defining a kernel has to follow rules. One of these rules specifies that the kernel function 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.

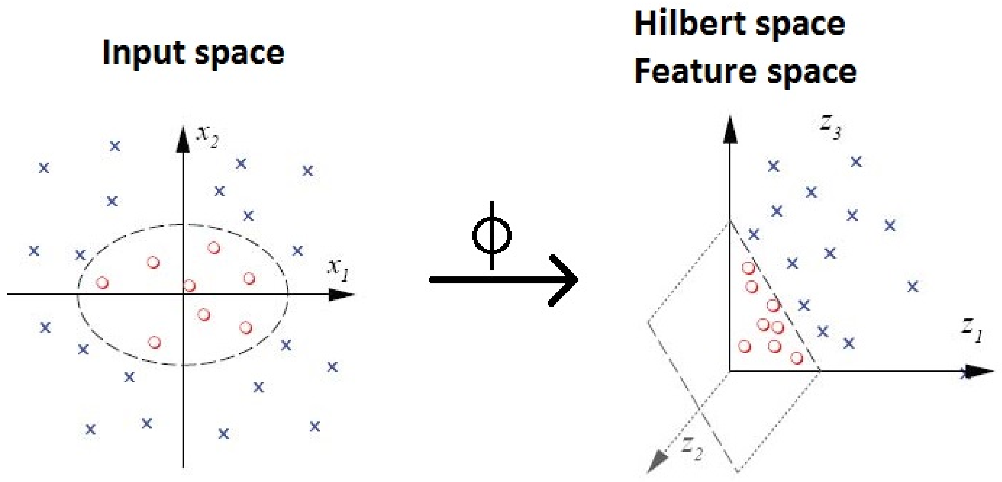


Figure 1.11: Left: in two dimensions these two classes of data are mixed together, and it is not possible to separate them by a line: the data is not linearly separable. Right: using a Gaussian kernel, these two classes of data (cross and circle) become separable by a hyperplane in feature space, which maps to the nonlinear boundary shown, back in input space.[4]

Many kernels have been proposed in the literature such as the polynomial, sigmoid, exponential or wavelet 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:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i.\mathbf{x}_j \tag{1.36}$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{||\mathbf{x}_j - \mathbf{x}_i||_2^2}{2\sigma^2}) = \exp(-\gamma||\mathbf{x}_j - \mathbf{x}_i||_2^2) \tag{1.37}$$

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 the identity transformation. In practice, for large scale problem (when p is high), using a Linear kernel is sufficient [FCH08].

The Gaussian kernel computed between a sample $\mathbf{x}_j$ and 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. 1.12 [5]. The parameter γ controls the decreasing speed of the sphere. In practice, this parameter is learnt during the training phase.

[4]source: `http://users.sussex.ac.uk/~christ/crs/ml/lec08a.html`

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

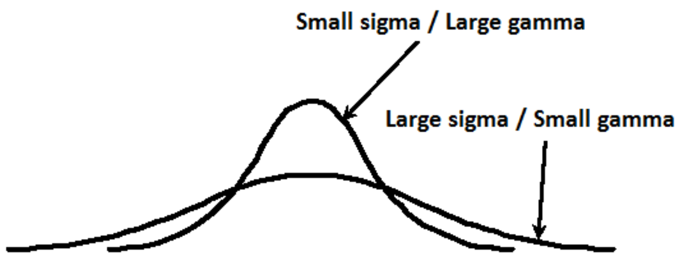


Figure 1.12: Illustration of the Gaussian kernel in the 1-dimensional input space for a small and large γ.

By applying the kernel trick to the soft margin formulation in Eq. 1.28, 1.29 and 1.31, the following optimization problem allows to learn non-linear classifiers:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i.\mathbf{x}_j) \right) \tag{1.38}$$

$$\textbf{s.t.} \ \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{1.39}$$

$$0 \leq \alpha_i \leq C \tag{1.40}$$

The decision function f becomes:

$$f(\mathbf{x}_j) = sign(\sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}_i.\mathbf{x}_j) + b^*) \tag{1.41}$$

Note that in this case, we can't recover the weight vector $\mathbf{w}^*$. Let n_{SV} be the number of support vectors ($n_{SV} \leq n$). To recover b^*, we recall that for support vectors $\mathbf{x}_i$:

$$y_j \left(\sum_{i=1}^{n_{SV}} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}_j) + b^* \right) = 1 \tag{1.42}$$

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

$$b^* = \frac{1}{y_j} - \sum_{i=1}^{n_{SV}} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}_j) \tag{1.43}$$

1.2.2.e Interpretation

Interpretation in the primal

We recall that $\mathbf{x}_i$ is a sample in p dimensions: $\mathbf{X}_1, \ldots, \mathbf{X}_p$. Geometrically, the vector $\mathbf{w}$ represents the direction of the hyperplane (Fig. 1.13). The bias b is equal to the distance of the hyperplane to the origin point $\mathbf{x} = \mathbf{0}$[6]. The orthogonal projection of a sample $\mathbf{x}_i$ on the direction $\mathbf{w}$ is $P_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i$. In the soft margin problem, the slack variables ξ_i of the samples $\mathbf{x}_i$ that lies within the two canonical hyperplanes are equal to zero. Outside of these canonical hyperplanes, the slack variables $\xi_i > 0$ are equal to the distance to the hyperplane.

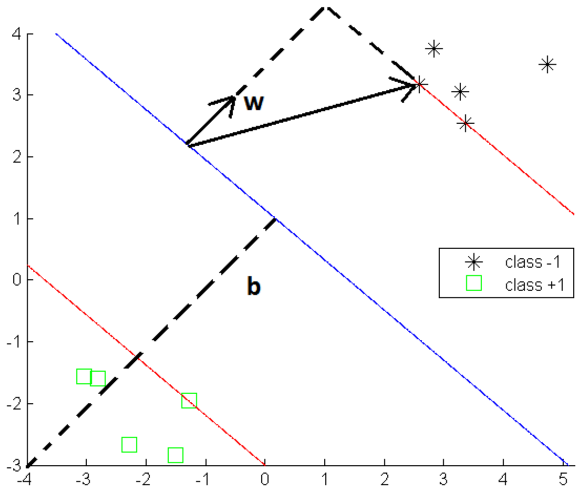


Figure 1.13: Geometric representation of SVM.

In the primal, the weight vector $\mathbf{w} = [w_1, \ldots, w_p]^T$ contains as many elements as there are dimensions in the dataset, i.e., $\mathbf{w} \in \mathbb{R}^p$. 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 used for the classification problem.

In order to visualize the above interpretation of the weight vector $\mathbf{w}$, let us examine several hyperplanes $\mathbf{w}^T\mathbf{x}+b = 0$ shown in Fig. 1.14 with $^T = 2$. 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 $\mathbf{w}$ 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 ($p = 3$) where an element of $\mathbf{w}$ 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.

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 $\mathbf{X}_j$ of the time series are normalized before learning the SVM model, they evolves in the same range. Thus, the ratio $\frac{w_j}{||\mathbf{w}||_2}.100$ defines the percentage of contribution for each variable $\mathbf{X}_j$ in the SVM model.

Interpretation in the dual

As a constrained optimization, the dual form satisfies the Karush-Kuhn-Tucker (KKT) con-

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

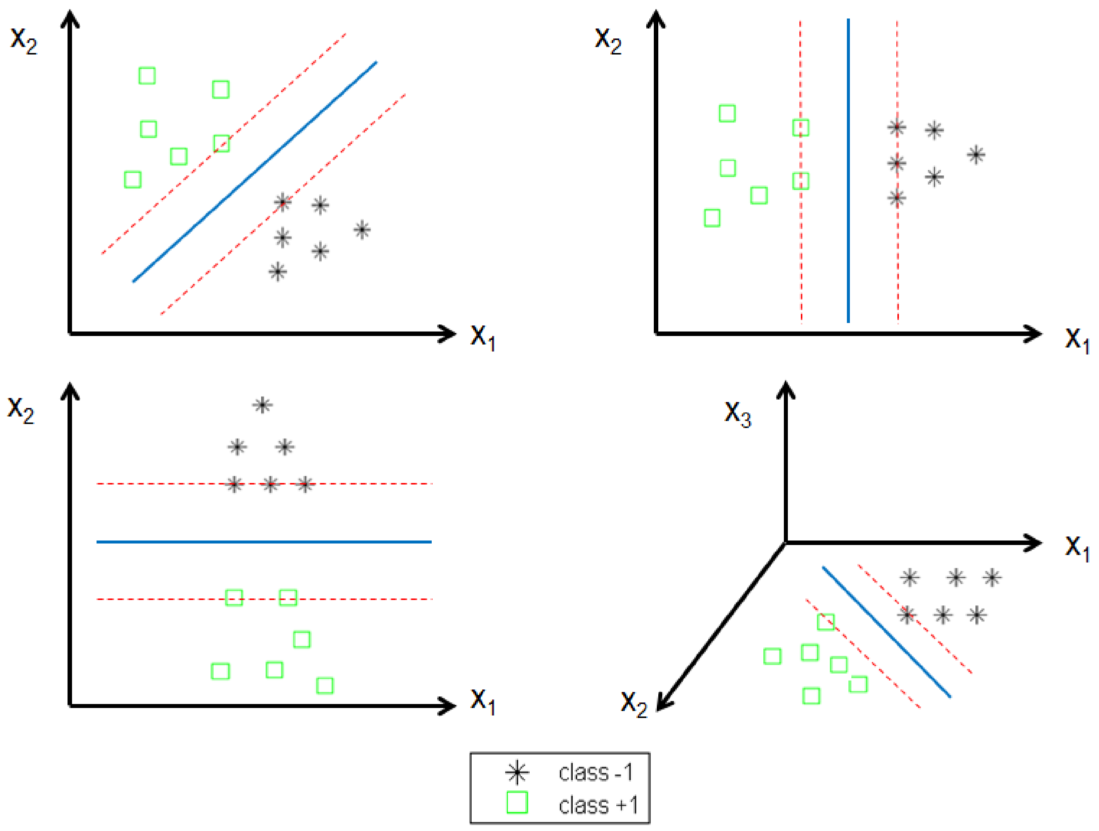


Figure 1.14: Example of several SVMs and how to interpret the weight vector **w**

ditions (Eq. 1.23, 1.24 and 1.25). We recall Eq. 1.25:

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

From this, for every datapoint $\mathbf{x}_i$, either $\alpha_i^* = 0$ or $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$. Any datapoint with $\alpha_i^* = 0$ do not appear in the sum of the decision function f in Eq. 1.34 or 1.41. Hence, they play no role for the classification decision of a new sample $\mathbf{x}_j$. The others $\mathbf{x}_i$ such that $\alpha_i^* > 0$ 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. In the soft margin formulation, by constraining α_i^* to be inferior to C (Box constraints) the effect of outliers can be reduced and controlled.

1.2.2.f Extensions of SVM

SVM has received many interests in recent years. Many extensions has been developed such as ν-SVM, asymmetric soft margin SVM or multiclass SVM [KU02]; [CS01]. One interesting extension is 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 sparseness, 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. 1.15.

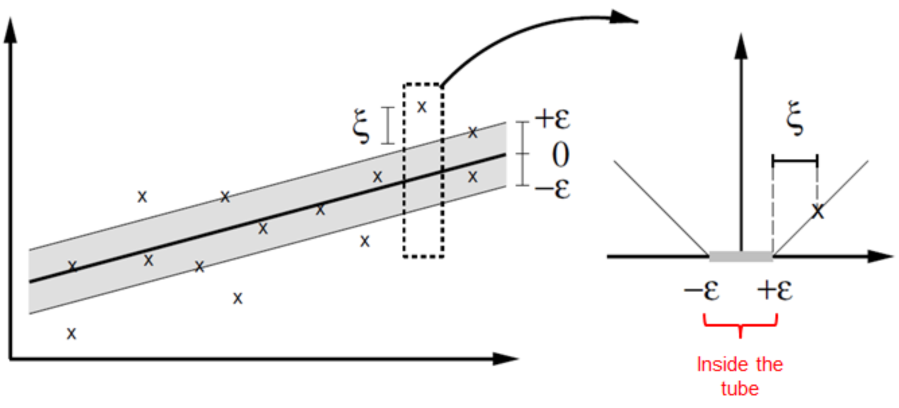


Figure 1.15: Illustration of SVM regression (left), showing the regression curve with the ϵ-insensitive "tube" (right). Samples $\mathbf{x}_i$ above the ϵ-tube have $\xi_1 > 0$ and $\xi_1 = 0$, points below the ϵ-tube have $\xi_2 = 0$ and $\xi_2 > 0$, and points inside the ϵ-tube have $\xi = 0$.

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

$$E_\epsilon(f(\mathbf{x}_i) - y_i) = \begin{cases} 0 & \text{if} \quad |f(\mathbf{x}_i) - y_i| < \epsilon \\ |f(\mathbf{x}_i) - y_i| - \epsilon & \text{otherwise} \end{cases} \tag{1.44}$$

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

$$\underset{\mathbf{w},b}{\text{argmin}} \left(\overbrace{\frac{1}{2}||\mathbf{w}||_2^2}^{\textit{Regularization}} + C \overbrace{\sum_{i=1}^{n}(\xi_{i_1} + \xi_{i_2})}^{\textit{Loss}} \right) \tag{1.45}$$

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

$$y_i - (\mathbf{w}.\mathbf{x}_i + b) \geq \epsilon - \xi_{i_1} \tag{1.46}$$

$$(\mathbf{w}.\mathbf{x}_i + b) - y_i \geq \epsilon - \xi_{i_2} \tag{1.47}$$

$$\xi_{i_1} \geq 0 \tag{1.48}$$

$$\xi_{i_2} \geq 0 \tag{1.49}$$

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

$$\underset{\boldsymbol{\alpha}}{\text{argmax}} \left(\sum_{i=1}^{n} y_i(\alpha_{i_1} - \alpha_{i_2}) - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_{i_1} - \alpha_{i_2})(\alpha_{j_1} - \alpha_{j_2})(\mathbf{x}_i.\mathbf{x}_j) \right) \tag{1.50}$$

$$\textbf{s.t.}\ \forall i = 1 \ldots n :$$

$$\sum_{i=1}^{n} \alpha_{i_1} = \sum_{i=1}^{n} \alpha_{i_2} \tag{1.51}$$

$$0 \leq \alpha_{i_1} \leq C \tag{1.52}$$

$$0 \leq \alpha_{i_2} \leq C \tag{1.53}$$

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:

$$f(\mathbf{x}_j) = \mathbf{w}^T \mathbf{x}_j + b \tag{1.54}$$

$$f(\mathbf{x}_j) = \sum_{i=1}^{n} (\alpha^*_{i_1} - \alpha^*_{i_2})(\mathbf{x}_i.\mathbf{x}_j) + b \tag{1.55}$$

$$f(\mathbf{x}_j) = \sum_{i=1}^{n} (\alpha^*_{i_1} - \alpha^*_{i_2}) K(\mathbf{x}_i.\mathbf{x}_j) + b \tag{1.56}$$

More informations about the calculation development can be found in [Bis06].

1.2.3 Other classification algorithms

Partie non encore rédigée. A faire à la fin.

- Positionner les travaux par rapport aux autres méthodes d'apprentissage supervisé
- 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 Conclusion of the chapter

This chapter has presented two machine learning algorithms used in our proposition: the k-Nearest Neighbors (k-NN) and the Support Vector Machine (SVM). We review the different steps in a machine learning framework: data normalization, model selection and model evaluation. In the following, we consider the k-NN as our classifier. The SVM will be used in our work for its large margin concept.

Our objective being the learning of a metric that optimizes the performances of the k-NN classifier, we review in the next section some metrics proposed for time series as well as metric learning concept for static data.

Chapter 2

Time series metrics and Metric Learning

Sommaire

In this chapter, we first present the definition of time series. Then, we recall the general properties of a metric and introduce some metrics proposed for time series. In particular, we focus on amplitude-based, behavior-based and frequential-based metrics. As real time series are subjected to varying delays, we recall the concept of alignment and dynamic programming. Then, we present some proposed combined metrics for time series. Finally, we review the concept of metric learning.

2.1 Definition of a time series

Time series are frequently data that 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. [PAN+08]; [Ram+08].

In **clustering** problems, one would like to organize similar time series together into homogeneous groups. In **classification** problems, the aim is to assign time series to one of several predefined categories (e.g., 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 and temperature sensors). Due to their temporal and structured nature, time series constitute complex data to be analyzed by classic machine learning approaches.

For physical systems, a time series of length T can be seen as a signal, sampled at a frequency f_e, in a temporal window $[0; \frac{T}{f_e}]$. From a mathematical perspective, a time series is a collection of a finite number of normalized observations made sequentially at discrete time instants $t = 1, ..., Q$. Note that when $f_e = 1$, $Q = T$.

Let $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{iQ})$ be a univariate time series of length Q. 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., the temperature measured by one sensor). When the observations are made at the same time from p variables (several sensors such as the temperature, the pressure, etc.), the time series is said multivariate. One possible representation is $\mathbf{x}_i = (\mathbf{x}_{i,1}, ..., \mathbf{x}_{i,p}) = (x_{i1,1}, ..., x_{iQ,1}, x_{i1,2}, ..., x_{i1,p}, ..., x_{iQ,p})$, where $\mathbf{x}_{i,j} = (\mathbf{x}_{i1,j}, \ldots, \mathbf{x}_{iQ,j})$. For simplification purpose, we consider in the following univariate time series.

Some authors propose to extract representative features from time series. Fig. 2.1 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).

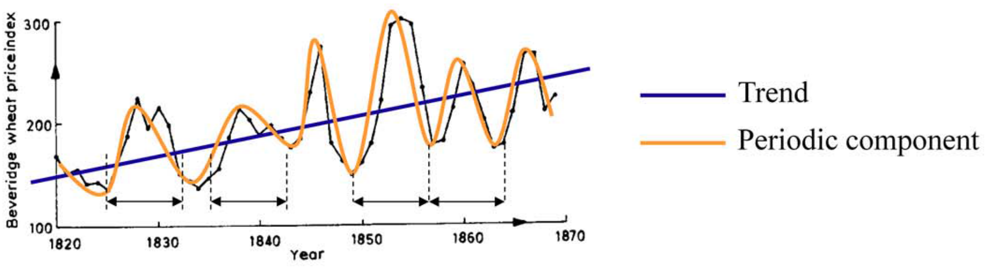


Figure 2.1: The Beveridge wheat price index is the average in nearly 50 places in various countries measured in successive years from 1500 to 1869. [1]

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. In practice, these 3 features are rarely sufficient for the classification or regression of real time series.

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

Other authors made the hypothesis of time independency between the observations x_{it}. They consider time series as a static vector data and use classic machine learning algorithms [Lia+12]; [CT01]; [HWZ13]; [HHK12]. Our work focus on classification and regression problems, and on time series comparison through metrics.

2.2 Properties of a metric

A mapping $D : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^+$ over a vector space $\mathbb{R}^p$ is called a metric or a distance if for all vectors $\forall \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_l \in \mathbb{R}^p$, it satisfies the properties:

Comment [CTD8]: je préfère garder l'espace pour + de visibilité

1. $D(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ (positivity)
2. $D(\mathbf{x}_i, \mathbf{x}_j) = D(\mathbf{x}_j, \mathbf{x}_i)$ (symmetry)
3. $D(\mathbf{x}_i, \mathbf{x}_j) = 0 \Leftrightarrow \mathbf{x}_i = \mathbf{x}_j$ (distinguishability)
4. $D(\mathbf{x}_i, \mathbf{x}_j) + D(\mathbf{x}_j, \mathbf{x}_l) \geq D(\mathbf{x}_i, \mathbf{x}_l)$ (triangular inequality)

A mapping D that satisfies at least properties 1, 2, 3 is called a dissimilarity, and the one that satisfies at least properties 1, 2, 4 a pseudo-metric. Note that for a metric, a dissimilarity and a pseudo metric, if a time series $\mathbf{x}_i$ is expected to be closer to $\mathbf{x}_j$ than to $\mathbf{x}_l$, then $D(\mathbf{x}_i, \mathbf{x}_j) \leq D(\mathbf{x}_i, \mathbf{x}_l)$. On the contrary, the mapping is called a similarity S when the time series $\mathbf{x}_i$ is expected to be closer to $\mathbf{x}_j$ than to $\mathbf{x}_l$ and then $S(\mathbf{x}_i, \mathbf{x}_j) \geq S(\mathbf{x}_i, \mathbf{x}_l)$. To simplify the discussion in the following, we refer to pseudo-metric and dissimilarity as metrics, pointing out the distinction only when necessary.

2.3 Unimodal metrics for time series

Defining and evaluating metrics for time series has become an active area of research for a wide variety of problems in machine learning [Din+08]; [Naj+12]. In the following, we suppose that time series have the same lengths Q and have been regularly sampled at the frequency f_e. Let $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{iQ})$ and $\mathbf{x}_j = (x_{i1}, x_{i2}, ..., x_{iQ})$ be two univariate time series of length Q.

A large number of distance measures have been proposed in the literature [MV14]. Contrary to static data, time series may exibit modalities and specificities due to their temporal nature (e.g., value, shape, frequency, delay, temporal locality). In this section, we review 3 categories of time series metrics used in our work: amplitude-based, frequential-based and behavior-based.

2.3.1 Amplitude-based metrics

The most usual comparison measures are amplitude-based metrics, where time series are compared in the temporal domain on their amplitudes regardless of their behaviors or frequential characteristics. Among these metrics, there are the commonly used Euclidean distance that compares elements observed at the same time [Din+08]:

$$d_E(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{t=1}^{Q}(x_{it} - x_{jt})^2} \tag{2.1}$$

Note that the Euclidean distance is a particular case of the Minkowski L_p norm ($p = 2$). An other amplitude-based metric is the Mahalanobis distance [PL12], defined as a dissimilarity measure between two random vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ of the same distribution with the covariance matrix $\mathbf{M}$:

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}^{-1} (\mathbf{x}_i - \mathbf{x}_j) \tag{2.2}$$

If the covariance matrix $\mathbf{M}$ is the identity matrix, the Mahalanobis distance is equal to the Euclidean distance. If the covariance matrix $\mathbf{M}$ is diagonal, then the resulting distance measure is called a normalized Euclidean distance:

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{l=1}^{Q} \frac{(x_{il} - x_{jl})^2}{m_l}} \tag{2.3}$$

where m_l is the variance of the x_{il} and x_{jl} over the sample set. In the following of the work, we consider the standard Euclidean distance d_E as the amplitude-based distance d_A.

In the example of Fig. 2.2, the aim is to determined which time series ($\mathbf{x}_2$ or $\mathbf{x}_3$) is the closest to $\mathbf{x}_1$. The amplitude-based distance d_A states that $\mathbf{x}_2$ is closer to $\mathbf{x}_1$ than $\mathbf{x}_3$ since $d_A(\mathbf{x}_1, \mathbf{x}_2) = 7.8816 < d_A(\mathbf{x}_1, \mathbf{x}_3) = 31.2250$.

2.3.2 Frequential-based metrics

The second category, commonly used in signal processing, relies on comparing time series based on their frequential properties (e.g. Fourier Transform, Wavelet, Mel-Frequency Cepstral Coefficients [SS12]; [TC98]; [BM67]). In our work, we limit the frequential comparison to Discrete Fourier Transform [Lhe+11], but other frequential properties can be used as well. Thus, for time series comparison, first the time series $\mathbf{x}_i$ are transformed into their Fourier representation $\tilde{\mathbf{x}}_i = [\tilde{x}_{i1}, ..., \tilde{x}_{iF}]$, with $\tilde{x}_{if}$ the complex component at frequential index f. The Euclidean distance is then used between their respective complex number modules $\tilde{x}_{if}$, noted $|\tilde{x}_{if}|$:

$$d_F(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{f=1}^{F}(|\tilde{x}_{if}| - |\tilde{x}_{jf}|)^2} \tag{2.4}$$

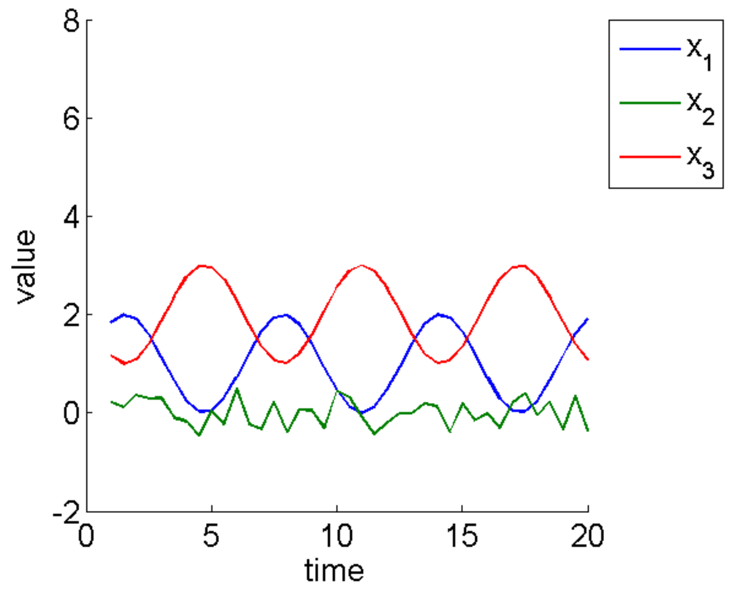


Figure 2.2: 3 toy time series. Time series in blue and red are two sinusoïdal signals. Time series in green is a random signal.

In the example of Fig. 2.2, the frequential-based distance d_F states that the time series $\mathbf{x}_3$ is closer to $\mathbf{x}_1$ than $\mathbf{x}_2$ since $d_F(\mathbf{x}_1, \mathbf{x}_3) = 0.8519 < d_F(\mathbf{x}_1, \mathbf{x}_2) = 0.9250$. This can be illustrated in the Frequency domain (Fig. ??)

Missing figure

Ajouter la représentation fréquentielle des signaux

2.3.3 Behavior-based metrics

The third category of metrics aims to compare time series based on their shape or behavior despite the range of their amplitudes. By time series of similar behavior, it is generally intended that for all temporal window $[t, t']$, they increase or decrease simultaneously with the same growth rate. On the contrary, they are said of opposite behavior if for all $[t, t']$, if one time series increases, the other one decreases and (vise-versa) with the same growth rate in absolute value. Finally, time series are considered of different behaviors if they are not similar, nor opposite. Many applications refer to the Pearson correlation [AT10]; [Ben+09] for behavior comparison. A generalization of the Pearson correlation is introduced in [DCA11]:

$$cort_r(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum (x_{it} - x_{it'})(x_{jt} - x_{jt'})}{\sqrt{\sum (x_{it} - x_{it'})^2}\sqrt{\sum (x_{jt} - x_{jt'})^2}} \tag{2.5}$$

where $|t-t'| \leq r$, $r \in [1, ..., T-1]$. The parameter r can be tuned or fixed a priori. It measures the importance of noise in data. For non-noisy data, low orders r is generally sufficient. For noisy data, the practitioner can either use de-noising data technics (Kalman or Wiener filtering [Kal60]; [Wie42]), or fix a high order r.

The temporal correlation *cort* computes the sum of growth rate between $\mathbf{x}_i$ and $\mathbf{x}_j$ between all pairs of values observed at $[t, t']$ for $t' \leq t+r$ (r-order differences). The value $cort_r(\mathbf{x}_i, \mathbf{x}_j) = 1$ means that $\mathbf{x}_i$ and $\mathbf{x}_j$ have similar behavior. The value $cort_r(\mathbf{x}_i, \mathbf{x}_j) = -1$ means that $\mathbf{x}_i$ and $\mathbf{x}_j$ have opposite behavior. Finally, $cort_r(\mathbf{x}_i, \mathbf{x}_j) = 0$ expresses that their growth rates are stochastically linearly independent (different behaviors).

When $r = T-1$, it leads to the Pearson correlation. As $cort_r$ is a similarity measure, it can be transformed into a dissimilarity measure:

$$d_B(\mathbf{x}_i, \mathbf{x}_j) = \frac{1 - cort_r(\mathbf{x}_i, \mathbf{x}_j)}{2} \tag{2.6}$$

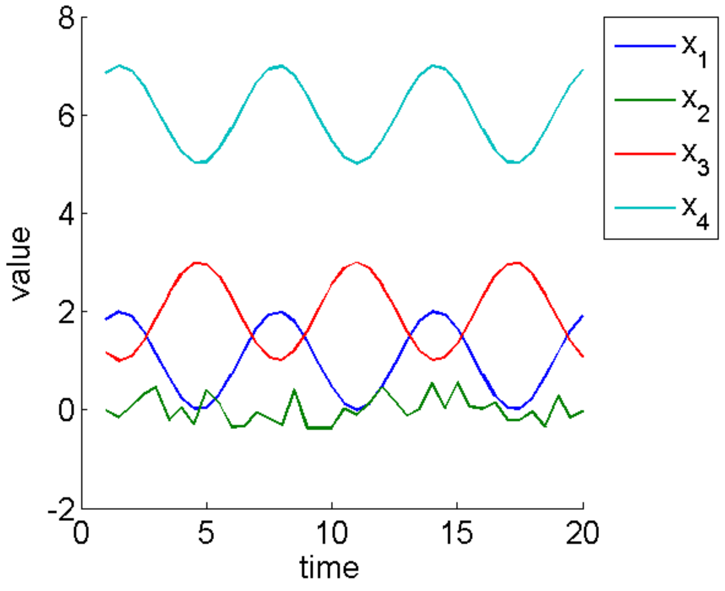


Figure 2.3: The signal from Fig. 2.2 and a signal $\mathbf{x}_4$ which is signal $\mathbf{x}_1$ and an added translation. Based on behavior comparison, $\mathbf{x}_4$ is the closest to $\mathbf{x}_1$.

Now considering Fig. 2.3

$$\begin{aligned} d_B(\mathbf{x}_1, \mathbf{x}_2) &= 0.477 \\ d_B(\mathbf{x}_1, \mathbf{x}_3) &= 1 \\ d_B(\mathbf{x}_1, \mathbf{x}_4) &= 0 \end{aligned}$$

2.3.4 Other metrics and Kernels for time series

A faire à la fin, pas urgent

- 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

2.4 Time series alignment and dynamic programming approach

In some applications, time series needs to be compared at different time t (i.e. energy data [Naj+12]) whereas in others, comparing time series on the same time t is essential (i.e. gene expression [DCN07]). When time series are asynchronous (i.e. varying delays or dynamic changes), they must be aligned before any analysis process. The asynchronous effects can be of various natures: time shifting (phase shift in signal processing), time compression or time dilatation. For example, in the case of voice recognition (Fig. 2.4), it is straightforward that a same sentence said by two different speakers will produce different time series: one speaker may speak faster than the other; one speaker may take more time on some vowels, etc.

To cope with delays and dynamic changes, dynamic programming approach has been introduced [BC94]. An alignment $\boldsymbol{\pi}$ of length $|\boldsymbol{\pi}| = m$ between two time series $\mathbf{x}_i$ and $\mathbf{x}_j$ of length T is defined as the set of m ($T \leq m \leq 2T-1$) couples of aligned elements of $\mathbf{x}_i$ to m elements of $\mathbf{x}_j$:

$$\boldsymbol{\pi} = ((\pi_i(1), \pi_j(1)), (\pi_i(2), \pi_j(2)), \ldots, (\pi_i(m), \pi_j(m))) \tag{2.7}$$

Comment [MR9]: Modifier figure. enlever 'one' et mettre la même échelle temporelle

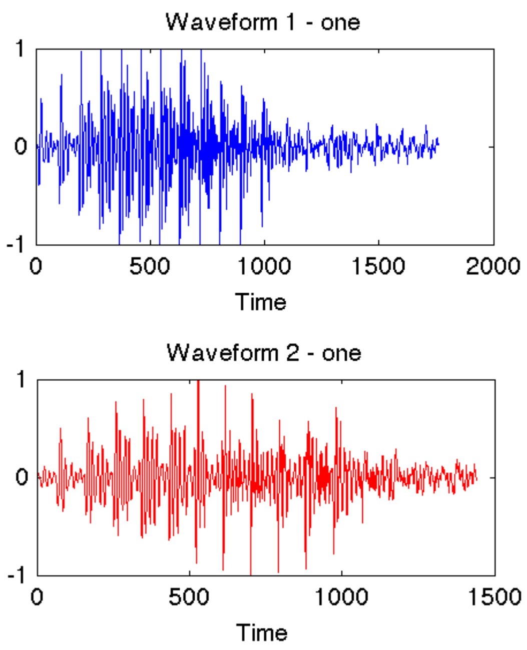


Figure 2.4: Example of a same sentence said by two different speakers. Time series are shifted, compressed and dilatated in the time.

where the applications π_i and π_j defined from $\{1, ..., m\}$ to $\{1, ..., T\}$ obey the following boundary monotonicity conditions:

Comment [AD10]: Ahla pas fan des notations

$$1 = \pi_i(1) \leq \pi_i(2) \leq ... \leq \pi_i(m) = T \tag{2.8}$$

$$1 = \pi_j(1) \leq \pi_j(2) \leq ... \leq \pi_j(m) = T \tag{2.9}$$

$\forall l \in \{1, ..., m\}$,

$$\pi_i(l+1) \leq \pi_i(l) + 1 \tag{2.10}$$

$$\text{and} \quad \pi_j(l+1) \leq \pi_j(l) + 1 \tag{2.11}$$

$$\text{and} \quad (\pi_i(l+1) - \pi_i(l)) - (\pi_j(l+1) - \pi_j(l)) \geq 1. \tag{2.12}$$

Intuitively, an alignment $\boldsymbol{\pi}$ defines a way to associate elements of two time series. Alignments can be described by paths in the $T \times T$ grid that crosses the elements of $\mathbf{x}_i$ and $\mathbf{x}_j$ (Fig. 2.5). We denote $\boldsymbol{\pi}$ a valid alignment and A, the set of all possible alignments between $\mathbf{x}_i$ and $\mathbf{x}_j$ ($\boldsymbol{\pi} \in A$). To find the best alignment $\boldsymbol{\pi}^*$ between two time series $\mathbf{x}_i$ and $\mathbf{x}_j$, the Dynamic Time Warping (DTW) algorithm has been proposed [KR04]; [SC].

DTW requires to choose a cost function φ to be optimised, such as a dissimilarity function (d_A, d_B, d_F, etc.). Classical DTW uses the Euclidean distance d_A (Eq. 2.1) as the cost function [BC94]. The warp path $\boldsymbol{\pi}$ is optimized for the chosen cost function φ:

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi} \in A}{\operatorname{argmin}} \frac{1}{|\boldsymbol{\pi}|} \sum_{(t,t') \in \boldsymbol{\pi}} \varphi(x_{it}, x_{jt'}) \tag{2.13}$$

When the cost function φ is a similarity measure, the optimization involves maximization instead of minimization. When other constraints are applied on $\boldsymbol{\pi}$, Eq. (2.13) leads to other variants of DTW (Sakoe-Shiba [SC78], Itakura parallelogram [RJ93]). Finally, the warped signals $\mathbf{x}_{i,\boldsymbol{\pi}}$ and $\mathbf{x}_{j,\boldsymbol{\pi}}$ are defined as:

$$\mathbf{x}_{i,\boldsymbol{\pi}} = (x_{i\pi_i(1)}, ..., x_{i\pi_i(m)}) \tag{2.14}$$

$$\mathbf{x}_{j,\boldsymbol{\pi}} = (x_{j\pi_j(1)}, ..., x_{j\pi_j(m)}) \tag{2.15}$$

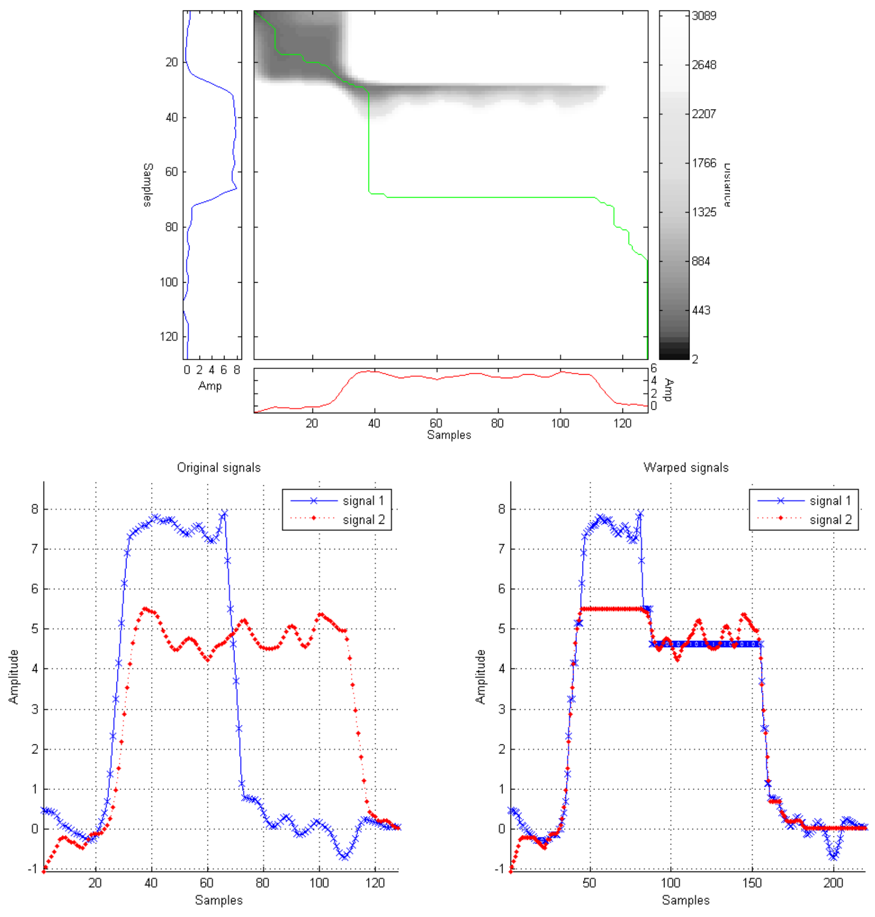


Figure 2.5: Example of DTW grid between 2 time series $\mathbf{x}_i$ and $\mathbf{x}_j$ (top) and the signals before and after warping (bottom). On the DTW grid, the two signals can be represented on the left and bottom of the grid. The optimal path $\boldsymbol{\pi}^*$ is represented in green line and show to associate elements of $\mathbf{x}_i$ to element of $\mathbf{x}_j$. Background show in grey scale the value of the considered metric (amplitude-based distance d_A in classical DTW)

The previous metric (amplitude-based d_A, behavior-based d_B) can be then computed on the warped signals $\mathbf{x}_{i,\boldsymbol{\pi}^*}$ and $\mathbf{x}_{j,\boldsymbol{\pi}^*}$. In the following, we suppose that the best alignment $\boldsymbol{\pi}^*$ is found. For simplification purpose, we refer $\mathbf{x}_{i,\boldsymbol{\pi}^*}$ and $\mathbf{x}_{j,\boldsymbol{\pi}^*}$ as $\mathbf{x}_i$ and $\mathbf{x}_j$.

2.5 Combined metrics for time series

In most classification problems, it is not known a priori if time series of a same class exhibits same characteristics based on their amplitude, behavior or frequential components alone. In some cases, several components (amplitude, behavior and/or frequential) may be implied.

A first technic considers a classifier for each p metric and combines the decision of the ref p resulting classifiers. This methods is referred as post-fusion , not considered in our work. Other propositions show the benefit of involving both behavior and amplitude components through a combination function. They combines the unimodal metrics together to obtain a single metric used after that in a classifier. This is called pre-fusion. The most classical combination functions combines the unimodal metrics (mainly d_A and d_B) through linear and geometric functions:

$$D_{Lin}(\mathbf{x}_i, \mathbf{x}_j) = \alpha d_B(\mathbf{x}_i, \mathbf{x}_j) + (1-\alpha) d_A(\mathbf{x}_i, \mathbf{x}_j) \tag{2.16}$$

$$D_{Geom}(\mathbf{x}_i, \mathbf{x}_j) = (d_B(\mathbf{x}_i, \mathbf{x}_j))^{\alpha} (d_A(\mathbf{x}_i, \mathbf{x}_j))^{1-\alpha} \tag{2.17}$$

where $\alpha \in [0;1]$ defines the trade-off between the amplitude d_A and the behavior d_B components, and is thus application dependent. In general, it is learned through a grid search procedure. Without being restrictive, these combinations can be extended to take into account more unimodal metrics.
More specific work on d_A and $cort$ propose to combine the two components through a sigmoid combination function [DCA11]:

$$D_{Sig}(\mathbf{x}_i, \mathbf{x}_j) = \frac{2 d_A(\mathbf{x}_i, \mathbf{x}_j)}{1 + \exp(\alpha cort_r(\mathbf{x}_i, \mathbf{x}_j))} \tag{2.18}$$

where α is a parameter that defines the compromise between behavior and amplitude components. When α is fixed to 0, the metric only includes the value proximity component. For $\alpha \geq 6$, the metric completely includes the behavior proximity component.

Fig.2.6 illustrates the value of the resulting combined metrics (D_{Lin}, D_{Geom} and D_{Sig}) in 2-dimensional space using contour plots for different values of the trade-off α. For small value of α ($\alpha = 0$), the three metrics only includes d_A. For high value of α ($\alpha = 1$), D_{Lin} and D_{Geom} only includes d_B. For $\alpha = 6$, D_{Sig} doesn't include completely $cort$. Note that these combinations are fixed and defined independently from the analysis task at hand. Moreover, in the case of D_{Sig}, only two variables are taking into account in these combined metrics and the component $cort_r$ can be seen as a penalizing factor of d_A. It doesn't represent a real compromise between value and behavior components. Finally, by adding metrics, the grid search to find the best parameters can become time consuming.

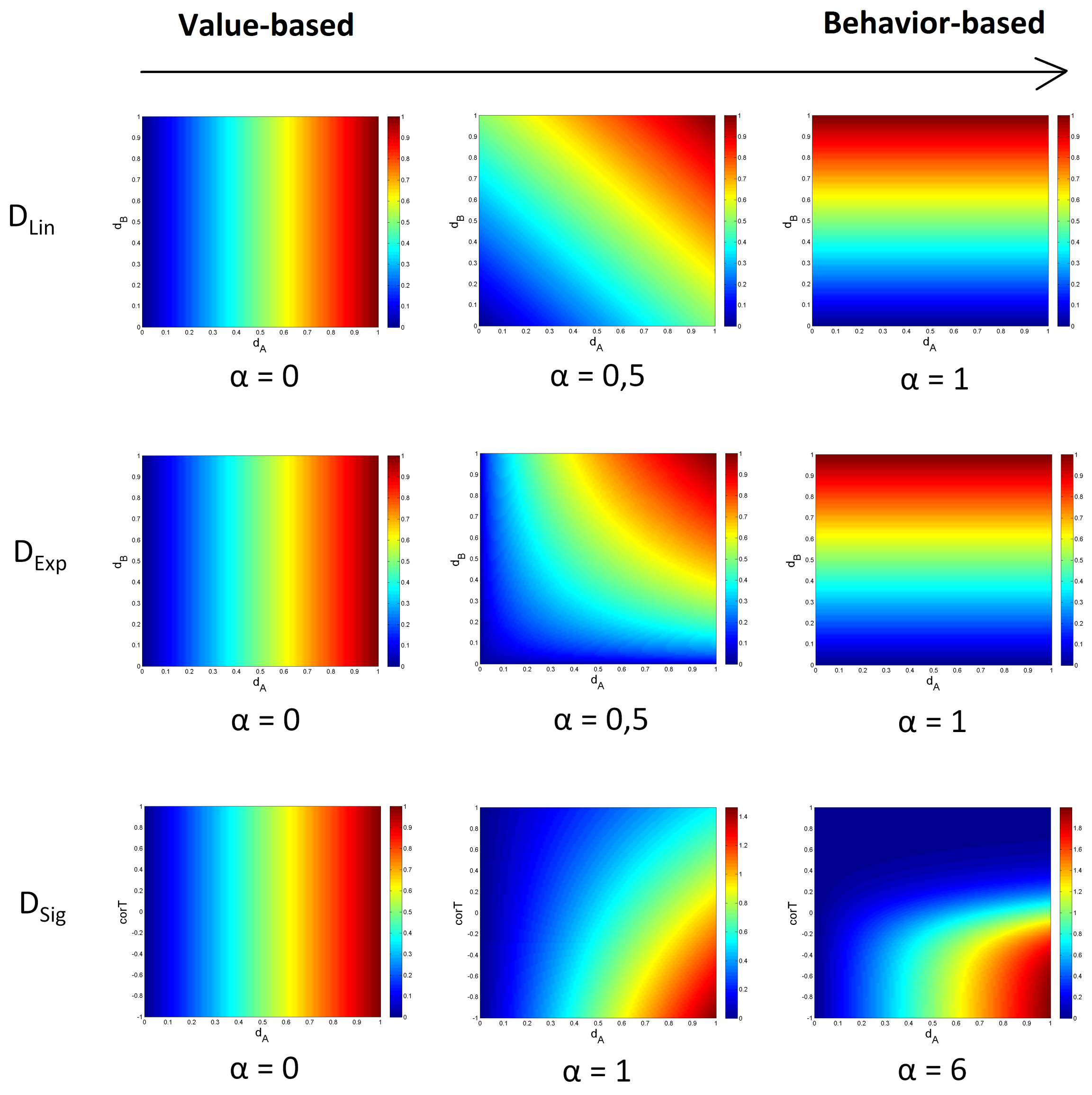


Figure 2.6: Contour plot of the resulting combined metrics: D_{Lin} (1st line), D_{Geom} (2nd line) and D_{Sig} (3rd line), for different value of α (D_{Sig}: $\alpha = 0; 1; 6$ and D_{Lin} and D_{Geom}: $\alpha = 0; 0.5; 1$). For D_{Sig}, the first and second dimensions are respectively the amplitude-based metrics d_A and the temporal correlation $corT$; for D_{Lin} and D_{Geom}, they correspond to d_A and the behavior-based metric d_B.

2.6 Metric learning

As our objective is to learn a metric in order to optimize the performance of the k-NN classifier, we review first metric learning concepts. Then, we focus on the framework proposed by

Weinberger & Saul for Large Margin Nearest Neighbor (LMNN) classification [WS09].

2.6.1 Review on metric learning work

In the case of static data, many work have demonstrated that k-NN classification performances depends highly on the considered metric and can be improved by learning an appropriate metric [She+02]; [Gol+04]; [CHL05]. Metric Learning can be defined as a process that aims to learn a distance from labeled examples by making closer samples that are expected to be similar, and far away those expected to be dissimilar.

A faire, avec papier PRL et papier Aurélien Bellet

2.6.2 Large Margin Nearest Neighbors (LMNN)

Let $\mathbf{X} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ be a set of N static vector samples, $\mathbf{x}_i \in \mathbb{R}^p$, p being the number of descriptive features and y_i the class labels. Weinberger & Saul proposed in [WS09] an approach to learn a dissimilarity metric D for a large margin k-NN in the case of static data.

Large Margin Nearest Neighbor (LMNN) approach is based on two intuitions: first, each training sample $\mathbf{x}_i$ should have the same label y_i as its k nearest neighbors; second, training samples with different labels should be widely separated. For this, the concept of **target** and **imposters** for each training sample $\mathbf{x}_i$ is introduced. The training sample $\mathbf{x}_i$ is referred as a **center point**. Target neighbors of $\mathbf{x}_i$, noted $j \rightsquigarrow i$, are the k closest $\mathbf{x}_j$ of the same class ($y_j = y_i$), while imposters of $\mathbf{x}_i$, denoted, $l \not\rightsquigarrow i$, are the $\mathbf{x}_l$ of different class ($y_l \neq y_i$) that invade the perimeter defined by the farthest targets of $\mathbf{x}_i$. Mathematically, for a sample $\mathbf{x}_i$, an imposter $\mathbf{x}_l$ is defined by an inequality related to the targets $\mathbf{x}_j$: $\forall l, \exists j \in j \rightsquigarrow i /$

$$D(\mathbf{x}_i, \mathbf{x}_l) \leq D(\mathbf{x}_i - \mathbf{x}_j) + 1 \tag{2.19}$$

Geometrically, an imposter $\mathbf{x}_l$ is a sample that invades the target neighborhood plus one unit margin as illustrated in Fig. 2.7. The target neighborhood is defined with respect to an initial metric. Without prior knowledge, L2-norm is often used. Metric Learning by LMNN aims to minimize the number of impostors invading the target neighborhood. By adding a margin of safety of one, the model is ensured to be robust to small amounts of noise in the training sample (large margin). The learned metric D pulls the targets $\mathbf{x}_j$ and pushes the imposters $\mathbf{x}_l$ as shown in Fig. 2.7.

LMNN approach learns a Mahalanobis distance D for a robust k-NN. We recall that the k-NN decision rule will correctly classify a sample if its k nearest neighbors share the same label (Section 1.2.1). The objective of LMNN is to increase the number of samples with this property by learning a linear transformation $\mathbf{L}$ of the input space ($\mathbf{x}_i = \mathbf{L}.\mathbf{x}_i$) before applying the k-NN classification:

$$D(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{L}(\mathbf{x}_i, \mathbf{x}_j)||_2^2 \tag{2.20}$$

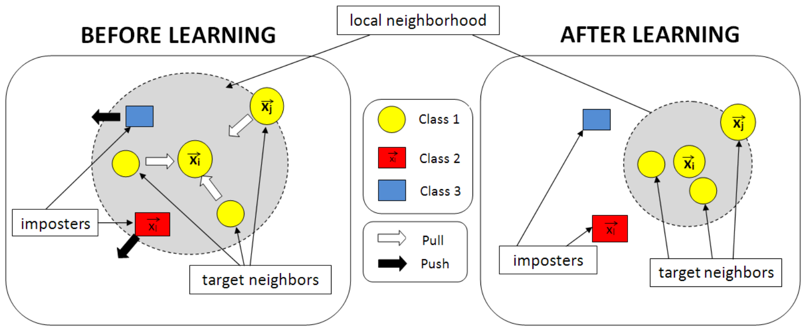


Figure 2.7: Pushed and pulled samples in the $k = 3$ target neighborhood of $\mathbf{x}_i$ before (left) and after (right) learning. The pushed (vs. pulled) samples are indicated by a white (vs. black) arrows (Weinberger & Sault [WS09]).

Commonly, the squared distances can be expressed in terms of the square matrix:

$$\mathbf{M} = \mathbf{L}'\mathbf{L} \tag{2.21}$$

It is proved that any matrix $\mathbf{M}$ formed as below from a real-valued matrix $\mathbf{L}$ is positive semidefinite (i.e., no negative eigenvalues) [WS09]. Using the matrix $\mathbf{M}$, squared distances can be expressed as:

$$D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)\mathbf{M}(\mathbf{x}_i - \mathbf{x}_j) \tag{2.22}$$

The computation of the learned metric $D_{\mathbf{M}}$ can thus be seen as a two steps procedure: first, it computes a linear transformation of the samples $\mathbf{x}_i$ given by the transformation $\mathbf{L}$; second, it computes the Euclidean distance in the transformed space:

$$D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = D^2(\mathbf{L}\mathbf{x}_i, \mathbf{L}\mathbf{x}_j) \tag{2.23}$$

Learning the linear transformation $\mathbf{L}$ is thus equivalent to learn the corresponding Mahalanobis metric D parametrized by $\mathbf{M}$. This equivalence leads to two different approaches to metric learning: we can either estimate the linear transformation $\mathbf{L}$, or estimate a positive semidefinite matrix $\mathbf{M}$. LMNN solution refers on the latter one.

Mathematically, it can be formalized as an optimization problem involving two competiting terms for each sample $\mathbf{x}_i$: one term penalizes large distances between nearby inputs with the same label (pull), while the other term penalizes small distances between inputs with different labels (push). For all samples $\mathbf{x}_i$, this implies a minimization problem:

$$\begin{aligned}
&\operatorname*{argmin}_{\mathbf{M},\xi} \underbrace{\sum_{i,j\rightsquigarrow i} D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)}_{pull} + C \underbrace{\sum_{i,j\rightsquigarrow i, l \not\rightsquigarrow i} \frac{1+y_{il}}{2} . \xi_{ijl}}_{push} \\
&\text{s.t. } \forall j \rightsquigarrow i, l \not\rightsquigarrow i, \\
&D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) - D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - \xi_{ijl} \\
&\xi_{ijl} \geq 0 \\
&\mathbf{M} \succeq 0
\end{aligned} \tag{2.24}$$

where C is a trade-off between the push and pull term and $y_{il} = -1$ if $y_i = y_l$ (same class) and $+1$ otherwise (different classes). Generally, the parameter C is tuned via cross validation and grid search. Similarly to Support Vector Machine (SVM) approach, slack variables ξ_{ijl} are introduced to relax the optimization problem.

2.6.3 Parallels between LMNN and SVM

Many connections can be made between LMNN and SVM: both are convex optimization problem based on a regularized and a loss term. In particular, Do & al. investigate this relationship and have shown that SVM can be formulated as a metric learning problem [Do+12]. The Mahalanobis distance $\mathbf{M}$ learned by LMNN can be expressed as a quadratic mapping $\boldsymbol{\phi}$. For a center point $\mathbf{x}_i$, for any sample $\mathbf{x}$, we have [Do+12]:

$$D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}) = D^2(\mathbf{L}\mathbf{x}_i, \mathbf{L}\mathbf{x}) \tag{2.25}$$
$$D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}) = \mathbf{w}_i^T \boldsymbol{\phi}(\mathbf{x}) + b_i \tag{2.26}$$

where $\mathbf{w}_i$ and b_i are the coefficient of the hyperplane H_i in the quadratic space $\boldsymbol{\phi}$.

Do & al. show that LMNN can be seen as a set of local SVM classifiers in the quadratic space induced by $\boldsymbol{\phi}$. For each center point $\mathbf{x}_i$, LMNN tries in its objective function to have its target neighbors $\mathbf{x}_j$ to have small value $\mathbf{w}_i^T \boldsymbol{\phi}(\mathbf{x}_j) + b_i$, i.e. be at the small distance from the hyperplane H_i. Minimizing the target neighbor distances from the hyperplane H_i makes the distance between support vectors and H_i small. Fig. 2.8 gives the equivalent point of view from the original space (Fig. 2.8(a)) into the quadratic space (Fig. 2.8(b)). The circle $\mathbf{C}_i$ with the center $\mathbf{L}\mathbf{x}_i$ in Fig. 2.8(a) corresponds to the hyperplane H_i in Fig. 2.8(b).

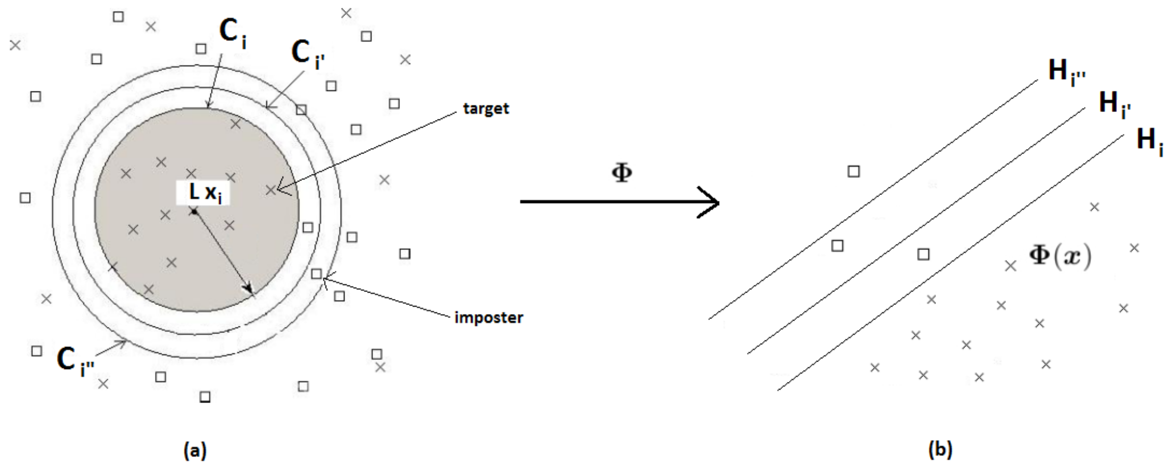


Figure 2.8: (a) Standard LMNN model view (b) LMNN model view under an SVM-like interpretation [Do+12]

Geometrically, SVM margin is defined globally with respect to a hyperplane, while LMNN margin is defined locally with respect to a center point $\mathbf{x}_i$. Fig. 2.9(a) illustrates the different local linear models in the quadratic space. The optimization process of LMNN combines the different local SVM hyperplane by bringing each point $\phi(\mathbf{x}_i)$ around a consensus hyperplane H.

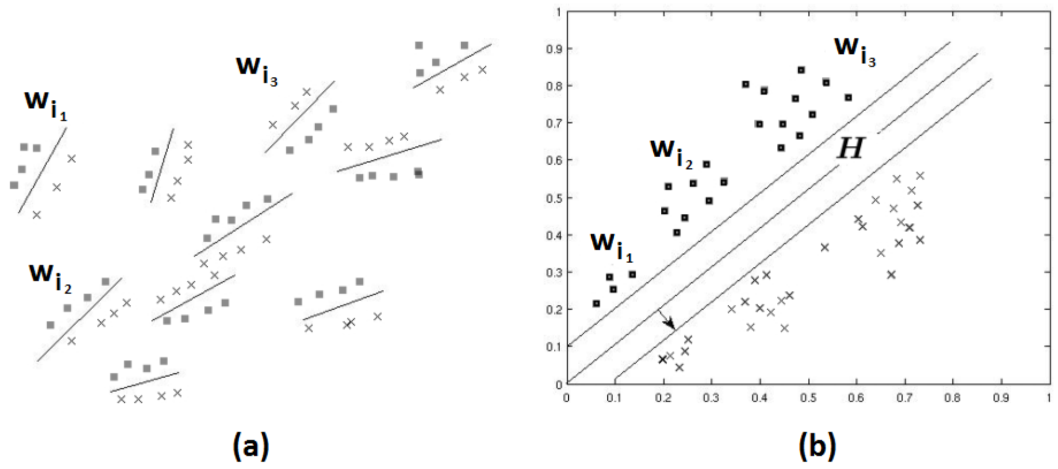


Figure 2.9: (a) LMNN in a local SVM-like view (b) LMNN in an SVM metric learning view [Do+12]

From these connections, some authors extends the LMNN approach to work in non-linear feature spaces by using the "kernel trick" . Finally, note that LMNN differs from SVM in which LMNN requires no modification for multiclass problems.

ref

2.7 Conclusion of the chapter

To cope with modalities inherent to time series (amplitude, behavior, frequency, etc.), we review in this chapter several unimodal metrics for time series, in particular, the Euclidean distance d_A, the Temporal correlation d_B or the Fourier-based distance d_F. In practice, real time series may be subjected to delays and need to be re-aligned before any analysis task. For that, the Dynamic Time Warping (DTW) algorithm is used in practice. However, these metrics (d_A, d_B, d_F) only include one modality. In general, several modalities may be implied and authors proposed to combine temporal metrics together. They mainly combine the Euclidean distance d_A and the Temporal correlation d_B.

As k-NN performances is impacted by the choice of the metric, other work propose in the case of static data to learn the metric in order to optimize the k-NN classification. In the following, we extend this framework to learn a combined metric for large margin k-NN classifier of time series.

Conclusion of Part I

In order to make the classification or regression of time series, a lot of technics exist in the literature. Our work focus on the k-NN classifier and the SVM will be used in the following for its large margin concept. We note that the k-Nearest Neighbors algorithm is based on the comparison of time series through distance measures.

Considering time series as static data lead to the only comparison based on their amplitude and the same time. To take into account other specificities of time series (behavior, frequential components), other metrics (e.g., the temporal correlation d_B, the frequential-based distance d_F, etc.) and other methods (Dynamic Time Warping DTW, dichotomy) have been proposed in the literature to cope with temporal characteristics.

Learning an adequate metric is a key challenge to well classify time series. Inspired by Metric Learning work for static data, we propose in the following a framework to learn a Multi-modal and Multi-scale Metric for a robust nearest neighbor classifier of time series.

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. 21).

[Alt92] Ns Altman. “An introduction to kernel and nearest-neighbor nonparametric regression.” In: *The American Statistician* 46.3 (1992), pp. 175–185 (cit. on p. 16).

[AT10] Z. Abraham and P.N. Tan. “An Integrated Framework for Simultaneous Classification and Regression of Time-Series Data.” In: *ACM SIGKDD*. 2010 (cit. on p. 35).

[BC94] Donald Berndt and James Clifford. “Using dynamic time warping to find patterns in time series.” In: *Workshop on Knowledge Knowledge Discovery in Databases* 398 (1994), pp. 359–370 (cit. on pp. 37, 39).

[Ben+09] J. Benesty et al. “Pearson correlation coefficient.” In: *Noise Reduction in Speech Processing* (2009) (cit. on p. 35).

[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. 17, 22).

[Bis06] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Vol. 4. 4. 2006, p. 738. arXiv: `0-387-31073-8` (cit. on pp. 8, 28).

[BM67] E. O. Brigham and R. E. Morrow. “The fast Fourier transform.” In: *Spectrum, IEEE* 4.12 (1967), pp. 63 –70 (cit. on p. 34).

[BMP02] Serge Belongie, Jitendra Malik, and Jan Puzicha. “Shape Matching and Object Recognition Using Shape Contexts.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (2002), pp. 509–522 (cit. on p. 16).

[CH67] T. Cover and P. Hart. “Nearest neighbor pattern classification.” In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27 (cit. on p. 15).

[Cha04] Christopher Chatfield. *The analysis of time series : an introduction*. 2004, xiii, 333 p. (Cit. on p. 32).

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. “Learning a similarity metric discriminatively, with application to face verification.” In: *CVPR*. Vol. 1. 2005, pp. 539–546 (cit. on p. 42).

[CHY96] Ming Syan Chen, Jiawei Han, and Philip S. Yu. *Data mining: An Overview from a Database Perspective*. 1996 (cit. on p. 8).

[Coc77] William C Cochran. “Snedecor G W & Cochran W G. Statistical methods applied to experiments in agriculture and biology. 5th ed. Ames, Iowa: Iowa State University Press, 1956.” In: *Citation Classics* 19 (1977), p. 1 (cit. on p. 12).

[CS01] Koby Crammer and Yoram Singer. “On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines.” In: *Journal of Machine Learning Research* 2 (2001), pp. 265–292 (cit. on p. 27).

[CT01] Lijuan Cao and Francis E H Tay. “Financial Forecasting Using Support Vector Machines.” In: *Neural Computing & Applications* (2001), pp. 184–192 (cit. on p. 33).

[CV95] Corinna Cortes and Vladimir Vapnik. “Support-vector networks.” In: *Machine Learning* 20.3 (1995), pp. 273–297. arXiv: `arXiv:1011.1669v3` (cit. on p. 17).

[CY11] Colin Campbell and Yiming Ying. *Learning with Support Vector Machines.* Vol. 5. 1. 2011, pp. 1–95 (cit. on pp. 17, 21).

[DCA11] A. Douzal-Chouakria and C. Amblard. “Classification trees for time series.” In: *Pattern Recognition journal* (2011) (cit. on pp. 36, 40).

[DCN07] A. Douzal-Chouakria and P. Nagabhushan. “Adaptive dissimilarity index for measuring time series proximity.” In: *Advances in Data Analysis and Classification* (2007) (cit. on p. 37).

[Den95] T. Denoeux. “A k-nearest neighbor classification rule based on Dempster-Shafer theory.” In: *IEEE Transactions on Systems, Man, and Cybernetics* 25.5 (1995), pp. 804–813 (cit. on p. 16).

[DHB95] Thomas G. Dietterich, Hermann Hild, and Ghulum Bakiri. “A comparison of ID3 and backpropagation for English text-to-speech mapping.” In: *Machine Learning* 18.1 (1995), pp. 51–80 (cit. on p. 12).

[Die97] T. Dietterich. “Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms.” In: (1997) (cit. on p. 12).

[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 pp. 16, 33, 34).

[Do+12] Huyen Do et al. “A metric learning perspective of SVM: on the relation of LMNN and SVM.” In: *Proceedings of the 15th International Con- ference on Artificial Intelligence and Statistics (AISTAS '12)* (2012), pp. 308–317. arXiv: `arXiv:1201.4714v1` (cit. on pp. 44, 45).

[Dud76] Sahibsingh a. Dudani. “DISTANCE-WEIGHTED k-NEAREST-NEIGHBOR RULE.” In: *IEEE Transactions on Systems, Man and Cybernetics* SMC-6.4 (1976), pp. 325–327 (cit. on p. 16).

[FCH08] RE Fan, KW Chang, and CJ Hsieh. “LIBLINEAR: A library for large linear classification.” In: *The Journal of Machine Learning* (2008) (cit. on p. 23).

[FRM94] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. “Fast subsequence matching in time-series databases.” In: *ACM SIGMOD Record* 23.2 (1994), pp. 419–429.

[Gol+04] Jacob Goldberger et al. “Neighbourhood Components Analysis.” In: *Advances in Neural Information Processing Systems* (2004), pp. 513–520 (cit. on p. 42).

[HCL08] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. “A Practical Guide to Support Vector Classification.” In: *BJU international* 101.1 (2008), pp. 1396–400. arXiv: `0-387-31073-8` (cit. on p. 14).

[HHK12] Seok Hwan Hwang, Dae Heon Ham, and Joong Hoon Kim. “Forecasting performance of LS-SVM for nonlinear hydrological time series.” In: *KSCE Journal of Civil Engineering* 16.5 (2012), pp. 870–882 (cit. on p. 33).

[HHP01] B Heisele, P Ho, and T Poggio. “Face recognition with support vector machines: global versus component-based approach.” In: *IEEE International Conference on Computer Vision, ICCV*. Vol. 2. July. 2001, pp. 688–694 (cit. on p. 17).

[HWZ13] Jianming Hu, Jianzhou Wang, and Guowei Zeng. “A hybrid forecasting approach applied to wind speed time series.” In: *Renewable Energy* 60 (2013), pp. 185–194 (cit. on p. 33).

[JMF99] a. K. Jain, M. N. Murty, and P. J. Flynn. “Data clustering: a review.” In: *ACM Computing Surveys* 31.3 (1999), pp. 264–323. arXiv: `arXiv:1101.1881v2` (cit. on p. 8).

[Kal60] R E Kalman. “A New Approach to Linear Filtering and Prediction Problems.” In: *Transactions of the ASME Journal of Basic Engineering* 82.Series D (1960), pp. 35–45 (cit. on p. 36).

[KGG85] James M. Keller, Michael R. Gray, and James a. Givens. *A fuzzy K-nearest neighbor algorithm.* 1985 (cit. on p. 16).

[KR04] Eamonn Keogh and Chotirat Ann Ratanamahatana. “Exact indexing of dynamic time warping.” In: *Knowledge and Information Systems* 7.3 (2004), pp. 358–386 (cit. on p. 38).

[KU02] B Kijsirikul and N Ussivakul. “Multiclass Support Vector Machines using Adaptive Directed Acyclic Graph.” In: *Neural Networks, 2002. IJCNN ’02. Proceedings of the 2002 International Joint Conference on* 1 (2002), pp. 980–985 (cit. on p. 27).

[Lhe+11] S. Lhermitte et al. “A comparison of time series similarity measures for classification and change detection of ecosystem dynamics.” In: *Remote Sensing of Environment* 115.12 (2011), pp. 3129–3152 (cit. on p. 34).

[Lia+12] Chunquan Liang et al. “Learning very fast decision tree from uncertain data streams with positive and unlabeled samples.” In: *Information Sciences* 213 (2012), pp. 50–67 (cit. on p. 33).

[MV14] Pablo Montero and José Vilar. “TSclust : An R Package for Time Series Clustering.” In: *Journal of Statistical Software November* 62.1 (2014) (cit. on p. 33).

[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 pp. 31, 33, 37).

[Ngu+12] L. Nguyen et al. “Predicting collective sentiment dynamics from time-series social media.” In: *WISDOM*. 2012 (cit. on p. 31).

[OE73] Richard O Duda and Peter E Hart. *Pattern Classification and Scene Analysis.* Vol. 7. 1973, p. 482 (cit. on pp. 8, 11, 16).

[PAN+08] COSTAS PANAGIOTAKIS et al. “SHAPE-BASED INDIVIDUAL/GROUP DETECTION FOR SPORT VIDEOS CATEGORIZATION.” In: *International Journal of Pattern Recognition and Artificial Intelligence* 22.06 (2008), pp. 1187–1213 (cit. on p. 31).

[PL12] Zoltán Prekopcsák and Daniel Lemire. “Time series classification by class-specific Mahalanobis distance measures.” In: *Advances in Data Analysis and Classification* 6.3 (2012), pp. 185–200. arXiv: `1010.1526` (cit. on p. 34).

[Ram+08] E. Ramasso et al. “Human action recognition in videos based on the transferable belief model : AAAApplication to athletics jumps.” In: *Pattern Analysis and Applications* 11.1 (2008), pp. 1–19 (cit. on p. 31).

[RJ93] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition.* Vol. 103. 1993 (cit. on p. 39).

[SC] Stan Salvador and Philip Chan. “FastDTW : Toward Accurate Dynamic Time Warping in Linear Time and Space.” In: () (cit. on p. 38).

[SC78] H. Sakoe and S. Chiba. “Dynamic Programming Algorithm Optimization for Spoken Word Recognition.” In: *IEEE transactions on acoustics, speech, and signal processing* (1978) (cit. on p. 39).

[She+02] Noam Shental et al. “Adjustment Learning and Relevant Component Analysis.” In: *European Conference on Computer Vision (ECCV)* 2353 (2002), pp. 776–790 (cit. on p. 42).

[SJ89] B W Silverman and M C Jones. “E. Fix and J.L. Hodges (1951): An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges (1951).” In: *International Statistical Review / Revue Internationale de Statistique* 57.3 (1989), pp. 233–238 (cit. on p. 15).

[SS12] Md Sahidullah and Goutam Saha. “Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition.” In: *Speech Communication* 54.4 (2012), pp. 543–565 (cit. on p. 34).

[SS13] Bernhard Schlkopf and Alexander J. Smola. *Learning with Kernels.* Vol. 53. 2013, pp. 1689–1699. arXiv: `arXiv:1011.1669v3` (cit. on pp. 17, 23).

[SSB03] Javad Sadri, Ching Y Suen, and Tien D. Bui. “Application of Support Vector Machines for recognition of handwritten Arabic/Persian digits.” In: *Second Conference on Machine Vision and Image Processing & Applications (MVIP 2003)* 1 (2003), pp. 300–307 (cit. on p. 17).

[TC98] Christopher Torrence and Gilbert P. Compo. “A Practical Guide to Wavelet Analysis.” In: *Bulletin of the American Meteorological Society* 79.1 (1998), pp. 61–78 (cit. on p. 34).

[Wan02] Jung-Ying Wang. “Support Vector Machines (SVM) in bioinformatics Bioinformatics applications.” In: *Bioinformatics* (2002), pp. 1–56 (cit. on p. 17).

[WS09] K. Weinberger and L. Saul. “Distance Metric Learning for Large Margin Nearest Neighbor Classification.” In: *Journal of Machine Learning Research* 10 (2009), pp. 207–244 (cit. on pp. 42, 43).

[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. 16).

[YG08] J. Yin and M. Gaber. “Clustering distributed time series in sensor networks.” In: *ICDM.* 2008 (cit. on p. 31).

[YL99] Yiming Yang and Xin Liu. “A re-examination of text categorization methods.” In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval SIGIR 99.* 1999, pp. 42–49 (cit. on p. 17).

[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. 8, 11).

[Wie42] Wiener N. *Extrapolation, Interpolation & Smoothing of Stationary Time Series - With Engineering Applications.* Tech. rep. Report of the Services 19, Research Project DIC-6037 MIT, 1942 (cit. on p. 36).

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