Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : **Mathématiques et Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Cao Tri Do

Thèse dirigée par **Ahlame Douzal,**
co-dirigée par **Michèle Rombaut** et
co-encadrée 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)**

Apprentissage de métriques multi-modales et multi-échelles pour la classification robuste de séries temporelles par plus proches voisins

Thèse soutenue publiquement le **6 mai 2016**,
devant le jury composé de :

Mr Stéphane Canu
Professeur des Universités, Université Normandie, Rapporteur

Mr Marc Sebban
Professeur des Universités, Université Hubert Curien, Rapporteur

Mr Patrick Gallinari
Professeur des Universités, Université Pierre et Marie Curie, Président

Mr Gustavo Camps-Valls
Associate Professor, Parc Científic Universitat de València, Examinateur

Mme Ahlame Douzal
Maître de conférences HDR, Université Grenoble Alpes, Directeur de thèse

Mme Michèle Rombaut
Professeur des Universités, Université Grenoble Alpes, Co-Directeur de thèse

Mr Sylvain Marié
Ingénieur de recherche, Schneider Electric, Co-Encadrant de thèse

UNIVERSITÉ DE GRENOBLE ALPES

ÉCOLE DOCTORALE MSTII

Institut Fourier, 100 rue des Mathématiques, 38400 Saint Martin d'Hères

T H È S E

pour obtenir le titre de

docteur en sciences

de l'Université de Grenoble Alpes

Spécialité : Informatique et Mathématiques

Présentée et soutenue par

Cao Tri Do

Multi-modal and Multi-scale Temporal Metric Learning for robust nearest neighbors classification

Thèse dirigée par Ahlame Douzal

préparée au Laboratoire d'Informatique de Grenoble (LIG)

soutenue le 6 mai 2016

Jury :

Rapporteurs :	Stéphane Canu	-	Laboratoire LITIS
	Marc Sebban	-	Laboratoire LAHC
Président :	Patrick Gallinari	-	Laboratoire LIP6
Examinateur :	Gustavo Camps-Valls	-	Laboratoire IPL
Directeur :	Ahlame Douzal	-	Laboratoire LIG
Co-Directeur :	Michèle Rombaut	-	Laboratoire GIPSA-Lab
Co-Encadrant :	Sylvain Marié	-	Schneider Electric

Acknowledgements

C'est au moment des remerciements que vient la plus grand peur, celle d'oublier quelqu'un ... Pour parer à cette éventualité, je commencerai par remercier toutes les personnes qui m'ont aidé et soutenu pendant ces années.

Je remercie tout d'abord mes directeurs de thèse, Ahlame, Michèle et Sylvain, qui m'ont encadré pendant ces 3 années de thèse, et même avant, dès mon stage de Master 2, pour m'avoir donné les connaissances en Machine Learning, la patience et la rigueur en recherche. Malgré les moments difficiles que nous avons pu rencontrer lors de la thèse, ils ont toujours su m'aider à tenir le bon cap pour arriver aujourd'hui à ce travail d'une excellente qualité. Un grand merci à Ahlame pour m'avoir suivi sur le plan scientifique en me guidant sur la rigueur de recherche et d'écriture; à Sylvain, pour nous avoir partagé ses idées bouillonnantes et sa motivation en tant qu'ingénieur et chercheur; et enfin à Michèle pour son encadrement, son écoute et ses conseils généraux qu'elle a pu me donner pendant ces 3 années.

Je remercie ensuite les membres du jury, Stéphane Canu et Marc Sebban. Je mesure la chance que j'ai de vous avoir comme rapporteurs. Je remercie à vous et aux autres membres, Patrick Gallinari et Gustavo Camps-Valls, pour vos remarques sur le manuscrit, les questions posées et l'intérêt que vous avez porté à ces travaux.

Je remercie également mes collègues de Gipsa, Laetitia, Lyuba, Stefen, Thibault, Jaume, Victor et leurs petit(e)s ami(e)s, Alexandre et Lilia, pour ces années passées avec vous dans le bureau 1131 où on a pu partager des bons moments de thèse et aussi des moments de doute. Votre soutien moral, nos discussions scientifiques et nos sortis Gipsa ont été pour moi un élément que je n'oublierai jamais de cette expérience !

Je n'oublie pas mes collègues de AMA qui cette fois sont nombreux (Saeid, Georgios, Adrian, Bikash, Simon, Irina, Jidong, Fabien et tous les autres) qui ont su m'accueillir alors que je n'ai pas pu être là très souvent. Votre gentillesse et les moments passés lors des séminaires au vert de l'équipe ont été des moments de partage que je garderai en souvenir.

De même, je remercie chacun des permanents des 2 équipes (très nombreux que je ne peux tous citer): Gipsa et AMA pour leur accueil, leurs conseils et leurs retours qu'ils ont pu m'apporter sur ma thèse pendant ces 3 années et qui m'ont permis de construire mon projet de thèse.

A tous mes collègues de Schneider, permanents (Daniel, Franck, Jean Louis, Vincent, Véronique, Henri, Olivier, Rodolph, Alfredo, Laurent, Patrick, Yvon, France, Yassine, Bartosz), doctorants (Peter, Chloé, Benoit, Thibaud), presta (Yvon, Pierre, Nelly, Romain, Sophie, Gregory), stagiaires (Léa, Blaise) et alternants (Matthieu, Thomas, Amadou, Omar) qui ont su me faire découvrir l'environnement Schneider pendant ces 4 ans au sein de l'équipe A4S. Un remerciement tout particulier à Didier, Claude et François pour nous avoir soutenu dans ce projet de thèse et pour nous avoir toujours fait des retours pertinents sur le plan

scientifique de la thèse. Et enfin, je n'oublierai jamais le A4S Band avec Matthieu et Peter où on a pu apporter la petite touche musicale de variété internationale dans Schneider. Il y a bien sûr tous les autres collègues de Schneider des autres équipes que je tenais à remercier pour ces années, notamment Benoît, qui a été à l'origine avec Patrice de ce projet.

Je remercie tous les amis qui sont venus de loin (Lyon, Avignon, Marseille, Montpellier, Pau, Paris et de la Suisse) pour le jour de la soutenance. Cela m'a beaucoup touché que vous ayez pu faire le déplacement. Un remerciement également à mes amis de Grenoble pour avoir pu partager ce moment avec vous, et un remerciement tout particulier à ma petite élève Candice et ses parents, Pascale et Claude, qui sont venus voir leur professeur de guitare présenté pour une fois autre chose que de la musique.

De manière générale, je remercie tous mes amis (français, québécois, allemands, vietnamiens, chinois, japonais, péruviens, espagnols), mes colocataires (Lauren, Laurence et Quentin) et ma famille qui ont contribué tout au long de ma thèse à veiller à son bon déroulement et à mes parents en particulier pour leur soutien moral et leur aide qui m'ont donné dans ces derniers jours avec le buffet de thèse et le déménagement.

Enfin, je terminerai ces remerciements à ma chère Louise, celle sans qui rien n'aurait pu être possible et qui m'a soutenu très largement pendant ces 3 ans !

Un grand merci à tous !

Contents

List of Figures

List of Tables

List 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	*Analytics for Solutions*
CIFRE	*Conventions Industrielles de Formation par la REcherche*
k-NN	*k-Nearest Neighbors*
SVM	*Support Vector Machines*
SVR	*Support Vector Regression*
KKT	*Karush-Kuhn-Tucker*
DTW	*Dynamic Time Warping*
SAX	*Symbolic Aggregate approXimation*
ARIMA	*AutoRegressive Integrated Moving Average*
ARMA	*AutoRegressive Moving Average*
IoT	*Internet of Things*
Acc	*Classification accuracy*
Err	*Classification error rate*
MAE	*Mean Absolute Error*
RMSE	*Root Mean Square Error*
MDS	*MultiDimensional Scaling*
FAQ	*Frequently Asked Questions*
LMNN	*Large Margin Nearest Neighbors*
M^{2}TML	*Multi-modal and Multi-scale Temporal Metric Learning*

Introduction

Motivation

The work of this PhD is in the context of a CIFRE[1] thesis with Schneider Electric and two public research laboratories, LIG[2] and GIPSA-lab[3]. Within Schneider Electric, the PhD took place in the Analytics for Solutions (A4S) team, part of the Technology and Strategy entity. Among the wide rage of activities of the A4S team, in the context of system modeling (*e.g.*, buildings, sensor networks, Internet of Things), two topics are at least studied: modeling by physical models (white/grey-box) and modeling by machine learning algorithms (black-box). With the increase of the amount of data and sensors that collect data, modeling accurately systems through *a priori* equations (white/grey-box) for some prediction tasks has become more and more difficult. Within the vast amount of applications in Schneider Electric, some applications involve in particular temporal data, *e.g.*, forecasting the energy consumption in a building, virtual sensors for industrial processes, fault detection and prediction for assets maintenance. More generally, Schneider Electric, like many other companies in various application domains (medicine, marketing, meteorology, etc.) has taken a growing interest these last decades in machine learning problems (classification, regression, clustering) that involve time series of one or several dimensions, of different samplings, etc. A time series can be seen in signal processing and in control theory as the response of a dynamic system. Contrary to static data, time series are more challenging in the sense that the temporal aspect (*i.e.*, order of appearance of the observations) is an additional key information.

Problem statement and contributions

In this work, we focus on classification problems of monovariate time series (data of 1 dimension) with a fixed sampling rate and of same fixed lengths. Among the wide variety of algorithms that exist in machine learning, some approaches (*e.g.*, k-Nearest Neighbors (k-NN)) classify samples using a concept of neighborhood based on the comparison between samples. In general, the concepts of 'near' and 'far' between samples is expressed through a distance measure. Time series can be compared based not only on their amplitudes like static data but also on other characteristics, called modalities, such as their dynamics or frequency components. Many metrics for time series have been proposed in the literature such as the Euclidean distance [Din+08], the temporal correlation [FDCG13], the Fourier-based distance [SS12a]. A detailed review of the major metrics is proposed in [MV14]. In general, existing metrics involve one modality (characteristic) at the global scale (*i.e.*, involving systematically

[1] Conventions Industrielles de Formation par la REcherche
[2] Laboratoire d'Informatique de Grenoble
[3] Grenoble Images Parole Signal Automatique

all the time series observations). We believe also that the multi-scale aspect of time series (*i.e.*, involving a temporal part of the observations), not present in static data, could enrich the definition of the existing metrics.

In this work, our objective is to learn a combined multi-modal and multi-scale time series metric for a robust k-NN classifier. The main contributions of the PhD are:

- The definition of a new space representation: the pairwise dissimilarity space where each pair of time series is embedded as a vector described by basic temporal metrics.
- The definition of basic temporal metrics that involve one modality at one specific scale.
- The learning of a multi-modal and multi-scale temporal metric for a large margin k-NN classifier of univariate time series.
- The definition of the general problem of learning a combined metric as a metric learning problem using the dissimilarity representation.
- The proposition of a framework based on Support Vector Machine (SVM) and a linear and non-linear solution to define the combined metric that satisfies at least the properties of a dissimilarity measure.
- The comparison of the proposed approach with standard metrics on a large number of public datasets.
- The analysis of the proposed approach to extract the discriminative features that are involved in the definition of the learned combined metric.

Organization of the manuscript

The first part makes a review of existing methods in machine learning and metrics for time series. The first chapter presents classical approaches in machine learning. We recall the general principle and framework in supervised learning and focus on two machine learning algorithms: the k-Nearest Neighbors (k-NN) and the Support Vector Machine (SVM). In the second chapter, we review some basic terminology for time series and recall at least three types of metrics proposed for time series: amplitude-, behavior- and frequential-based.
The second part of the manuscript proposes a Multi-modal and Multi-scale Temporal Metric Learning (M^2TML) approach for a robust k-NN classifier of time series. In the third chapter, we first review the concept of metric learning for static data and focus on a framework of metric learning for nearest neighbors classification proposed by Weinberger & Saul [WS09]. We then present a new space representation, the pairwise dissimilarity space based on a multi-modal and multi-scale time series description and their corresponding basic metrics. Then, we formalize the general M^2TML optimization problem using the pairwise dissimilarity space representation. From the general formalization, we propose at least three different formalizations. The first and second propositions involve different regularizers, allowing to

learn an *a priori* linear or non-linear form of the combined metric. The third proposition presents a framework based on SVM and a solution to build the combined metric, in the linear and non-linear context, satisfying at least the properties of a dissimilarity measure. Finally, Chapter 4 presents the experiments conducted on a wide range of 30 public and challenging datasets, and discusses the results obtained.

Notations

$\mathbf{x}_i$	a vector sample / a time series
y_i	a label (discrete or continous)
$\hat{y}_i$	a predicted label (discrete or continous)
$X = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$	a training set of $n \in \mathbb{N}$ labeled time series
$X_{Test} = \{\mathbf{x}_j, y_j\}_{j=1}^{m}$	a test set of $m \in \mathbb{N}$ labeled time series
$X_{op} = \{\mathbf{x}_l, y_l\}_{l=1}^{L}$	an operational set of $L \in \mathbb{N}$ labeled time series
d_E	Euclidean distance
d_A	amplitude-based distance
d_B	behavior-based distance
d_F	frequential-based distance
L_q	Minkovski q-norm
$\|\|\mathbf{x}\|\|_q$	q-norm of the vector $\mathbf{x}$
$cort$	temporal correlation
D	linear learned combined distance
$D_{\mathscr{H}}$	non-linear learned combined distance
κ	kernel function
$\phi(\mathbf{x}_i)$	embedding function from the original space to the Hilbert space
$\mathbf{x}_{ij}$	a pair of time series $\mathbf{x}_i$ and $\mathbf{x}_j$ in the pairwise dissimilarity space
y_{ij}	the pairwise label of $\mathbf{x}_{ij}$
t	time stamp/index with $t = 1, ..., q$
q	length of the time series (supposed fixed)
f	frequential index / function to learn
F	length of the Fourier transform
ξ	slack variable
p	number of metric measures considered in the metric learning process
r	order of the temporal correlation
k	number of nearest neighbors
C	hyper-parameter of the SVM (trade-off)
α	parameter to control the size of the neighborhood
λ	parameter to control the push term
$\mathbf{w}$	weight vector
v	number of folds in cross-validation
μ	parameter to control the overlap in the binary segmentation of the multi-scale description

Chapter 1

Related work

Contents

In this chapter, we recall some concepts of machine learning. First, we review the principles, 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 Machines (SVM).

1.1 Classification, Regression

In this section, we review some terminology used 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 known 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 [Dre+06].

Let $X = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$ be a training set of n vector samples $\mathbf{x}_i \in \mathbb{R}^p$ and y_i their corresponding labels. The aim of supervised machine learning is to learn a relationship (model) f between the samples $\mathbf{x}_i$ and their labels y_i based on examples [Bis06]; [Dre+06]; [DH73]. 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 $\hat{y}_j$ 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. For both problems, when a part of the labels y_i are known and another part of y_i is unknown during training, learning f is a semi-supervised problem [Zhu07]. Note that when the labels y_i are totally unknown, learning f refers to a clustering problem (unsupervised learning) [JMF99]; [CHY96]. Semi-supervised and unsupervised learning problems are out of the scope of this work.

1.1.2 Model selection in supervised learning

A key objective of supervised learning algorithms is to build models f with good generalization capabilities, *i.e.*, models f that correctly predict the labels y_j of new unknown samples $\mathbf{x}_j$. There exist 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 situation is known as model overfitting (Fig. 1.1). In general, the complexity in learning can be measured through 2 measures: the information complexity and the computational complexity. The information complexity concerns the generalization performances of the learner: how many samples are needed? How much time the learner will take to converge to its optimal solution? Etc. The computational complexity deals with the computational resources needed to make a new prediction based on the training data.

In most cases, learning algorithms require to tune some hyper-parameters. A first approach could consist in trying all the possible combinations of hyper-parameters values and keep the one with the lowest training error. However, as discussed above, the model with the lowest training error is not always the one with the best generalization error. To avoid overfitting, the training set can be divided into 2 sets: a learning and a validation set. Suppose that 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. 1.2). For each combination (a cell of the grid), the model is learned on the learning set and evaluated on the

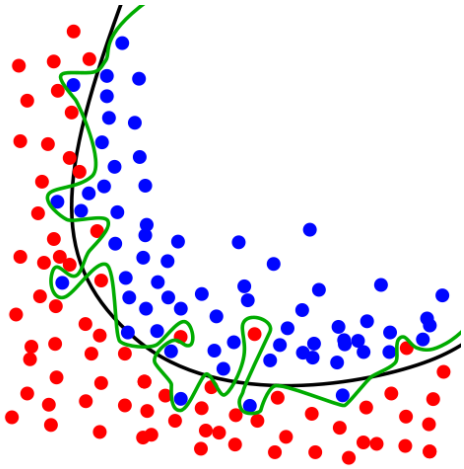


Figure 1.1: An example of overfitting in the case of classification. The objective is to separate blue points from red points. Black line shows a classifier f_1 with low complexity where as green line illustrates a classifier f_2 with high complexity. On training examples (blue and red points), the model f_2 separates all the classes perfectly but may lead to poor generalization on new unseen examples. Model f_1 is often preferred.

validation set. At the end, the hyper-parameters with the lowest error on the validation set are retained and the model f is learned on all training data using these optimal hyper-parameters. This process is referred to as the **model selection**.

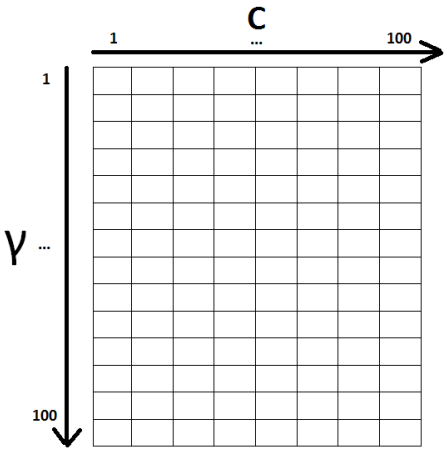


Figure 1.2: Example of a 2 dimensional grid search for parameters C and γ. It defines a grid where each cell of the grid contains a combination (C, γ). Each combination is used to learn the model and is evaluated on the validation set.

An alternative is **cross-validation** with v folds, illustrated in Fig. 1.3. 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

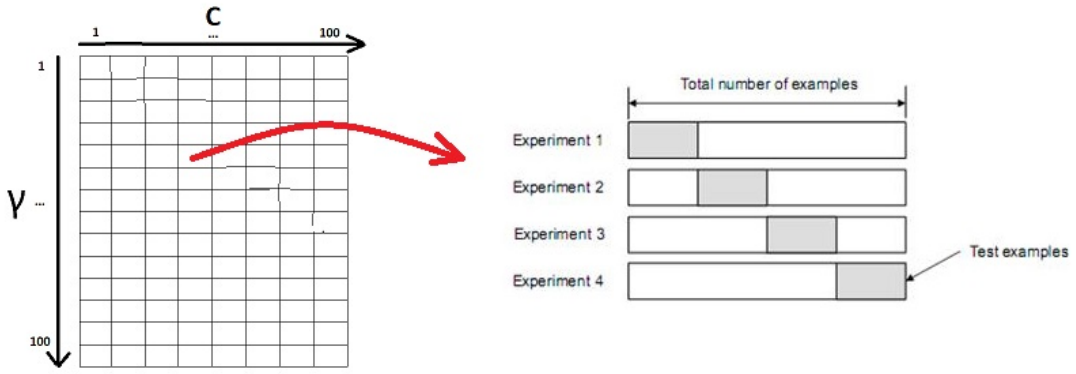


Figure 1.3: v-fold Cross-validation for one combination of parameters. For each of v experiments, use $v-1$ folds for training and a different fold for Testing, then the training error for this combination of parameter is the mean of all testing errors. This procedure is illustrated for $v = 4$.

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 that as much data as possible are used 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 of the training set, is small. There exist other methods such as sub-sampling or bootstrap [DH73]; [Dre+06]. We only use cross-validation in our experiments.

To sum up, Fig. 1.4 shows a general approach for solving machine learning problems. In general, a dataset can be divided into 3 sub-datasets (illustrated in Fig. 1.5):

- A **training set** $X = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$, which consists 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 requires some hyper-parameters to be tuned, there exists a risk of model overfitting. To avoid such risk, the training set X has to be divided into two subsets :
  - A **learning set** which is used to build the supervised model f for each value of the hyper-parameters.
  - A **validation set** which is used to evaluate the supervised model f for each value of the hyper-parameters. The model f with the lowest error on the validation set is kept, thus ensuring that it has the best generalization abilities.
- 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 are not used during the training step. The model f is applied to predict the

label $\hat{y}_j$ of samples $\mathbf{x}_j$ to evaluate the performance of the learnt model by comparing $\hat{y}_j$ 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.

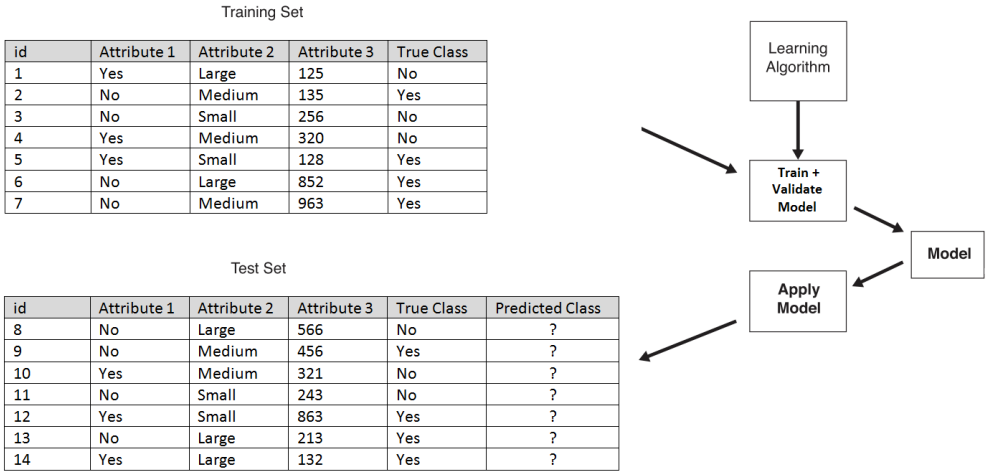


Figure 1.4: General framework for building a supervised (classification/regression) model. Example with 3 features and 2 classes ('Yes' and 'No').

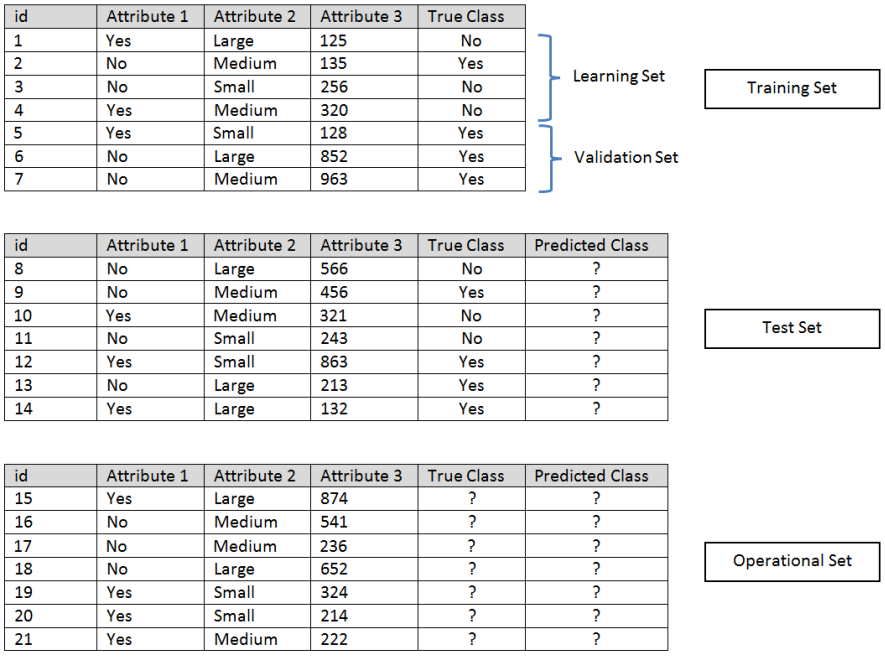


Figure 1.5: Division of a dataset into 3 datasets: training, test and operational.

1.1.3 Model evaluation

As seen in the previous section, model selection is inherently based on the ability to quantify its error on the validation set. In this section, we recall how this error is computed for classification and regression problems.

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 g_{ij} of 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 $\sum_{i=1}^{C} g_{ii}$, where C is the number of classes.

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

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

For binary classification problems, g_{11} is the number of true positives, g_{10} is the number of false negatives, g_{01} is the number of false positives and g_{00} is the number of true negatives.

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

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

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

Using these performance metrics allows one to compare the performance of different classifiers f. It allows one to determine in particular whether one learning algorithm outperforms another on a particular learning task on a given test set 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 [Die98]; [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 m_A (resp. m_B) of misclassification of m test examples made by classifier f_A (resp. f_B) 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 with a mean $m(p_A - p_B)$. Thus, the quantity $m_A - m_B$ is a normally distributed random variable. Under the null hypothesis (the two algorithms 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

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 L_1-norm loss.

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

A commonly used performance metrics is the Root Mean Squared Error function ($RMSE$) that computes the root of the mean square error:

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

Many works rely on the R^2 measure, the coefficient of determination [Nag91]. It provides

a measure of how well future samples are likely to be predicted by the model[1]. It can also be interpreted as a measure of how the model f is better than a constant model.

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

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 subject to uneven scaling, noise, outliers, etc. Before applying any learning protocol, it is often necessary to pre-process the data: data scaling, data filtering (*e.g.*, de-noising), outlier removal, etc. We focus on data normalization in this work.

Let $X = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$ be a training set, $\mathbf{x}_i$ being a sample described by p attributes $x_1, \ldots, x_p$. Part 2 of Sarle's Neural Networks FAQ (1997) [2] explains the importance of data normalization for neural networks 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 or Z-normalization [MU13]. Let μ_j and σ_j as the mean and the standard deviation of an attribute x_j, applying the Z-normalized attribute x_j^{norm} is given by:

$$x_j^{norm} = \frac{x_j - \mu_j}{\sigma_j} \tag{1.8}$$

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. Secondly, training and testing data must not be scaled separately. Thirdly, the whole dataset must not be scaled together at the same time as it often leads to poorer results. A proper way to do normalization is to scale the training data, store the parameters of the normalization (*e.g.*, μ_i and σ_i for Z-normalization), then apply the same normalization parameters to the testing data.

[1] `http://scikit-learn.org/stable/modules/model_evaluation.html`
[2] `http://www.faqs.org/faqs/ai-faq/neural-nets/`

1.2 Machine learning algorithms

Many algorithms have been proposed in the context of supervised learning, such as Deep Neural Networks, Decision Trees, k-Nearest Neighbors (k-NN) or Support Vector Machine (SVM). In Decision Trees, the aim is to build a decision tree by recursively partitioning the sample space [Qui86]. The tree consists of nodes that split the sample space into sub-spaces, and leaf nodes that are associated to classes. In Deep Neural Networks, most of the propositions aim to learn a representation of the data by extracting features using a cascade of layers [Lee+09], then to use a neural network algorithm to learn a model from the learned features. One main advantage of Decision Trees over Deep Neural Networks is that by using the features from the sample space, the model is still interpretable.
In the following of the work, the k-Nearest Neighbors (k-NN) will be considered as our classifier. The Support Vector Machine (SVM) will be used for its large margin concept, a key part of one of our algorithms. In this section, we focus and detail these two approaches.

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 use the class y_i of its nearest neighbor $\mathbf{x}_i$ in the training set in order to predict its labels: $\hat{y}_j = y_i$.
More generally, we can consider the k nearest neighbors of $\mathbf{x}_j$. The class y_j of a 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 referred to as the k-Nearest Neighbors algorithm (k-NN) [SJ89]; [CH67]. Fig. 1.6 illustrates the concept for a neighborhood of $k = 3$ and $k = 5$.

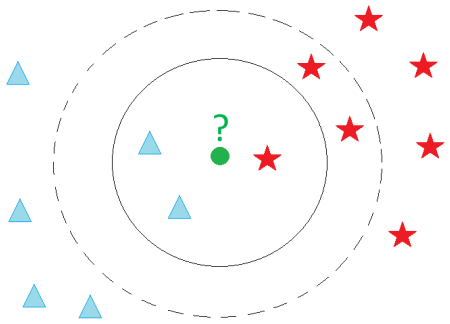


Figure 1.6: Example of k-NN classification. The test sample (green circle) is classified either to the first class (red stars) or to the second class (blue triangles). If $k = 3$ (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 star inside the inner circle. If $k = 5$ (dashed line circle) it is assigned to the first class (3 stars vs. 2 triangles inside the outer circle).

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, frequently used metrics are the Euclidean

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

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 following optimization problem: for a new sample $\mathbf{x}_j$, $\forall i \in \{1, \ldots, n\}$,

$$y_j = y_{i^*} \tag{1.9}$$

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

The k-NN algorithm can be extended to estimate continous labels (regression problems). In that case, the label y_j is defined as :

$$y_j = \frac{1}{k} \sum_{i=1}^{k} y_i \tag{1.10}$$

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 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 some variants such as the fuzzy k-NN or the belief k-NN. In a fuzzy k-NN, the membership degree in each class of an unseen sample $\mathbf{x}_j$ is obtained by combining the memberships of its neighbors [KGG85]. In a belief k-NN, the approach relies on the Dempster-Shafer theory to modify the belief concerning the class membership of a pattern, and quantify the uncertainty attached to each sample [Den95].

Despite its implementation simplicity, the k-NN algorithm has been shown to be successful on time series classification problems [BMP02]; [Xi+06]; [Din+08]. Ding & al. in [Din+08] presents the benefice of using a framework based on 1-NN classifier to evaluate the performance of metrics for time series. He states that accuracy evaluations should answer the question: why is this a good measure for describing the (dis)similarity between time series? First, the underlying distance metric is critical to the performance of 1NN classifier . Thus, the accuracy of the 1-NN classifier directly reflects the effectiveness of the metric. Second, 1-NN classifier is easy to implement and doesn't need to learn any hyper-parameters, which make it straightforward for anyone to reproduce results. Other methods to compare metrics exists such as clustering with small data sets which are not statistically significant, or compare the compactness of the metric . Third, it has been proved that the error ratio of 1NN classifier is at most twice the Bayes error ratio.
However, the k-NN algorithm presents some disadvantages, mainly due to its computational complexity, both in memory space (storage of the training samples $\mathbf{x}_i$) and time (search of the neighbors) [DH73]. Suppose that 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 $\mathbf{x}_i$ ($i = 1, \ldots, n$) one by one, calculate its distance to $\mathbf{x}_j$ ($D(\mathbf{x}_i,\mathbf{x}_j)$) 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 $O(pn)$. Finally, note that using standard metrics

refer to all of them as metrics.

(such as the Euclidean distance) in the k-NN relies on all p dimensions in the computation of the metric and thus assumes that all dimensions have the same effect on the metric. This assumption may be wrong and may impact the classification performances.
To overcome these limitations (memory space, computation complexity), some authors proposed algorithms that structures the data: Ball Tree, k-d Tree, etc. In a ball tree, the aim is to build a binary tree that improves the kNN in term of speed. Each node of the tree partitions the data into two disjoint sets which are associated to different hyperspheres, called balls. While the balls themselves may intersect, each sample is assigned to one or the other ball in the partition according to its distance from the ball's center. Each leaf node in the tree defines a ball and enumerates all samples inside that ball. In a k-d Tree, the aim is to divide the training data into two parts, right node and left node. Left or right side of tree is searched according to query records. After reaching the terminal node, records in terminal node are examined to find the closest data node to query record. A more detailed reviews of other algorithms to improve the kNN speed can be found in [BA10].

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 has demonstrated 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 thus allows non-linear problems to be solved. This section gives a brief overview of the mathematical key points and interpretation of the method. For more information, 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 a 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 equation[4] is $\mathbf{w}^T\mathbf{x} + b = 0$, that can separate samples of class +1 from the ones of class -1. When the problem is linearly separable such as in Fig. 1.7, there exists an infinite number of valid hyperplanes.

Cortes & Vapnik [CV95] propose to choose the separating hyperplane that maximizes the margin, *i.e.*, the hyperplane that leaves as much distance as possible between the hyperplane

[4] $\mathbf{w}^T$ denotes the transpose of the vector $\mathbf{w}$

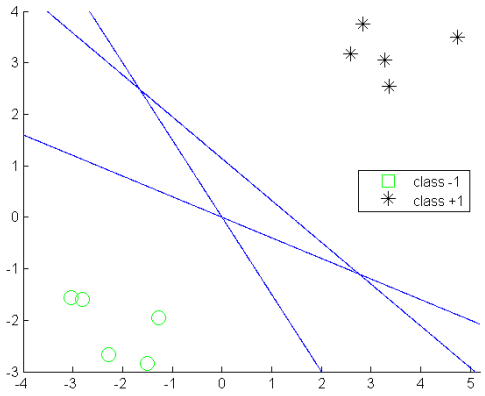


Figure 1.7: Example of linear classifiers (blue lines) in a 2-dimensional classification problem. For a set of samples of classes +1 (stars) and -1 (circles) that are linearly separable, there exists an infinite number of separating hyperplanes corresponding to $\mathbf{w}^T\mathbf{x} + b = 0$.

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 L_2-norm of the vector $\mathbf{w}$ and $||\mathbf{w}||_1$ the L_1-norm of $\mathbf{w}$:

$$||\mathbf{w}||_2 = \sqrt{\mathbf{w}^T\mathbf{w}} = \sqrt{\sum_{h=1}^{p} w_h^2} \tag{1.11}$$

$$||\mathbf{w}||_1 = \sum_{h=1}^{p} |w_h| \tag{1.12}$$

where $\mathbf{w} = [w_1, \ldots, w_p]^T$ denotes the weight vector.
The two hyperplanes passing through the support vectors of each class are referred to as the **canonical hyperplanes**, and the region between the canonical hyperplanes is called the **margin band** (Fig. 1.8). From this, for a binary classification problem, to classify a new sample $\mathbf{x}_j$, the decision function is:

$$f(\mathbf{x}_j) = sign(\mathbf{w}^T\mathbf{x}_j + b) \tag{1.13}$$

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:

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \frac{1}{2}||\mathbf{w}||_2^2 \tag{1.14}$$

$$\textbf{s.t. } \forall i = 1, \ldots, n: \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \tag{1.15}$$

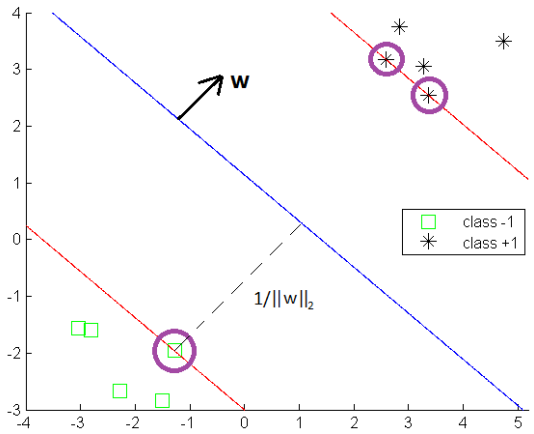


Figure 1.8: The argument inside the decision function of a SVM classifier is $\mathbf{w}^T\mathbf{x}+b$. The separating hyperplane corresponding to $\mathbf{w}^T\mathbf{x}+b=0$ is shown as a blue line in this 2-dimensional plot. This hyperplane separates the two classes of data with points on one side labeled $y_i=+1$ ($\mathbf{w}^T\mathbf{x}_i+b\geq 0$) and points on the other side labeled $y_i=-1$ ($\mathbf{w}^T\mathbf{x}_i+b<0$). Support vectors are circled in purple and lie on the hyperplanes $\mathbf{w}^T\mathbf{x}+b=+1$ and $\mathbf{w}^T\mathbf{x}+b=-1$ (red lines)

This is a constrained optimization problem in which we minimize an objective function (Eq. 1.14) subject to constraints (Eq. 1.15). This formulation is referred to 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:

$$\underset{\mathbf{w},b,\xi}{\operatorname{argmin}} \left(\overbrace{\frac{1}{2}||\mathbf{w}||_2^2}^{Regularization} + C \overbrace{\sum_{i=1}^{n} \xi_i}^{Loss} \right) \tag{1.16}$$

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

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \tag{1.17}$$

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

where $C>0$ is a trade-off hyper-parameter. This formulation is referred to as the **primal soft margin problem**. It is a quadratic programming optimization problem subject to constraints. Thus, it is a convex problem: any local solution is a global solution. The objective function in Eq. 1.16 is made of two terms. The first one, the regularization term, penalizes the complexity of the model, controlling 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 $+\infty$, all the slack variables ξ_i have to be equal to zero in order to not have an infinite loss term. The problem is thus equivalent to the primal hard margin problem. The hyper-parameter C is learnt during the training phase (Section 1.1.2).

1.2.2.c Dual formulation

From the primal formulation, 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 Eqs. 1.14 and 1.15. As a constrained optimization problem, the formulation is equivalent to the maximization of a Lagrange function $L(\mathbf{w}, b)$, consisting of the sum of the objective function (Eq. 1.14) and the n constraints (Eq. 1.15) multiplied by their respective Lagrange multipliers $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^T$:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \left(L(\mathbf{w}, b) = \frac{1}{2}(\mathbf{w}^T\mathbf{w}) - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) \right) \tag{1.19}$$

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

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

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0 \tag{1.21}$$

$$\alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) = 0 \tag{1.22}$$

where $\alpha_i \geq 0$ are the Lagrange multipliers. In optimization theory, Eqs. 1.20, 1.21 and 1.22 are called the Karush-Kuhn-Tucker (KKT) conditions [Bis06]. 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 maximum value of $L(\mathbf{w}, b)$, 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:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{1.23}$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{1.24}$$

By substituting $\mathbf{w}$ into $L(\mathbf{w}, b)$ in Eq. 1.19, we obtain the **dual formulation** (*Wolfe dual*):

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) \right) \tag{1.25}$$

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

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

$$\alpha_i \geq 0 \tag{1.27}$$

The dual objective in Eq. 1.25 is quadratic in the parameters α_i. Adding the constraints in Eqs. 1.26 and 1.27, it is a constrained quadratic programming optimization problem (QP). Note that while the primal formulation is a minimization problem, the equivalent dual formulation is a maximization problem. It can be shown that the objective functions of both formulations (primal and dual) reach the same value when the solution is found [CY11].
In the same spirit, it can be shown that the soft margin primal problem leads to the same formulation to the ones in Eqs. 1.25 and 1.26, except that the Lagrange multipliers α_i are upper bounded by the trade-off C in the soft margin formulation [CY11]:

$$0 \leq \alpha_i \leq C \tag{1.28}$$

The constraints in Eq. 1.28 are called the Box constraints. 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:

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \tag{1.29}$$

$$b^* = \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i) \tag{1.30}$$

At the optimality point, Eq. 1.22 leads $\alpha_i^* = 0$ for all datapoints that are well classified and that are not on the margin. Hence, only a few number of datapoints have $\alpha_i^* > 0$ as shown as in Fig. 1.9. These samples are the support vectors. All other datapoints have $\alpha_i^* = 0$, and the decision function is independent of them. Thus, the representation is said sparse.

From Eqs. 1.13 & 1.29, to classify a new sample $\mathbf{x}_j$, the decision function for a binary classification problem is:

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

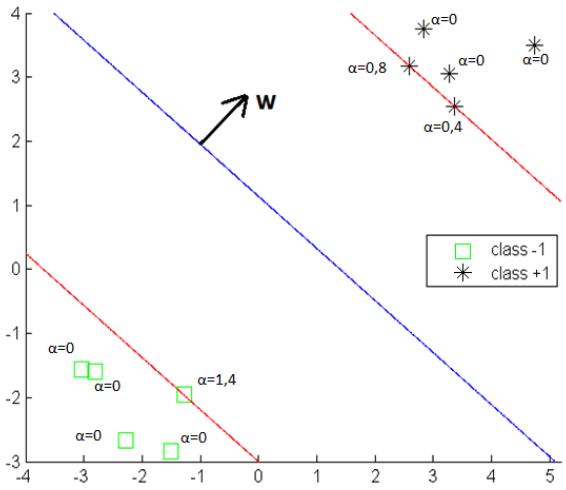


Figure 1.9: Hyperplane obtained after a dual resolution (blue line). The 2 canonical hyperplanes (red lines) contain the support vectors whose $\alpha_i > 0$. Other points have their $\alpha_i = 0$ and the equation of the hyperplane is only affected by the support vectors.

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 & 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. 1.25, we note that the samples $\mathbf{x}_i$ are only involved in a dot-product. Therefore, if we map these samples $\mathbf{x}_i$ into a higher dimensional hyperspace, called the feature space, we only need to know the dot product in the feature space:

$$(\mathbf{x}_i.\mathbf{x}_j) \rightarrow \Phi(\mathbf{x}_i).\Phi(\mathbf{x}_j) \tag{1.32}$$

where Φ is the mapping function.

The intuition behind using such mapping 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. Fig. 1.10 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 $+1$ (circle) labeled points are moved forward and the -1 (cross) labeled moved back the two classes become separable.

In most of the case, the mapping function Φ does not need to be known since we only need the dot product $\Phi(\mathbf{x}_i).\Phi(\mathbf{x}_j)$. Therefore, we can use any kernel function κ such that:

$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i).\Phi(\mathbf{x}_j)$. We call Gram matrix G, the matrix containing all $\kappa(\mathbf{x}_i, \mathbf{x}_j)$:

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

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 spaces**.

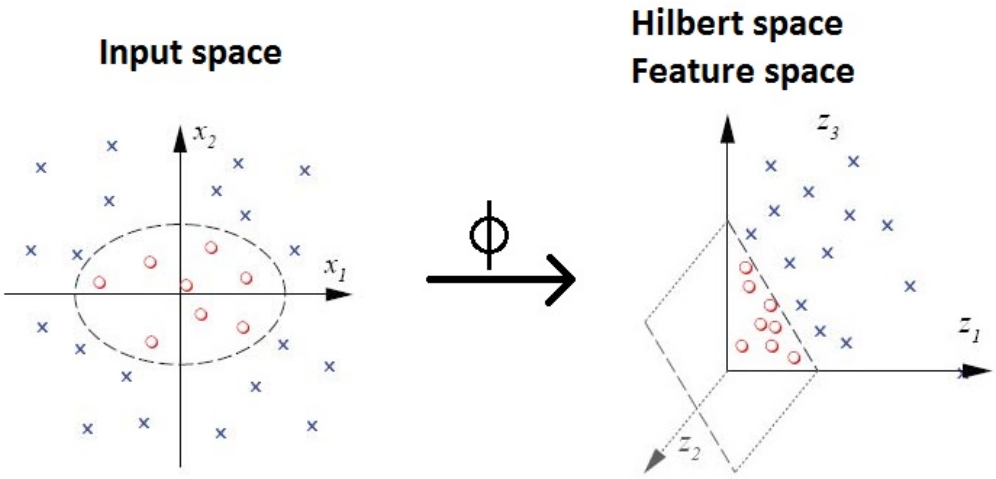


Figure 1.10: Left: in two dimensions the two classes of data (-1 for cross and +1 for circle) are mixed together, and it is not possible to separate them by a line: the data is not linearly separable. Right: using a kernel, these two classes of data become separable by a hyperplane in feature space, which maps to the nonlinear boundary shown, back in input space.[5]

Many kernels have been proposed in the literature such as the polynomial, 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:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \tag{1.33}$$

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_j - \mathbf{x}_i||_2^2}{2\sigma^2}\right) = \exp\left(-\gamma||\mathbf{x}_j - \mathbf{x}_i||_2^2\right) \tag{1.34}$$

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 the number of dimensions 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 exponen-

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

tially decaying function in the input space. The maximum value of the kernel ($\kappa(\mathbf{x}_i, \mathbf{x}_j)$=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. 1.11 [6]. The parameter γ controls the decreasing speed of the sphere. In practice, this parameter is learned during the training phase (Section 1.1.2).

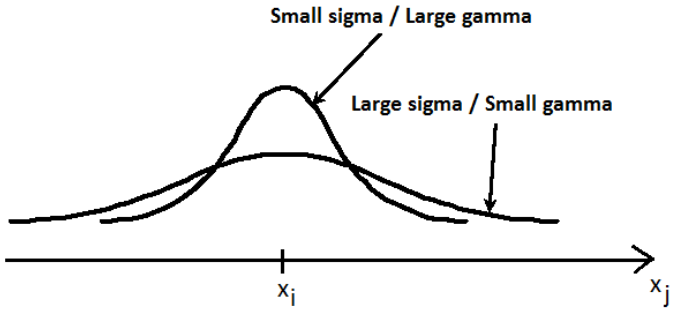


Figure 1.11: Illustration of the Gaussian kernel in the 1-dimensional input space for a small and large γ when $\mathbf{x}_i$ is fixed and $\mathbf{x}_j$ varies.

By applying the kernel trick to the soft margin formulation in Eqs. 1.25, 1.26 and 1.28, the following optimization problem allows non-linear classifiers to be learned:

$$\underset{\boldsymbol{\alpha}}{\text{argmax}} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \right) \tag{1.35}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{1.36}$$

$$0 \leq \alpha_i \leq C \tag{1.37}$$

The decision function f becomes:

$$f(\mathbf{x}_j) = sign \left(\sum_{i=1}^{n} \alpha_i^* y_i \kappa(\mathbf{x}_i, \mathbf{x}_j) + b^* \right) \tag{1.38}$$

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

$$y_j \left(\sum_{i=1}^{n_{SV}} \alpha_i^* y_i \kappa(\mathbf{x}_i, \mathbf{x}_j) + b^* \right) = 1 \tag{1.39}$$

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

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

$$b^* = \frac{1}{y_j} - \sum_{i=1}^{n_{SV}} \alpha_i^* y_i \kappa(\mathbf{x}_i, \mathbf{x}_j) \tag{1.40}$$

Note that in this case, we can't recover the weight vector $\mathbf{w}^*$ but it is not useful here for the decision function.

1.2.2.e Interpretation

Interpretation in the primal

We recall that $\mathbf{x}_i$ is a sample in p dimensions: $x_1, \ldots, x_p$. Geometrically, the vector $\mathbf{w}$ represents the direction of the hyperplane and points towards the direction of positive decision function $f(\mathbf{x}) \geq 0$ (Fig. 1.12). The absolute value of the bias $|b|$ is equal to the distance of the hyperplane to the origin $\mathbf{x} = \mathbf{0}$[7] if the norm of the vector $\mathbf{w}$ is equal to 1. In the soft margin problem, the slack variables ξ_i can be interpreted as follows:

- $\xi_i = 0$ implies that $\mathbf{x}_i$ is correctly classified and is either on the margin or on the correct side of the margin.
- $0 < \xi_i \leq 1$ implies that $\mathbf{x}_i$ lies inside the margin, but on the correct side of the decision boundary.
- $\xi_i \geq 1$ implies that $\mathbf{x}_i$ lies on the wrong side of the decision boundary and is misclassified.

In the primal formulation, 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. 1.13 with $p = 2$. Fig. 1.13(a) shows a hyperplane where elements of $\mathbf{w}$ are the same for both variables x_1 and x_2. The interpretation is that both variables contribute equally for classification of objects into positive and negative. Fig. 1.13(b) shows a hyperplane where the element of $\mathbf{w}$ for x_1 is 1, while that for x_2 is 0. This is interpreted as that x_1 is important but x_2 is not. An opposite example is shown in Fig. 1.13(c) where x_2 is considered to be important but x_1 is not. Finally, Fig. 1.13(d) provides a 3-dimensional example ($p = 3$) where an element of $\mathbf{w}$ for x_3 is 0 and all other elements are equal to 1. The interpretation is that x_1 and x_2 are important but x_3 is not.

Another way to interpret how much a variable contributes to the vector $\mathbf{w}$ is to express the contribution in percentage: the ratio $\frac{w_j}{||\mathbf{w}||_2}.100$ defines the percentage of contribution for each variable x_j in the SVM model. The interpretation is only valid if the variables x_j of the samples are normalized before learning the SVM model, they evolve in the same range.

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

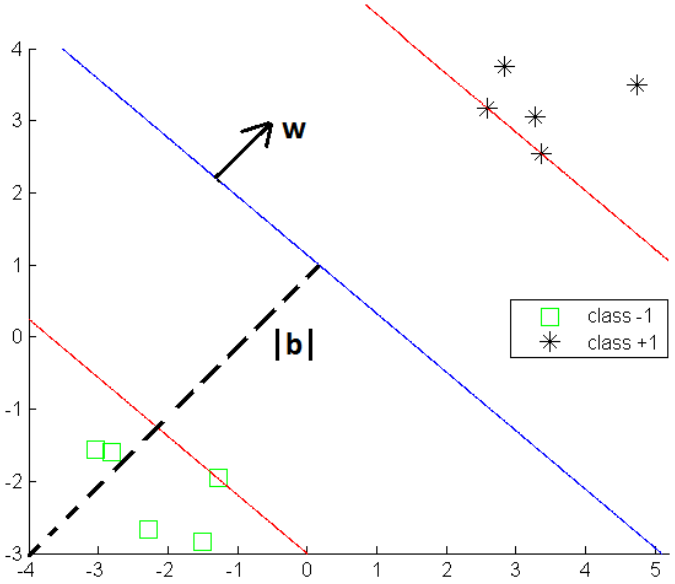


Figure 1.12: Geometric representation of SVM.

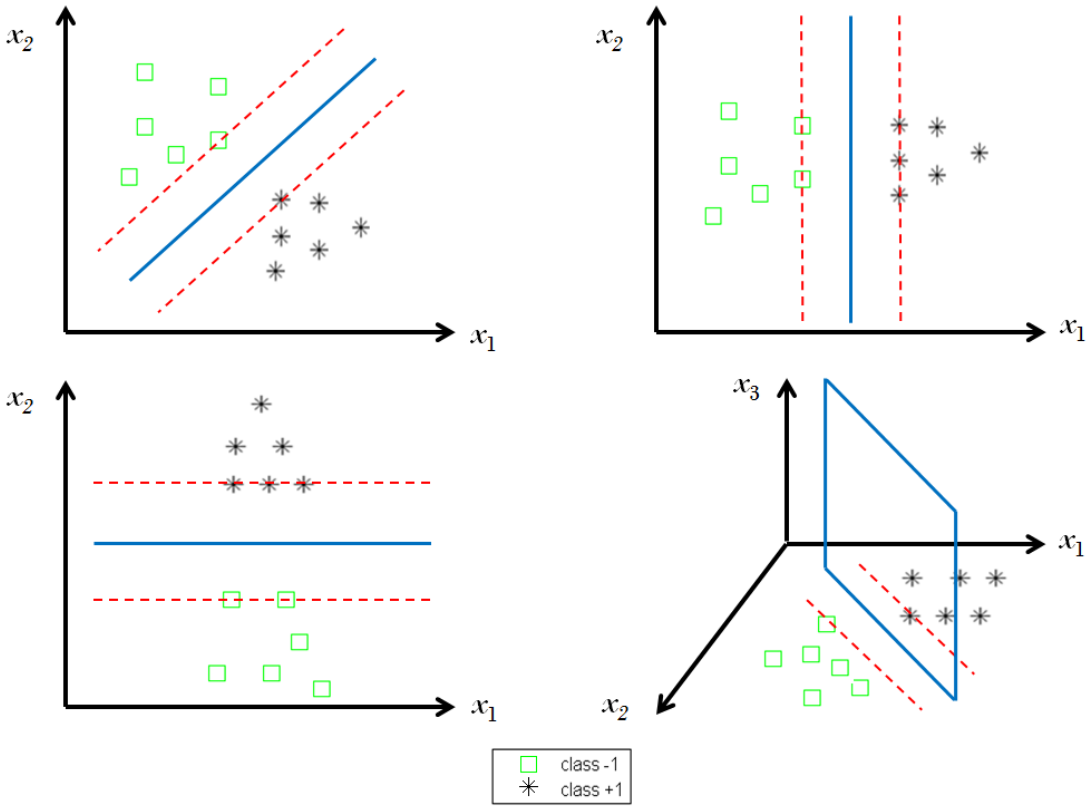


Figure 1.13: Example of several SVMs and how to interpret the weight vector **w**

Interpretation in the dual

As a constrained optimization, the dual form satisfies the Karush-Kuhn-Tucker (KKT) conditions (Eqs. 1.20, 1.21 and 1.22). We recall Eq. 1.22:

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

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. 1.31 or 1.38. Hence, they play no role for the classification decision of a new sample $\mathbf{x}_j$. The other $\mathbf{x}_i$ such that $\alpha_i^* > 0$ correspond to the support vectors. 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 the coefficient α_i^* for a sample $\mathbf{x}_i$ is, the more the sample $\mathbf{x}_i$ impacts on the decision function f. However, an unusually high value of α_i^* among the samples can lead to two interpretations: either this point is a critical point to the decision, or 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 Variants of SVM

From the primal formulation of SVM (Eqs. 1.14 & 1.15), some works investigate the effect of modifications in the regularization and loss term [HCL08].

First, the two common regularizers are $||\mathbf{w}||_1$ and $||\mathbf{w}||_2$. The former is referred to as L_1-Regularizer while the latter is L_2-Regularizer. L_1-Regularizer is used to obtain sparser models than L_2-Regularizer, *i.e.*, the vector $\mathbf{w}$ will contain many elements w_i that will be equal to zero. Thus, it can be used for variable selection.

Secondly, the two common loss functions ξ_i are $\max(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b), 0)$ and $[\max(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b), 0)]^2$. The former is referred to as L_1-Loss and the latter is L_2-Loss function. L_2-loss function will penalize more slack variables ξ_i during training and would be more sensitive to outliers. Theorically, it should lead to less error in training and poorer generalization in most of the case [HCL08]. In general, L_1-Loss is preferred.

1.2.2.g Extensions of SVM

SVM has received many interest 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 $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$. 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. 1.14.

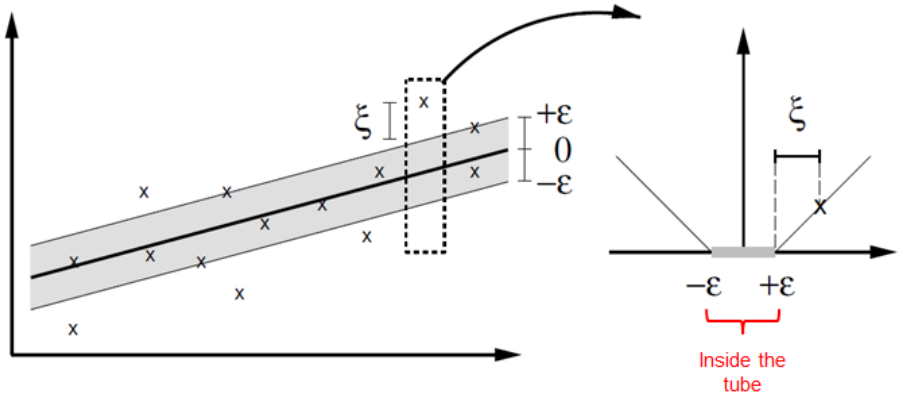


Figure 1.14: Illustration of SVM regression (left), showing the regression curve with the ϵ-insensitive "tube" (right) [CGS05]. Samples $\mathbf{x}_i$ above the ϵ-tube have $\xi_1 > 0$ and $\xi_1 = 0$, points below the ϵ-tube have $\xi_2 = 0$ and $\xi_2 > 0$, and points inside the ϵ-tube have $\xi = 0$.

The ϵ-insensitive error function E_ϵ is defined by:

$$E_\epsilon(f(\mathbf{x}_i) - y_i) = \begin{cases} 0 & \text{if} \quad |f(\mathbf{x}_i) - y_i| < \epsilon \\ |f(\mathbf{x}_i) - y_i| - \epsilon & \text{otherwise} \end{cases} \tag{1.41}$$

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

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \left(\overbrace{\frac{1}{2}||\mathbf{w}||_2^2}^{Regularization} + C \overbrace{\sum_{i=1}^{n} (\xi_{i_1} + \xi_{i_2})}^{Loss} \right) \tag{1.42}$$

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

$$y_i - (\mathbf{w}^T \mathbf{x}_i + b) \leq \epsilon + \xi_{i_1} \tag{1.43}$$

$$(\mathbf{w}^T \mathbf{x}_i + b) - y_i \leq \epsilon + \xi_{i_2} \tag{1.44}$$

$$\xi_{i_1} \geq 0 \tag{1.45}$$

$$\xi_{i_2} \geq 0b \tag{1.46}$$

The slack variables are divided into 2 kind of slacks variables, one for samples above the decision function f (ξ_{i_1}), and one for samples under the decision function f (ξ_{i_2}). As for SVM,

it is possible to have a dual formulation:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \left(\sum_{i=1}^{n} y_i(\alpha_{i_1} - \alpha_{i_2}) - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_{i_1} - \alpha_{i_2})(\alpha_{j_1} - \alpha_{j_2})(\mathbf{x}_i.\mathbf{x}_j) \right) \tag{1.47}$$

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

$$\sum_{i=1}^{n} \alpha_{i_1} = \sum_{i=1}^{n} \alpha_{i_2} \tag{1.48}$$

$$0 \leq \alpha_{i_1} \leq C \tag{1.49}$$

$$0 \leq \alpha_{i_2} \leq C \tag{1.50}$$

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

$$f(\mathbf{x}_j) = \mathbf{w}^T \mathbf{x}_j + b \tag{1.51}$$

$$f(\mathbf{x}_j) = \sum_{i=1}^{n} (\alpha^*_{i_1} - \alpha^*_{i_2})(\mathbf{x}_i.\mathbf{x}_j) + b \tag{1.52}$$

$$f(\mathbf{x}_j) = \sum_{i=1}^{n} (\alpha^*_{i_1} - \alpha^*_{i_2}) \kappa(\mathbf{x}_i, \mathbf{x}_j) + b \tag{1.53}$$

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

1.3 Conclusion of the chapter

This chapter reviews the different steps in a machine learning framework: data normalization, model selection and model evaluation. We focus on two machine learning algorithms: the k-Nearest Neighbors (k-NN) and the Support Vector Machine (SVM).

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.

Chapter 2

Time series metrics

Contents

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 subject to varying size and delays, we recall the concept of alignment and dynamic programming. Finally, we present some proposed combined metrics for time series.

2.1 Definition of time series

Time series are data that can be frequently 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, detecting 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 duration T can be seen as a signal, sampled at a frequency f_e, in a temporal window $[0;T]$. From a mathematical perspective, a time series of length q is a collection of a finite number of observations made sequentially at discrete time instants $t = 1, ..., q$. Note that $q = Tf_e$.

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} $(t = 1, ..., q)$ comes from the observation of one variable (*e.g.*, the temperature measured by one sensor). When simultaneous observation of p variables (several sensors such as the temperature, the pressure, etc.) are made at the same time, the time series is said to be multivariate. From this, one possible representation would be $\mathbf{x}_i = (\mathbf{x}_{i,1}, ..., \mathbf{x}_{i,p}) = (x_{i1,1}, ..., x_{iq,1}, x_{i1,2}, ..., x_{i1,p}, ..., x_{iq,p})$.

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).

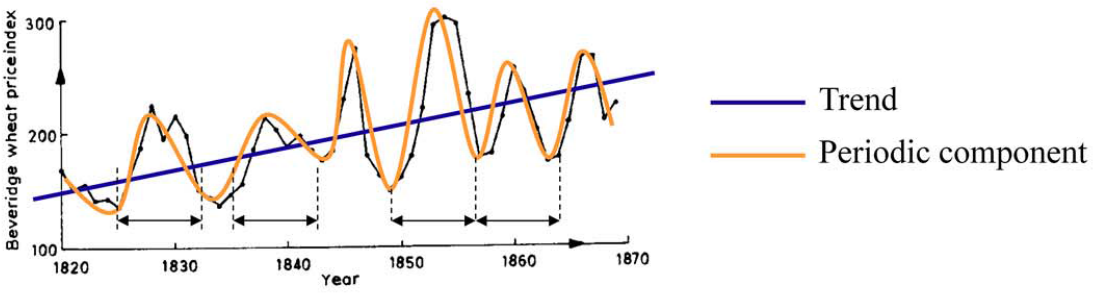


Figure 2.1: The Beveridge wheat price index is the average in nearly 50 places in various countries measured in successive years from 1500 to 1869[1].

According to Chatfield, most time series exhibit either or both a long term change in the mean (trend) and a periodic (cyclic) component. The trend can be linear, quadratic, etc. The cyclic component is a variation at a fixed period of time (seasonality) such as for example the seasonal variation of temperature. In practice, the 3 features (trend, cycle, residuals) are rarely sufficient for the classification or regression of real time series.

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 focuses on classification problems, and on time series comparison through metrics.

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

2.2 Properties and representation 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 1 to 4[DD09]:

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)
5. $D(\mathbf{x}_i, \mathbf{x}_i) = 0$ (reflexivity)

A mapping D that satisfies at least properties 1, 2 and 5 is called a **dissimilarity**, and the one that satisfies at least properties 1, 2, 4 a **pseudo-metric**. A metric, a dissimilarity and a pseudo metric can be both interpretated as a measure of how "different" two samples are: if a sample $\mathbf{x}_i$ is expected to be closer to $\mathbf{x}_j$ than to $\mathbf{x}_l$, then $D(\mathbf{x}_i, \mathbf{x}_j) \leq D(\mathbf{x}_i, \mathbf{x}_l)$. A mapping $S : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is called a **similarity** on $\mathbb{R}^p$ if S satisfies the properties of positivity, symmetry and the inequality: $\forall\ \mathbf{x}_i, \mathbf{x}_j,\ S(\mathbf{x}_i, \mathbf{x}_j) \leq S(\mathbf{x}_i, \mathbf{x}_i)$. To simplify the discussion in the following, we refer to pseudo-metric and dissimilarity as metrics, pointing out the distinction only when necessary.

Metric can be represented in two ways (Fig. 2.2). First, in Fig. (a), data points (samples $\mathbf{x}_i$ and $\mathbf{x}$) can be fixed and the distance sphere is shown for each metric (*e.g.*, the Manhattan, Euclidean and Infinite distance). Secondly, by fixing a data point $\mathbf{x}_i$, the distance sphere is fixed and the data point $\mathbf{x}$ changes according to the considered distance at hand ($\mathbf{x}_{Manhattan}$ for the Manhattan distance, $\mathbf{x}_{Euclidean}$ for the Euclidean distance, $\mathbf{x}_{Infinite}$ for the Infinite distance). For example, the latter representation is used by Weinberger and Saul in [WS09] to illustrate the effect of an initial Euclidean distance and a learned Mahalanobis metric to purify the neighborhood of data points. This concept will be explored in Chapter 3.

Given the pairwise dissimilarities between samples, some algorithms such as MultiDimensional Scaling (MDS) [CA80] or Isomap [GZZ05] have been proposed to visualize the proximity between samples in a dataset. Briefly, a MDS algorithm aims to place each sample in a P-dimensional space (in general, $P = 2$ or 3) such that the between-object distances are preserved as well as possible. Classical MDS takes an input matrix giving dissimilarities between pairs of samples and outputs a coordinate for each sample whose configuration minimizes a loss function called stress. An example of applications is given in Fig. 2.3: the distances between pairs of cities is given and the aim is to reconstruct a two dimensional map that reproduces the best the given distances. More generally, we can take benefice of this algorithm for any other type of data (samples, time series, etc.) if the given dissimilarity matrix is given.

[2]source: `http://www.bristol.ac.uk/media-library/sites/cmm/migrated/documents/chapter3.pdf`

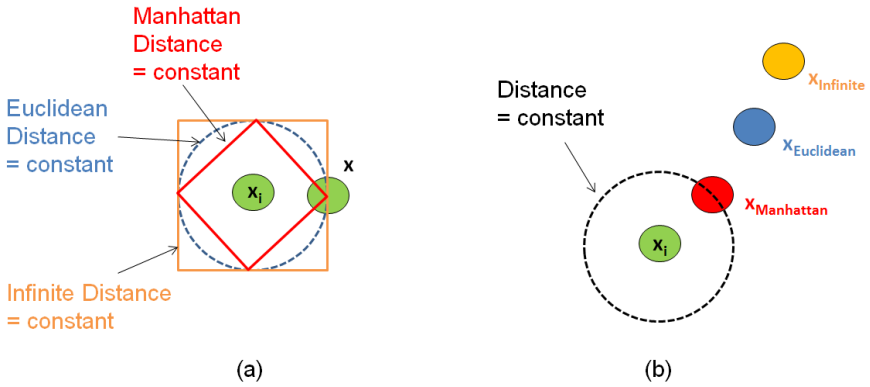
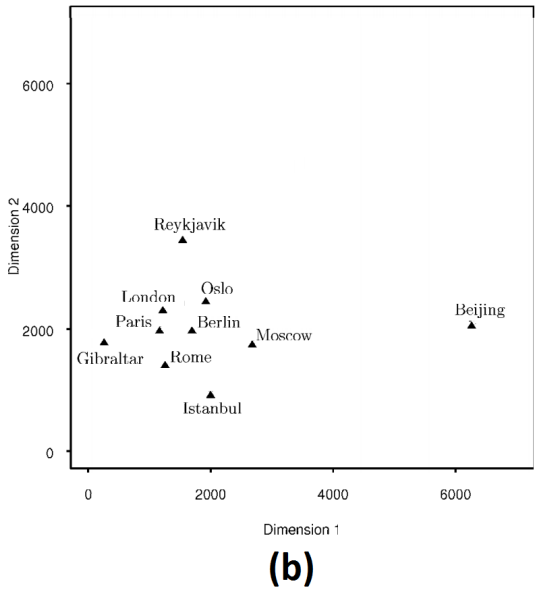


Figure 2.2: Example of metric representation: (a) Data points are fixed and the distance sphere is shown for each metric given a constant. (b) The distance sphere is fixed and the data points **x** are moving according to each distance.

	London	Berlin	Oslo	Moscow	Paris	Rome	Beijing	Istanbul	Gibraltar	Reykjavik
London	–									
Berlin	570	–								
Oslo	710	520	–							
Moscow	1550	1000	1020	–						
Paris	210	540	830	1540	–					
Rome	890	730	1240	1470	680	–				
Beijing	5050	4570	4360	3600	5100	5050	–			
Istanbul	1550	1080	1520	1090	1040	850	4380	–		
Gibraltar	1090	1450	1790	2410	960	1030	6010	1870	–	
Reykjavik	1170	1480	1080	2060	1380	2040	4900	2560	2050	–

(a)



(b)

Figure 2.3: Example of MDS. (a) Distances between ten cities in miles. (b) Two dimensional plot of the ten cities from a classical MDS[2].

2.3 Unimodal metrics for time series

In the following, we suppose that time series have the same duration T and have been regularly sampled at the frequency f_e. Therefore, they have the same length $q = Tf_e$. Let $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{iq})$ and $\mathbf{x}_j = (x_{j1}, x_{j2}, ..., x_{jq})$ be two univariate time series of length q.

2.3.1 General review of 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]. For example, as explained in [MV14], a crucial question in clustering is to determine what we mean by "similar" samples, *i.e.*, defining a suitable metrics between two samples. The idea is not restricted to clustering and can be naturally extended to other machine learning problems (supervised, semi-supervised). Due to their temporal nature, time series may be compared based on different characteristics, called **modalities**, such as their amplitude, shape or frequency. Contrary to static data, time series may be also subject to temporal specificities such as delays. From the surge of research, one can identify at least three categories: metrics in the time domain, metrics in a feature-extracted domain, metrics based on models.

The first category (metrics in the time domain) aims to compare the closeness of time series observations in the temporal domain. Without being exhaustive, many variants cover the Minkowski distance or the Frechet distance [Mau06] for amplitude comparison, and the correlation-based distances [DC03]; [DCA12]; [DCN07]; [Ben+09]for behavior comparison.
The second category (metrics in a feature extracted domain) aims to project the time series in an other domain, such as the frequential domain or the symbolic representation, and to compute a distance in the projected domain. For frequential-based distance, many measures have been proposed such as the periodogram-based distance [CCP06] or the dissimilarity measure based on the discrete wavelet transform [Cha+08]. Based on the symbolic representation SAX (Symbolic Aggregate approXimation), a dissimilarity measure between symbols have been proposed in [Lin+03].
In the third category (metrics based on models), the aim is to assume a model of the time series and to compute the dissimilarity between the evaluated models. Most of the propositions assume that time series are generated by either an ARIMA (AutoRegressive Integrated Moving Average) or ARMA (AutoRegressive Moving Average) process [KGP01]; [Mar00]. Some work have considered alternative models such as the Markov chains [RSC02] or the hidden Markov models [Smy97]. Based on the ARIMA or ARMA model, a number of metrics have been proposed, such as the Piccolo distance [Pic90] or the Maharaj distance [Mah96], which evaluate the distance between the parameters of the estimated model. Other distances between series based on their evaluated models have been proposed such as the kernel of Binet-Cauchy [VSV07].

In the following, we focus on three types of metrics for time series, two in the time domain (amplitude-based and behavior-based distance) and one in the frequential domain (frequential-

based distance).

2.3.2 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 weighted by a matrix $\mathbf{M}$:

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

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 the normalized Euclidean distance:

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{t=1}^{q} \frac{(x_{it} - x_{jt})^2}{m_t}} \tag{2.3}$$

where m_t is the variance of the x_{it} and x_{jt} over the sample set. Note that this is equivalent to normalize each feature: $x'_{it} = x_{it}/\sqrt{m_t}$ and use the Euclidean distance on the normalized features x'_{it}. 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. 2.4, let's try to determine 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 $\mathbf{x}_1$ is closer to $\mathbf{x}_3$ than to $\mathbf{x}_2$ since $d_A(\mathbf{x}_1, \mathbf{x}_3) = 24.1909 < d_A(\mathbf{x}_1, \mathbf{x}_2) = 29.0818$.

2.3.3 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 [SS12b]; [TC98]; [BM67]). In this 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 components at frequential index $f =$

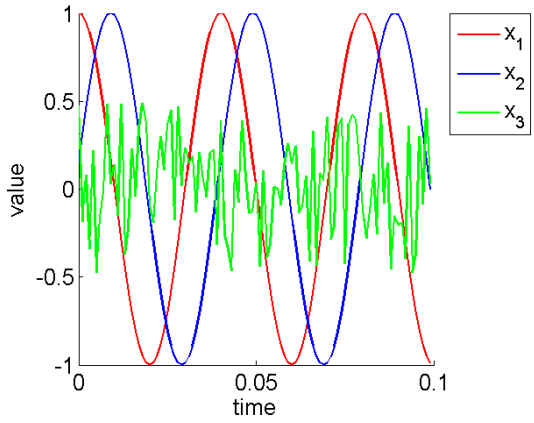


Figure 2.4: 3 toy time series in the temporal domain. Time series in blue and red are two sinusoidal signals. Time series in green is a random signal.

$\{1, \ldots, F\}$. Inspired from the amplitude-based metric in the temporal domain (Section 2.3.2), we propose a frequential-based metric based on the Euclidean distance between the complex number modules $|\tilde{x}_{if}|$ of the time series projected in the Fourier domain:

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

In the example of Fig. 2.4, recall that the Euclidean distance d_A states that $\mathbf{x}_1$ is closer to $\mathbf{x}_3$ than $\mathbf{x}_2$. However, in the frequency domain (Fig. 2.5), the frequential-based distance d_F states that $\mathbf{x}_1$ is closer to $\mathbf{x}_2$ than to $\mathbf{x}_3$ since $d_F(\mathbf{x}_1, \mathbf{x}_2) = 0.0835 < d_F(\mathbf{x}_1, \mathbf{x}_3) = 1.0124$.

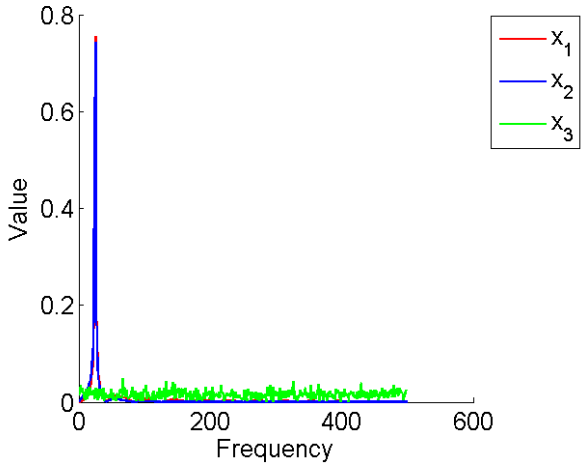


Figure 2.5: 3 toy time series in the frequency domain: blue and red are the spectrum of the Fourier transform of two sinusoidal signals; green is the spectrum of the Fourier transform of a random signal.

2.3.4 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, called the temporal correlation $cort_r$, is introduced in [DC03]; [DCA12]; [DCN07]:

$$cort_r(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{t,t'=1}^{q} (x_{it} - x_{it'})(x_{jt} - x_{jt'})}{\sqrt{\sum_{t,t'=1}^{q} (x_{it} - x_{it'})^2} \sqrt{\sum_{t,t'=1}^{q} (x_{jt} - x_{jt'})^2}} \tag{2.5}$$

where $|t-t'| \leq r$, $r \in [1, ..., q-1]$. The parameter r can be tuned or fixed *a priori* and depends on 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. This parameter can be seen as a temporal window where the time series are compared.

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

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

Considering Fig. 2.6, the behavior-based metric d_B states that $\mathbf{x}_1$ is closer to $\mathbf{x}_4$ than to $\mathbf{x}_2$ or $\mathbf{x}_3$ since $d_B(\mathbf{x}_1, \mathbf{x}_2) = 0.477$, $d_B(\mathbf{x}_1, \mathbf{x}_3) = 1$ and $d_B(\mathbf{x}_1, \mathbf{x}_4) = 0$.

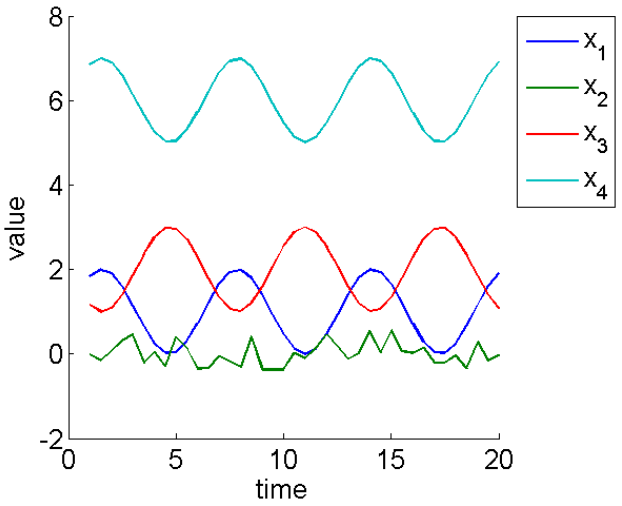


Figure 2.6: The signal from Fig. 2.4 and a signal $\mathbf{x}_4$ which is signal $\mathbf{x}_1$ and an added translation. Based on behavior comparison, $\mathbf{x}_4$ is the closest to $\mathbf{x}_1$.

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. 2.7), 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 $|\boldsymbol{\pi}_{ij}| = m$ between two time series $\mathbf{x}_i$ and $\mathbf{x}_j$ of length q is defined as the set of m ($q \leq m \leq 2q-1$) couples of aligned elements of $\mathbf{x}_i$ to m elements of $\mathbf{x}_j$:

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

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

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

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

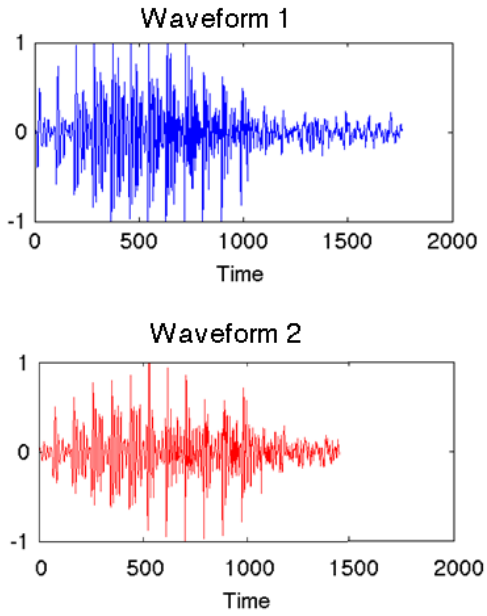


Figure 2.7: Example of a same sentence said by two different speakers. Time series are shifted, compressed and dilatated in the time.

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

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

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

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

In the following, we denote $\boldsymbol{\pi} = \boldsymbol{\pi}_{ij}$ to simplify the notation. Intuitively, an alignment $\boldsymbol{\pi}$ defines a way to associate elements of two time series. Alignments can be described by paths in the $q \times q$ grid that crosses the elements of $\mathbf{x}_i$ and $\mathbf{x}_j$ (Fig. 2.8). We denote $\boldsymbol{\pi}$ a valid alignment and A_{ij}, 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]; [SC04].

DTW requires to choose a cost function φ to be optimised, such as a dissimilarity function (d_A, d_B, d_F, etc.). Standard 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:

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

When the cost function φ is a similarity measure (Section 2.2), 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:

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

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

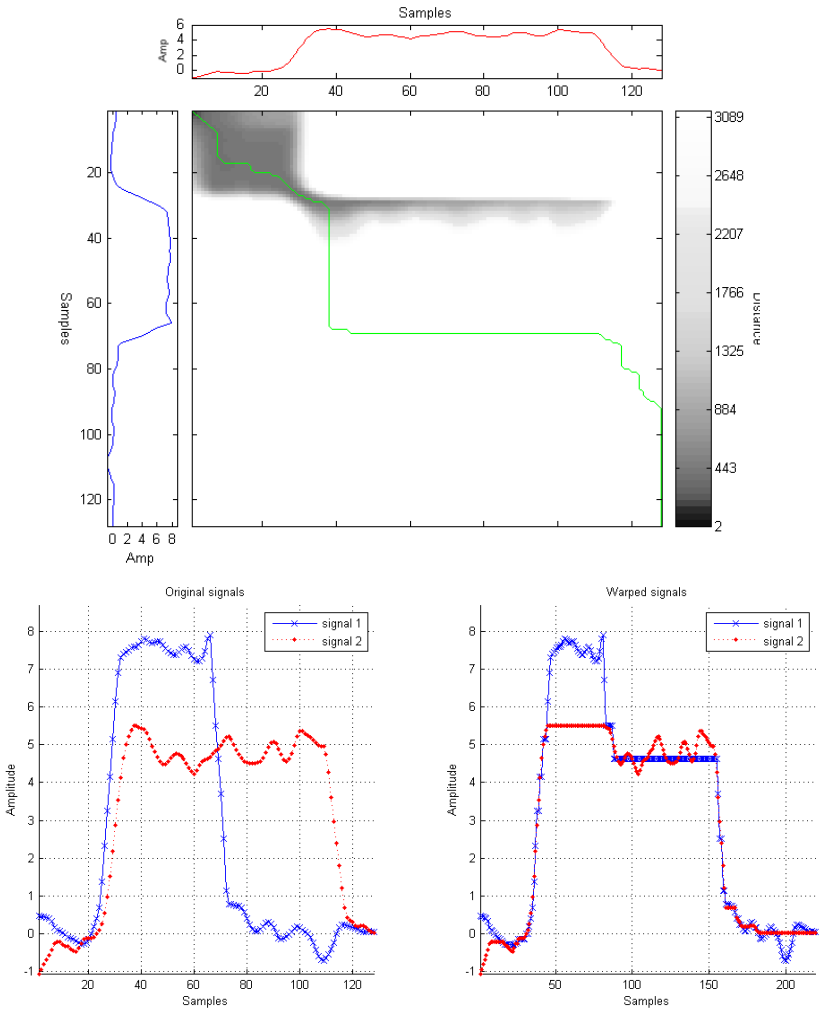


Figure 2.8: Example of DTW grid between 2 time series $\mathbf{x}_i$ and $\mathbf{x}_j$ (top) and the signals before and after warping (bottom). On the DTW grid, the two signals can be represented on the left and bottom of the grid. The optimal path $\boldsymbol{\pi}^*$ is represented in green line and shows how 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)

Once an optimal alignment $\boldsymbol{\pi}^*$ has been found, and whatever cost function φ have been chosen to find it, the metric presented in Section 2.3 (amplitude-based d_A, behavior-based d_B, frequential-based d_F) 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 to $\mathbf{x}_{i,\boldsymbol{\pi}^*}$ 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 modalities (characteristics) based on their amplitude, behavior or frequential components alone. In some cases, several components (amplitude, behavior and/or frequential) may be implied.

2.5.1 Combination functions

A first technic considers a classifier for each p metric and combines the decision of the p resulting classifiers. This methods is referred to as post-fusion [Zha+13], 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 basic combination functions that we could use combines two unimodal metrics through a linear or geometric function. For example, with d_A and d_B, we obtain:

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

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

where $\beta \in [0; 1]$ defines the trade-off between the amplitude d_A and the behavior d_B components, and is thus application dependent. For example, in classification problems, this parameter can be learned through a grid search procedure (Section 1.1.2). 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 [DCA12]; [DCN07]:

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

where β is a parameter that defines the compromise between behavior and amplitude components.

Fig. 2.9 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 β. For small value of β ($\beta = 0$), the three metrics only includes d_A. For high value of β ($\beta = 1$), D_{Lin} and D_{Geom} only includes d_B. For $\beta = 6$ and for small values of d_B, D_{Sig} mostly includes d_B while for large value of d_B, D_{Sig} mostly includes d_A.
Note that these combinations are fixed and defined independently from the analysis task at hand. Moreover, in the case of D_{Sig}, the component $cort_r$ can be seen as a penalizing factor of d_A. Finally, one could extend D_{Lin} and D_{Geom} by adding metrics, but that would imply to add parameters. The grid search to find the best parameters would become time consuming.

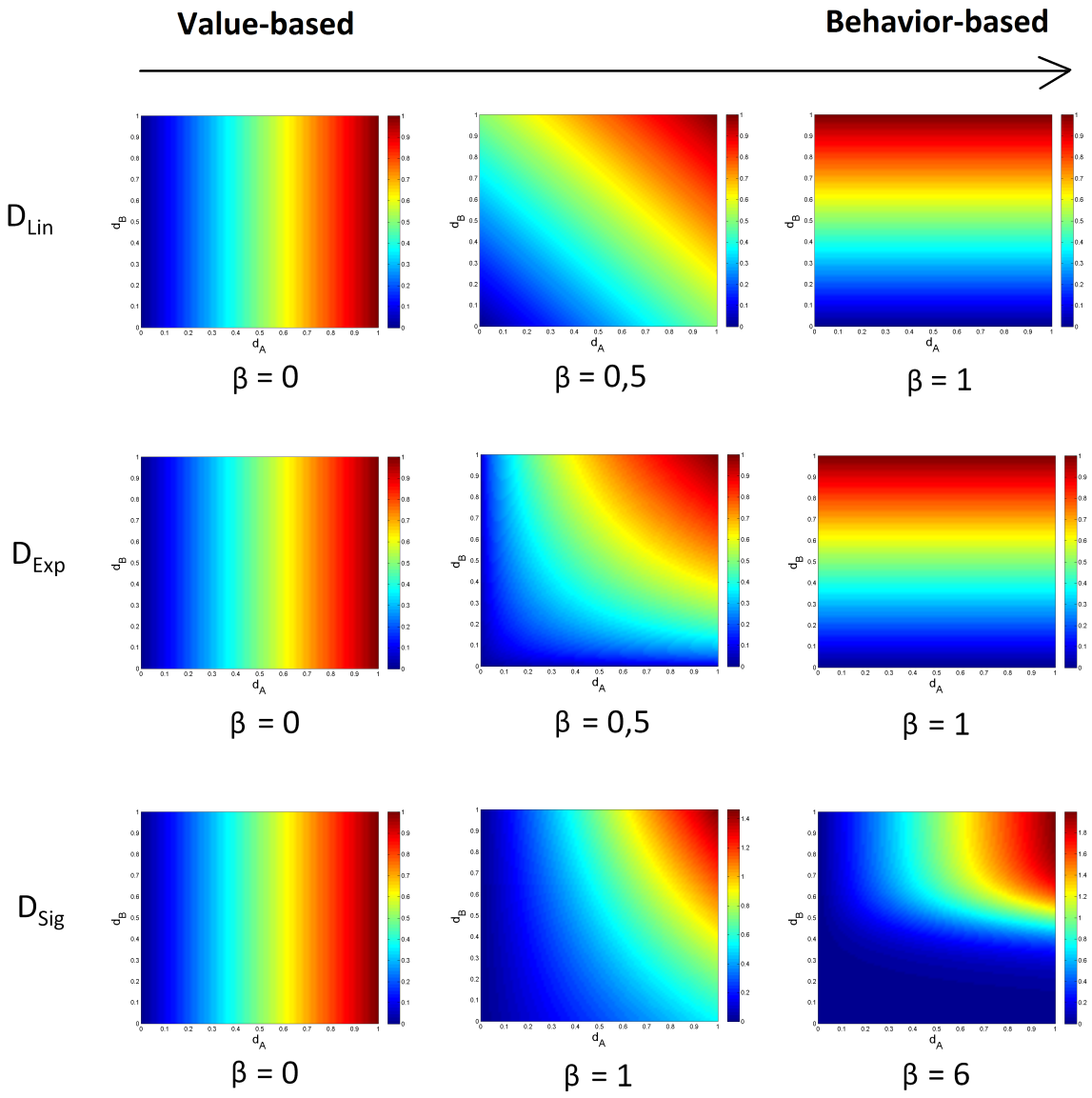


Figure 2.9: Contour plot of the resulting combined metrics: D_{Lin} (1^{st} line), D_{Geom} (2^{nd} line) and D_{Sig} (3^{rd} line), for different values of β. For the three combined metrics, the first and second dimensions are respectively the amplitude-based metrics d_A and the behavior-based metric d_B.

2.5.2 Proposition of normalization of distances

When combining several metrics into a single metric, it is necessary to normalize the metrics involved in the combination to avoid one metric from another to have a larger impact in the combination. Classically, to normalize data, Z-normalization is used (Section 1.1.4). In that case, we suppose that the variables x_j are normally distributed: data evolves between $[-\infty; +\infty]$ and are coming from a Gaussian process.

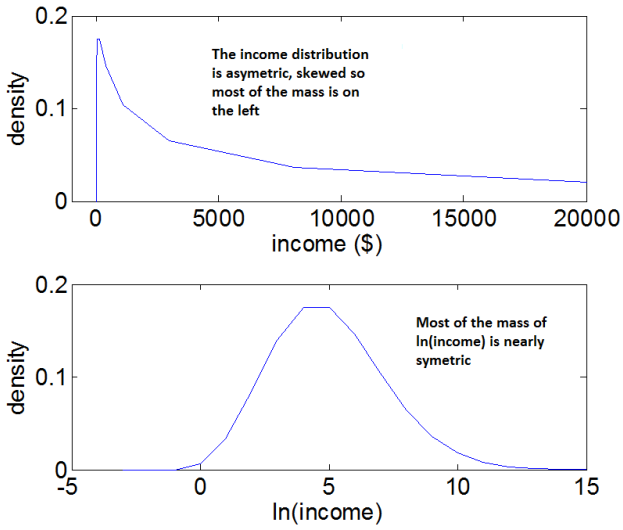


Figure 2.10: A nearly log-normal distribution, and its log transform[3]

In some cases, the data are skewed such as monetary amounts, incomes or distances. These data may be log-normally distributed, *e.g.*, the log of the data is normally distributed (Fig. 2.10). Some works proposes to take the log of the data (x_j^{ln}) to restore the symmetry, and then, to apply a Z-normalization of this transformation [ZMP14]:

$$x_j^{ln} = \ln(x_j); \tag{2.19}$$

$$x_j^{ln,norm} = \frac{x_j^{ln} - \mu_j^{ln}}{\sigma_j^{ln}} \tag{2.20}$$

$$x_j^{norm} = \exp(x_j^{ln,norm}) \tag{2.21}$$

where ln denotes the Natural Logarithm function, μ_j^{ln} and σ_j^{ln} the mean and the standard deviation of a variable x_j^{ln}.

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

2.6 Conclusion of the chapter

To cope with modalities (characteristics) 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 behavior-based distance d_B and the Fourier-based distance d_F. In practice, real time series may be subject 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, the metrics d_A, d_B and d_F only include one modality. In general, several modalities may be implied and some combined metric have been proposed, but the propositions are often limited to two modalities and the metrics are defined independently from the analysis task.

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, inspired from these ideas, we propose framework to learn a combined metric for a large margin k-NN classifier of time series.

Chapter 3

Multi-modal and Multi-scale Time series Metric Learning (M^2TML)

Contents

In this chapter, we first motivate the problem of Multi-modal and Multi-scale Temporal Metric Learning (M^2TML) for nearest neighbors classification. Secondly, we recall the Large Margin Nearest Neighbors (LMNN) framework proposed by Weinberger & Saul. Thirdly, we introduce the concept of dissimilarity space. Then, we formalize the general problem of M^2TML. After that, we propose three different formalizations (Linear, Quadratic and SVM-based), each involving a different regularization term. We give an interpretation of the solution and study the properties of the obtained metric. Finally, we give the algorithm.

3.1 Motivations

This work focuses on defining a 'good' metric for k-NN classification of time series. The definition of a metric to compare samples is a fundamental issue in data analysis or machine learning. As seen in Chapter 2, temporal data may be compared based on one or several characteristics, called **modalities** (amplitude, behavior, frequency) and they might be subject to delays. In some classification problems, the sharing of common features at the global level (implying all the time series observations) characterizes the class membership. In other problems, the presence of saliencies or local events is a key characteristic to discriminate the classes. Thus, there is a need to take into account local and global aspects, depending on the complexity of the considered data. We refer this notion as scale in our work. We believe that the definition of a temporal metric should consider at least these different aspects (modality, delay, scale) in order to improve the performance of a classifier. Fig. 3.1 illustrates a result obtained with our proposition. There is a significant improvement in classification performances by taking into account in the metric definition, several modalities (behavior d_B, frequential d_F) located at different scales (illustrated by black rectangles in the figure), one at the global scale (d_F) and one at a more locally scale (d_B). The performance of the learned combined metric is compared with the ones of the standard metrics that take into account for each, only one modality on a global scale (involving all time series elements).

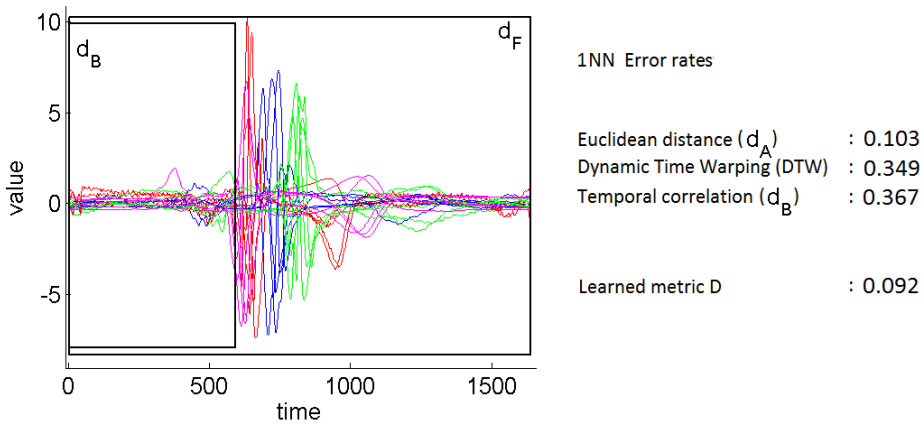


Figure 3.1: CinCECGtorso dataset and error rate using a kNN classifier ($k = 1$) with standard metrics (Euclidean distance, Dynamic Time Warping, temporal correlation) and a learned combined metric D. The figure shows the 2 major metrics involve in the combined metric D and their respective temporal scale (black rectangles).

Our aim is to take leverage from the metric learning framework [WS09]; [BHS12] to learn a multi-modal and multi-scale temporal metric for time series nearest neighbors classification. Specifically, our objective is to learn from the data a linear or non linear function that combines several temporal modalities at several temporal scales, that satisfies metric properties (Section 2.2), and that generalizes the case of unimodal metrics at the global scale.
Metric learning can be defined as learning, from the data and for a task, a pairwise function (*i.e.*, a similarity, dissimilarity or a distance) that brings closer samples that are expected to be

similar, and pushes far away those expected to be dissimilar. Such similarity and dissimilarity expectations, is inherently task- and application-dependent, generally given *a priori* and fixed during the learning process. Metric learning has become an active area of research in the last decade for various machine learning problems (supervised, semi-supervised, unsupervised, online learning) and has received many interests in its theoretical background (generalization guarantees) [BHS13]. From the surge of recent researches in metric learning, one can identify mainly two categories: the linear and non linear approaches. The former is the most popular, it defines the majority of the propositions, and focuses mainly on the Mahalanobis distance learning [WS09]. The latter addresses non linear metric learning which aims at capturing non linear structures in the data, *e.g.*, Kernel Principal Component Analysis (KPCA) and Support Vector Metric Learning (SVML). In both cases, the metric is learned in the original space (*i.e.*, space described by the features of the samples). In KPCA, the aim is to project the data into a non linear feature space and learn the metric in that projected space [ZY10]; [Cha+10]. In SVML, the Mahalanobis distance is learned jointly with the learning of the SVM model in order to minimize the validation error [XWC12]. In general, the optimization problems in non linear approaches is more expensive to solve that in linear approaches, and the methods tend to favor overfitting as the constraints are generally easier to satisfy in a nonlinear kernel space. A more detailed review on metric learning is done in [BHS13].
Contrary to static data, metric learning for structured data (*e.g.* sequence, time series, trees, graphs, strings) is less frequent. While for sequence data most of the works focus on string edit distance to learn the edit cost matrix [OS06]; [BHS12], metric learning for time series is still in its infancy. Without being exhaustive, major recent proposals rely on weighted variants of dynamic time warping to learn alignments under phase or amplitude constraints [Rey11]; [JJO11]; [ZLL14]; [Mei+15], enlarging alignment learning framework to multiple temporal matching guided by both global and local discriminative features [FDCG13]. For most of these propositions, temporal metric learning process is systematically: a) Uni-modal (amplitude-based), the divergence between aligned elements being either the Euclidean or the Mahalanobis distance and b) Uni-scale (global level), involving all time series elements at once, which restricts its potential to capture local characteristics. We believe that perpectives for metric learning, in the case of time series, should include multi-modal and multi-scale aspects.

We propose in this work to learn a multi-modal and multi-scale temporal metric for a robust k-NN classifier. For this, the main idea is to embed time series into a pairwise dissimilarity space where a linear function combining several modalities at different temporal scales can be learned, driven by a large margin optimization process inspired from the nearest neighbors metric learning framework [WS09]. Thanks to the "kernel trick", the proposed solution is extended to non-linear temporal metric learning context. A sparse and interpretable variant of the proposed metrics confirms its ability to localize finely discriminative modalities as well as their temporal scales.

In this chapter, we first recall the Large Margin Nearest Neighbors (LMNN) framework proposed by Weinberger & Saul. Secondly, we introduce the concept of pairwise dissimilarity space. We formalize the general problem of learning a combined metric for a robust k-NN as the learning a function in the dissimilarity space. From the general formalization, we propose three formalizations (Linear, Quadratic and SVM-based), give an interpretation of the solutions

and study the properties of the learned metrics. Finally, we give the algorithm. Note that these formalizations don't only concern time series and could be applied to learn a combined metric on any type of data.

3.2 A recall on 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 metric D for a large margin k-NN classifier in the case of static data. The M^{2}TML concept and optimization problem will be inspired in the following by this approach.

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 **impostors** for each training sample $\mathbf{x}_i$ is introduced. Given a metric D, 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 impostors of $\mathbf{x}_i$, denoted, $l \nrightarrow i$, 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 $\mathbf{x}_j$: $\forall l, \exists j \in j \rightsquigarrow i/$

$$D(\mathbf{x}_i, \mathbf{x}_l) \leq D(\mathbf{x}_i, \mathbf{x}_j) + 1 \tag{3.1}$$

Geometrically, an imposter $\mathbf{x}_l$ is a sample that invades the target neighborhood plus one unit margin as illustrated in Fig. 3.2. The target neighborhood is defined with respect to an initial metric D_0. Without prior knowledge, L2-norm is often used. Metric learning by LMNN aims at minimizing the number of impostors invading the target neighborhood. By adding a margin 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 impostors $\mathbf{x}_l$ as illustrated in Fig. 3.2.

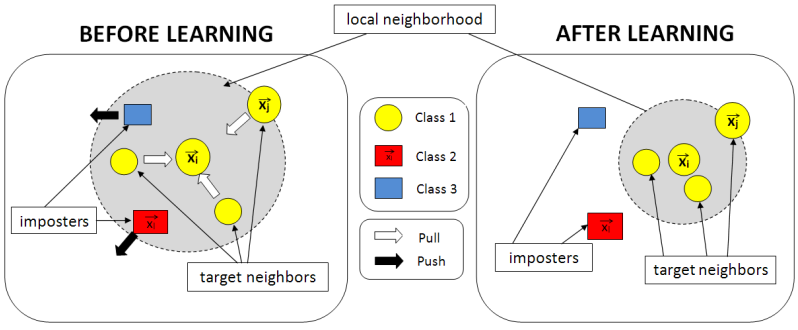


Figure 3.2: Pushed and pulled samples in the $k = 3$ target neighborhood of $\mathbf{x}_i$ before (left) and after (right) learning. The pushed (vs. pulled) samples are indicated by a white (vs. black) arrows (Weinberger & Saul [WS09]). Note: the representation of the metric here is the one where the distance sphere is fixed and the data points are moving according to the considered distance (Section 2.2).

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 the majority of 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:

$$D^2_{\mathbf{L}}(\mathbf{x}_i, \mathbf{x}_j) = D^2(\mathbf{L}\mathbf{x}_i, \mathbf{L}\mathbf{x}_j) \tag{3.2}$$
$$D^2_{\mathbf{L}}(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{L}(\mathbf{x}_i - \mathbf{x}_j)||_2^2 \tag{3.3}$$

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

$$D^2_{\mathbf{L}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)\mathbf{L}^T\mathbf{L}(\mathbf{x}_i - \mathbf{x}_j) \tag{3.4}$$

Let $\mathbf{M} = \mathbf{L}^T\mathbf{L}$. 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:

$$D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)\mathbf{M}(\mathbf{x}_i - \mathbf{x}_j) \tag{3.5}$$

The computation of the learned metric $D_{\mathbf{M}}$ can thus be seen as a two-stepped 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:

$$D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = D^2_{\mathbf{L}}(\mathbf{L}\mathbf{x}_i, \mathbf{L}\mathbf{x}_j) \tag{3.6}$$

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 to the latter one.

Mathematically, the metric learning problem can be formalized as an optimization problem involving two 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:

$$\begin{aligned}
&\underset{\mathbf{M},\xi}{\operatorname{argmin}} \left\{ \underbrace{\sum_{i,j \rightsquigarrow i} D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)}_{pull} + C \underbrace{\sum_{i,j \rightsquigarrow i, l \nrightarrow i} \xi_{ijl}}_{push} \right\} \\
&\text{s.t. } \forall i = 1, \ldots, n, \forall j \rightsquigarrow i, l \nrightarrow i, \\
&D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) - D^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - \xi_{ijl} \\
&\xi_{ijl} \geq 0 \\
&\mathbf{M} \succeq 0
\end{aligned} \tag{3.7}$$

where ξ_{ijl} are slack variables, C is a trade-off between the push and pull term and $\mathbf{M} \succeq 0$ means that $\mathbf{M}$ is a positive semidefinite matrix. Generally, the parameter C is tuned via cross validation and grid search (Section 1.1.2). Similarly to Support Vector Machine (SVM) approach, slack variables ξ_{ijl} are introduced to relax the optimization problem.

3.3 Multi-modal and multi-scale pairwise dissimilarity space

In this section, we first present the concept of pairwise dissimilarity space for multi-modal description. Then, in the case of time series, we enrich this representation with a multi-scale description.

3.3.1 Pairwise embedding

Let $\{\mathbf{x}_i, y_i\}_{i=1}^n$ be a set of n time series $\mathbf{x}_i = [x_{i1}, \ldots, x_{iq}] \in \mathbb{R}^q$ labeled y_i. Let $d_1, \ldots, d_p$ be p given metrics that allow one to compare samples $\mathbf{x}_i$. As discussed in Chapter 2, three naturally modalities are involved for time series comparison: amplitude-based d_A, behavior-based d_B and frequential-based d_F. Our objective is to learn a metric D that combines the p basic temporal metrics for a robust k-NN classifier.

The computation of a metric d, and D, always takes into account a pair of samples $(\mathbf{x}_i, \mathbf{x}_j)$. We introduce a new space representation referred to as the **pairwise dissimilarity space**. We note φ an embedding function that maps each pair of time series $(\mathbf{x}_i, \mathbf{x}_j)$ to a vector $\mathbf{x}_{ij}$ in a pairwise dissimilarity space $\mathscr{E} = \mathbb{R}^p$ whose dimensions are $d_1, \ldots, d_p$ (Fig. 3.3):

$$\begin{aligned} \varphi : \mathbb{R}^q \times \mathbb{R}^q &\to \mathscr{E} = \mathbb{R}^p \\ (\mathbf{x}_i, \mathbf{x}_j) &\to \mathbf{x}_{ij} = [d_1(\mathbf{x}_i, \mathbf{x}_j), \ldots, d_p(\mathbf{x}_i, \mathbf{x}_j)]^T \end{aligned} \tag{3.8}$$

$$\begin{aligned} \varphi : \mathbb{R}^q \times \mathbb{R}^q &\to \mathscr{E} = \mathbb{R}^p \\ (\mathbf{x}_i, \mathbf{x}_j) &\to \mathbf{x}_{ij} = [d_A(\mathbf{x}_i, \mathbf{x}_j), d_B(\mathbf{x}_i, \mathbf{x}_j), d_F(\mathbf{x}_i, \mathbf{x}_j)]^T \end{aligned} \tag{3.9}$$

A metric D that combines the p metrics $d_1, \ldots, d_p$ can be seen as a function of the dissimilarity space:

$$\begin{aligned} D : \mathbb{R}^p &\to \mathbb{R} \\ \mathbf{x}_{ij} &\to D(\mathbf{x}_{ij}) = f(d_1(\mathbf{x}_i, \mathbf{x}_j), \ldots, d_p(\mathbf{x}_i, \mathbf{x}_j)) \end{aligned} \tag{3.10}$$

In that space, the norm of a pairwise vector $||\mathbf{x}_{ij}||$ refers to the proximity between the time series $\mathbf{x}_i$ and $\mathbf{x}_j$. In particular, if $||\mathbf{x}_{ij}|| = 0$ then $\mathbf{x}_j$ is identical to $\mathbf{x}_i$ according to all metrics d_h.

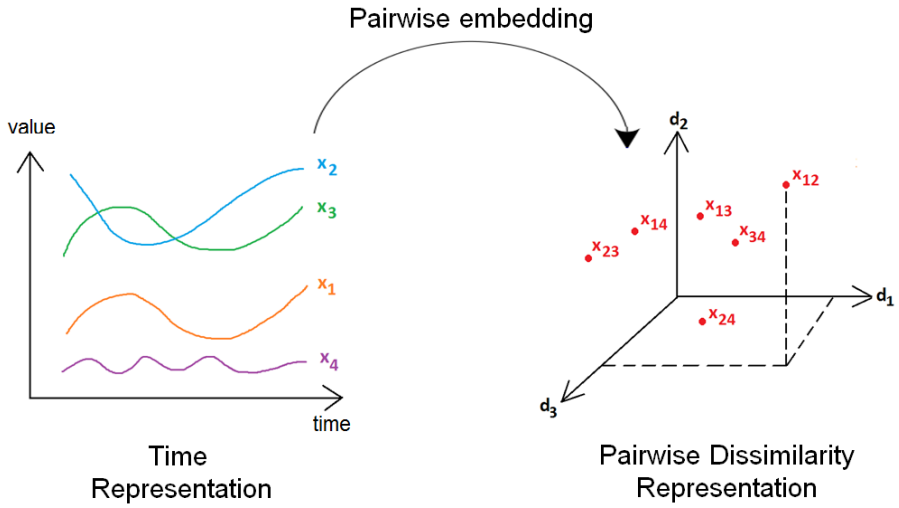


Figure 3.3: Example of embedding of four time series $\mathbf{x}_i$ from the temporal space (left) into the dissimilarity space (right) for $p = 3$ basic metrics.

3.3.2 Interpretation in the pairwise dissimilarity space

In this section, we give more detailed interpretations in the dissimilarity space. We recall that the norm of a pairwise vector is given by:

$$||\mathbf{x}_{ij}||_2 = \sqrt{\sum_{h=1}^{p} (d_h(\mathbf{x}_i, \mathbf{x}_j))^2} \tag{3.11}$$

In the following, we denote the norm $||\mathbf{x}_{ij}||$ as an initial distance in the dissimilarity space and call it D_0. Any other initial metric could have been chosen. The norm of a pairwise vector $\mathbf{x}_{ij}$ can be interpreted as a proximity measure: the lower the norm of $\mathbf{x}_{ij}$ is, the closer are the time series $\mathbf{x}_i$ and $\mathbf{x}_j$. Two pairwise vectors $\mathbf{x}_{ij}$ and $\mathbf{x}_{kl}$ that are on a same line that passes through the origin $\mathbf{x}_{ii} = \mathbf{0}$ represent differences in the the same proportions between their respective modalities (Fig. 3.4).

The Euclidean distance $\sqrt{\sum_{h=1}^{p} (d_h(\mathbf{x}_i, \mathbf{x}_j) - d_h(\mathbf{x}_k, \mathbf{x}_l))^2}$ between two pairwise vectors $\mathbf{x}_{ij}$ and $\mathbf{x}_{kl}$ represents the similarity between the differences among the same modalities, in the same proportions. Note that if the Euclidean distance is close to 0 ($\mathbf{x}_{ij}$ and $\mathbf{x}_{kl}$ are close in the dissimilarity space), it doesn't mean that the time series $\mathbf{x}_i$, $\mathbf{x}_j$, $\mathbf{x}_k$ and $\mathbf{x}_l$ are similar. Fig 3.5 shows an example of two pairwise vectors $\mathbf{x}_{ij}$ and $\mathbf{x}_{kl}$ close together in the pairwise space. However, in the temporal space, the time series $\mathbf{x}_1$ and $\mathbf{x}_3$ are not similar for example. It means that $\mathbf{x}_i$ is as similar to $\mathbf{x}_j$ as $\mathbf{x}_k$ is to $\mathbf{x}_l$, *i.e.*, the distance D_0 between $\mathbf{x}_i$ and $\mathbf{x}_j$ is nearly the same than the distance D_0 between $\mathbf{x}_k$ and $\mathbf{x}_l$.

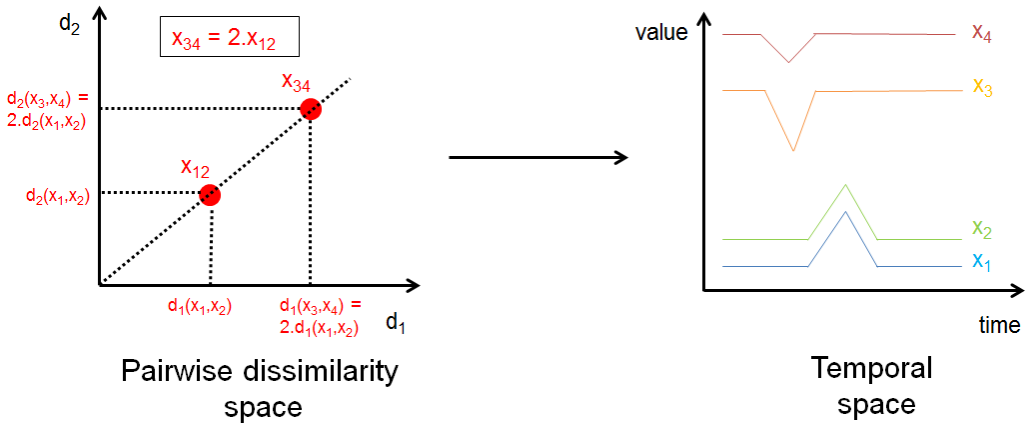


Figure 3.4: Example of interpretation of two pairwise vectors $\mathbf{x}_{12}$ and $\mathbf{x}_{34}$ on a same line passing through the origin in the pairwise dissimilarity space.

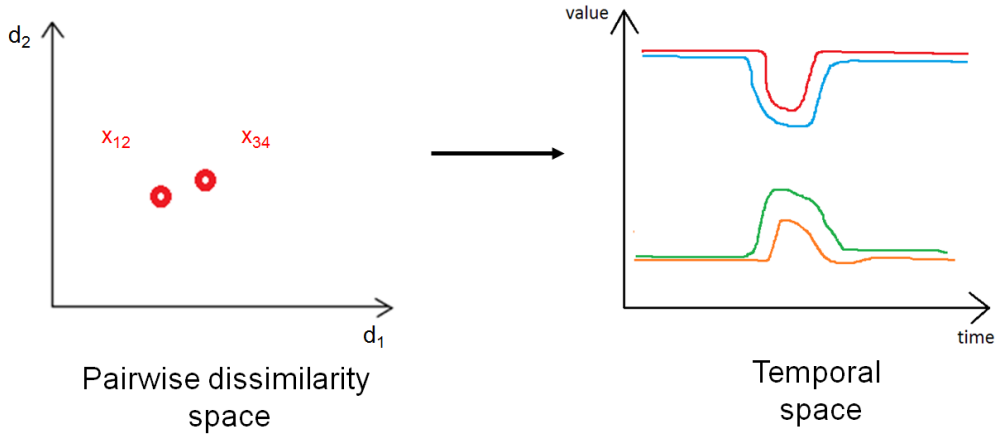


Figure 3.5: Example of two pairwise vectors $\mathbf{x}_{12}$ and $\mathbf{x}_{34}$ close in the pairwise dissimilarity space. However, the time series $\mathbf{x}_1$ and $\mathbf{x}_3$ are not similar in the temporal space.

3.3.3 Multi-scale description for time series

The multi-modal representation in the dissimilarity space can be enriched for time series by measuring each unimodal metric d_h at different temporal localization, called in this work scales. Note that the distance measures (amplitude-based d_A, frequential-based d_F, behavior-based d_B) in Eqs. 2.1, 2.4 and 2.6 imply systematically the total time series elements x_{it} and thus, restricts the distance measures to capture local temporal differences. In this work, we provide a multi-scale framework for time series comparison using a hierarchical structure. Many methods exist in the literature such as the sliding window [Keo+03] or the dichotomy [DCA12]. We detail here the latter one.

A multi-scale description can be obtained by repeatedly segmenting a time series expressed at a given temporal scale to induce its description at a more local level. Many approaches have been proposed assuming fixed either the number of the segments or their lengths [Fu11].

In this work, we consider a binary segmentation at each level. Let $I = [a; b]$ be a temporal interval of size $(b - a)$. The interval I is decomposed into two equal overlapped intervals I_L (left interval) and I_R (right interval). A parameter μ allows the two intervals I_L and I_R to be overlapped, covering discriminating subsequences in the central region of I (around $\frac{b+a}{2}$): $I = [a; b]; I_L = [a; a + \mu(b - a)]; I_R = [b - \mu(b - a); b]$. For $\mu = 0.6$, the overlap covers 10% of the size of the interval I. Then, the process is repeated on the intervals I_L and I_R. We obtain a set of intervals I_s illustrated in Fig. 3.6.
A multi-scale dissimilarity description between two time series is obtained by computing the usual time series metrics (d_A, d_B, d_F) on each localized temporal resulting segments I_s. Note that for two time series $\mathbf{x}_i$ and $\mathbf{x}_j$, the comparison between $\mathbf{x}_i$ and $\mathbf{x}_j$ is done on the same interval I_s. For a multi-scale amplitude-based comparison based on binary segmentation, the set of involved amplitude-based measures d_A^{Is} is $\{d_A^{I_1}, d_A^{I_2}, \ldots\}$ where d_A^{Is} is defined as:

$$d_A^{Is}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{t \in Is} (x_{it} - x_{jt})^2} \tag{3.12}$$

The local behaviors- and frequential- based measures d_B^{Is} and d_F^{Is} are obtained similarly. In

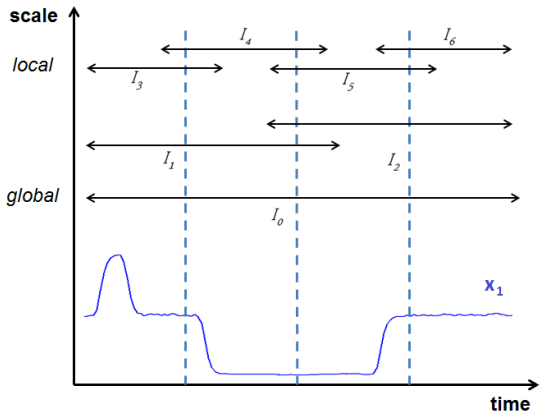


Figure 3.6: Multi-scale decomposition

the following, for simplification purpose, we consider $d_1, \ldots, d_p$ as the set of multi-modal and multi-scale metrics.

3.4 M²TML general problem

In this section, we propose to define the Multi-modal and Multi-scale Time series Metric Learning (M²TML) problem in the initial space as a general problem of learning a function in the pairwise dissimilarity space. First, we give the intuition and formalize the general optimization problem. Secondly, we propose different strategies to define the neighborhood. Thirdly, we give some more detailed interpretations of the M²TML problem in the pairwise dissimilarity space.

3.4.1 General formalization for M²TML

Our objective is to learn a dissimilarity $D = f(d_1, \ldots, d_p)$ in $\mathscr{E}$, the embedding space, that combines the p dissimilarities $d_1, \ldots, d_p$ for a robust k-NN classifier. The function f can be linear or non-linear and must satisfy at least the properties of a dissimilarity, *i.e.*, positivity ($D(\mathbf{x}_{ij} \geq 0)$), reflexivity ($D(\mathbf{x}_{ii}) = 0 \ \forall i$) and symmetry ($D(\mathbf{x}_{ij}) = D(\mathbf{x}_{ji}) \ \forall i, j$) (Section 2.2). In the following, the term metric is used to reference both a distance or a dissimilarity measure.

The proposition is based on two standard intuitions in metric learning, *i.e.*, for each time series $\mathbf{x}_i$, the metric D should bring closer the time series $\mathbf{x}_j$ of the same class ($y_j = y_i$) while pushing the time series $\mathbf{x}_l$ of different classes ($y_l \neq y_i$). These two sets are called respectively $Pull_i$ and $Push_i$. In addition, in order to have a robust k-NN, a safety margin between the resulting metric values between the sets $Pull_i$ and $Push_i$ must be considered. Our proposition is inspired from the LMNN framework where the optimization problem involves a pull term that penalizes large distances between sample of same labels ($Pull_i$). It can be interpreted as a regularization term on $Pull_i$. In LMNN, the push term penalizes small distances between samples of different labels ($Push_i$). It can be interpreted as a loss term on $Push_i$. To ensure a safety margin between similar and dissimilar samples, in LMNN, a constraint is added: $D^2(\mathbf{x}_{ij}) - D^2(\mathbf{x}_{il}) \geq 1 - \xi_{ijl}$.

Similarly, we formalize the M²TML problem as an optimization problem involving both a **regularization term** on D and the pull set $Pull_i$, denoted $R_{Pull}(D)$, and a **loss term** on ξ and the push set $Push_i$, denoted $L_{Push}(\xi)$. **A set of constraints** is added to control the push term in order to have a large margin between $Pull_i$ and $Push_i$:

$$\begin{aligned} &\underset{D,\xi}{\operatorname{argmin}} \{R_{Pull}(D) + L_{Push}(\xi)\} \\ &\text{s.t. } \forall i, j \in Pull_i, l \in Push_i, \\ &\quad D(\mathbf{x}_{il}) - D(\mathbf{x}_{ij}) \geq 1 - \xi_{ijl} \\ &\quad \xi_{ijl} \geq 0 \end{aligned} \tag{3.13}$$

Note that for the M²TML problem, the square of the distances in the constraints are not needed. Among the possibilities for the regularization term, we decide to choose to minimize the sum of the distances of the pull pairs. Among the possibilities for the loss term, we decide to choose to minimize the sum of the slack variables on the push pairs:

$$R_{Pull}(D) = \sum_{\substack{i \\ j \in Pull_i}} D(\mathbf{x}_{ij}) \tag{3.14}$$

$$L_{Push}(\xi) = \sum_{\substack{i \\ j \in Pull_i \\ l \in Push_i}} \xi_{ijl} \tag{3.15}$$

with $D = f(d_1, \ldots, d_p)$ combination of the metrics $d_1, \ldots, d_p$.

The M²TML problem for large margin k-NN classification can be written as the following

optimization problem:

$$\begin{aligned}
&\underset{D,\xi}{\operatorname{argmin}} \left\{ \underbrace{\sum_{\substack{i \\ j \in Pull_i}} D(\mathbf{x}_{ij})}_{pull} + C \underbrace{\sum_{\substack{i \\ j \in Pull_i \\ l \in Push_i}} \xi_{ijl}}_{push} \right\} \\
&\text{s.t. } \forall i = 1, \ldots, n, \forall j \in Pull_i, l \in Push_i, \\
&\quad D(\mathbf{x}_{il}) - D(\mathbf{x}_{ij}) \geq 1 - \xi_{ijl} \\
&\quad \xi_{ijl} \geq 0
\end{aligned} \tag{3.16}$$

where ξ_{ijl} are the slack variables and C, the trade-off between the pull (regularization) and push (loss) costs. In the next section, we detail different strategies to define the $Pull_i$ and $Push_i$ sets.

3.4.2 Push and pull set definition

To build the pairwise training set, we associate for each $\mathbf{x}_i$, two sets, $Pull_i$ and $Push_i$, where the two sets are chosen according to one of the following strategies, illustrated in Fig 3.7. Recall that the norm $D_0(\mathbf{x}_{ij}) = ||\mathbf{x}_{ij}||_2$ is set as our initial distance D_0.

1. **k-NN vs impostors**: for a given $\mathbf{x}_i$, the sets of pairs to pull and to push corresponds respectively to:

$$\forall i \in 1, \ldots, n, \quad Pull_i = \{\mathbf{x}_{ij} \ / \ y_j = y_i, \ D_0(\mathbf{x}_{ij}) \text{ is among the } k\text{-lowest distance}\} \tag{3.17}$$

$$Push_i = \{\mathbf{x}_{il} \ / \ y_l \neq y_i, \ D_0(\mathbf{x}_{il}) \leq \max_{\mathbf{x}_{ij} \in Pull_i} D_0(\mathbf{x}_{ij})\} \tag{3.18}$$

   Note that it corresponds to the definition of neighborhood defined by Weinberger & Saul.

2. **k-NN vs all**: for a given $\mathbf{x}_i$, the sets of pairs to pull and to push corresponds respectively to:

$$\forall i \in 1, \ldots, n, \quad Pull_i = \{\mathbf{x}_{ij} \ / \ y_j = y_i, \ D_0(\mathbf{x}_{ij}) \text{ is among the } k\text{-lowest distance}\} \tag{3.19}$$

$$Push_i = \{\mathbf{x}_{il} \ / \ y_l \neq y_i\} \tag{3.20}$$

   Note that by considering all samples of different classes, we ensure that a pair $\mathbf{x}_{il}$ doesn't become an imposter during the optimization process.

3. m-**NN**$^+$ **vs** m-**NN**$^-$: for a given $\mathbf{x}_i$, the pull and push sets are defined respectively as the set of the m-nearest neighbors of the same class ($y_j = y_i$), and the m-nearest neighbor of $\mathbf{x}_i$ of a different class ($y_j \neq y_i$) More precisely, our proposition states: $m = \alpha.k$ with $\alpha \geq 1$. Other propositions for m are possible:

$$\forall i \in 1, \ldots, n, \quad Pull_i = \{\mathbf{x}_{ij} \ / \ y_j = y_i, \ D_0(\mathbf{x}_{ij}) \text{ is among the } m\text{-lowest distance}\} \tag{3.21}$$

$$Push_i = \{\mathbf{x}_{il} \ / \ \text{s.t. } y_l \neq y_i, \ D_0(\mathbf{x}_{il}) \text{ is among the } m\text{-lowest distance}\} \tag{3.22}$$

In the following, we denote $m\text{-NN}^+ = \bigcup_i Pull_i$ and $m\text{-NN}^- = \bigcup_i Push_i$

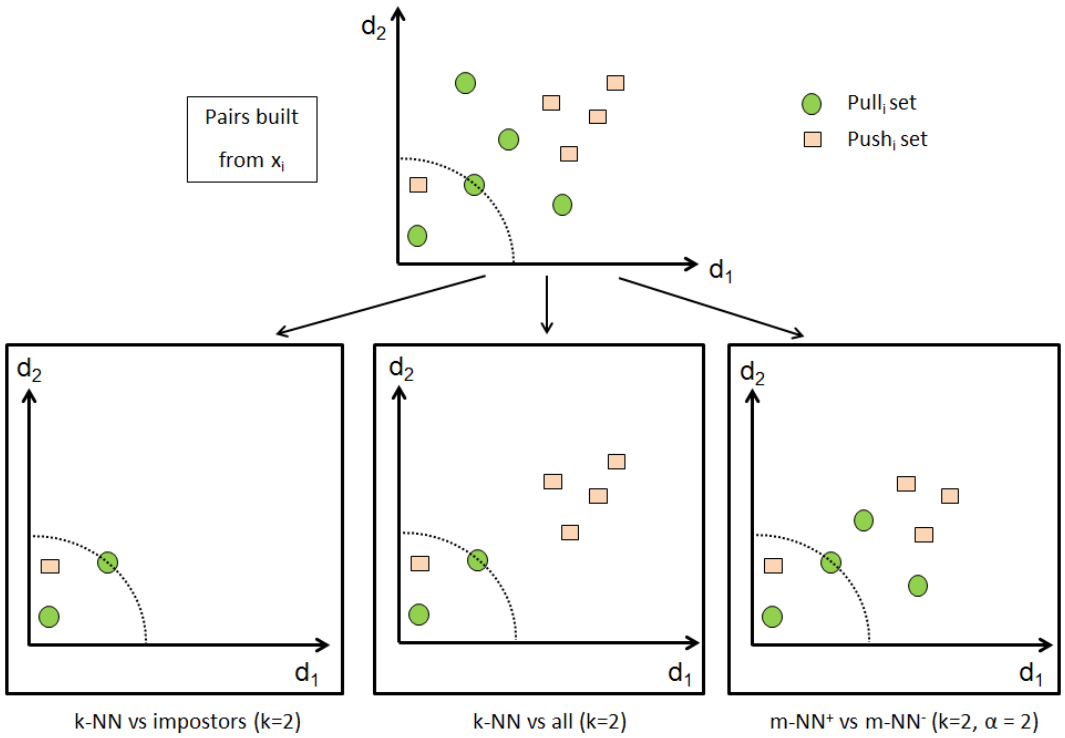


Figure 3.7: Example of different strategies to build $Pull_i$ and $Push_i$ sets for a $k = 2$ neighborhood.

Finally, let discuss about the similarities and differences between LMNN (Weinberger & Saul [WS09]) and our M²TML proposition. In LMNN, the sets $Pull_i$ and $Push_i$ are defined according the k-**NN vs impostors** strategy (Eqs. 3.17 & 3.18) and may be unbalanced. The sets are defined and fixed during the optimization process according to the initial metric D_0. In M²TML the sets $Pull_i$ and $Push_i$ are defined according the m-**NN**$^+$ **vs** m-**NN**$^-$ strategy (Eqs. 3.21 & 3.22) and are balanced. The sets are defined and fixed during the optimization process according to the initial metric D_0, but the m-neighborhood is larger than the k-neighborhood. By considering a neighborhood larger than the k-neighborhood, we believe that the generalization properties of the learned metric D will be improved.

3.4.3 Interpretation in the pairwise dissimilarity space

In this section, we give more detailed interpretations of the M²TML problem in the dissimilarity space. Our objective is to learn a metric D as a linear or non-linear combination of the p unimodal metrics $d_1, \ldots, d_p$. The metric D can be seen as a function of the dissimilarity space that should:

- **pull** to the origin $\mathbf{x}_{ii} = \mathbf{0}$ the pairs $\mathbf{x}_{ij}$ of $Pull_i$
- **push** away from the origin all the pairs $\mathbf{x}_{il}$ of $Push_i$

Fig. 3.8 illustrates the idea in the original space and in the pairwise dissimilarity space: first, we build the sets $Pull_i$ and $Push_i$ according to an initial metric D_0; secondly, we optimize the metric D so that the pairs $Pull_i$ are pulled to the origin and the pairs $Push_i$ are pushed away from the origin.

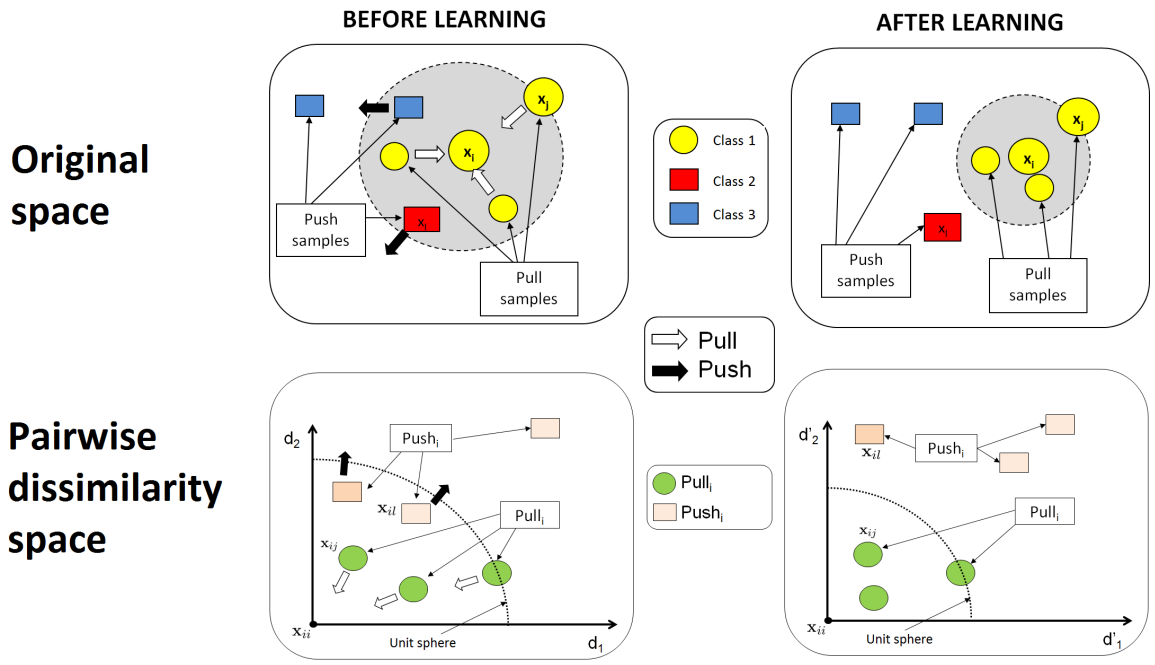


Figure 3.8: Metric learning problem in the original space (top) and the pairwise dissimilarity space (bottom) for a $k = 3$ neighborhood of $\mathbf{x}_i$. Before learning (left), push samples $\mathbf{x}_l$ invade the targets perimeter $\mathbf{x}_j$. In the dissimilarity pairwise space, this is equivalent to have push pairwise vectors $\mathbf{x}_{il}$ with an initial distance D_0 lower than the distance of pull pairwise vectors $\mathbf{x}_{ij}$. The aim of M²TML is to learn a metric D to push $\mathbf{x}_{il}$ (black arrow) and pull $\mathbf{x}_{ij}$ from the origin (white arrow).

Note that by considering a larger neighborhood, we ensure that pairs $Push_i$ doesn't invade the perimeter defined by pairs $Pull_i$ during the optimization process. Similarly to the interpretation of slack variables in SVM, if a push pair invade the perimeter defined by pairs $Pull_i$,

then in Eq. 3.16, it will violate the constraints and the slack variables ξ_{ijl} will be penalized in the objective function:

- If $D(\mathbf{x}_{il}) < D(\mathbf{x}_{ij})$, then the pair $\mathbf{x}_{il}$ is an imposter pair that invades the neighborhood of the target pairs $\mathbf{x}_{ij}$. The slack variable $\xi_{ijl} > 1$ will be penalized in the objective function.

- If $D(\mathbf{x}_{ij}) \leq D(\mathbf{x}_{il}) \leq D(\mathbf{x}_{ij}) + 1$, the pair $\mathbf{x}_{il}$ is within the safety margin of the target pairs $\mathbf{x}_{ij}$. The slack variable $\xi_{ijl} \in [0; 1]$ will have a small penalization effect in the objective function.

- If $D(\mathbf{x}_{il}) > D(\mathbf{x}_{ij}) + 1$, $\xi_{ijl} = 0$ and the slack variable has no effect in the objective function.

In the following, we propose different regularizers for the pull term $R_{Pull}(D)$. First, we use a linear regularization. Secondly, we use a quadratic regularization that enables to extend the approach to learn non-linear functions for D by using the "kernel" trick. Thirdly, we formulate the problem as a SVM problem to solve a large margin problem between $Pull_i$ and $Push_i$ sets, and then, we define the combined metric D based on the SVM solution. Finally, we sum up the retained solution (SVM-based solution) and give the main steps of the algorithm.

3.5 Linear formalization for M^{2}TML

In this section, we define the problem of learning a combined metric D as a linear combination in the dissimilarity space using a linear regularizer. First, we give the M^{2}TML optimization problem for a linear regularizer. Then, we discuss the properties of the learned metric D.

Let $\{\mathbf{x}_{ij}\}_{i,j=1}^{n}$ be a set of pairwise vectors $\mathbf{x}_{ij} = [d_1(\mathbf{x}_i, \mathbf{x}_j), ..., d_p(\mathbf{x}_i, \mathbf{x}_j)]^T$ described by p metrics $d_1, \ldots, d_p$. We consider a linear combination of the p metrics:

$$D(\mathbf{x}_{ij}) = \mathbf{w}^T \mathbf{x}_{ij} = \sum_{h=1}^{p} w_h . d_h(\mathbf{x}_i, \mathbf{x}_j) \tag{3.23}$$

where $\mathbf{w} = [w_1, \ldots, w_p]^T$ is the vector of weights w_h. From Eq. 3.16, by choosing $R_{Pull}(D) =$

$R_{Pull}(\mathbf{w}) = \sum_{i,j \in Pull_i} \mathbf{w}^T \mathbf{x}_{ij}$, learning a linear combined metric D can be formalized as follow:

$$\underset{\mathbf{w},\xi}{\operatorname{argmin}} \left\{ \underbrace{\sum_{\substack{i \\ j \in Pull_i}} \mathbf{w}^T \mathbf{x}_{ij}}_{pull} + C \underbrace{\sum_{\substack{i \\ j \in Pull_i \\ l \in Push_i}} \xi_{ijl}}_{push} \right\}$$

$$\text{s.t. } \forall i = 1, \ldots, n, \forall j \in Pull_i, l \in Push_i, \tag{3.24}$$
$$\mathbf{w}^T(\mathbf{x}_{il} - \mathbf{x}_{ij}) \geq 1 - \xi_{ijl}$$
$$\xi_{ijl} \geq 0$$

$$\forall h = 1, \ldots, p, \quad w_h \geq 0 \tag{3.25}$$

where ξ_{ijl} are the slack variables, C the trade-off between the pull and push costs, and $Pull_i$ and $Push_i$ are defined in Eqs. 3.21 & 3.22.

Note that the problem is very similar to a C-SVM classification problem. When C is infinite, we have a "strict" problem: the solver will try to find a direction $\mathbf{w}$ in the dissimilarity space $\mathscr{E}$ for which all $\xi_{ijl} = 0$, that means that only pull samples should be in the close neighborhood of each $\mathbf{x}_i$. Let denote $\mathbf{x}^*_{ij}$ and $\mathbf{x}^*_{il}$, the vectors for which $\xi_{ijl} = 0$. In that case, if a solution is found, the margin $\min_{i,j,l}(||\mathbf{x}^*_{il} - \mathbf{x}^*_{ij}||_2)$ can be derived from the tightest constraint, for which equality holds:

$$\begin{aligned} \mathbf{w}^T(\mathbf{x}^*_{il} - \mathbf{x}^*_{ij}) &= 1 \\ ||\mathbf{w}||_2 ||\mathbf{x}^*_{il} - \mathbf{x}^*_{ij}||_2 &= 1 \\ ||\mathbf{x}^*_{il} - \mathbf{x}^*_{ij}||_2 &= \frac{1}{||\mathbf{w}||_2} \end{aligned}$$

Concerning the properties of D, positivity is ensured with the constraints $w_h \geq 0$ (Eq. 3.25) and because $d_1, \ldots, d_p$ are dissimilarity measures ($d_h \geq 0$). As the metric D is defined as a linear combination of dissimilarity measures $d_1, \ldots, d_p$, it can be shown that symmetry and reflexivity is verified.

3.6 Quadratic formalization for M^{2}TML

In this section, we define the problem of learning D as a linear or non-linear combination in the dissimilarity space using a quadratic regularizer. First, we give the optimization problem and its dual formulation form involving only dot products. Then, we discuss on the properties

of the learned metric D. Finally, we study a link between SVM and the quadratic formalization.

3.6.1 Primal and dual formalization

The formulation in Eq. 3.24 supposes that the metric D is a linear combination of the metrics d_h. The linear formalization being similar to the one of a L_1-regularized SVM, it can be derived into a dual form involving only dot-products to extend the method to find non-linear solutions for D. For that, we propose to change the linear regularizer $R_{Pull}(\mathbf{w})$ in the objective function of Eq. 3.24 into a quadratic regularizer. Two solutions for $R_{Pull}(\mathbf{w})$ are at least possible:

$$1. \quad R_{Pull}(\mathbf{w}) = \frac{1}{2}\sum_{h=1}^{p}\sum_{\substack{i \\ j\in Pull_i}} (w_h d_h(\mathbf{x}_{ij}))^2 \tag{3.26}$$

$$2. \quad R_{Pull}(\mathbf{w}) = \frac{1}{2}\sum_{h=1}^{p}\left(\sum_{\substack{i \\ j\in Pull_i}} w_h d_h(\mathbf{x}_{ij})\right)^2 = \frac{1}{2}m.n\sum_{h=1}^{p}\left(w_h \bar{d_h}\right)^2 \tag{3.27}$$

where $\bar{d_h} = \frac{1}{mn}\sum\limits_{\substack{i \\ j\in Pull_i}} d_h(\mathbf{x}_{ij})$ denotes the mean of the distances $d_h(\mathbf{x}_{ij})$ for each metric d_h.

Other regularizations are possible. We focus on these two propositions that can be reduced to the following formula:

$$R(Pull) = \frac{1}{2}\mathbf{w}^T\mathbf{M}\mathbf{w} \tag{3.28}$$

where $\mathbf{M}$ denotes respectively the following matrix for each regularizer:

$$1. \quad \mathbf{M} = Diag(\mathbf{X}_{pull}^T\mathbf{X}_{pull}) = \begin{bmatrix} \sum\limits_{\substack{i \\ j\in Pull_i}} d_1^2(\mathbf{x}_{ij}) & & 0 \\ & \ddots & \\ 0 & & \sum\limits_{\substack{i \\ j\in Pull_i}} d_p^2(\mathbf{x}_{ij}) \end{bmatrix} \tag{3.29}$$

$$2. \quad \mathbf{M} = Diag(\bar{\mathbf{x}}).Diag(\bar{\mathbf{x}}) = \begin{bmatrix} \bar{d_1}^2 & & 0 \\ & \ddots & \\ 0 & & \bar{d_p}^2 \end{bmatrix} \tag{3.30}$$

where $\mathbf{X}_{pull} = \bigcup_i Pull_i$ be a $(m.n) \times p$ matrix containing the vector $\mathbf{x}_{ij} \in Pull_i$ and $\bar{\mathbf{x}} = [\bar{d_1}, \ldots, \bar{d_p}]^T$ is a vector of size p containing the mean of the metrics $\bar{d_1}, \ldots, \bar{d_p}$.

From this, the optimization problem can be written using a quadratic regularization for the

pull term:

$$\begin{aligned}\underset{\mathbf{w},\xi}{\operatorname{argmin}} \left\{ \underbrace{\frac{1}{2}\mathbf{w}^T\mathbf{M}\mathbf{w}}_{pull} + C \underbrace{\sum_{i} \sum_{\substack{j\in Pull_i \\ l\in Push_i}} \xi_{ijl}}_{push} \right\} \\ \text{s.t. } \forall i = 1,\dots,n, \forall j \in Pull_i, l \in Push_i, \\ \mathbf{w}^T(\mathbf{x}_{il} - \mathbf{x}_{ij}) \geq 1 - \xi_{ijl} \\ \xi_{ijl} \geq 0\end{aligned} \tag{3.31}$$

Note that in this case, the constraint $w_h \geq 0$ (Eq. 3.25) is not considered because the following development would not allow us to obtain a formulation with only dot-product.
Similarly to SVM, the formulation in Eq. 3.31 can be reduced to the maximization of the following Lagrange function $L(\mathbf{w}, \xi, \boldsymbol{\alpha}, \mathbf{r})$, consisting of the sum of the objective function and the constraints multiplied by their respective Lagrange multipliers $\boldsymbol{\alpha}$ and $\mathbf{r}$:

$$\begin{aligned}L(\mathbf{w}, \xi, \boldsymbol{\alpha}, \mathbf{r}) =& \frac{1}{2}\mathbf{w}^T\mathbf{M}\mathbf{w} + C\sum_{ijl}\xi_{ijl} - \sum_{ijl} r_{ijl}\xi_{ijl} \\ & - \sum_{ijl} \alpha_{ijl}\left(\mathbf{w}^T(\mathbf{x}_{il} - \mathbf{x}_{ij}) - 1 + \xi_{ijl}\right)\end{aligned} \tag{3.32}$$

where $\alpha_{ijl} \geq 0$ and $r_{ijl} \geq 0$ are the Lagrange multipliers. At the maximum value of $L(\mathbf{w}, \xi, \boldsymbol{\alpha}, \mathbf{r})$, the derivatives with respect to $\mathbf{w}$ and ξ_{ijl} are set to zero:

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{w}} &= \mathbf{M}\mathbf{w} - \sum_{ijl}\alpha_{ijl}(\mathbf{x}_{il} - \mathbf{x}_{ij}) = 0 \\ \frac{\partial L}{\partial \xi_{ijl}} &= C - \alpha_{ijl} - r_{ijl} = 0\end{aligned}$$

The matrix $\mathbf{M}$ being diagonal in both case (Eqs. 3.29 & 3.30), it is thus invertible. The equations lead to:

$$\mathbf{w} = \mathbf{M}^{-1}\sum_{ijl}\alpha_{ijl}(\mathbf{x}_{il} - \mathbf{x}_{ij}) \tag{3.33}$$

$$r_{ijl} = C - \alpha_{ijl} \tag{3.34}$$

Substituting Eq. 3.33 and 3.34 back into $L(\mathbf{w}, \xi, \boldsymbol{\alpha}, \mathbf{r})$ in Eq. 3.32, we get the dual formulation

(details of the development can be found in Appendix A):

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \left\{ \sum_{ijl} \alpha_{ijl} - \frac{1}{2} \sum_{ijl} \sum_{i'j'l'} \alpha_{ijl}\alpha_{i'j'l'} (\mathbf{x}_{il} - \mathbf{x}_{ij})^T \mathbf{M}^{-1} (\mathbf{x}_{i'l'} - \mathbf{x}_{i'j'}) \right\} \\ &\text{s.t. } \forall i = 1, \ldots, n, \forall j \in Pull_i, l \in Push_i, \\ &0 \leq \alpha_{ijl} \leq C \end{aligned} \tag{3.35}$$

For any new pair of samples $\mathbf{x}_{i'}$ and $\mathbf{x}_{j'}$, the resulting metric D writes:

$$D(\mathbf{x}_{i'j'}) = \underbrace{\sum_{ijl} \alpha_{ijl} (\mathbf{x}_{il} - \mathbf{x}_{ij})^T \mathbf{M}^{-1}}_{\mathbf{w}^T} \mathbf{x}_{i'j'} \tag{3.36}$$

By developing Eq. 3.35, the dual formulation is equivalent to:

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \left\{ \sum_{ijl} \alpha_{ijl} - \frac{1}{2} \sum_{ijl} \sum_{i'j'l'} \alpha_{ijl}\alpha_{i'j'l'} \left(\mathbf{x}_{il}^T \mathbf{M}^{-1} \mathbf{x}_{i'l'} - 2\mathbf{x}_{ij}^T \mathbf{M}^{-1} \mathbf{x}_{i'l'} + \mathbf{x}_{ij}^T \mathbf{M}^{-1} \mathbf{x}_{i'j'} \right) \right\} \\ &\text{s.t. } \forall i = 1, \ldots, n, \forall j \in Pull_i, l \in Push_i, \\ &0 \leq \alpha_{ijl} \leq C \end{aligned} \tag{3.37}$$

And the metric in Eq. 3.36 writes:

$$D(\mathbf{x}_{i'j'}) = \underbrace{\sum_{ijl} \alpha_{ijl} \mathbf{x}_{il}^T \mathbf{M}^{-1} \mathbf{x}_{i'j'}}_{\text{similarity of } \mathbf{x}_{i'j'} \text{ to } Push \text{ set}} - \underbrace{\sum_{ijl} \alpha_{ijl} \mathbf{x}_{ij}^T \mathbf{M}^{-1} \mathbf{x}_{i'j'}}_{\text{similarity of } \mathbf{x}_{i'j'} \text{ to } Pull \text{ set}} \tag{3.38}$$

3.6.2 Non-linear combined metric

The above formula (Eqs. 3.37 and 3.38) can be extended to find non-linear function for the metric D. As $\mathbf{M}$ is a diagonal matrix, it is invertible and can be written $\mathbf{M}^{-1} = \mathbf{M}^{-\frac{1}{2}}\mathbf{M}^{-\frac{1}{2}}$. For each regularization, we give below the matrix $\mathbf{M}^{-\frac{1}{2}}$:

$$1.\quad \mathbf{M}^{-\frac{1}{2}} = \begin{bmatrix} \frac{1}{\sqrt{\sum_{j \in Pull_i} \sum_i d_1^2(\mathbf{x}_{ij})}} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\sqrt{\sum_{j \in Pull_i} \sum_i d_p^2(\mathbf{x}_{ij})}} \end{bmatrix} \tag{3.39}$$

$$2.\quad \mathbf{M}^{-\frac{1}{2}} = \begin{bmatrix} \frac{1}{d_1} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{d_p} \end{bmatrix} \tag{3.40}$$

Then, the formulation in Eqs. 3.37 and 3.38 can be written to involve only an inner product between pairs:

$$\begin{aligned} \mathbf{x}_{il}^T \mathbf{M}^{-1} \mathbf{x}_{i'j'} &= \mathbf{x}_{il}^T \mathbf{M}^{-\frac{1}{2}} \mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{i'j'} \\ &= \left(\mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{il}\right)^T \left(\mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{i'j'}\right) \\ &=< \mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{il}; \mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{i'j'} > \end{aligned}$$

The inner product can be easily kernelized using the "kernel" trick:

$$< \mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{il}; \mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{i'j'} > = \kappa(\mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{il}; \mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{i'j'})$$

The matrix $\mathbf{M}^{-\frac{1}{2}}$ can be thus interpreted in the first regularization proposition as a normalization by the variance of the distance for each metric d_h. In the second regularization, it can be interpreted as a normalization by the mean of the distance for each metric d_h.

By replacing the inner product by a kernel back into Eq. 3.37, the kernelized dual formulation becomes:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_{ijl} \alpha_{ijl} - \frac{1}{2} \sum_{ijl} \sum_{i'j'l'} \alpha_{ijl} \alpha_{i'j'l'} (\kappa(\mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{il}; \mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{i'l'}) - 2\kappa(\mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{ij}; \mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{i'l'}) \\ + \kappa(\mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{ij}; \mathbf{M}^{-\frac{1}{2}} \mathbf{x}_{i'j'})) \end{aligned} \tag{3.41}$$

s.t. $\forall i = 1, \ldots, n, \forall j \in Pull_i, l \in Push_i$,

$0 \leq \alpha_{ijl} \leq C$

By replacing the inner product by a kernel back into Eq. 3.38, we obtain:

$$D(\mathbf{x}_{i'j'}) = \overbrace{\sum_{ijl} \alpha_{ijl} \kappa(\mathbf{M}^{-\frac{1}{2}}\mathbf{x}_{il}; \mathbf{M}^{-\frac{1}{2}}\mathbf{x}_{i'j'})}^{\text{similarity of } \mathbf{x}_{i'j'} \text{ to } Push \text{ set}} - \overbrace{\sum_{ijl} \alpha_{ijl} \kappa(\mathbf{M}^{-\frac{1}{2}}\mathbf{x}_{ij}; \mathbf{M}^{-\frac{1}{2}}\mathbf{x}_{i'j'})}^{\text{similarity of } \mathbf{x}_{i'j'} \text{ to } Pull \text{ set}} \tag{3.42}$$

Let's give some interpretation and discussion about the properties of D. Similarly to SVM, from Eq. 3.36, at the optimality, only the triplets $(\mathbf{x}_{il} - \mathbf{x}_{ij})$ with $\alpha_{ijl} > 0$ are considered as the support vectors and the computation of the metric D depends only on these support vectors. Note that in this case, there exist two categories of support vectors (Eqs. 3.38 & 3.42): the vectors $\mathbf{x}_{il}$ from the push set $Push_i$ and the vectors $\mathbf{x}_{ij}$ from the pull set $Pull_i$ which $\alpha_{ijl} > 0$. The resulting metric D can be interpreted as the difference involving two similarity terms: a new pair $\mathbf{x}_{i'j'}$ is dissimilar when its similarity to the $Push$ set is high while its similarity to the $Pull$ set is low. Inversely, the pair $\mathbf{x}_{i'j'}$ is similar when its similarity to the $Push$ set is low while its similarity to the $Pull$ set is high.

Concerning the property of D, it is not a dissimilarity as non positive: D is a difference of two similarities, the similarity of the pull term is greater than the similarity of the push term.

3.6.3 Link between SVM and the quadratic formalization

Parallels between Large Margin Nearest Neighbors (LMNN) and SVM have been studied in the literature [Do+12]. SVM is a well known framework: its has been well implemented in many libraries (*e.g.*, LIBLINEAR [FCH08] and LIBSVM [HCL08]), well studied for its generalization properties and extension to non-linear solutions (Section 1.2.2).

Similarly, we study in this section a link between the quadratic formalization of M^2TML and a SVM problem when the form of the metric D is defined *a priori*. For that, let consider the following SVM problem that aims to separate the set $Pull_i$ and $Push_i$:

$$\begin{aligned} &\underset{\mathbf{w},b,\boldsymbol{\xi}}{\operatorname{argmin}} \left\{ \frac{1}{2}||\mathbf{w}||_2^2 + C \sum_{\substack{i \\ j \in Pull_i \text{ or} \\ j \in Push_i}} p_i \xi_{ij} \right\} \\ &\text{s.t. } \forall i, j \in Pull_i \text{ or } j \in Push_i : \\ &y_{ij}(\mathbf{w}^T \mathbf{x}_{ij} + b) \geq 1 - \xi_{ij} \\ &\xi_{ij} \geq 0 \end{aligned} \tag{3.43}$$

where p_i is a weight factor for each slack variable ξ_{ij} (in classical SVM, $p_i = 1$).

The loss part in the SVM formulation can be split into 2 terms involving the sets $Pull_i$ and $Push_i$:

$$\begin{aligned}
&\underset{\mathbf{w},b,\boldsymbol{\xi}}{\operatorname{argmin}} \left\{ \frac{1}{2}||\mathbf{w}||_2^2 + C \sum_{\substack{i \\ j \in Pull_i}} p_i^+ \xi_{ij} + C \sum_{\substack{i \\ l \in Push_i}} p_i^- \xi_{il} \right\} \\
&\text{s.t. :} \\
&\forall i,j \in Pull_i : y_{ij}(\mathbf{w}^T \mathbf{x}_{ij} + b) \geq 1 - \xi_{ij} \\
&\forall i,l \in Push_i : y_{il}(\mathbf{w}^T \mathbf{x}_{il} + b) \geq 1 - \xi_{il} \\
&\forall i,j \in Pull_i : \xi_{ij} \geq 0 \\
&\forall i,l \in Push_i : \xi_{il} \geq 0
\end{aligned} \tag{3.44}$$

where p_i^+ and p_i^- are the weight factors for pull pairs $Pull_i$ and push pairs $Push_i$.

We show in Appendix B that solving the SVM problem in Eq. 3.44 for $\mathbf{w}$ and b solves a similar problem with a quadratic regularization in Eq. 3.31 for $D(\mathbf{x}_{ij}) = -\frac{1}{2}(\mathbf{w}^T \mathbf{x}_{ij} + b)$ and where p_i^- and p_i^+ are defined as:

$$p_i^- = \frac{Card(Pull_i)}{2} = \sum_{j \in Pull_i} \frac{1}{2} \tag{3.45}$$

$$p_i^+ = \frac{Card(Push_i)}{2} = \sum_{l \in Push_i} \frac{1}{2} \tag{3.46}$$

where $Card(Pull_i)$ and $Card(Push_i)$ denotes respectively the cardinal of the set $Pull_i$ and $Push_i$ (equal to m in the m-**NN**$^+$ **vs** m-**NN**$^-$ strategy). p_i^- can be interpreted as the half of the number pairs in $Pull_i$ and p_i^+ as the half of the number of time series in $Push_i$. Let define $\xi_{ijl} = \frac{\xi_{ij}+\xi_{il}}{2}$ and $\xi_{ijkl} = \frac{\xi_{ij}+\xi_{kl}}{2}$.

Let's underline below the main similarities and differences between the SVM problem in Eq. 3.44 and the quadratic formalization of M^{2}TML in Eq. 3.31:

- Both problems suppose at first a linear combination for D.
- Both problems can be extended to learn non-linear combinations for D thanks to the kernel trick.
- The two problems involve different regularization terms: in the quadratic formalization, the regularizer involves a pull action (Eqs. 3.26 & 3.27), not present in SVM.
- Concerning the constraints and the slack variables:
  - Both problems share a same set of constraints between triplets:

$$\forall i,j \in Pull_i, l \in Push_i : D(\mathbf{x}_{il}) - D(\mathbf{x}_{ij}) \geq 1 - \xi_{ijl}$$

  - The SVM problem includes an additional set of constraints that is not present in the quadratic formalization. SVM takes into account pull pairs $\mathbf{x}_{ij}$ and push pairs

$\mathbf{x}_{kl}$ that don't belong to the same neighborhood:

$$\forall i, j \in Pull_i, k, l \in Push_k, i \neq k : D(\mathbf{x}_{kl}) - D(\mathbf{x}_{ij}) \geq 1 - \xi_{ijkl}$$

Geometrically, the global SVM margin includes both local neighborhoods and "inter-neighborhood" between pull and push pairs of different neighborhood.

– The SVM problem includes in the loss term additional slack variables ξ_{ijkl} that are not present in the quadratic formalization because of the additional set of constraints. It is not only a push term.

Concerning the properties of the metric D, positivity is not ensured in the primal and dual formulation as there are no constraint on $\mathbf{w}$. Symmetry and reflexivity is ensured.

3.7 SVM-based formalization for M^2TML

In this section,we formulate the problem as a SVM problem to solve a large margin problem between $Pull_i$ and $Push_i$ sets, and then, induce a combined metric D for the obtained SVM solution. Thanks to the SVM framework, the proposition can be naturally extended to learn both, linear or non-linear functions for the metric D.

3.7.1 Support Vector Machine (SVM) resolution

Let $\{\mathbf{x}_{ij}, y_{ij} = \pm 1\}$, $\mathbf{x}_{ij} \in Pull_i \cup Push_i$ be the training set, with $y_{ij} = -1$ for $\mathbf{x}_{ij} \in Push_i$ and $+1$ for $\mathbf{x}_{ij} \in Pull_i$. For a maximum margin between the sets $Pull_i$ and $Push_i$, the problem is formalized in the dissimilarity space $\mathscr{E}$:

$$\begin{aligned} &\underset{\mathbf{w},b,\boldsymbol{\xi}}{\operatorname{argmin}} \left\{ \frac{1}{2}||\mathbf{w}||_2^2 + C \sum_{i,j} \xi_{ij} \right\} \\ &\text{s.t. } y_{ij}(\mathbf{w}^T \mathbf{x}_{ij} + b) \geq 1 - \xi_{ij} \\ &\xi_{ij} \geq 0 \end{aligned} \tag{3.47}$$

In the linear case, a L_1 regularization ($||\mathbf{w}||_1$) in Eq. 3.47 leads to a sparse and interpretable $\mathbf{w}$ that uncovers the modalities, periods and scales that differentiate best pull from push pairs for a robust nearest neighbors classification. In practice, the local neighborhoods for each sample $\mathbf{x}_i$ can have very different scales. Thanks to the unit radii normalization $\mathbf{x}_{ij}/r_i$, where r_i denotes the norm of the m-th neighbors in $Pull_i$, the SVM ensures a global large margin solution involving equally local neighborhood constraints (*i.e.*, local margins). This point will be detailed in Section 3.8.

3.7.2 Solution for the linearly separable Pull and Push sets

Let $\mathbf{x}_{test}$ be a new sample, $\mathbf{x}_{i,test} \in \mathscr{E}$ gives the proximity between $\mathbf{x}_i$ and $\mathbf{x}_{test}$ based on the p multi-modal and multi-scale metrics d_h. The objective being to predict the label y_{test} of $\mathbf{x}_{test}$ using a k-NN classifier, it is necessary to define the metric $D(\mathbf{x}_{i,test})$. We review in this section different interpretations in the dissimilarity space.

M^{2}TML metric definition

Given a test pair $\mathbf{x}_{i,test}$, the norm $||\mathbf{x}_{i,test}||$ of the pair allows to estimate the proximity between $\mathbf{x}_i$ and $\mathbf{x}_{test}$. In particular, for M^{2}TML, two quantities are used to define the dissimilarity measure: the projected norm and the distance to the margin.

Let denote $\mathbf{P}_{\mathbf{w}}(\mathbf{x}_{i,test})$, the orthogonal projection of $\mathbf{x}_{i,test}$ on the axis of direction $\mathbf{w}$:

$$\mathbf{P}_{\mathbf{w}}(\mathbf{x}_{i,test}) = \frac{< \mathbf{w}, \mathbf{x}_{i,test} >}{||\mathbf{w}||^2}\mathbf{w} = \frac{\mathbf{w}^T\mathbf{x}_{i,test}}{||\mathbf{w}||^2}\mathbf{w} \tag{3.48}$$

The projected norm $||\mathbf{P}_{\mathbf{w}}(\mathbf{x}_{i,test})||$ of $\mathbf{x}_{i,test}$ on the direction $\mathbf{w}$ limits the comparison of $\mathbf{x}_i$ and $\mathbf{x}_{test}$ to the features separating pull and push sets (Fig. 3.9), it is defined as:

$$||\mathbf{P}_{\mathbf{w}}(\mathbf{x}_{i,test})|| = \frac{|\mathbf{w}^T\mathbf{x}_{i,test}|}{||\mathbf{w}||_2} \tag{3.49}$$

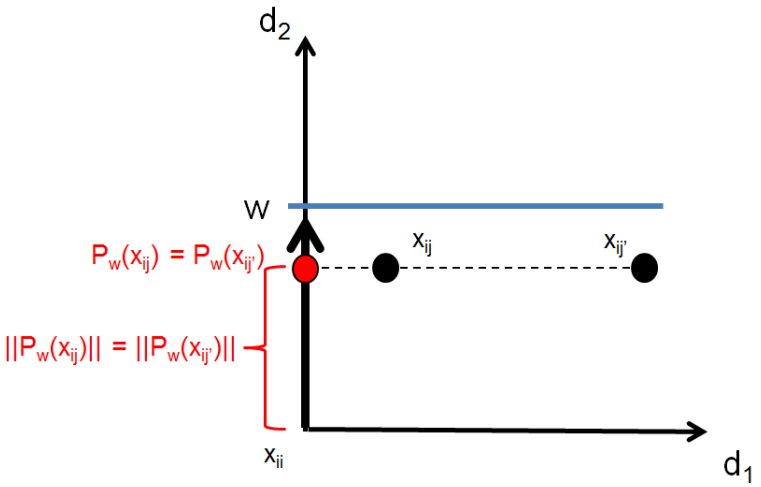


Figure 3.9: The projected vector $\mathbf{P}_{\mathbf{w}}(\mathbf{x}_{ij})$ and $\mathbf{P}_{\mathbf{w}}(\mathbf{x}_{ij'})$

Although the norm $||\mathbf{P}_{\mathbf{w}}(\mathbf{x}_{i,test})||$ satisfies positivity, it doesn't guarantee lower distances for pull pairs than for push pairs as illustrated in Fig 3.10.

Note that the distance of the projection to the margin $\mathbf{w}^T\mathbf{P}_{\mathbf{w}}(\mathbf{x}_{i,test}) + b$ gives the membership of the projected vector $\mathbf{P}_{\mathbf{w}}(\mathbf{x}_{i,test})$ in the pull or push side. However, it can't used as a dissimilarity (non-positivity).

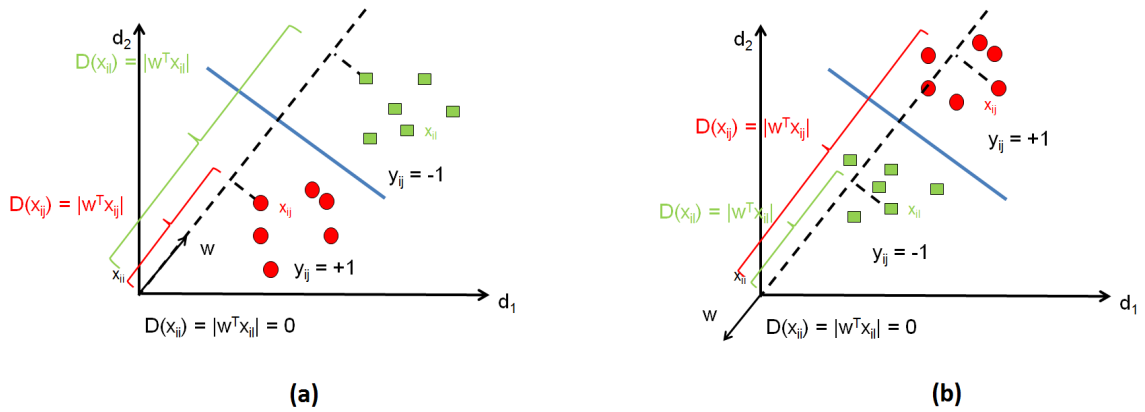


Figure 3.10: Example of SVM solutions and of the resulting metric D defined by the norm of the projection on $\mathbf{w}$. Fig. (a) represents common expected configuration where pull pairs $Pull_i$ are situated in the same side as the origin $\mathbf{x}_{ii} = \mathbf{0}$. In Fig. (b), the vector $\mathbf{w} = [-1 - 1]^T$ indicates that push pairs $Push_i$ are on the side of the origin point. One problem arises in Fig. (b): distance of push pairs $D(\mathbf{x}_{il})$ is lower than the distance of pull pairs $D(\mathbf{x}_{ij})$.

We propose to add an exponential term to operate a "push" on push pairs based on their distances to the separator hyperplan, that leads to the dissimilarity measure D of required properties:

$$D(\mathbf{x}_{i,test}) = ||\mathbf{P}_{\mathbf{w}}(\mathbf{x}_{i,test})||.\exp(\lambda[-(\mathbf{w}^T\mathbf{P}_{\mathbf{w}}(\mathbf{x}_{i,test}) + b)]_+) \quad \lambda \geq 0 \tag{3.50}$$

where λ controls the "push" term and $\mathbf{w}^T\mathbf{P}_{\mathbf{w}}(\mathbf{x}_{i,test}) + b$ defines the distance between the orthogonal projected vector and the separator hyperplane; $[t]_+ = \max(0; t)$ being the positive operator. Note that, for a pair lying into the pull side ($y_{ij} = +1$), $[-(\mathbf{w}^T\mathbf{P}_{\mathbf{w}}(\mathbf{x}_{i,test})+b)]_+ = 0$, the exponential term is vanished (i.e. no "pull" action) and the dissimilarity leads to the norm term. For a pair situated in the push side ($y_{ij} = -1$), the norm is expanded by the push term, all the more the distance to the hyperplane is high.

Fig. 3.11, illustrates for $p = 2$ the behavior of the learned dissimilarity according to two extreme cases.
The first one (Fig. 3.11-a), represents common expected configuration where pairs $Pull_i$ are situated in the same side as the origin. The dissimilarity increases proportionally to the norm in the pull side, then exponentially on the push side. Although the expansion operated in the push side is dispensable in that case, it doesn't affect nearest neighbors classification.
Fig. 3.11-b, shows a challenging configuration where pairs $Push_i$ are situated in the same side as the origin. The dissimilarity behaves proportionally to the norm on the pull side, and increases exponentially from the hyperplane until an abrupt decrease induced by a norm near 0. Note that the region under the abrupt decrease mainly uncovers false pairs $Push_i$, *i.e.*, pairs of norm zero labeled differently.

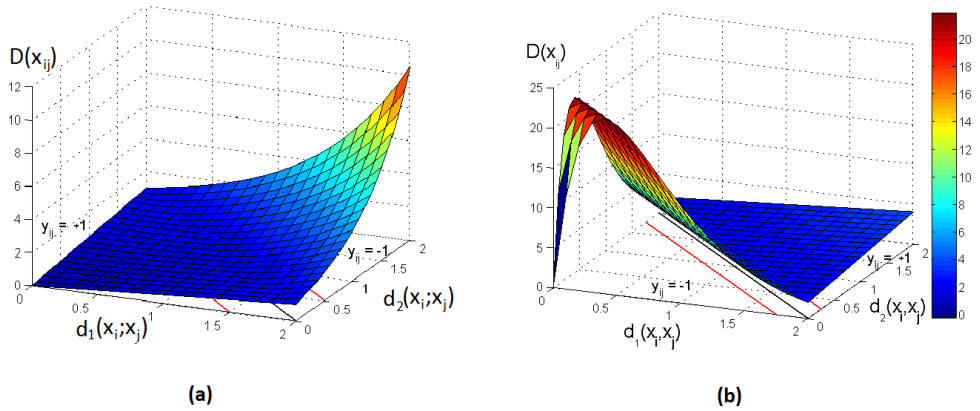


Figure 3.11: The behavior of the learned metric D ($p = 2$; $\lambda = 2.5$) with respect to common (a) and challenging (b) configurations of pull and push pairs.

3.7.3 Solution for the non-linearly separable Pull and Push sets

The above solution holds true for any kernel κ and allows us to extend the dissimilarity D given in Eq. 3.50 to non linearly separable pull and push pairs. Let κ be a kernel defined in the dissimilarity space $\mathscr{E}$ and the related Hilbert space (feature space) $\mathscr{H}$. For a non linear combination function of the metrics $d_h, h = 1, \ldots, p$ in $\mathscr{E}$, we define the dissimilarity measure $D_{\mathscr{H}}$ in the feature space $\mathscr{H}$ as:

$$D_{\mathscr{H}}(\mathbf{x}_{i,test}) = \mid ||\mathbf{P}_{\mathbf{w}}(\Phi(\mathbf{x}_{i,test}))|| - ||\mathbf{P}_{\mathbf{w}}(\Phi(\mathbf{0}))|| \mid . \exp\left(\lambda \left[- \left(\sum_{ij} y_{ij}\alpha_{ij}\kappa(\mathbf{x}_{ij}, \mathbf{x}_{i,test}) + b \right) \right]_+ \right) \quad \lambda \geq 0 \tag{3.51}$$

with $\Phi(\mathbf{x}_{i,test})$ and $\Phi(\mathbf{0})$ denotes the image of $\mathbf{x}_{i,test}$ and $\mathbf{0}$ into the feature space $\mathscr{H}$. Based on Eq. 3.48, from the known SVM equations (Section 1.2.2), the inner product gives $< \mathbf{w}; \Phi(\mathbf{x}_{i,test}) >= \sum_{ij} y_{ij}\alpha_{ij}\kappa(\mathbf{x}_{ij}, \mathbf{x}_{i,test})$ and the norm of $\mathbf{w}$ gives $||\mathbf{w}|| = \sqrt{\sum_{ijkl} \alpha_{ij}\alpha_{kl}y_{ij}y_{kl}\kappa(\mathbf{x}_{ij}, \mathbf{x}_{kl})}$. Replacing back into Eq. 3.49, the norm of the orthogonal projection of $\Phi(\mathbf{x}_{i,test})$ on $\mathbf{w}$ gives:

$$||\mathbf{P}_{\mathbf{w}}(\Phi(\mathbf{x}_{i,test}))|| = \frac{\sum_{ij} y_{ij}\alpha_{ij}\kappa(\mathbf{x}_{ij}, \mathbf{x}_{i,test})}{\sqrt{\sum_{ijkl} \alpha_{ij}\alpha_{kl}y_{ij}y_{kl}\kappa(\mathbf{x}_{ij}, \mathbf{x}_{kl})}} \tag{3.52}$$

Note that as $\Phi(\mathbf{0})$ is not guaranteed to be the origin in the feature space $\mathscr{H}$, the norms in Eq. 3.51 are centered with respect to $\Phi(\mathbf{0})$ to ensure the reflexivity property. It is easy to show that both D and $D_{\mathscr{H}}$ ensure the properties of a dissimilarity (positivity, reflexivity, symmetry).

Note that the framework to define the metric D and $D_{\mathscr{H}}$ can also be used in the linear and quadratic formalization. However, the obtained solution for D and $D_{\mathscr{H}}$ can be far away from the original form of D that was optimized in the optimization problem.

3.8 SVM-based solution and algorithm for M²TML

In this section, we review the main steps of the retained SVM solution. In particular, we detail two pre-processing steps needed to adapt the SVM framework to our metric learning problem that are the pairwise space normalization and the neighborhood scaling.

Pairwise space normalization. The scale between the p basic metrics d_h can be different. Thus, there is a need to scale the data within the pairwise space and ensure comparable ranges for the p basic metrics d_h. In our experiment, we use dissimilarity measures with values in $[0; +\infty[$. Therefore, we propose to Z-normalize their log distributions as explained in Section 2.5.2.

Neighborhood scaling. As exposed in Section 3.7.1, in real datasets, local neighborhoods may have very different scales as illustrated in Fig. 3.12. To make the pull neighborhood spreads comparable, we propose for each $\mathbf{x}_i$ to scale each pair $\mathbf{x}_{ij}$ such that the L_2 norm (radius) of the farthest m-th nearest neighbor is 1:

$$\mathbf{x}_{ij}^{norm} = \left[\frac{d_1(\mathbf{x}_i, \mathbf{x}_j)}{r_i}, \ldots, \frac{d_p(\mathbf{x}_i, \mathbf{x}_j)}{r_i}\right]^T \tag{3.53}$$

where r_i is the radius associated to $\mathbf{x}_i$ corresponding to the maximum norm of its m-th nearest neighbor of same class in $Pull_i$:

$$r_i = \max_{\mathbf{x}_{ij} \in Pull_i} D_0(\mathbf{x}_{ij}) \tag{3.54}$$

For simplification purpose, we denote $\mathbf{x}_{ij}$ as $\mathbf{x}_{ij}^{norm}$. Fig. 3.12 illustrates the effect of neighborhood scaling in the dissimilarity space.

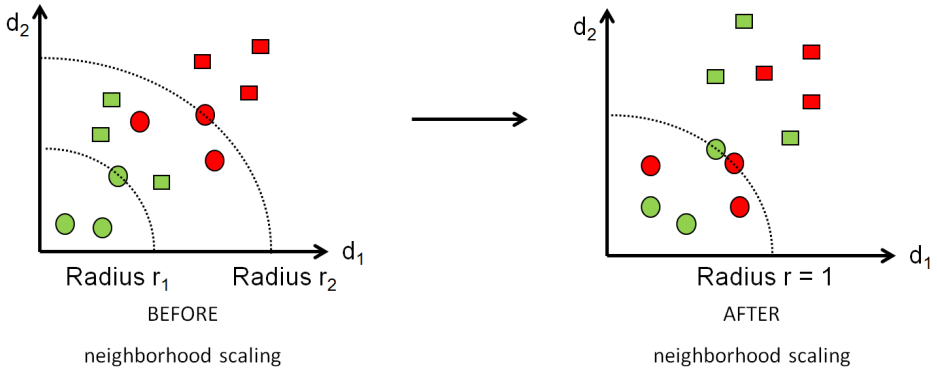


Figure 3.12: Effect of neighborhood scaling before (left) and after (right) on the neighborhood of two time series $\mathbf{x}_1$ (green) and $\mathbf{x}_2$ (red). Circles represent pairs $Pull_i$ and squares represent pairs $Push_i$ for $m = 3$ neighbors. Before scaling, the problem is not linearly separable with a global SVM approach and the spread of each neighborhood are not comparable. After scaling, the target neighborhood becomes comparable and in this example, the problem becomes linearly separable.

Finally, Algorithm 1 summarizes the main steps to learn a multi-modal and multi-scale temporal metric D for a robust nearest neighbors classifier of time series. Algorithm 2 details the steps to classify a new sample $\mathbf{x}_{test}$ using the learned metric D.

Algorithm 1 Multi-modal and Multi-scale Temporal Metric Learning (M^{2}TML) for k-NN classification of time series

1: Input: $\{\mathbf{x}_i, y_i\}_{i=1}^n$ n labeled time series
$d_1, ..., d_p$ metrics as described in Eqs. 2.1, 2.4, 2.6, 3.12
a kernel κ
2: Output: the learned dissimilarity D or $D_{\mathscr{H}}$ depending of κ
3: *Pairwise dissimilarity embedding and normalization*
Embed pairs $(\mathbf{x}_i, \mathbf{x}_j)$ $i, j \in 1, ..., n$ into $\mathscr{E}$ as described in Eq. 3.9 and normalize d_hs (Section 2.5.2)
4: *Build $Pull_i$ and $Push_i$ sets and neighborhood scaling*
Build the sets of pairs $Pull_i$ and $Push_i$ as described in Eq. 3.21 & 3.22 and scale the radii to 1 (Eq. 3.53).
5: SVM *learning*
Train a SVM for a large margin classifier between $Pull_i$ and $Push_i$ sets (Eq. 3.47)
6: *Dissimilarity definition*
Consider Eq. 3.50 (resp. Eq. 3.51) to define D (resp. $D_{\mathscr{H}}$) a linear (resp. non linear) combination function of the normalized metrics d_hs.

Algorithm 2 k-NN classification using the learned metric D or $D_{\mathscr{H}}$

1: Input: $\{\mathbf{x}_i, y_i\}_{i=1}^n$ n labeled time series
$\mathbf{x}_{test}$ a time series to test
$d_1, ..., d_p$ metrics as described in Eqs. 2.1, 2.4, 2.6, 3.12
the learned dissimilarity D or $D_{\mathscr{H}}$ depending of the kernel κ
2: Output: Predicted label $\hat{y}_{test}$
3: *Dissimilarity embedding*
Embed pairs $(\mathbf{x}_i, \mathbf{x}_{test})$ $i \in 1, ..., n$ into $\mathscr{E}$ as described in Eq. 3.9 and normalize d_hs using the same normalization parameters than Algorithm 1
4: *Combined metric computation*
Consider Eq. 3.50 (resp. Eq. 3.51) to compute $D(\mathbf{x}_i, \mathbf{x}_{test})$ (resp. $D_{\mathscr{H}}(\mathbf{x}_i, \mathbf{x}_{test})$) a linear (resp. non linear) combination function of the metrics $d_h(\mathbf{x}_i, \mathbf{x}_{test})$.
5: *Classification*
Consider the k lowest dissimilarities $D(\mathbf{x}_i, \mathbf{x}_{test})$ (resp. $D_{\mathscr{H}}(\mathbf{x}_i, \mathbf{x}_{test})$). Extract the labels y_i of the considered $\mathbf{x}_i$ and make a vote scheme to predict the label $\hat{y}_{test}$ of $\mathbf{x}_{test}$

3.9 Conclusion of the chapter

To learn a multi-modal and multi-scale temporal combined metric, we propose in this chapter to embed time series into a pairwise dissimilarity space. The multi-modal and multi-scale metric learning (M^{2}TML) problem can be formalized as a problem of learning a function in the pairwise dissimilarity space, that ensures the properties of a dissimilarity.
To learn a metric for a robust k-NN, we formulate the M^{2}TML problem into a general regularized large margin optimization problem involving a regularization (pull) and loss (push) term. Choosing a m-neighborhood, greater than the k-neighborhood allows the learned metric to be generalized better. From the general formalization, we propose three different formalizations (Linear, Quadratic, SVM-based). Table 3.1 sums up the characteristics of each formalization and the induced dissimilarities.

	Linear formalization	Quadratic formalization	SVM-based formalization
D	Linear	Linear/Non-linear	Linear/Non-linear
Sparcity	Yes	No	Yes (L_1 regularized SVM)
Dissimilarity properties	Yes	No (non-positivity)	Yes

Table 3.1: The different formalizations for M^{2}TML

The adaptation of SVM in the dissimilarity space to learn the multi-modal and multi-scale metric D have brought us to propose a pre-processing step before solving the problem such as the neighborhood scaling. Note that any multi-class problem is transformed in the pairwise dissimilarity space as a binary classification problem.

As we have defined all functions components of our algorithms (learning, testing), we test our proposed algorithms M^{2}TML in the next chapter on large public datasets.

Chapter 4

Experiments

Contents

In this chapter, we evaluate the efficiency of the proposed M^{2}TML algorithm on public datasets for classification problems of univariate time series. First, we describe the datasets. Then, we detail the experimental protocol. Finally, we present and discuss the obtained results.

4.1 Description

The efficiency of the learned multi-modal and multi-scale dissimilarities D and $D_{\mathscr{H}}$ is evaluated through a 1-NN classification on 30 public datasets[1] [Keo+11]. The 1-NN classifier is used to make the results comparable with the results of the UCR time series data mining archive[2]. Time series come from several fields (simulated data, medical data, electrical data, etc.), are from variable lengths (from small ($q = 24$) to long lengths ($q = 1882$)) and the number of classes to discriminate evolves between 1 and 37 classes. Note that some of the datasets have a small number of time series in the training set ($n < 30$) and others have a large number of time series in the training set ($n > 100$). The results using standard metrics (Euclidean distance, Dynamic time warping) show both easy and challenging classifications problems, the latter being opened for improvements.

[1] PowerCons: `https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption`, BME and UMD: `http://ama.liglab.fr/~douzal/tools.html`.

[2] Note: the datasets and results are the ones before the update of August 2015

Table 4.1 gives a description of the datasets considered in the experiments and Fig. 4.1 gives the temporal representation for some of the datasets. Note that for some datasets (*e.g.*, SonyAIBO, ECG200, FaceFour, PowerConsumption), it is visually difficult to discriminate the classes using one modality (value, behavior, frequential).

Dataset	Nb. Class	Nb. Train	Nb. Test	TS length
1 ItalyPowerD	2	67	1029	24
2 CinCECGtorso	4	40	1380	1639
3 BME	3	300	1500	128
4 ECG200	2	100	100	96
5 SonyAIBOII	2	27	953	65
6 Coffee	2	28	28	286
7 ECG5Days	2	23	861	136
8 SonyAIBO	2	20	601	70
9 Adiac	37	390	391	176
10 Beef	5	30	30	470
11 Trace	4	100	100	275
12 CBF	3	30	900	128
13 CC	6	300	300	60
14 DiatomSizeReduc	4	16	306	345
15 Symbols	6	25	995	398
16 GunPoint	2	50	150	150
17 FacesUCR	14	200	2050	131
18 TwoLeadECG	2	23	1139	82
19 UMD	3	360	1440	150
20 MoteStrain	2	20	1252	84
21 Lighting2	2	60	61	637
22 OliveOil	4	30	30	570
23 FISH	7	175	175	463
24 FaceFour	4	24	88	350
25 SwedishLeaf	15	500	625	128
26 MedicalImages	10	381	760	99
27 Lighting7	7	70	73	319
28 PowerCons	2	73	292	144
29 OSULeaf	6	200	242	427
30 InlineSkate	7	100	550	1882

Table 4.1: Dataset table description providing the number of classes (Nb. Class), the number of time series for the training (Nb. Train) and the testing (Nb. Test) sets, and the length of each time series (TS length).

The results of the learned metrics D and $D_{\mathscr{H}}$ are compared to those of three *a priori* combined metrics $D_{Lin}, D_{Geom}, D_{Sig}$ (Eqs. 2.16, 2.17, 2.18) and five alternative uni-modal metrics covering:

1. The standard Euclidean distance d_A (Eq. 2.1) and Dynamic time warping[3] DTW (Eq. 2.13)
2. The behavior-based measures d_B (Eq. 2.6) and $d_{B-\mathrm{DTW}}$ its counterpart for asynchronous time series, that is d_B is evaluated once time series are synchronized using dynamic programing
3. The frequential-based metric d_F (Eq. 2.4).

[3]In this chapter, the term DTW denotes the classically value-based metric computed after an alignment of the time series obtained with the DTW algorithm with a value-based cost function.

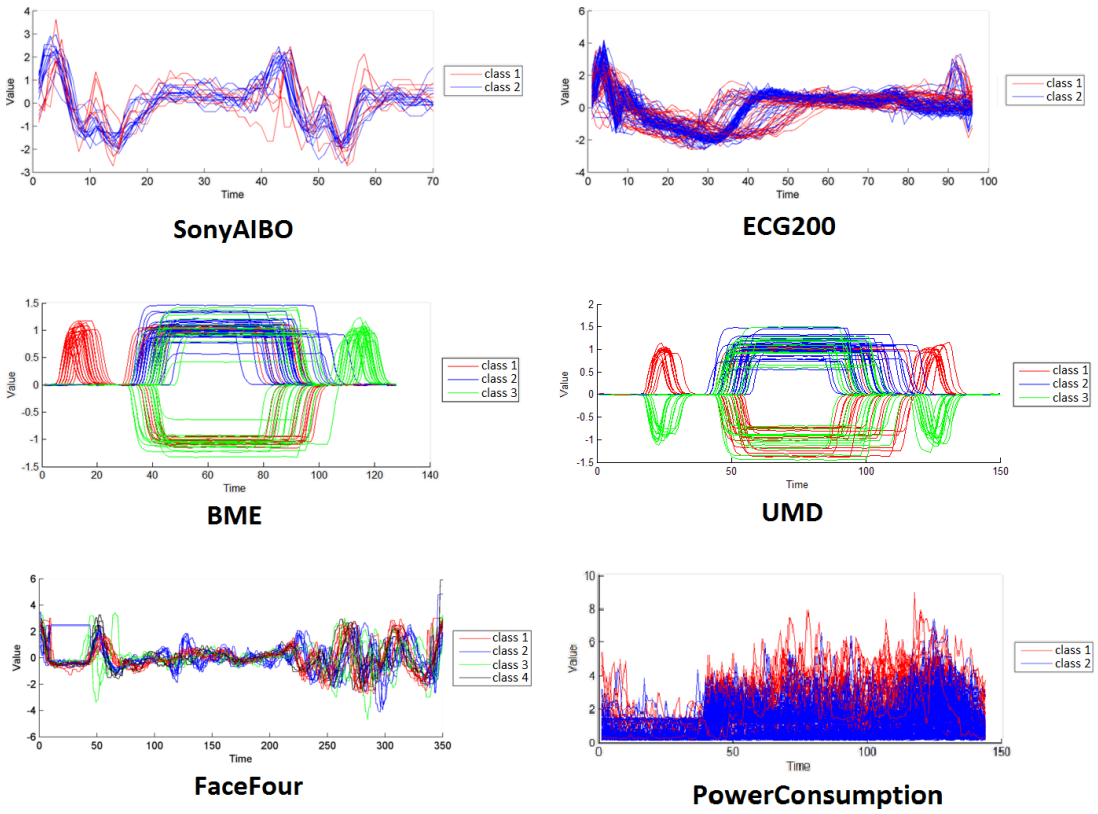


Figure 4.1: Temporal representation of some datasets (SonyAIBO, ECG200, BME, UMD, FaceFour, PowerConsumption) considered in the experiments.

Symbol	Name	Equation reference	Description
d_A	Value-based dissimilarity	Eq. 2.1	Euclidean distance
d_B	Behavior-based dissimilarity	Eq. 2.6	Behavior metric based on cort
DTW	Dynamic time warping	Eqs. 2.13 & 2.1	Euclidean distance after alignment
$d_{B-\mathrm{DTW}}$	Behavior-based aligned dissimilarity	Eqs. 2.13 & 2.6	Behavior metric based on cort after alignment
d_F	Frequential-based dissimilarity	Eq. 2.4	Frequential metric based on Fourier transform
D_{Lin}	Linear combined metric	Eq. 2.16	Combines d_A and d_B (resp. DTW and $d_{B-\mathrm{DTW}}$)
D_{Geom}	Geometric combined metric	Eq. 2.17	Combines d_A and d_B (resp. DTW and $d_{B-\mathrm{DTW}}$)
D_{Sig}	Sigmoid combined metric	Eq. 2.18	Combines d_A and d_B (resp. DTW and $d_{B-\mathrm{DTW}}$)
D	Linear learned metric	Eq. 3.50	M^2TML linear combined metric
$D_{\mathscr{H}}$	Non-linear learned metric	Eq. 3.51	M^2TML non-linear combined metric with a Gaussian kernel

Table 4.2: Considered metric in the experiments

Table 4.2 recalls briefly the considered metrics in the experiments. The *a priori* combined metrics ($D_{Lin}, D_{Geom}, D_{Sig}$) rely, on 2 log-normalized dissimilarities d_A, d_B (resp. DTW, $d_{B-\mathrm{DTW}}$ for asynchronous time series). The alternative metrics and the *a priori* combined metrics are evaluated as usual by involving all time series elements (*i.e.*, at the global scale). For D and $D_{\mathscr{H}}$, we consider a 21-dimensional embedding space $\mathscr{E}$ that relies, for synchronous (resp. asynchronous) data, on 3 log-normalized dissimilarities d_A^s, d_B^s (resp. DTW^s, $d_{B-\mathrm{DTW}}^s$), and d_F^s, at 7 temporal granularities $s \in \{0, ..., 6\}$ obtained by binary segmentation, described in Section 3.3.

4.2 Experimental protocol

The different metrics can be split into two categories. For those without parameters to tune (d_A, DTW), the 1−NN classifier is applied directly on the test set. For those that require to tune parameters (d_B, $d_{B-\mathrm{DTW}}$, D_{Lin}, D_{Geom}, D_{Sig}, D, $D_{\mathscr{H}}$), we recall briefly the grid search and cross-validation procedure (Section 1.1.2). When a learning algorithm requires to tune some parameters, to avoid overfitting, the training set can be divided into two sets: a learning and a validation set. The model is learnt for each combination of parameters (grid search) on the learning set and evaluated on the validation set. The model with the lowest error on the validation set is retained. An other alternative is cross-validation, which partitions the training set into v folds, performs the learning on one subset, and validates on the $v-1$ other subsets. To take into account of variability within the data, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds. Note that for unbalanced datasets in classification problems, it is recommended to use stratified sampling. Table 4.3 resumes the parameter ranges for each metric. We recall that the parameters retained are those that:

- **First**, minimize the average classification error on the validation set.
- **Secondly**, in the case of multiple solutions leading to equal performances, the most discriminant one is retained (*i.e.*, making closer pull pairs and far away push pairs). Precisely, it minimizes the ratio $\frac{d_{intra}}{d_{inter}}$ where d_{intra} and d_{inter} stands respectively to the mean of all intraclass and interclass distances.

As D and $D_{\mathscr{H}}$ involves several parameters to be tuned, we detail hereafter the procedure. The combined metrics D and $D_{\mathscr{H}}$ (κ as the Gaussian kernel) are learned respectively under L_1 and L_2 regularization, using LIBLINEAR and LIBSVM libraries [FCH08]; [HCL08]. The parameters are estimated on a validation set by line/grid search. A cross-validation and stratified sampling for unbalanced datasets are used. Particularly, for each couple (r, λ) $r \in \{1, 4, 10\}$ and $\lambda \in \{0, 10, 30\}$, the pairwise SVM parameters (C, α, γ) are learned by grid search as indicated in Table 4.3.

Dissimilarity	Parameter	Ranges	Description
d_B, $d_{B-\mathrm{DTW}}$	r	$\{1, 2, 3, , \ldots, q-1\}$	Order of behavior-based metric
$D_{Lin}, D_{Geom}, D_{Sig}$	β	$\{0, 0.1, \ldots, 1\}$	Trade-off between value and behavior components
D, $D_{\mathscr{H}}$	λ	$\{0, 10, 30\}$	Strength of the '*push*' term
D, $D_{\mathscr{H}}$	r	$\{1, 4, 10\}$	Order of behavior-based metrics
D, $D_{\mathscr{H}}$	C	$\{10^{-3}, 0.5, 1, 5, 10, 20, 30, ..., 150\}$	Parameter of SVM
D, $D_{\mathscr{H}}$	α	$\{1, 2, 3\}$	Size of the $m = \alpha.k$ neighborhood
$D_{\mathscr{H}}$	γ	$\{10^{-3}, 10^{-2}, \ldots, 10^{3}\}$	Parameter of the Gaussian kernel

Table 4.3: Parameter ranges

Note that the temporal order r for the behavior-based metrics d_B is noise-dependent, typically 1 is retained for noise-free data. The parameter λ corresponds to the strength of the '*push*' term; precisely, if no, moderate or strong '*push*' is required during the training process, a λ value of 0, 10 and 30 is learned, respectively.

4.3 Results and discussion

In this section, we first present a summary table of the quantitivative results obtained in the experiment. Secondly, we present an analysis of the performances of the different metrics. Finally, we present the ability of our proposed approach M^{2}TML to extract discriminative features.

4.3.1 Results

Table 4.4 reports the 1-NN classification test errors based on uni-modal metrics (first 5 columns), on three *a priori* combined metrics ($D_{Lin}, D_{Geom}, D_{Sig}$) and on D and $D_{\mathscr{H}}$. The results for each dataset that are statistically and significantly better than the best performance are indicated in bold (Z-test at 5% risk detailed in Section 1.1.3.a). The last column 'WARP' indicates the synchronous (✓) or asynchronous (×) data type.
Data that need 'WARP' are situated above the line. For each type of delay ('WARP' or non-'WARP'), the datasets are ordered from the less challenging datasets according to the performance of the classically used distances (d_A or DTW) to the most challenging datasets.

	Alternative uni-modal metrics					A priori combinations			M^{2}TML		WARP
Dataset	d_A	d_B	d_F	DTW	$d_{B-\mathrm{DTW}}$	D_{Lin}	D_{Geom}	D_{Sig}	$D(\lambda^*)$	$D_{\mathscr{H}}(\lambda^*)$	WARP
1 ItalyPowerD	0.045	**0.028**	0.078	0.050	0.055	**0.028**	**0.028**	**0.030**	**0.034** (0)	0.046 (0)	×
2 CinCECGtorso	0.103	0.367	0.167	0.349	0.367	**0.094**	**0.094**	**0.093**	**0.092** (0)	**0.088** (0)	×
3 BME	0.173	0.160	0.373	0.107	0.120	0.107	0.107	0.107	**0.007** (0)	**0.007** (0)	×
4 ECG200	**0.120**	**0.070**	0.160	0.230	0.190	**0.070**	**0.070**	**0.070**	**0.080** (0)	**0.080** (0)	×
5 SonyAIBOII	**0.141**	**0.142**	**0.128**	0.169	0.194	**0.142**	**0.142**	**0.144**	**0.162** (0)	**0.142** (0)	×
6 Coffee	0.250	**0.000**	0.357	0.179	0.143	**0.000**	**0.000**	**0.071**	0.143 (0)	**0.036** (10)	×
7 ECG5Days	0.203	0.153	**0.006**	0.232	0.236	0.203	0.203	0.203	**0.012** (10)	0.024 (0)	×
8 SonyAIBO	0.305	0.308	0.258	0.275	0.343	0.308	0.308	0.293	**0.188** (0)	**0.228** (0)	×
9 Adiac	0.389	**0.297**	**0.261**	0.396	0.338	0.373	0.363	0.402	0.358 (0)	0.361 (0)	×
10 Beef	0.467	0.300	0.500	0.500	0.500	0.367	0.267	0.467	**0.033** (0)	0.257 (0)	×
11 Trace	0.240	0.240	0.140	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000** (0)	**0.010** (0)	✓
12 CBF	0.148	0.140	0.382	**0.003**	**0.000**	**0.000**	**0.000**	**0.000**	0.097 (0)	**0.008** (0)	✓
13 CC	0.120	0.113	0.383	**0.007**	0.027	**0.007**	**0.007**	**0.007**	**0.007** (0)	**0.007** (0)	✓
14 DiatomSizeR	0.065	0.076	0.069	**0.033**	**0.029**	**0.033**	**0.033**	**0.042**	0.088 (0)	**0.029** (0)	✓
15 Symbols	0.101	0.111	0.080	**0.050**	**0.043**	**0.051**	**0.050**	**0.052**	0.102 (10)	**0.057** (0)	✓
16 GunPoint	0.087	0.113	**0.027**	0.093	**0.027**	**0.027**	**0.027**	**0.040**	**0.033** (0)	**0.053** (10)	✓
17 FacesUCR	0.231	0.227	0.175	0.095	0.102	0.098	0.098	0.099	**0.068** (10)	**0.068** (0)	✓
18 TwoLeadECG	0.253	0.153	0.103	0.096	**0.008**	**0.005**	**0.005**	0.018	**0.006** (0)	0.016 (10)	✓
19 UMD	0.194	0.222	0.229	0.118	**0.090**	0.111	0.111	0.118	0.104 (0)	**0.042** (0)	✓
20 MoteStrain	**0.121**	0.263	0.278	0.165	0.171	0.260	0.248	0.188	0.185 (0)	0.179 (0)	✓
21 Lighting2	**0.246**	**0.246**	**0.148**	**0.131**	**0.213**	**0.131**	**0.131**	**0.131**	**0.213** (0)	**0.131** (0)	✓
22 OliveOil	**0.133**	**0.133**	**0.167**	**0.200**	**0.100**	**0.133**	**0.133**	**0.133**	**0.167** (0)	**0.100** (10)	✓
23 FISH	0.217	**0.149**	0.229	**0.166**	**0.137**	**0.109**	**0.137**	**0.126**	**0.149** (0)	0.240 (0)	✓
24 FaceFour	0.216	0.216	0.239	0.170	0.136	0.170	0.170	0.170	**0.023** (0)	0.114 (0)	✓
25 SwedishLeaf	0.211	0.186	0.146	0.208	**0.109**	**0.115**	**0.110**	**0.125**	**0.142** (0)	**0.114** (0)	✓
26 MedicalImages	0.316	0.313	0.345	**0.263**	0.290	**0.263**	**0.263**	**0.263**	**0.237** (0)	**0.241** (10)	✓
27 Lighting7	**0.425**	**0.411**	**0.316**	**0.274**	**0.288**	**0.342**	**0.356**	**0.342**	**0.411** (0)	**0.233** (0)	✓
28 PowerCons	**0.366**	0.445	**0.315**	0.397	0.401	0.401	0.401	0.401	**0.318** (0)	**0.342** (0)	✓
29 OSULeaf	0.484	0.475	0.426	0.409	**0.265**	**0.264**	**0.264**	**0.322**	0.421 (0)	0.388 (0)	✓
30 InlineSkate	**0.658**	**0.658**	0.675	**0.616**	**0.623**	**0.605**	**0.605**	**0.602**	0.833 (10)	**0.625** (0)	✓

Table 4.4: 1-NN test error rates for standard, *a priori* combined and M^{2}TML measures.

4.3.2 Comparison of the classification performances on the test set

From Table 4.4, we can see first that the 1-NN classification reaches the best results in:

1. Less than one-third of the data when based on unimodal metrics d_A, d_B or d_F
2. Slightly more than one-third for unimodal metrics DTW and $d_{B-\text{DTW}}$
3. Two-thirds (19 - 20 times on 30) when based on *a priori* combined metrics D_{Lin}, D_{Geom} and D_{Sig}
4. More than two-thirds (21 times on 30) when based on learned metrics D or $D_{\mathscr{H}}$.

Particularly, note that for nearly all datasets for which an uni-modal metric succeeds, the M^{2}TML metrics succeed similarly or lead to equivalent results. However, for several challenging datasets (*e.g.* FaceFour, Beef, FaceUCR, SonyAIBO, BME, CinCECGTorso), M^{2}TML realizes drastic improvements, to the best of our knowledge never achieved before for these challenging public data. For instance, a score of 3% is obtained for Beef against an error rate varying from 30% to 50% for alternative metrics, and of 2.3% obtained for FaceFour v.s. 13% to 23% for alternative metrics. Finally, D and $D_{\mathscr{H}}$ are most datasets, either equivalent or better if only compared to the standard metrics d_A (the Euclidean distance) and DTW.

If we compare the *a priori* combined metrics (D_{Lin}, D_{Geom}, D_{Sig}) based on only the unimodal metrics involved in the combination (either d_A and d_B or DTW and $d_{B-\text{DTW}}$), we observe that *a priori* combined metrics achieved on two-third of the data with an equivalent or better score. Compared to the learned metrics (D, $D_{\mathscr{H}}$), the results are globally similar except for 8 datasets where the learned metrics perform better (FaceFour, Beef, ECG5Days, FaceUCR, SonyAIBO, PowerCons, BME, UMD) and one where the *a priori* combined metrics perform better (OSULeaf). Note that the combined metric D_{Sig} is limited to two components and can't be easily extend to other metrics in its combination. D_{Lin} and D_{Geom} could be easily extended and a proposition could be:

$$D_{Lin}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{h=1}^{p} \alpha_h d_h(\mathbf{x}_i, \mathbf{x}_j) \tag{4.1}$$

$$D_{Geom}(\mathbf{x}_i, \mathbf{x}_j) = \prod_{h=1}^{p} \alpha_h d_h(\mathbf{x}_i, \mathbf{x}_j) \tag{4.2}$$

However, by considering p metrics d_h the resulting models requires to optimize p parameters. The grid search to find the best parameters α_h can become time consuming. The M^{2}TML approach has been proposed to prevent such an exhaustive grid search.

In the second part, we perform a graphical analysis for a global comparison on the whole datasets. In Fig. 4.2-a, each dataset is projected according to, on the x-axis its best error rate obtained for D and $D_{\mathscr{H}}$, and on y-axis its best performance *w.r.t* the standard amplitude-based metrics d_A and DTW. In Fig. 4.2-b, the y-axis is related to the best error rate of the

behavior-based metrics d_B and and $d_{B-\mathrm{DTW}}$. In Fig. 4.2-c, the y-axis is related to the best error rate of the two "non-warp" uni-modal metrics d_A and d_B, . In Fig. 4.2-d, the y-axis is related to the best error rate of the two "warp" uni-modal metrics DTW and $d_{B-\mathrm{DTW}}$ that are also the two most performant uni-modal metrics. In Fig. 4.2-e, the y-axis is related to the error rate of the frequential-based metric d_F. In Fig. 4.2-f, the y-axis is related to the best error rate of the a priori-combined metrics $D_{Lin}, D_{Geom}, D_{Sig}$.
For all plots, let first give some interpretations. If the datasets are situated on the first bisector, it means that the considered metrics in x-axis and y-axis have equal performance. For datasets situated above the first bisector, it means in this case that M^{2}TML method is better that the considered metrics in y-axis. Similarly, for datasets situated below the first bisector, it means in this case that the considered metrics in y-axis are better than M^{2}TML. Less challenging datasets (low classification error rate) are situated near the origin and challenging dataset (high classification error rate) are situated far from the origin.
For all plots, we can note that the datasets are principally projected above the first bisector, indicating higher error rates mostly obtained for uni-modal and *a priori* combined metrics than for M^{2}TML. For the less challenging datasets (near the origin of each graph), although almost projected near the bisector denoting equal performances for the compared metrics, M^{2}TML still bring improvements with projections clearly positioned above the bisector. Finally, from all plots, note that some datasets (Adiac, OSULeaf, InlineSkate) remains challenging for all studied metrics.

4.3.3 Analysis of the discriminative features

For the learned metric D, thanks to the L_1 regularization, the learned SVM reveals the features that most differentiate pull from push pairs. We recall that the weight for each feature can be analyzed through the weight vector $\mathbf{w}$ obtained by learning the SVM classifier. Table 4.5 shows the sparse, muti-modal and multi-scale potential of M^{2}TML approach. It gives for each dataset, the weights of the top five 'discriminative' features that contribute to the definition of D. For instance, for FaceFour D reaches an error of 2.3% by combining, in the order of importance, the behavior $d_{B-\mathrm{DTW}}$, frequential d_F and amplitude DTW modalities, at the global (I^0) and local (I^4, I^5, I^2) scales. For Beef, the learned model is very sparse as D involves only the behavior modality based on the segment I^3 (d_B^3). Note that if we look at only the most discriminative feature (1st column), the M^{2}TML method helps to localize discriminative modality and a specific temporal scale (localization) that could not be easily guessed *a priori* (*e.g.*, Lightning7: behavior modality on the segment I_6 ($d^6_{B-\mathrm{DTW}}$), OliveOil: frequential modality on the segment I_5 (d^5_F), TwoLeadECG: behavior modality on the segment I_4 ($d^4_{B-\mathrm{DTW}}$)).

In Fig. 4.3, we plot the weights of all features for SonyAIBO, Beef, CincECGtorso and FaceFour cases as an example. It illustrates both the sparsity of the M^{2}TML approach (Beef, CincECGtorso and FaceFour) and the ability of the algorithm to combine all the features into the metric D (SonyAIBO). In particular, the approach is able to either select one single feature (Beef) or combine several selected features (CinCECGTorso, FaceFour). Fig. 4.4 illustrates the temporal locations of the most discriminative features for these datasets. Note

that from looking at the temporal representation, it is not easy to determine *a priori* which modality (value, behavior, frequential) and at which temporal scale (localization) is the most discriminative feature to separate the classes.

In summary, we can emphasize that for almost all datasets, the definition of D involves no more than five features (the most contributive ones), that assesses not only the model's sparsity but also the representativeness of the revealed features.

Dataset	Feature weights (%)				
ItalyPowerD	d_B^0 (27.5%)	d_F^4 (17.2%)	d_F^1 (12.3%)	d_A^1 (11.2%)	d_B^2 (9%)
CinCECGtorso	d_F^0 (38.4%)	d_A^5 (13.1%)	d_B^4 (11.5%)	d_F^1 (11.2%)	d_A^2 (9.8%)
BME	d_B^0 (75.2%)	d_F^4 (15.5%)	d_B^2 (5.8%)	d_B^1 (1.9%)	d_F^1 (0.7%)
ECG200	d_B^0 (89.6%)	d_B^6 (2.4%)	d_A^3 (2.3%)	d_B^1 (2.2%)	d_B^4 (2%)
SonyAIBOII	d_B^3 (100%)	-	-	-	-
Coffee	d_F^4 (59.4%)	d_B^6 (6.4%)	d_B^2 (5.6%)	d_B^3 (5%)	d_F^5 (4.4%)
ECG5Days	d_B^5 (44.9%)	d_B^6 (36.3%)	d_A^4 (7.9%)	d_F^6 (7.4%)	d_B^4 (2.7%)
SonyAIBO	d_F^3 (30.8%)	d_B^6 (27.3%)	d_B^5 (5%)	d_A^1 (4.1%)	d_B^0 (3.9%)
Adiac	d_F^0 (79.2%)	d_B^4 (13.8%)	d_A^4 (3.5%)	d_F^5 (1.7%)	d_B^5 (1.2%)
Beef	d_B^3 (100%)	-	-	-	-
Trace	DTW^0 (58.3%)	DTW^6 (6.9%)	$d_{B-\text{DTW}}^0$ (5.8%)	DTW^2 (5.6%)	DTW^5 (5.5%)
CBF	d_F^6 (18.5%)	d_F^3 (18.5%)	d_F^0 (15.2%)	DTW^4 (12.4%)	d_F^1 (9%)
CC	d_F^0 (17.1%)	DTW^3 (13.2%)	DTW^2 (11.4%)	$d_{B-\text{DTW}}^2$ (11%)	d_F^1 (7.1%)
DiatomSizeR	d_F^5 (100%)	-	-	-	-
Symbols	d_F^6 (38.2%)	DTW^0 (16.1%)	DTW^1 (12%)	d_F^0 (6.7%)	DTW^2 (4.7%)
GunPoint	$d_{B-\text{DTW}}^0$ (41.1%)	DTW^5 (14.7%)	DTW^2 (9.5%)	DTW^4 (6.1%)	d_F^4 (6%)
FacesUCR	d_F^2 (21.5%)	$d_{B-\text{DTW}}^0$ (19.5%)	d_F^4 (16.7%)	DTW^0 (12.6%)	$d_{B-\text{DTW}}^2$ (8.6%)
TwoLeadECG	$d_{B-\text{DTW}}^4$ (60%)	d_F^1 (12%)	DTW^4 (11.4%)	$d_{B-\text{DTW}}^6$ (7.6%)	$d_{B-\text{DTW}}^1$ (4.2%)
UMD	$d_{B-\text{DTW}}^0$ (99.8%)	$d_{B-\text{DTW}}^5$ (0.2%)	-	-	-
MoteStrain	$d_{B-\text{DTW}}^5$ (93.2%)	$d_{B-\text{DTW}}^6$ (6.8%)	-	-	-
Lighting2	$d_{B-\text{DTW}}^0$ (100%)	DTW^0 (0%)	d_F^0 (0%)	DTW^1 (0%)	$d_{B-\text{DTW}}^1$ (0%)
OliveOil	d_F^5 (97%)	$d_{B-\text{DTW}}^2$ (3%)	-	-	-
FISH	$d_{B-\text{DTW}}^5$ (17.9%)	d_F^0 (10.5%)	$d_{B-\text{DTW}}^6$ (9.9%)	$d_{B-\text{DTW}}^4$ (8.3%)	$d_{B-\text{DTW}}^3$ (7.8%)
FaceFour	$d_{B-\text{DTW}}^4$ (66.7%)	d_F^4 (22.4%)	$d_{B-\text{DTW}}^3$ (5.6%)	$d_{B-\text{DTW}}^1$ (5.3%)	-
SwedishLeaf	d_F^0 (23.9%)	$d_{B-\text{DTW}}^1$ (14.1%)	$d_{B-\text{DTW}}^2$ (10.5%)	$d_{B-\text{DTW}}^6$ (10%)	$d_{B-\text{DTW}}^5$ (6%)
MedicalImages	$d_{B-\text{DTW}}^1$ (53.3%)	d_F^3 (12.9%)	$d_{B-\text{DTW}}^2$ (10.7%)	$d_{B-\text{DTW}}^3$ (10.1%)	$d_{B-\text{DTW}}^0$ (3.8%)
Lighting7	$d_{B-\text{DTW}}^6$ (77.7%)	d_F^6 (20.8%)	$d_{B-\text{DTW}}^5$ (1.5%)	DTW^3 (0%)	DTW^1 (0%)
PowerCons	d_F^0 (26.1%)	DTW^0 (20.3%)	d_F^1 (19.3%)	$d_{B-\text{DTW}}^0$ (6.1%)	d_F^2 (5.1%)
OSULeaf	$d_{B-\text{DTW}}^2$ (84.7%)	d_F^6 (7.7%)	d_F^0 (2.7%)	d_F^5 (1.6%)	DTW^5 (1.2%)
InlineSkate	DTW^2 (25.7%)	d_F^4 (24.3%)	d_F^3 (16.5%)	$d_{B-\text{DTW}}^2$ (11.2%)	$d_{B-\text{DTW}}^3$ (5.2%)

Table 4.5: Top 5 multi-modal and multi-scale features involved in D

4.3.4 Effect on the neighborhood before and after learning

In the last part, we compare the global effect of the alternative and M^{2}TML metrics on the 1-NN neighborhood distribution and class discrimination. For that, a MultiDimensional Scaling[4] (MDS) is used to visualize the distribution of samples according to their pairwise dissimilarities. Briefly, we recall that MDS is a method of visualizing the proximity between samples in a dataset (Section 2.2). Given an input dissimilarity matrix, we can project the time series on a 2-dimensional plot whose configuration reproduces the best the dissimilarities between the time series. Note that the MDS representation has no link with the dissimilarity space representation whose dimensions are basic temporal metrics.

For FaceFour, Fig. 4.5 shows the first obtained plans and their corresponding stresses, the classes being indicated in different symbols and colors. We can see distinctly the effect of the learned D that leads to more compact and more isolated classes with robust neighborhoods for 1-NN classification (*i.e.*, closer pull pairs and far away push pairs) than the best alternative metric $d_{B-\mathrm{DTW}}$ that shows more overlapping classes and heterogeneous neighborhoods.

4.4 Conclusion of the chapter

The large conducted experiments and the impressive performances obtained attest the efficiency of the learned M^{2}TML metrics for time series nearest neighbors classification. As discussed, the datasets encompass time series that involve global or local temporal comparison, require or not time warping, with linearly or non linearly separable neighborhoods.
Finally, let us underline the merit of the M^{2}TML solution, that not only leads to equivalent or better performances from the standard metrics (Euclidean distance, Dynamic time warping), but also provides a comprehensive and fine-grained information about which modalities are mostly discriminant, how they should be combined and precisely at which temporal granularity (localization).

[4]Matlab function: mdscale for metrics and non metrics

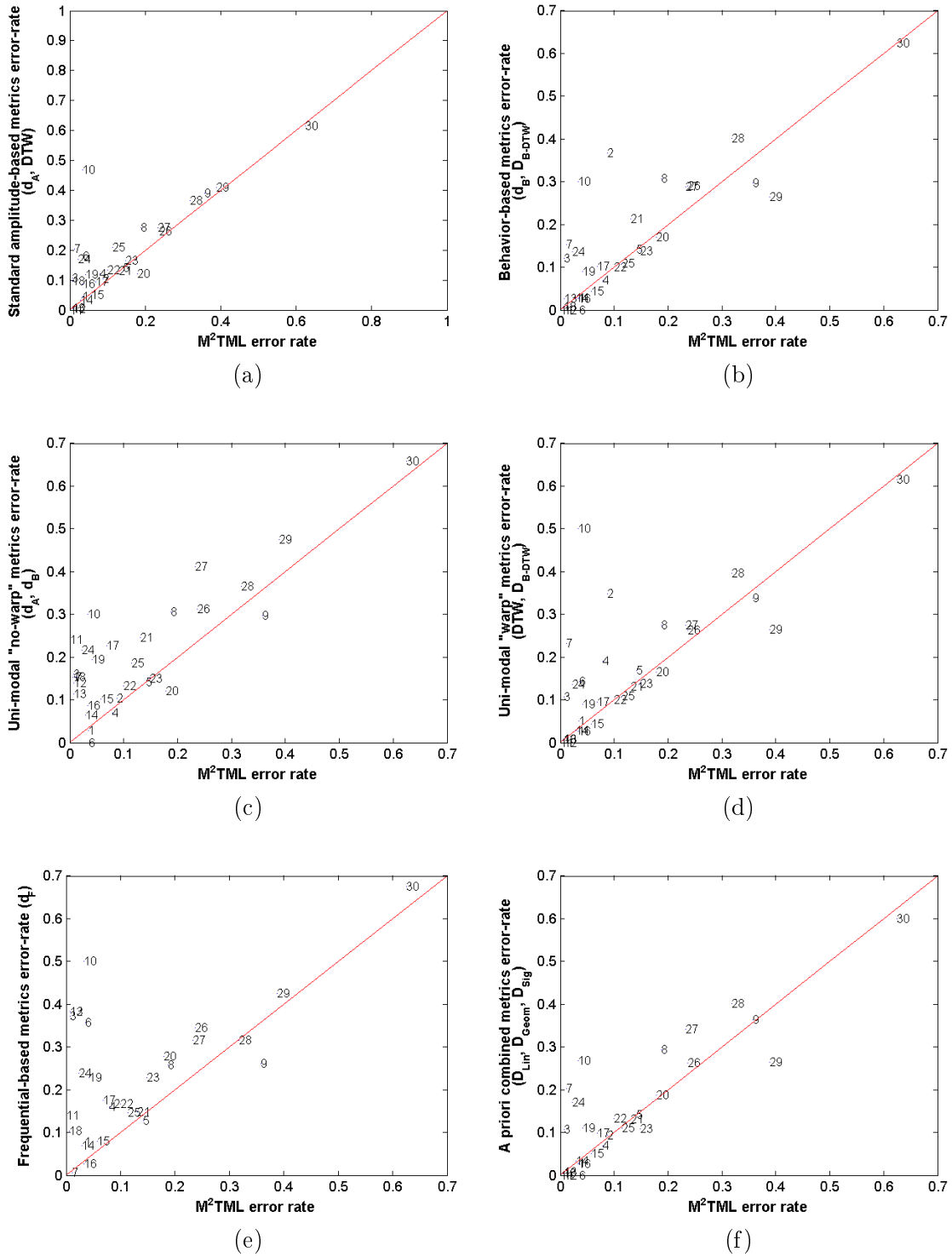
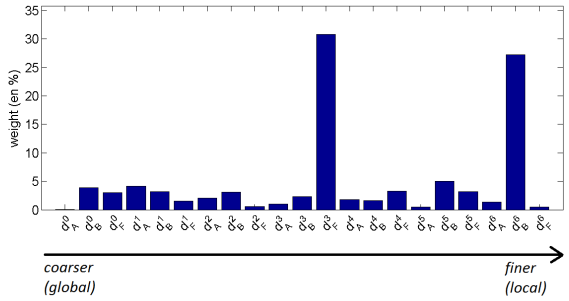
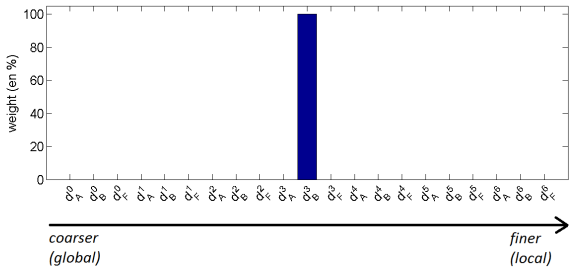


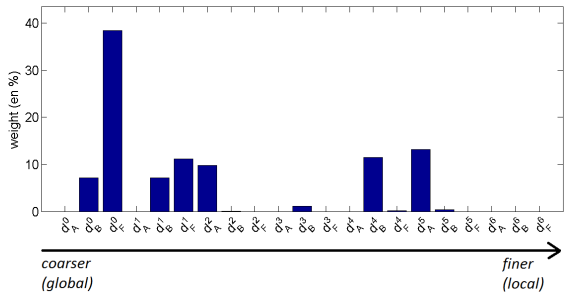
Figure 4.2: (a) Standard amplitude-based (Euclidean distance d_A and DTW) vs. M^{2}TML (D and $D_{\mathscr{H}}$) metrics. (b) Behavior-based (d_B and $d_{B-\mathrm{DTW}}$) vs. M^{2}TML metrics. (c) No-warp (d_A and d_B) vs. M^{2}TML metrics. (d) Warp (DTW and $d_{B-\mathrm{DTW}}$) vs. M^{2}TML metrics. (e) Frequential-based (d_F) vs. M^{2}TML metrics. (f) *A priori* (D_{Lin}, D_{Geom}, D_{Sig}) vs. M^{2}TML metrics.



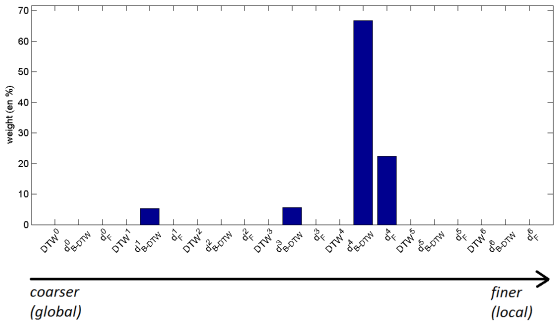
(a) SonyAIBO



(b) Beef

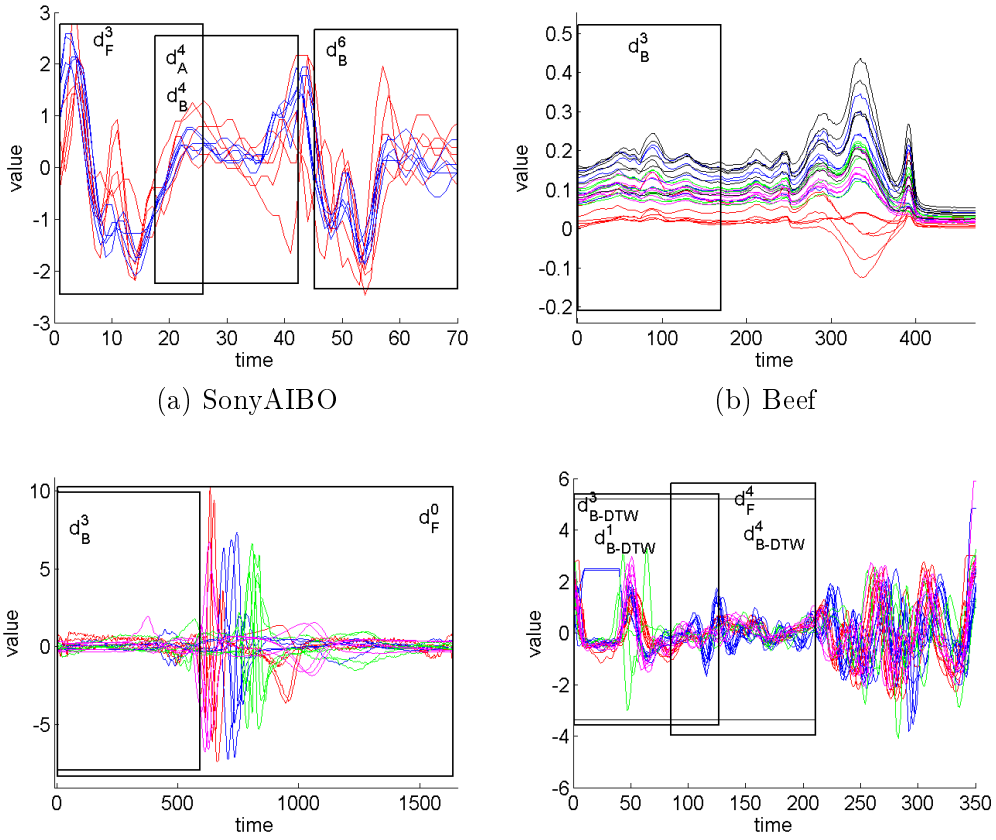


(c) CinC ECG torso



(d) FaceFour

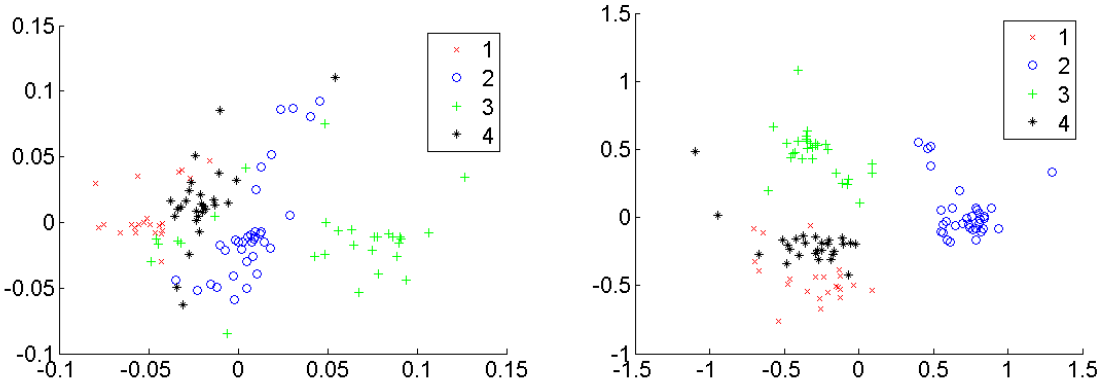
Figure 4.3: M^{2}TML feature weights for 4 datasets.



(a) SonyAIBO (b) Beef

(c) CinC ECG torso (d) FaceFour

Figure 4.4: Temporal representation of the top M^{2}TML feature weights for 4 datasets.



(a) FaceFour ($d_{B-\mathrm{DTW}}$, stress: 20,1%) (b) FaceFour (D, stress: 18,9%)

Figure 4.5: MDS visualization of the $d_{B-\mathrm{DTW}}$ (Fig. a) and D (Fig. b) dissimilarities for FaceFour

Conclusion and perspectives

Conclusion

By considering usual classifiers (*e.g.*, k-NN) that are based on distance between samples, we have proposed a Multi-modal and Multi-scale Temporal Metric Learning (M^{2}TML) framework for a robust k-NN classifier. It is based on a new space representation, the pairwise dissimilarity space, where pairs of time series are embedded as vectors described by different basic temporal metrics. A metric combining the basic metrics can be seen as a function (linear or non-linear) of the pairwise dissimilarity space, learned by using a large margin optimization process (SVM) inspired from the nearest neighbors metric learning framework. The obtained metric satisfies the properties of a dissimilarity (positivity, reflexivity, symmetry), leads to good performances on a large number of public datasets, and gives an interpretable solution that allows us to analyze the modalities and scales that are the most discriminant.

Temporal data may be compared based on various characteristics, called modalities. Time series can be compared not only on their amplitudes like static data, but also on other modalities such as their behavior, frequency, etc. To cope with delays in real time series, Dynamic time warping approach can be used to re-align the signals. Some authors propose to combine several modalities through a combination function but the combinations are either, limited to two modalities or in the case of multiple modalities (more than 2), the number of parameters to optimize for a classifier may become time consuming. In general, state of the art approaches compared the time series by involving all observations, restricting the potential of comparison measures (metrics) to capture local differences. We proposed to take into account local characteristics, that we named multi-scale. We have believed that all of these considerations (modality, scale, delays) should be taken into consideration in the definition of a metric in order to improve the performance of the classifier.

The objective is to learn a metric for a robust k-NN. For that, we propose a general formalization of the problem of learning a combined multi-modal and multi-scale temporal metric (M^{2}TML). Based on a pairwise dissimilarity representation of the pairs of time series, the metric learning problem can be reduced to the learning of a linear or non linear function of the dissimilarity space that satisfies the properties of a dissimilarity. Inspired from metric learning work, the problem is formalized as an optimization problem involving a regularization and loss term which aims to pull samples that are expected to be similar and push away samples that should be dissimilar. First, by considering a linear combination of the basic metrics, changing the regularization term leads the general formalization to a linear and quadratic formalization. The latter allows us to extend to the learning of non-linear functions thanks to the "kernel" trick. However, the methods can lead to functions that doesn't meet the properties of a dissimilarity (non-positivity). Secondly, we propose to formulate the problem as a SVM problem which aims to separate pull and push samples, then we define a metric that satisfies the required properties of a dissimilarity.

The efficiency of the proposed SVM-based solution has been tested in the case of classification of univariate time series, on a wide variety of datasets coming from various fields (simulated data, medicine, power consumption, etc.), diverse sizes of training and testing, various number of classes, etc. The M^{2}TML solution achieves not only, either equivalent or better performances compared to the standard global metrics (Euclidean distance, dynamic time warping, temporal correlation, Fourier-based distance), but it also provides a sparse and interpretable solution that allows us to give a comprehensive analysis of the most discriminative modalities and their respective temporal granularity that may not be always intuitive *a priori*.

Perspectives

Extension to other modalities, multivariate problems and other type of data

In this work, we only focus on three basic temporal metrics (euclidean distance, temporal correlation, Fourier-based distance). Montero & Vilar propose in [MV14] a review on a wide number of metrics dedicated to time series. For remaining challenging datasets in our experiments, it could be interesting to integrate other basic temporal metrics in our framework to see the obtained results.

The framework can be easily extend to multivariate problem. For each dimension, we consider the set of multi-modal and multi-scale description. Then, we consider the union over the dimensions as the pairwise dissimilarity description $d_1, \ldots, d_p$.

The proposed solution has been tested in the case of time series data but the framework is more general. It can be applied to any other type of data (strings, graphs, images) to learn a combined metric. These data might be compared on other characteristics. Deza & Deza makes a detailed review of metrics for various domains in [DD09].

Other possibilities for the multi-scale description

A second improvement is about the multi-scale description that denotes in this work as a temporal segmentation. We propose a multi-scale approach based on a binary segmentation using a dichotomy process. Other solutions could be proposed, in particular, in order to localize automatically finely events of interest. For example, in the case of the dataset SonyAIBO, the discriminative temporal locations of the signal is known *a priori* (Fig. 4.6). With the actual multi-scale description, it is not possible to extract exactly the two red patterns of interest. A solution based on a sliding window of variable lengths could be used to locate precisely these patterns.

[5] source: `http://www.cs.ucr.edu/~eamonn/LogicalShapelet.pdf`

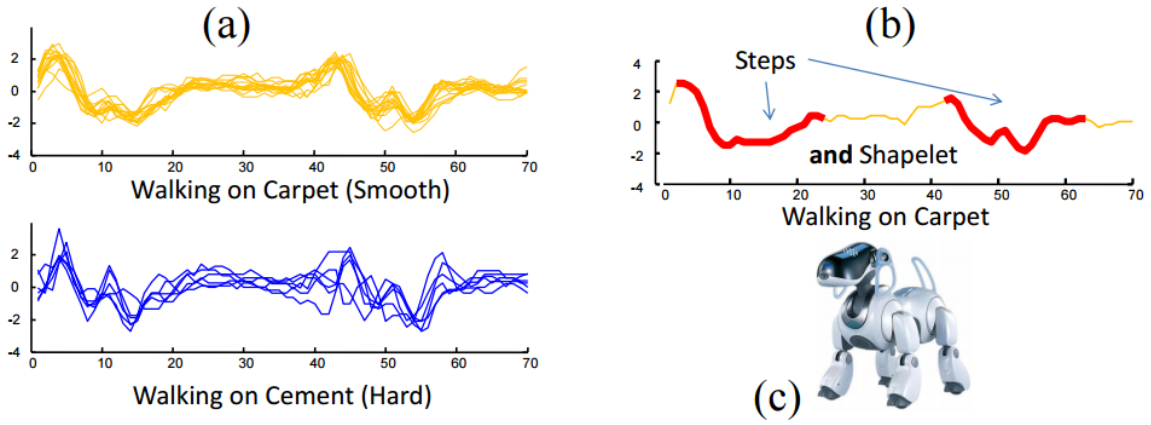


Figure 4.6: (a) Two classes of time series from the Sony AIBO accelerometer. (b) The and-shapelets from the walk cycle on carpet. (c) The Sony AIBO Robot.[5]

Learning of local metrics

Some authors suggest that in some datasets, global linear metric learning approach may not be sufficient to improve the accuracy of k-NN classification [WS09]; [WWK12]. Since the discriminatory power of the input features might vary between different neighborhoods, learning a global metric cannot fit well the distance over the data. To overcome this difficulty, they propose to learn a metric on each neighborhood, referred to as local metric learning. Similarly, the M^{2}TML framework could be extended to learn local combined temporal metrics for each neighborhood. The objective is to learn for each n set $Pull_i$ and $Push_i$ (n being the number of samples in the training set) a local metric using the same framework than the one we propose in this work. We obtain n local metrics D_i. Then, to classify a new sample $\mathbf{x}_{test}$, we compute the n metrics $D_i(\mathbf{x}_{i,test})$ and classify $\mathbf{x}_{test}$ using the k lowest distances $D_i(\mathbf{x}_{i,test})$.

Re-iteration of the initial metric

Similarly to Large Margin Nearest Neighbors (LMNN) approach proposed by Weinberger & Saul [WS09], the M^{2}TML approach might inherit the same problem of the initial distance, *i.e.*, fixing the set $Pull_i$ and $Push_i$ according to an initial distance (Euclidean distance in this work). Other initial distances could have been used. If the initial distance is far away from the optimal solution, the definition of the sets $Pull_i$ and $Push_i$ can impact the convergence to the optimal solution. In same spirit as the multi-pass LMNN approach proposed by Weinberger & Saul, we could re-iterate the learning process. At each step, we re-define the sets $Pull_i$ and $Push_i$ using the distance learned at the previous step. Then, we stop the learning when arriving at convergence (*e.g.*, the sets $Pull_i$ and $Push_i$ doesn't evolve anymore or evolve slightly between two steps).

Other propositions to define the combined metric

First, as said in Section 3.7, note that the framework to define the metric D and $D_{\mathscr{H}}$ can also be used in the linear and quadratic formalization. However, the obtained solution for D and $D_{\mathscr{H}}$ can be far away from the original form of D that has been optimized in the optimization problem.

Secondly, we have proposed a form for the metric D and $D_{\mathscr{H}}$ so that it satisfies the properties of a dissimilarity. Other solutions could have been proposed. In particular, instead of using a max operator in the definition of D and $D_{\mathscr{H}}$, an other variant could consider a parameter λ that can be either positive or negative. In the case of negative λ, the action of the exponential term would become a pull term instead of a push term. Note that in both cases, for extreme value of λ, there exists a risk to binarize the metric. In particular, for $\lambda \mapsto -\infty$, there exists a risk of having zero values for the nearest neighbors, which could lead to problems when classifying by the nearest neighbors.

Extension to regression problems

For the SVM-based solution, in the pairwise dissimilarity space, each vector $\mathbf{x}_{ij}$ is labeled y_{ij} by following the rule: if $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar, the vector $\mathbf{x}_{ij}$ is labeled -1; and +1 otherwise. For classification problems, the concept of similarity between samples $\mathbf{x}_i$ and $\mathbf{x}_j$ is driven by the class label y_i and y_j in the original space:

$$y_{ij} = \begin{cases} +1 & \text{if } y_i = y_j \\ -1 & \text{if } y_i \neq y_j \end{cases} \tag{4.3}$$

For regression problems, each sample $\mathbf{x}_i$ is assigned to a continuous value y_i. Two approaches are possible to define the similarity concept. The first one discretizes the continuous space of values of the labels y_i to create classes. One possible discretization bins the label y_i into Q intervals as illustrated in Fig. 4.7. Each interval becomes a class which associated value can be set for example as the mean or median value of the interval. Then, the classification framework is used to define the pairwise label y_{ij}.

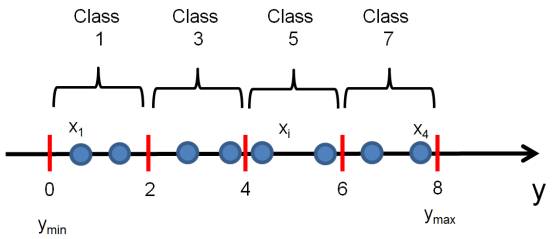


Figure 4.7: Example of discretization by binning a continuous label y into $Q = 4$ equal-length intervals. Each interval is associated to a unique class label. In this example, the class label for each interval is equal to the mean in each interval.

This approach may leads to border effects between the classes. For instance, two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ that are close to a frontier and that are on different sides of the border will be considered as different, as illustrated in Fig 4.8. Moreover, a new sample $\mathbf{x}_j$ will have its labels y_j assigned to a class and not a real continuous value.

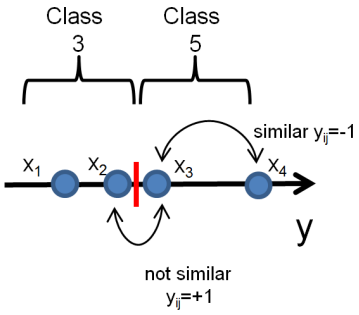


Figure 4.8: Border effect problems. In this example, $\mathbf{x}_2$ and $\mathbf{x}_3$ have closer value labels y_2 and y_3 than $\mathbf{x}_3$ and $\mathbf{x}_4$. However, with the discretization $\mathbf{x}_2$ and $\mathbf{x}_3$ don't belong to the same class and thus are consider as not similar.

The second approach considers the continuous value of y_i, computes a L_1-norm between the labels $|y_i - y_j|$ and compare this value to a threshold ϵ. Geometrically, a tube of size ϵ around each value of y_i is built. Two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ are considered as similar if the absolute difference between their labels $|y_i - y_j|$ is lower than ϵ (Fig. 4.9):

$$y_{ij} = \begin{cases} -1 & \text{if } |y_i - y_j| \leq \epsilon \\ +1 & \text{otherwise} \end{cases} \tag{4.4}$$

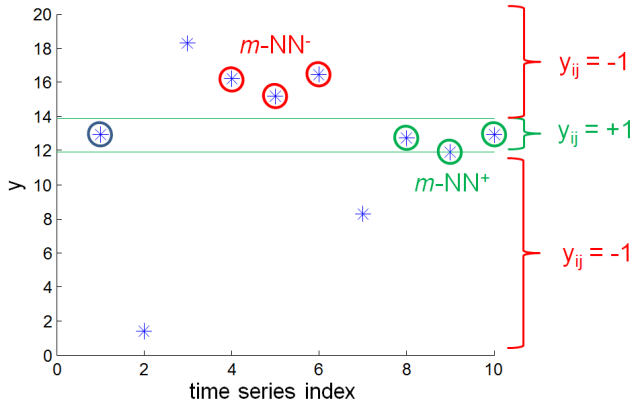


Figure 4.9: Example of pairwise label definition using an ϵ-tube (green lines) around the time series $\mathbf{x}_i$ (circled in blue). For, time series $\mathbf{x}_j$ that falls into the tube, the pairwise label is $y_{ij} = -1$ (similar) and outside of the tube, $y_{ij} = +1$ (not similar). m-NN$^+$ and m-NN$^-$ time series are indicated respectively in green and red circle for k-NN with $k = 1$ and $m = 3$ neighborhood.

Using the learned combined metric in other algorithms

In this work, we propose to learn a temporal metric for a robust k-NN classifier. As explained in Section 1.2.1, for industrial practical usage, the k-NN algorithm may present some disadvantages, mainly due to its computational complexity, both in memory space (storage of the training samples) and time (search of the neighbors).
Inspired from the work on temporal trees in [DCA12], we could use the learned metric in an other classifier such as a decision tree. For multivariate classification problems, an idea could be to learn in a first step a linear or non-linear multi-modal and multi-scale temporal metric for each dimension. Then, in a second step, given the set of multi-modal and multi-scale temporal metrics, we build a temporal tree based on the training samples: First, for each dimension, we split the data into two partitions using a clustering algorithm such as k-means ($k = 2$) with the learned temporal metric for the considered dimension. Secondly, similarly to classical decision tree, we compute a criterion (*e.g.*, a Gini coefficient or Information Gain coefficient) to select the best split, *i.e.*, the best learned temporal metric that minimize the criterion. Thirdly, we compute for each partition the centroid, *i.e.*, the time series that minimizes the mean distance over all the other time series in the same partition. Then, for each obtained partition, we re-iterate the process until the stopping condition is met, *e.g.*, all the sample in the partition have the same class label or the number of sample have fallen below some minimum threshold. It corresponds a leaf node and the label of the node is assigned to the one of the majority. To classify a new time series $\mathbf{x}_{test}$, similarly to classical decision tree, at each node, we compute the distance D or $D_{\mathscr{H}}$ of the new sample to each centroid. The time series $\mathbf{x}_{test}$ is then assigned to the node of the nearest centroid until it reaches a leaf node where its label is assigned.

List of publications

Journal

- C. Do, A. Douzal-Chouakria, S. Marié, M. Rombaut. Multi-modal and Multi-scale Temporal Metric Learning for Time Series Nearest Neighbors Classification, Pattern Recognition Letters 2016 (under submission)
- C. Do, A. Douzal-Chouakria, S. Marié, M. Rombaut. Temporal Metric Learning for robust kNN classification based on temporal and frequential characteristics of time series, Springer LNAI volume, "Time Series Analysis and Machine Learning" (under submission)

Conference

- C. Do, A. Douzal-Chouakria, S. Marié, M. Rombaut. Multiple Metric Learning for large margin kNN Classification of time series, EUSIPCO'2015
- C. Do, A. Douzal-Chouakria, S. Marié, M. Rombaut. Temporal and Frequential Metric Learning for Time Series kNN Classification. ECML-PKDD'2015 Workshop on Advanced Analytics and Learning from Temporal Data, 39-45

Appendix A

Quadratic formalization development

Objective

We recall the Lagrangian function:

$$\begin{aligned} L(\mathbf{w}, \xi, \boldsymbol{\alpha}, \mathbf{r}) =& \frac{1}{2}\mathbf{w}^T\mathbf{M}\mathbf{w} + C\sum_{ijl}\xi_{ijl} - \sum_{ijl} r_{ijl}\xi_{ijl} \\ & - \sum_{ijl}\alpha_{ijl}\left(\mathbf{w}^T(\mathbf{x}_{il} - \mathbf{x}_{ij}) - 1 + \xi_{ijl}\right) \end{aligned} \tag{A.1}$$

where $\alpha_{ijl} \geq 0$ and $r_{ijl} \geq 0$ are the Lagrange multipliers.

The objective is to show that the Lagrangian is equal to:

$$L(\mathbf{w}, \xi, \boldsymbol{\alpha}, \mathbf{r}) = \sum_{ijl}\alpha_{ijl} - \frac{1}{2}\sum_{ijl}\sum_{i'j'l'}\alpha_{ijl}\alpha_{i'j'l'}(\mathbf{x}_{il} - \mathbf{x}_{ij})^T\mathbf{M}^{-1}(\mathbf{x}_{i'l'} - \mathbf{x}_{i'j'}) \tag{A.2}$$

where:

$$\mathbf{w} = \mathbf{M}^{-1}\sum_{ijl}\alpha_{ijl}(\mathbf{x}_{il} - \mathbf{x}_{ij}) \tag{A.3}$$

$$r_{ijl} = C - \alpha_{ijl} \quad \Leftrightarrow \quad C - \alpha_{ijl} - r_{ijl} = 0 \tag{A.4}$$

Calcul development

$$\begin{aligned} L(\mathbf{w}, \xi, \boldsymbol{\alpha}, \mathbf{r}) =& \frac{1}{2}\mathbf{w}^T\mathbf{M}\mathbf{w} + C\sum_{ijl}\xi_{ijl} - \sum_{ijl} r_{ijl}\xi_{ijl} \\ & - \sum_{ijl}\alpha_{ijl}\left(\mathbf{w}^T(\mathbf{x}_{il} - \mathbf{x}_{ij}) - 1 + \xi_{ijl}\right) \end{aligned} \tag{A.5}$$

$$\begin{aligned} L(\mathbf{w}, \xi, \boldsymbol{\alpha}, \mathbf{r}) =& \frac{1}{2}\mathbf{M}^{-1}\sum_{i'j'l'}\alpha_{i'j'l'}(\mathbf{x}_{i'l'} - \mathbf{x}_{i'j'})\mathbf{M}\mathbf{M}^{-1}\sum_{ijl}\alpha_{ijl}(\mathbf{x}_{il} - \mathbf{x}_{ij}) \\ & + C\sum_{ijl}\xi_{ijl} - \sum_{ijl} r_{ijl}\xi_{ijl} \\ & - \sum_{ijl}\alpha_{ijl}\left(\mathbf{M}^{-1}\sum_{i'j'l'}\alpha_{i'j'l'}(\mathbf{x}_{i'l'} - \mathbf{x}_{i'j'})(\mathbf{x}_{il} - \mathbf{x}_{ij}) - 1 + \xi_{ijl}\right) \end{aligned} \tag{A.6}$$

$$
\begin{aligned}
L(\mathbf{w},\xi,\boldsymbol{\alpha},\mathbf{r}) =& \frac{1}{2}\mathbf{M}^{-1}\sum_{i'j'l'}\alpha_{i'j'l'}(\mathbf{x}_{i'l'}-\mathbf{x}_{i'j'})\mathbf{M}\mathbf{M}^{-1}\sum_{ijl}\alpha_{ijl}(\mathbf{x}_{il}-\mathbf{x}_{ij}) \\
&+ C\sum_{ijl}\xi_{ijl} - \sum_{ijl} r_{ijl}\xi_{ijl} \\
&- \sum_{ijl}\alpha_{ijl}\mathbf{M}^{-1}\sum_{i'j'l'}\alpha_{i'j'l'}(\mathbf{x}_{i'l'}-\mathbf{x}_{i'j'})(\mathbf{x}_{il}-\mathbf{x}_{ij}) \\
&+ \sum_{ijl}\alpha_{ijl} \\
&- \sum_{ijl}\alpha_{ijl}\xi_{ijl}
\end{aligned} \tag{A.7}
$$

$$
\begin{aligned}
L(\mathbf{w},\xi,\boldsymbol{\alpha},\mathbf{r}) =& \frac{1}{2}\sum_{ijl}\sum_{i'j'l'}\alpha_{ijl}\alpha_{i'j'l'}(\mathbf{x}_{il}-\mathbf{x}_{ij})\mathbf{M}^{-1}(\mathbf{x}_{i'l'}-\mathbf{x}_{i'j'}) \\
&+ \sum_{ijl}(C - r_{ijl} - \alpha_{ijl})\xi_{ijl} \\
&- \sum_{ijl}\sum_{i'j'l'}\alpha_{ijl}\alpha_{i'j'l'}(\mathbf{x}_{il}-\mathbf{x}_{ij})\mathbf{M}^{-1}(\mathbf{x}_{i'l'}-\mathbf{x}_{i'j'}) \\
&+ \sum_{ijl}\alpha_{ijl}
\end{aligned} \tag{A.8}
$$

$$
L(\mathbf{w},\xi,\boldsymbol{\alpha},\mathbf{r}) = \sum_{ijl}\alpha_{ijl} - \frac{1}{2}\sum_{ijl}\sum_{i'j'l'}\alpha_{ijl}\alpha_{i'j'l'}(\mathbf{x}_{il}-\mathbf{x}_{ij})^T\mathbf{M}^{-1}(\mathbf{x}_{i'l'}-\mathbf{x}_{i'j'}) \tag{A.9}
$$

Appendix B

Link between SVM and the Quadratic formalization

Objective
First, let consider the M^{2}TML quadratic formalization:

$$\begin{aligned}
&\underset{\mathbf{w},\xi}{\operatorname{argmin}} \left\{ \underbrace{\frac{1}{2}\mathbf{w}^T\mathbf{M}\mathbf{w}}_{pull} + \underbrace{C \sum_{\substack{i \\ j\in Pull_i \\ l\in Push_i}} \xi_{ijl}}_{push} \right\} \\
&\text{s.t. } \forall i = 1,\dots,n, \forall j \in Pull_i, l \in Push_i, \\
&\quad \mathbf{w}^T(\mathbf{x}_{il} - \mathbf{x}_{ij}) \geq 1 - \xi_{ijl} \\
&\quad \xi_{ijl} \geq 0
\end{aligned} \tag{B.1}$$

Secondly, let consider the SVM problem that separates $Pull_i$ and $Push_i$ sets:

$$\begin{aligned}
&\underset{\mathbf{w},b,\boldsymbol{\xi}}{\operatorname{argmin}} \left\{ \frac{1}{2}||\mathbf{w}||_2^2 + C \sum_{\substack{i \\ j\in Pull_i}} p_i^+ \xi_{ij} + C \sum_{\substack{i \\ l\in Push_i}} p_i^- \xi_{il} \right\} \\
&\text{s.t. :} \\
&\forall i,j \in Pull_i : y_{ij}(\mathbf{w}^T\mathbf{x}_{ij} + b) \geq 1 - \xi_{ij} \\
&\forall i,l \in Push_i : y_{il}(\mathbf{w}^T\mathbf{x}_{il} + b) \geq 1 - \xi_{il} \\
&\forall i,j \in Pull_i : \xi_{ij} \geq 0 \\
&\forall i,l \in Push_i : \xi_{il} \geq 0
\end{aligned} \tag{B.2}$$

where p_i^+ and p_i^- are the weight factors for pull pairs $Pull_i$ and push pairs $Push_i$.

We study here a link between the two problems when $D(\mathbf{x}_{ij}) = -\frac{1}{2}(\mathbf{w}^T\mathbf{x}_{ij} + b)$ and p_i^- and

p_i^+ are defined as:

$$p_i^- = \frac{Card(Pull_i)}{2} = \sum_{j \in Pull_i} \frac{1}{2} \tag{B.3}$$

$$p_i^+ = \frac{Card(Push_i)}{2} = \sum_{l \in Push_i} \frac{1}{2} \tag{B.4}$$

Similarities and differences in the constraints
First, we recall the SVM constraints in Eq. B.2:

$$\begin{aligned} +(\mathbf{w}^T \mathbf{x}_{ij} + b) &\geq 1 - \xi_{ij} \quad (Pull_i : y_{ij} = +1) \\ -(\mathbf{w}^T \mathbf{x}_{il} + b) &\geq 1 - \xi_{il} \quad (Push_i : y_{ij} = -1) \end{aligned}$$

By defining $D(\mathbf{x}_{ij}) = -\frac{1}{2}(\mathbf{w}^T \mathbf{x}_{ij} + b)$, the equations lead to:

$$\begin{aligned} -D(\mathbf{x}_{ij}) &\geq \frac{1}{2} - \frac{\xi_{ij}}{2} \\ D(\mathbf{x}_{il}) &\geq \frac{1}{2} - \frac{\xi_{il}}{2} \end{aligned}$$

By summing each constraint two by two, this set of constraints implies the following set of constraints:

$$\begin{cases} \bullet \forall i, j, k, l \text{ such that } y_{ij} = -1, \text{ and } y_{kl} = +1, i \neq j \text{ and } i \neq k : \\ D(\mathbf{x}_k, \mathbf{x}_l) - D(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - \frac{\xi_{kl} + \xi_{ij}}{2} \\ \bullet \forall i, j, l \text{ such that } y_{ij} = -1, \text{ and } y_{il} = +1, i \neq j : \\ D(\mathbf{x}_i, \mathbf{x}_l) - D(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - \frac{\xi_{il} + \xi_{ij}}{2} \end{cases} \tag{B.5}$$

By defining $\xi_{ijl} = \frac{\xi_{ij} + \xi_{il}}{2}$, the second constraint in Eq. B.5 from the M^{2}TML formulation is the same as the constraints in the SVM formulation in Eq. B.2.
By defining $\xi_{ijkl} = \frac{\xi_{ij} + \xi_{kl}}{2}$, we note that an additional set of constraints is present in the SVM formulation (first set of constraints in Eq. B.5) and not in M^{2}TML.

Similarities and differences in the objective function
Mathematically, from Eq. B.3, we write:

$$\sum_{\substack{i \\ l \in Push_i}} p_i^- \xi_{il} = \sum_{\substack{i \\ l \in Push_i}} \left(\sum_{j \in Pull_i} \frac{1}{2} \right) \xi_{il} \tag{B.6}$$

$$= \sum_{\substack{i \\ j \in Pull_i \\ l \in Push_i}} \frac{\xi_{il}}{2} \tag{B.7}$$

And from Eq. B.4, we write:

$$\sum_{\substack{i \\ j \in Pull_i}} p_i^+ \xi_{ij} = \sum_{\substack{i \\ j \in Pull_i}} \left(\sum_{l \in Push_i} \frac{1}{2} \right) \xi_{ij} \tag{B.8}$$

$$= \sum_{\substack{i \\ j \in Pull_i \\ l \in Push_i}} \frac{\xi_{ij}}{2} \tag{B.9}$$

By replacing Eqs. B.3 and B.4 back into Eq. B.2, the objective function becomes:

$$\underset{\mathbf{w},\xi}{\operatorname{argmin}} \left\{ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \sum_{\substack{i \\ j \in Pull_i \\ l \in Push_i}} \frac{\xi_{ij}}{2} + C \sum_{\substack{i \\ j \in Pull_i \\ l \in Push_i}} \frac{\xi_{il}}{2} \right\} \tag{B.10}$$

$$\underset{\mathbf{w},\xi}{\operatorname{argmin}} \left\{ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \sum_{\substack{i \\ j \in Pull_i \\ l \in Push_i}} \frac{\xi_{ij} + \xi_{il}}{2} + C \sum_{\substack{i \\ j \in Pull_i \\ k \\ l \in Push_k}} \frac{\xi_{ij} + \xi_{kl}}{2} \right\} \tag{B.11}$$

$$\underset{\mathbf{w},\xi}{\operatorname{argmin}} \left\{ \underbrace{\frac{1}{2}\mathbf{w}^T\mathbf{w}}_{Regularization} + C \underbrace{\left(\sum_{\substack{i \\ j \in Pull_i \\ l \in Push_i}} \xi_{ijl} + \sum_{\substack{i \\ j \in Pull_i \\ k \\ l \in Push_k}} \xi_{ijkl} \right)}_{Loss} \right\} \tag{B.12}$$

the SVM formulation (Eq. B.2) and the M^{2}TML formalization (Eq. B.1) share a same loss term involving the slack variables ξ_{ijl} (push cost). However, the SVM formulation includes additionnal slack variables ξ_{ijkl} due to the additional set of constraints.
In the SVM formulation (Eq. B.2), the regularization part tends to minimize the norm of $\mathbf{w}$ whereas in M^{2}TML (Eq. B.1), it tends to minimize the norm of $\mathbf{w}$ after a linear transformation through the matrix $\mathbf{M}$ (pull cost).

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. 22).

[Alt92] N.S. 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. 38).

[BA10] Nitin Bhatia and Corres Author. "Survey of Nearest Neighbor Techniques." In: *IJCSIS) International Journal of Computer Science and Information Security* 8.2 (2010), pp. 302–305. arXiv: `1007.0085` (cit. on p. 17).

[BC94] D. Berndt and J. 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. 39, 40).

[Ben+09] J. Benesty et al. "Pearson correlation coefficient." In: *Noise Reduction in Speech Processing* (2009) (cit. on pp. 35, 38).

[BGV92] B.E. Boser, I.M. Guyon, and V.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).

[BHS12] A. Bellet, A. Habrard, and M. Sebban. "Good edit similarity learning by loss minimization." In: *Machine Learning* 89.1-2 (2012), pp. 5–35 (cit. on pp. 48, 49).

[BHS13] A. Bellet, A. Habrard, and M. Sebban. *A Survey on Metric Learning for Feature Vectors and Structured Data*. Tech. rep. 2013. arXiv: `1306.6709` (cit. on p. 49).

[Bis06] C.M. Bishop. *Pattern Recognition and Machine Learning*. Vol. 4. 4. 2006, p. 738. arXiv: `0-387-31073-8` (cit. on pp. 8, 20, 29).

[BM67] E.O. Brigham and R.E. Morrow. "The fast Fourier transform." In: *Spectrum, IEEE* 4.12 (1967), pp. 63 –70 (cit. on p. 36).

[BMP02] S. Belongie, J. Malik, and J. 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).

[CA80] J.D. Carroll and P. Arabie. "Multidimensional scaling." In: *Annual review of psychology* 31.1 (1980), pp. 607–649 (cit. on p. 33).

[CCP06] J. Caiado, N. Crato, and D. Peña. "A periodogram-based metric for time series classification." In: *Computational Statistics and Data Analysis* 50.10 (2006), pp. 2668–2684 (cit. on p. 35).

[CGS05] Stella M. Clarke, Jan H. Griebsch, and Timothy W. Simpson. "Analysis of Support Vector Regression for Approximation of Complex Engineering Analyses." In: *Journal of Mechanical Design* 127.6 (2005), p. 1077 (cit. on p. 28).

[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).

[Cha+08] A. Chan et al. "Wavelet distance measure for person identification using electrocardiograms." In: *IEEE Transactions on Instrumentation and Measurement* 57.2 (2008), pp. 248–253 (cit. on p. 35).

[Cha+10] R. Chatpatanasiri et al. "A new kernelization framework for Mahalanobis distance learning algorithms." In: *Neurocomputing* 73.10-12 (2010), pp. 1570–1579 (cit. on p. 49).

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

[CHY96] M.S. Chen, J. Han, and P.S. Yu. *Data mining: An Overview from a Database Perspective*. 1996 (cit. on p. 8).

[Coc77] W. Cochran. "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. 13).

[CS01] K. Crammer and Y. 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] L. Cao and F.E.H. Tay. "Financial Forecasting Using Support Vector Machines." In: *Neural Computing & Applications* (2001), pp. 184–192 (cit. on p. 32).

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

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

[DC03] A. Douzal-Chouakria. "Compression technique preserving correlations of a multivariate temporal sequence." In: *Advances in Intelligent Data Analysis V*. Springer Berlin Heidelberg, 2003, pp. 566–577 (cit. on pp. 35, 38).

[DCA12] A. Douzal-Chouakria and C. Amblard. "Classification trees for time series." In: *Pattern Recognition* 45.3 (2012), pp. 1076–1091 (cit. on pp. 35, 38, 42, 54, 92).

[DCN07] A. Douzal-Chouakria and P. Nagabhushan. "Adaptive dissimilarity index for measuring time series proximity." In: *Advances in Data Analysis and Classification* (2007), pp. 5–21 (cit. on pp. 35, 38, 39, 42).

[DD09] M.M. Deza and E. Deza. *Encyclopedia of Distances*. Vol. 2006. 2009, p. 590. arXiv: `0505065 [arXiv:gr-qc]` (cit. on pp. 33, 88).

[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).

[DH73] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Vol. 7. 1973, p. 482 (cit. on pp. 8, 10, 16).

[DHB95] T. Dietterich, H. Hild, and G. 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. 13).

[Die98] T. Dietterich. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms." In: *Neural Computation* 10 (1998), pp. 1895–1923 (cit. on p. 13).

[Din+08] H. 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. 1, 16, 35, 36).

[Do+12] H. 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 p. 66).

[Dre+06] G. Dreyfus et al. *Apprentissage Apprentissage statistique*. Eyrolles. 2006, p. 471 (cit. on pp. 8, 10).

[Dud76] S. Dudani. "Distance weighed k-Nearest Neighbor rule." In: *IEEE Transactions on Systems, Man and Cybernetics* SMC-6.4 (1976), pp. 325–327 (cit. on p. 16).

[FCH08] R.E. Fan, K.W. Chang, and C.J. Hsieh. "LIBLINEAR: A library for large linear classification." In: *The Journal of Machine Learning* (2008) (cit. on pp. 23, 66, 78).

[FDCG13] C. Frambourg, A. Douzal-Chouakria, and E. Gaussier. "Learning multiple temporal matching for time series classification." In: *Intelligent Data Analysis*. 2013, pp. 198–209 (cit. on pp. 1, 49).

[Fu11] T. Fu. *A review on time series data mining*. 2011 (cit. on p. 54).

[GZZ05] X. Geng, D. Zhan, and Z. Zhou. "Supervised nonlinear dimensionality reduction for visualization and classification." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 35.6 (2005), pp. 1098–1107 (cit. on p. 33).

[HCL08] C. Hsu, C. Chang, and C. Lin. "A Practical Guide to Support Vector Classification." In: *BJU international* 101.1 (2008), pp. 1396–400. arXiv: `0-387-31073-8` (cit. on pp. 14, 27, 66, 78).

[HHK12] S. Hwang, D. Ham, and J. 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. 32).

[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] J. Hu, J. Wang, and G. Zeng. "A hybrid forecasting approach applied to wind speed time series." In: *Renewable Energy* 60 (2013), pp. 185–194 (cit. on p. 32).

[JJO11] Y. Jeong, M.K. Jeong, and O.A. Omitaomu. "Weighted dynamic time warping for time series classification." In: *Pattern Recognition* 44.9 (2011), pp. 2231–2240 (cit. on p. 49).

[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. 38).

[Keo+03] E. Keogh et al. "Segmenting Time Series: A Survey and Novel Approach." In: *Data Mining in Time Series Databases* (2003), pp. 1–21 (cit. on p. 54).

[Keo+11] E. Keogh et al. *The UCR Time Series Classification/Clustering Homepage.* 2011 (cit. on p. 75).

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

[KGP01] K. Kalpakis, D. Gada, and V. Puttagunta. "Distance Measures for Effective Clustering of ARIMA Time-Series." In: *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM'01).* San Jose, CA, 2001, pp. 273–280 (cit. on p. 35).

[KR04] E. Keogh and C.A. Ratanamahatana. "Exact indexing of dynamic time warping." In: *Knowledge and Information Systems* 7.3 (2004), pp. 358–386 (cit. on p. 40).

[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).

[Lee+09] H. Lee et al. "Unsupervised feature learning for audio classification using convolutional deep belief networks." In: *Nips* (2009), pp. 1–9 (cit. on p. 15).

[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. 36).

[Lia+12] C. 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. 32).

[Lin+03] J. Lin et al. "A Symbolic Representation of Time Series, with Implications for Streaming Algorithms." In: *SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery* (2003), pp. 2–11 (cit. on p. 35).

[Mah96] E.A. Maharaj. "A Significance Test for Classifying ARMA Models." In: *Journal of Statistical Computation and Simulation* 54 (1996), pp. 305–331 (cit. on p. 35).

[Mar00] R.J. Martin. "Metric for ARMA processes." In: *IEEE Transactions on Signal Processing* 48.4 (2000), pp. 1164–1170 (cit. on p. 35).

[Mau06] F.M. Maurice. "Sur quelques points du calcul fonctionnel." In: *Rendiconti del Circolo Matematico di Palermo (1884-1940)* 22.1 (1906), pp. 1–72 (cit. on p. 35).

[Mei+15] Jiangyuan Mei et al. *Learning a Mahalanobis Distance-Based Dynamic Time Warping Measure for Multivariate Time Series Classification.* 2015 (cit. on p. 49).

[MU13] I.B. Mohamad and D. Usman. "Standardization and its effects on K-means clustering algorithm." In: *Research Journal of Applied Sciences, Engineering and Technology* 6.17 (2013), pp. 3299–3303 (cit. on p. 14).

[MV14] P. Montero and J. Vilar. "TSclust : An R Package for Time Series Clustering." In: *Journal of Statistical Software November* 62.1 (2014) (cit. on pp. 1, 35, 88).

[Nag91] N. Nagelkerke. "A note on a general definition of the coefficient of determination." In: *Biometrika* 78.3 (1991), pp. 691–692 (cit. on p. 13).

[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, 35, 39).

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

[OS06] J. Oncina and M. Sebban. "Learning stochastic edit distance: Application in handwritten character recognition." In: *Pattern Recognition* 39 (2006), pp. 1575–1587 (cit. on p. 49).

[Pan+08] C. 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).

[Pic90] D. Piccolo. "A distance measure for classifying ARIMA models." In: *Journal of Time Series Analysis* (1990) (cit. on p. 35).

[PL12] Z. Prekopcsák and D. 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. 36).

[Qui86] J.R. Quinlan. "Induction of Decision Trees." In: *Machine Learning* 1.1 (1986), pp. 81–106 (cit. on p. 15).

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

[Rey11] M. Reyes. "Feature weighting in dynamic time warping for gesture recognition in depth data." In: *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on* (2011), pp. 1182–1188 (cit. on p. 49).

[RJ93] L.R. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Vol. 103. 1993 (cit. on p. 41).

[RSC02] M. Ramoni, P. Sebastiani, and P. Cohen. "Bayesian clustering by dynamics." In: *Machine Learning* 47.1 (2002), pp. 91–121 (cit. on p. 35).

[SC04] S. Salvador and P. Chan. "FastDTW : Toward Accurate Dynamic Time Warping in Linear Time and Space." In: *3rd Workshop on Mining Temporal and Sequential Data*. 2004 (cit. on p. 40).

[SC78] H. Sakoe and S. Chiba. "Dynamic Programming Algorithm Optimization for Spoken Word Recognition." In: *IEEE transactions on acoustics, speech, and signal processing* ASSP-26.1 (1978), pp. 43–49 (cit. on p. 41).

[SJ89] B. Silverman and M. Jones. "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).

[Smy97] P. Smyth. "Clustering sequences with hidden Markov models." In: *Advances in neural information processing systems* 9 (1997), pp. 648–654 (cit. on p. 35).

[SS12a] M. Sahidullah and G. 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. 1).

[SS12b] M. Sahidullah and G. 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. 36).

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

[SSB03] J. Sadri, C. Suen, and T. 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] C. Torrence and G. Compo. "A Practical Guide to Wavelet Analysis." In: *Bulletin of the American Meteorological Society* 79.1 (1998), pp. 61–78 (cit. on p. 36).

[VSV07] S V N Vishwanathan, A J Smola, and R Vidal. "Binet-cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes." In: *International Journal of Computer Vision* 73.1 (2007), pp. 95–119 (cit. on p. 35).

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

[Wie42] N. Wiener. *Extrapolation, Interpolation & Smoothing of Stationary Time Series - With Engineering Applications*. 1942 (cit. on p. 38).

[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. 2, 33, 48–51, 58, 89).

[WWK12] J. Wang, A. Woznica, and A. Kalousis. "Parametric local metric learning for nearest neighbor classification." In: *arXiv preprint arXiv:1209.3056* (2012), pp. 1–9 (cit. on p. 89).

[Xi+06] X. Xi et al. "A 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).

[XWC12] Z. Xu, K. Weinberger, and O. Chapelle. "Distance Metric Learning for Kernel Machines." In: *Arxiv* (2012), pp. 1–17. arXiv: `1208.3422` (cit. on p. 49).

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

[YL99] Y. Yang and X. 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).

[Zha+13] Y. Zhang et al. "Multi-metric learning for multi-sensor fusion based classification." In: *Information Fusion* 14.4 (2013), pp. 431–440 (cit. on p. 42).

[Zhu07] X. Zhu. "Semi-Supervised Learning Literature Survey." In: *Sciences-New York* (2007), pp. 1–59 (cit. on p. 8).

[ZLL14] X.L. Zhang, Z.G. Luo, and M. Li. "Merge-weighted dynamic time warping for speech recognition." In: *Journal of Computer Science and Technology* 29.6 (2014), pp. 1072–1082 (cit. on p. 49).

[ZMP14] N. Zumel, J. Mount, and J. Porzak. *Practical Data Science with R*. 2014 (cit. on p. 44).

[ZY10] Y. Zhang and D. Yeung. "Transfer metric learning by learning task relationships." In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10* (2010), p. 1199 (cit. on p. 49).

Résumé — La définition d'une métrique entre des séries temporelles est un élément important pour de nombreuses tâches en analyse ou en fouille de données, tel que le clustering, la classification ou la prédiction. Les séries temporelles présentent naturellement différentes caractéristiques, que nous appelons modalités, sur lesquelles elles peuvent être comparées, comme leurs valeurs, leurs formes ou leurs contenus fréquentielles. Ces caractéristiques peuvent être exprimées avec des délais variables et à différentes granularités ou localisations temporelles —exprimées globalement ou localement. Combiner plusieurs modalités à plusieurs échelles pour apprendre une métrique adaptée est un challenge clé pour de nombreuses applications réelles impliquant des données temporelles. Cette thèse propose une approche pour l'Apprentissage d'une Métrique Multi-modal et Multi-scale (M^{2}TML) en vue d'une classification robuste par plus proches voisins. La solution est basée sur la projection des paires de séries temporelles dans un espace de dissimilarités, dans lequel un processus d'optimisation à vaste marge est opéré pour apprendre la métrique. La solution M^{2}TML est proposée à la fois dans le contexte linéaire et non-linéaire, et est étudiée pour différents types de régularisation. Une variante parcimonieuse et interprétable de la solution montre le potentiel de la métrique temporelle apprise à pouvoir localiser finement les modalités discriminantes, ainsi que leurs échelles temporelles en vue de la tâche d'analyse considérée. L'approche est testée sur un vaste nombre de 30 bases de données publiques et challenging, couvrant des images, traces, données ECG, qui sont linéairement ou non-linéairement séparables. Les expériences montrent l'efficacité et le potentiel de la méthode M^{2}TML pour la classification de séries temporelles par plus proches voisins.

Summary — The definition of a metric between time series is inherent to several data analysis and mining tasks, including clustering, classification or forecasting. Time series data present naturally several characteristics, called modalities, covering their amplitude, behavior or frequential spectrum, that may be expressed with varying delays and at different temporal granularity and localization —exhibited globally or locally. Combining several modalities at multiple temporal scales to learn a holistic metric is a key challenge for many real temporal data applications. This PhD proposes a Multi-modal and Multi-scale Temporal Metric Learning (M^{2}TML) approach for robust time series nearest neighbors classification. The solution is based on the embedding of pairs of time series into a pairwise dissimilarity space, in which a large margin optimization process is performed to learn the metric. The M^{2}TML solution is proposed for both linear and non linear contexts, and is studied for different regularizers. A sparse and interpretable variant of the solution shows the ability of the learned temporal metric to localize accurately discriminative modalities as well as their temporal scales. A wide range of 30 public and challenging datasets, encompassing images, traces and ECG data, that are linearly or non linearly separable, are used to show the efficiency and the potential of M^{2}TML for time series nearest neighbors classification.

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