Part I

Work positioning

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

CHAPTER 1

Related work

Sommaire

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

1.1 Classification, Regression

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

1.1.1 Machine learning principle

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

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

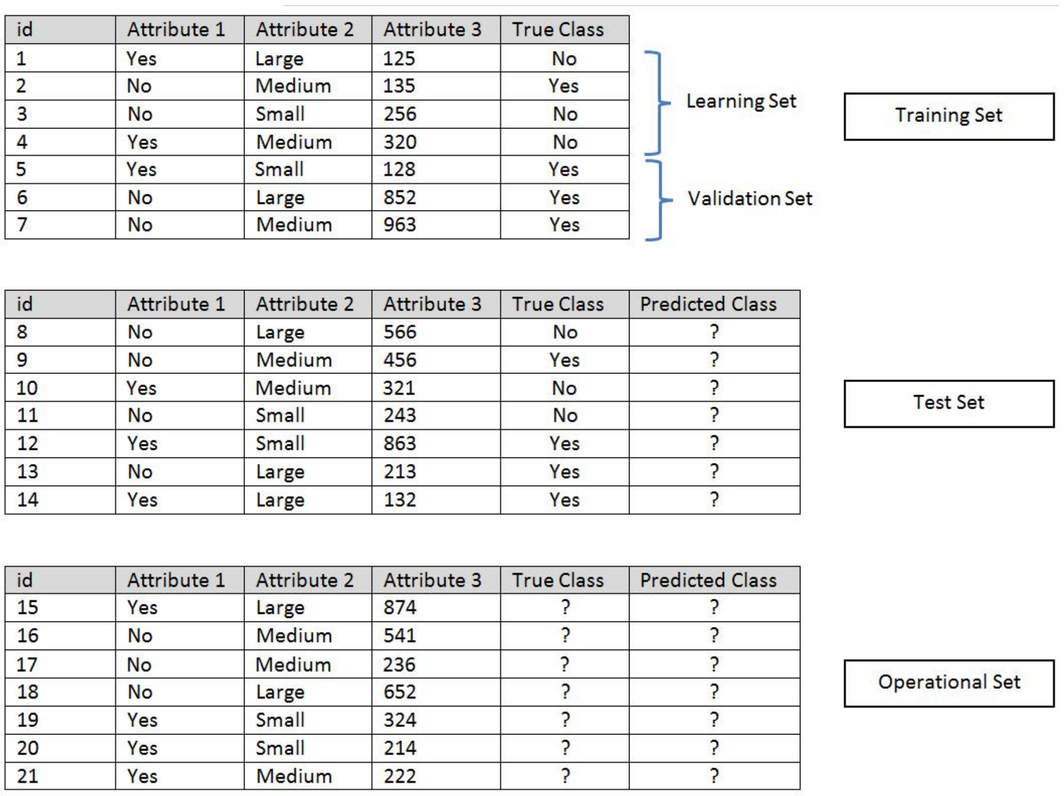
When y_i are class labels (e.g., class 'A', 'B', 'C' in the case of child's reading), learning the model f is a classification problem; when y_i is a continuous value (e.g., the energy consumption in a building), learning f is a regression problem. Both problems corresponds to supervised learning as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

[biblio semi-supervisé]

1.1.2 Model selection

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

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

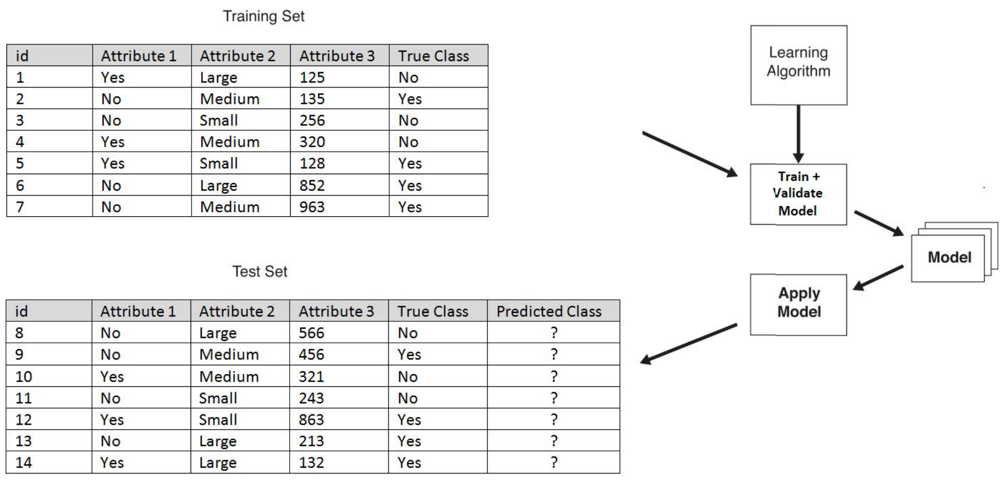


id	Attribute 1	Attribute 2	Attribute 3	True Class
1	Yes	Large	125	No
2	No	Medium	135	Yes
3	No	Small	256	No
4	Yes	Medium	320	No
5	Yes	Small	128	Yes
6	No	Large	852	Yes
7	No	Medium	963	Yes

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

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

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



Training Set

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

Test Set

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

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

There exists two types of errors committed by a classification or regression model f: training error and generalization error. **Training error** is the error on the training set and **generalization error** is the error on the testing set. A good supervised model f must not only fit the training data X well, it must also accurately classify records it has never seen before (test set X_{Test}). In other words, a good model f must have low training error as well as low generalization error. This is important because a model that fits the training data too much can have a poorer generalization error than a model with a higher training error. Such a situation is known as model overfitting (Fig. 1.3).

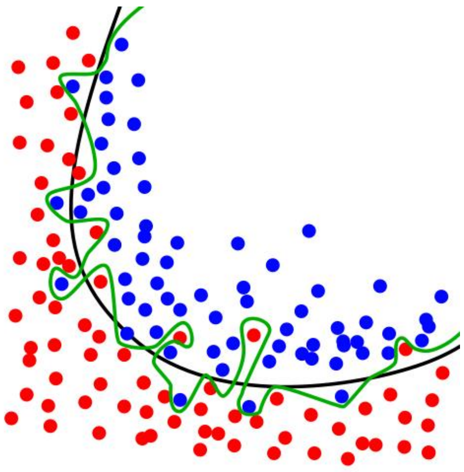


Figure 1.3: An example of overfitting in the case of classification. The objective is to separate blue points from red points. Black line shows a classifier f_1 with low complexity where as green line illustrates a classifier f_2 with high complexity. On training examples (blue and red points), the model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

In most cases, learning algorithms requires to tune some hyper-parameters. For that, the training set can be divided into 2 sets: a learning and a validation set. Suppose we have two hyper-parameters to tune: C and γ. We make a grid search for each combination (C, γ) of the hyper-parameters, that is in this case a 2-dimensional grid (Fig. 1.4). For each combination (a cell of the grid), the model is learnt on the learning set and evaluated on the validation set. At the end, the model with the lowest error on the validation set is retained. This process is refered as the model selection.

An alternative is cross-validation with v folds, illustrated in Fig. 1.5. In this approach, we partition the training data into v equal-sized subsets. The objective is to evaluate the error for each combination of hyper-parameters. For each run, one fold is chosen for validation, while the $v - 1$ remaining folds are used as the learning set. We repeat the process for each fold, thus v times. Each fold gives one validation error and thus we obtain v errors. The total error for the current combination of hyper-parameters is obtained by summing up the errors for all v folds. When $v = n$, the size of training set, this approach is called leave-one-out or Jackknife. Each test set contains only one sample. The advantage is much data are used as possible for training. Moreover, the validation sets are exclusive and they cover the entire data set. The drawback is that it is computationally expensive to repeat the procedure n times. Furthermore, since each validation set contains only one record, the variance of the estimated performance metric is usually high. This procedure is often used when n, the size training set,

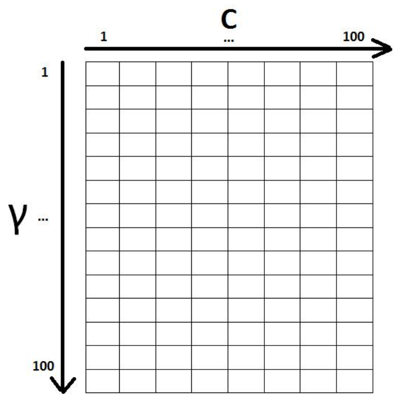


Figure 1.4: Example of a 2 dimensional grid search for parameters C and γ. It defines a grid where each cell of the grid contains a combination (C, γ). Each combination is used to learn the model and is evaluated on the validation set.

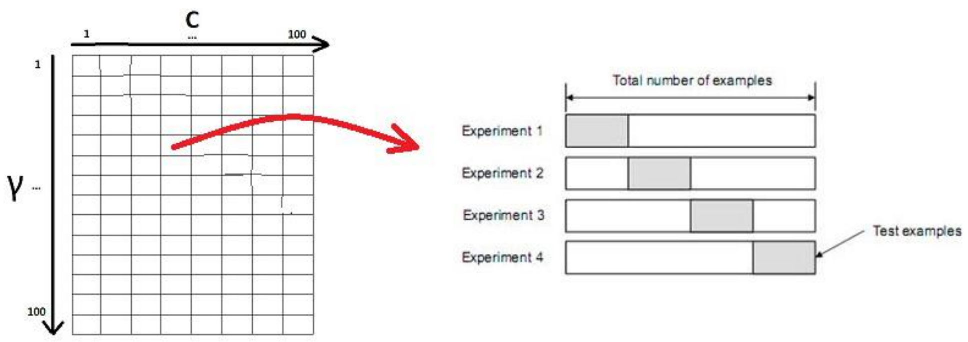


Figure 1.5: v-fold Cross-validation for one combination of parameters. For each of v experiments, use $v - 1$ folds for training and a different fold for Testing, then the training error for this combination of parameter is the mean of all testing errors. This procedure is illustrated for $v = 4$.

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

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

1.1.3 Model evaluation

1.1.3.a Classification evaluation

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

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

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

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

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

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

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

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

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

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

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

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

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

1.1.3.b Regression evaluation

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

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

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

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

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

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

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

1.1.4 Data normalization

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

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

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

références normalization

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

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

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

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

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

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

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

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

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

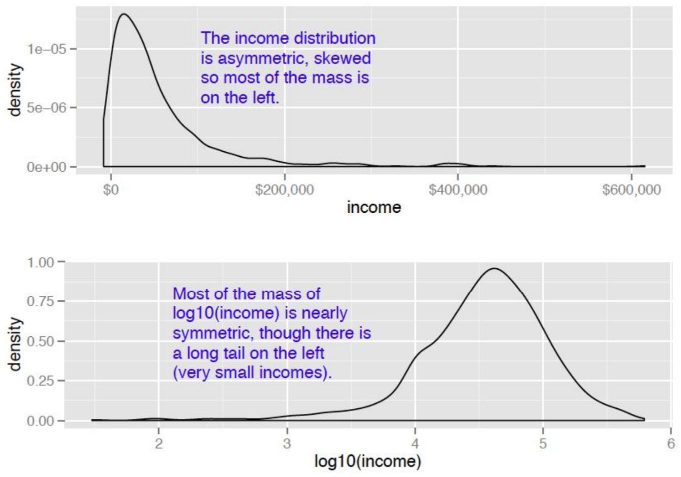


Figure 1.6: A nearly log-normal distribution, and its log transform [2]

1.2 Machine learning algorithms

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

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

1.2.1 k-Nearest Neighbors (k-NN) classifier

A simple approach to classify samples is to consider that "close" samples have a great probability to belong to the same class. Given a test sample $\mathbf{x}_j$, one can decide that $\mathbf{x}_j$ belong to the class y_i of its nearest neighbor $\mathbf{x}_i$ in the training set.

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

More generally, we can consider the k nearest neighbors of $\mathbf{x}_j$. The class y_j of the test sample $\mathbf{x}_j$ is assigned with a voting scheme among them, i.e., using the majority of the class of nearest neighbors. This algorithm is refer as the k-Nearest Neighbors algorithm (k-NN) [SJ89]; [CH67]. Fig. 1.7 illustrates the concept for a neighborhood of $k = 3$ and $k = 5$.

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

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

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

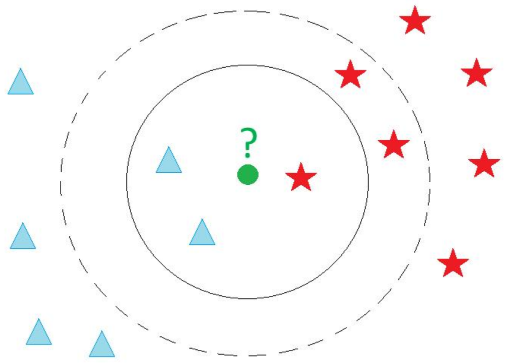


Figure 1.7: Example of k-NN classification. The test sample (green circle) is classified either to the first class (red stars) or to the second class (blue triangles). If $k = 3$ (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 star inside the inner circle.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

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

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

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

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

Comment [AD6]: Expliquer d'avantage

Despite its simplicity, the k-NN algorithm has been shown to be successful on time series classification problems [BMP02]; [Xi+06]; [Din+08]. However, the k-NN algorithm presents some disadvantages, mainly due to its computational complexity, both in space (storage of the training samples $\mathbf{x}_i$) and time (search of the neighbors) [OE73]. Suppose we have n labeled training samples in p dimensions, and find the closest neighbors to a test sample $\mathbf{x}_j$ ($k = 1$). In the most simple approach, we look at each stored samples $\mathbf{x}_i$ ($i = 1...n$) one by one, calculate its metric to $\mathbf{x}_i$ (D($\mathbf{x}_i$,$\mathbf{x}_j$)) and retain the index of the current closest one. For the standard Euclidean distance, each metric computation is $O(p)$ and thus the search is $O(pn)$. Moreover, using standard metrics (such as the Euclidean distance) uses all the p dimensions in its computation and thus assumes that all dimensions have the same effect on the metric. This assumption may be wrong and can impact the classification performances.

1.2.2 Support Vector Machine (SVM) algorithm

Support Vector Machine (SVM) is a classification method introduced in 1992 by Boser, Guyon, and Vapnik [BGV92]; [CV95] to solve at first linearly separable problems. The SVM classifier have demonstrate high accuracy, ability to deal with high-dimensional data, good generalization properties and interpretation for various applications from recognizing handwritten digits, to face identification, text categorization, bioinformatics and database marketing [Wan02]; [YL99]; [HHP01]; [SSB03]; [CY11]. SVMs belong to the category of kernel methods, algorithms that depends on the data only through dot-products [SS13]. It allows thus to solve non-linear problem. This section gives a brief overview of the mathematical key points and interpretation of the method. For more informations, the reader can consult [SS13]; [CY11]; [CV95].

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

1.2.2.a Intuition

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

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

Vapnik & al. [CV95] propose to choose the separating hyperplane that maximizes the margin, e.g. the hyperplane that leaves as much distance as possible between the hyperplane and the closest samples $\mathbf{x}_i$ of each class, called the support vectors. This distance is equal to $\frac{1}{||\mathbf{w}||_2}$. We denote $||\mathbf{w}||_2$, the L2-norm of the vector $\mathbf{w}$ and $||\mathbf{w}||_1$ the L1-norm of $\mathbf{w}$:

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

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

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

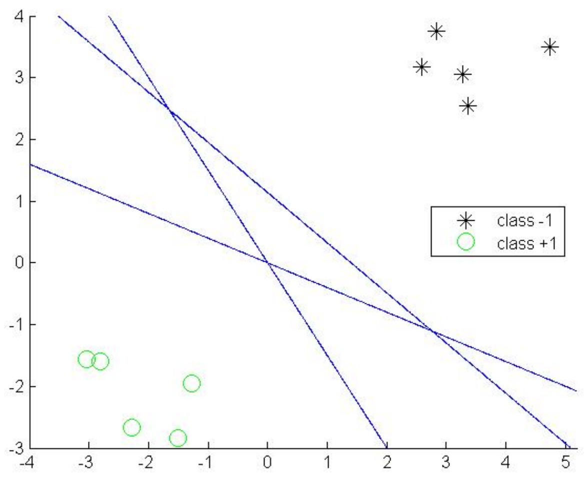


Figure 1.8: Example of linear classifiers in a 2-dimensional plot. For a set of points of classes $+1$ and -1 that are linearly separable, there exists an infinite number of separating hyperplanes corresponding to $\mathbf{w}^T\mathbf{x} + b = 0$.

1.2.2.b Primal formulation

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

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

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

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

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

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

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

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

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

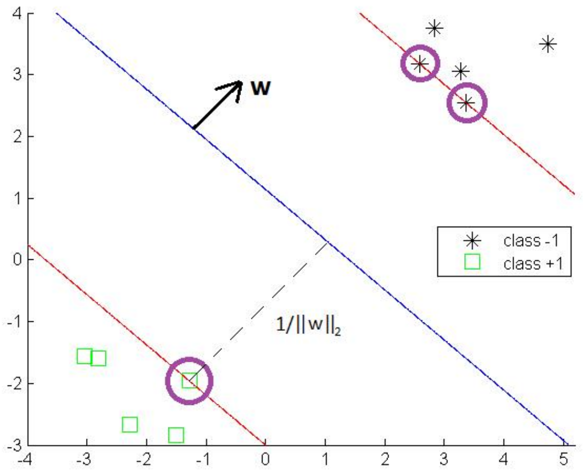


Figure 1.9: The argument inside the decision function of a classifier is $\mathbf{w}^T\mathbf{x}+b$. The separating hyperplane corresponding to $\mathbf{w}^T\mathbf{x}+b=0$ is shown as a line in this 2-dimensional plot. This hyperplane separates the two classes of data with points on one side labeled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

solutions is a global solution. The objective function in Eq. 1.18 is made of two terms. The first one, the regularization term, penalizes the complexity of the model and thus, controls the ability of the algorithm to generalize on new samples. The second one, the loss term, is an adaptation term to the data. The hyper-parameter C is a trade-off between the regularization and the loss term. When C tends to $+\infty$, the problem is equivalent to the primal hard margin problem. The hyper-parameter C is learnt during the training phase.

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

Two common regularizer are $||\mathbf{w}||_1$ and $||\mathbf{w}||_2$. The former is referred to as L1-Regularizer while the latter is L2-Regularizer. L1-Regularizer is used to obtain sparser models than L2-Regularizer. Thus, it can be used for variable selection.

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

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

1.2.2.c Dual formulation

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

First, to simplify the calculation development, let consider the hard margin formulation in Eq. 1.18, 1.19 and 1.20 with a L1-Loss function. As a constrained optimization problem, the formulation is equivalent to the minimization of a Lagrange function $L(\mathbf{w}, b)$, consisting of the sum of the objective function and the n constraints multiplied by their respective Lagrange multipliers $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_n]^T$:

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

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

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

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

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

where $\alpha_i \geq 0$ are the Lagrange multipliers. In optimization theory, Eq. 1.23, 1.24 and 1.25 are called the Karush-Kuhn-Tucker (KKT) conditions. It corresponds to the set of conditions which must be satisfied at the optimum of a constrained optimization problem. The KKT conditions will play an important role in the interpretation of SVM in Section 1.2.2.e.

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

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

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

that leads to:

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

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

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

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

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

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

The dual objective in Eq. 1.28 is quadratic in the parameters α_i. Adding the constraints in Eq. 1.29 and 1.30, it is a constrained quadratic programming optimization problem (QP). Note that while the primal formulation is minimization, the equivalent dual formulation is maximization. It can be shown that the objective functions of both formulations reach the same value when the solution is found [CY11].
In the same spirit, considering the soft margin primal problem, it can be shown that it leads to the same formulation [CY11] (Eqs. 1.28 and 1.29), except that the Lagrange multipliers α_i are upper bounded by the trade-off C in the soft margin formulation:

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

The constraints in Eq. 1.31 are called the Box constraints [CY11]. From the optimal value of α_i, denoted α_i^*, it is possible to compute the weight vector $\mathbf{w}^*$ and the bias b^* at the optimality:

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

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

At the optimality point, only a few number of datapoints have $\alpha_i^* > 0$ as shown as in Fig. 1.10. These samples are the vector supports. All other datapoints have $\alpha_i^* = 0$, and the decision function is independent of them. Thus, the representation is sparse.

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

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

1.2.2.d Kernel trick

The concept of kernels was introduced by Aizerman & Al in 1964 to design potential functions in the context of pattern recognition [ABR64]. The idea was re-introduced in 1992 by Boser &

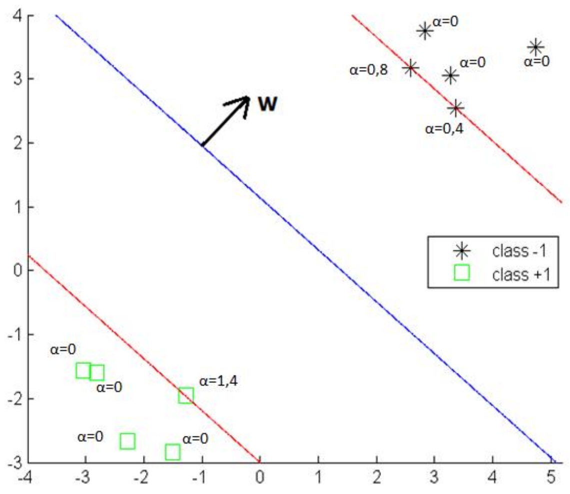


Figure 1.10: Obtained hyperplane after a dual resolution (full blue line). The 2 canonical hyperplanes (dash blue line) contains the support vectors whose $\alpha_i > 0$. Other points have their $\alpha_i = 0$ and the equation of the hyperplane is only affected by the support vectors.

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

From the dual objective in Eq. 1.28, we note that the samples $\mathbf{x}_i$ are only involves in a dot-product. Therefore, we can map these samples $\mathbf{x}_i$ into a higher dimensional hyperspace, called the feature space, through the replacement:

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

where Φ is the mapping function. The intuition behind is that for many datasets, it is not possible to find a hyperplan that can separate the two classes in the input space if the problem is not linearly separable. However, by applying a transformation Φ, data might become linearly separable in a higher dimensional space (feature space). Fig. 1.11 illustrates the idea: in the original 2-dimensional space (left), the two classes can't be separated by a line. However, with a third dimension such that the $+1$ labeled points are moved forward and the -1 labeled moved back the two classes become separable.

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

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

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

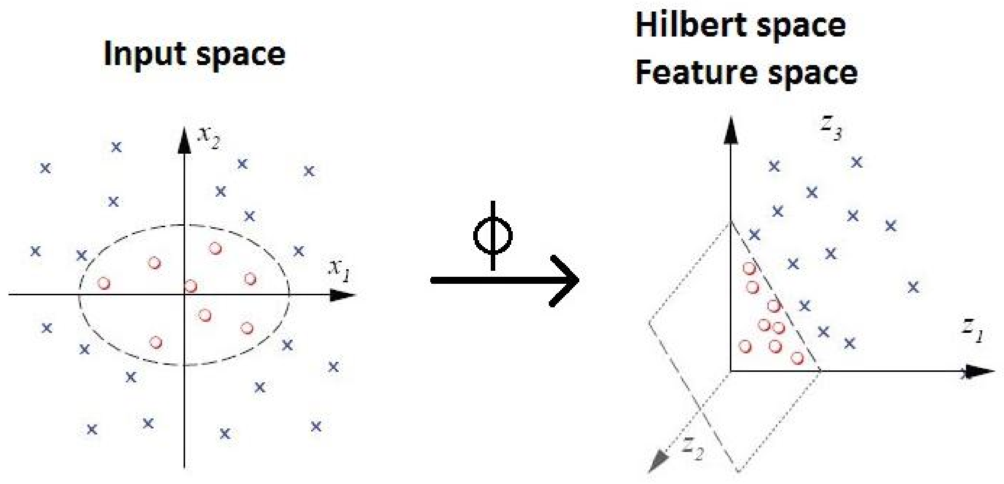


Figure 1.11: Left: in two dimensions these two classes of data are mixed together, and it is not possible to separate them by a line: the data is not linearly separable. Right: using a Gaussian kernel, these two classes of data (cross and circle) become separable by a hyperplane in feature space, which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

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

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

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

where $\gamma = \frac{1}{2\sigma^2}$ is the parameter of the Gaussian kernel and $||\mathbf{x}_j - \mathbf{x}_i||_2$ is the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. Note that the Linear kernel is the identity transformation. In practice, for large scale problem (when p is high), using a Linear kernel is sufficient [FCH08].

The Gaussian kernel computed between a sample $\mathbf{x}_j$ and a support vector $\mathbf{x}_i$ is an exponentially decaying function in the input feature space. The maximum value of the kernel ($K(\mathbf{x}_i, \mathbf{x}_j)$=1) is attained at the support vector (when $\mathbf{x}_i = \mathbf{x}_j$). Then, the value of the kernel decreases uniformly in all directions around the support vector, with distance and ranges between zero and one. It can thus be interpreted as a similarity measure. Geometrically speaking, it leads to hyper-spherical contours of the kernel function as shown in Fig. 1.12 [5]. The parameter γ controls the decreasing speed of the sphere. In practice, this parameter is learnt during the training phase.

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

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

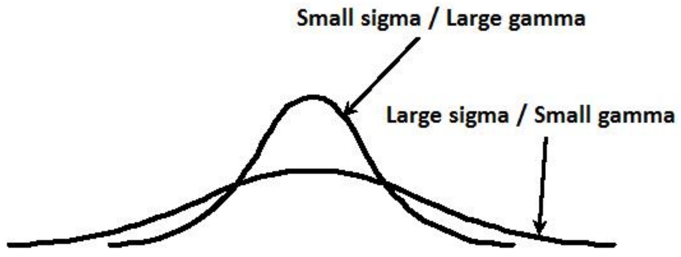


Figure 1.12: Illustration of the Gaussian kernel in the 1-dimensional input space for a small and large γ.

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

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

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

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

The decision function f becomes:

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

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

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

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

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

1.2.2.e Interpretation

Interpretation in the primal

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

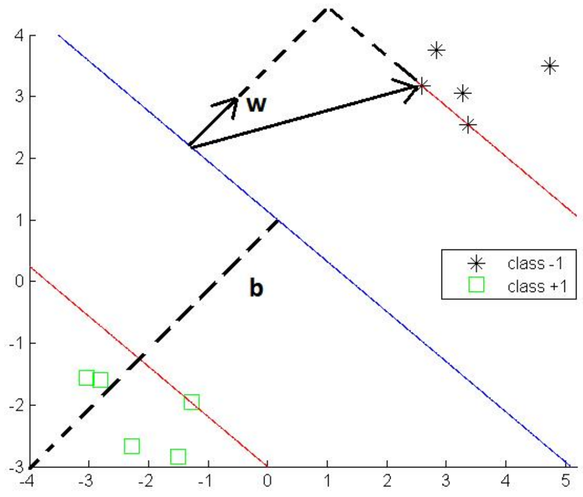


Figure 1.13: Geometric representation of SVM.

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

In order to visualize the above interpretation of the weight vector $\mathbf{w}$, let us examine several hyperplanes $\mathbf{w}^T\mathbf{x}+b = 0$ shown in Fig. 1.14 with $^T = 2$. Figure (a) shows a hyperplane where elements of $\mathbf{w}$ are the same for both variables $\mathbf{X}_1$ and $\mathbf{X}_2$. The interpretation is that both variables contribute equally for classification of objects into positive and negative. Figure (b) shows a hyperplane where the element of $\mathbf{w}$ for $\mathbf{X}_1$ is 1, while that for $\mathbf{X}_2$ is 0. This is interpreted as that $\mathbf{X}_1$ is important but $\mathbf{X}_2$ is not. An opposite example is shown in figure (c) where $\mathbf{X}_2$ is considered to be important but $\mathbf{X}_1$ is not. Finally, figure (d) provides a 3-dimensional example ($p = 3$) where an element of $\mathbf{w}$ for $\mathbf{X}_3$ is 0 and all other elements are equal to 1. The interpretation is that $\mathbf{X}_1$ and $\mathbf{X}_2$ are important but $\mathbf{X}_3$ is not.

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

Interpretation in the dual

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

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

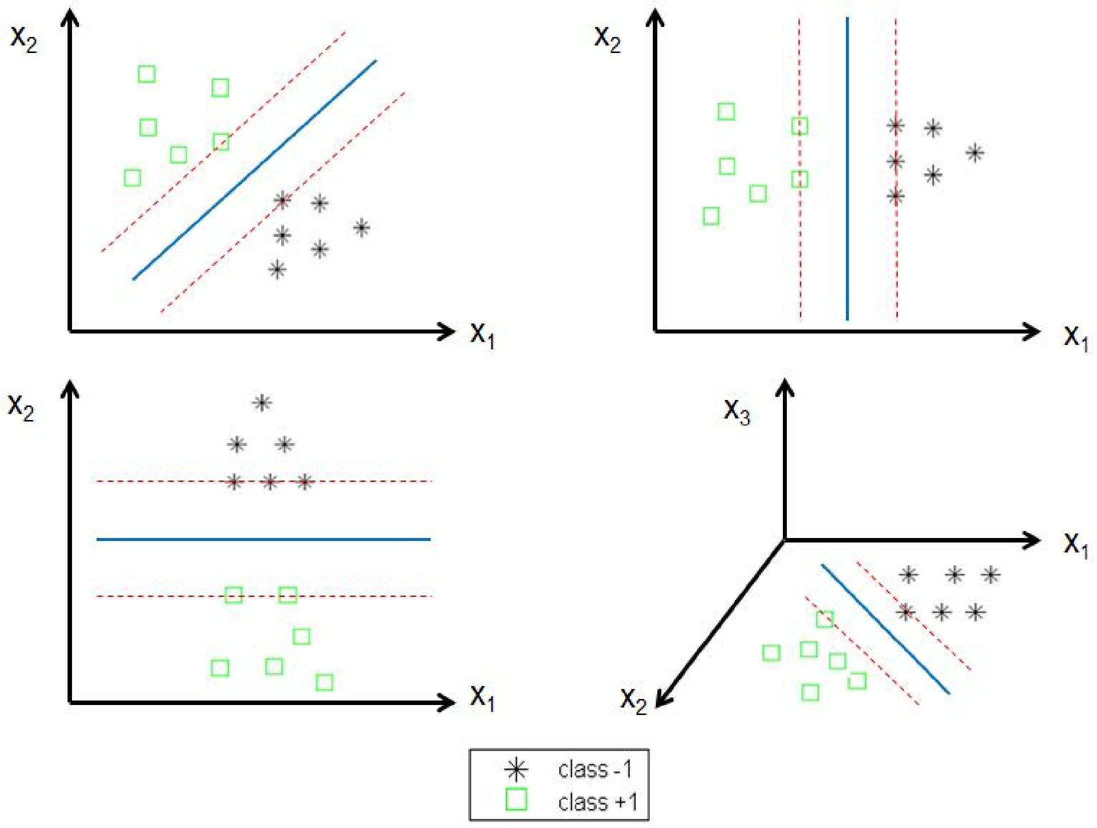


Figure 1.14: Example of several SVMs and how to interpret the weight vector $\mathbf{w}$

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

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

From this, for every datapoint $\mathbf{x}_i$, either $\alpha_i^* = 0$ or $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$. Any datapoint with $\alpha_i^* = 0$ do not appear in the sum of the decision function f in Eq. 1.34 or 1.41. Hence, they play no role for the classification decision of a new sample $\mathbf{x}_j$. The others $\mathbf{x}_i$ such that $\alpha_i^* > 0$ corresponds to the support vector. Looking at the distribution of α_i^* allows also to have either a better understanding of the datasets, or either to detect outliers. The higher is the coefficient α_i^* for a sample $\mathbf{x}_i$, the more the sample $\mathbf{x}_i$ impacts on the decision function f. However, unusual high value of α_i^* among the samples can lead to two interpretations: either this point is a critical point to the decision, either this point is an outlier. In the soft margin formulation, by constraining α_i^* to be inferior to C (Box constraints) the effect of outliers can be reduced and controlled.

1.2.2.f Extensions of SVM

SVM has received many interests in recent years. Many extensions has been developed such as ν-SVM, asymmetric soft margin SVM or multiclass SVM [KU02]; [CS01]. One interesting extension is the extension of Support Vector Machine to regression problems, also called Support Vector Regression (SVR). The objective is to find a linear regression model $f(\mathbf{x}) = \mathbf{w}.\mathbf{x} + b$. To preserve the property of sparseness, the idea is to consider an ϵ-insensitive error function. It gives zero error if the absolute difference between the prediction $f(\mathbf{x}_i)$ and the target y_i is less than ϵ where $\epsilon > 0$ penalize samples that are outside of a ϵ-tube as shown as in Fig. 1.15.

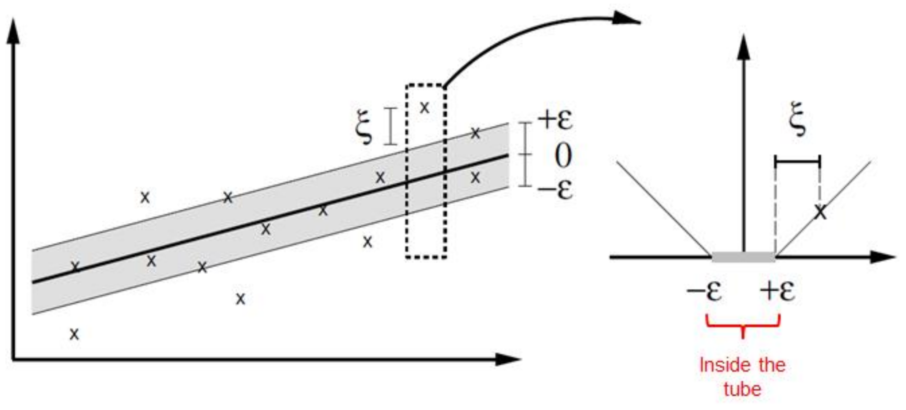


Figure 1.15: Illustration of SVM regression (left), showing the regression curve with the ϵ-insensitive "tube" (right). Samples $\mathbf{x}_i$ above the ϵ-tube have $\xi_1 > 0$ and $\xi_1 = 0$, points below the ϵ-tube have $\xi_2 = 0$ and $\xi_2 > 0$, and points inside the ϵ-tube have $\xi = 0$.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1.2.3 Other classification algorithms

Partie non encore rédigée. A faire à la fin.

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

1.3 Conclusion of the chapter

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

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