TELECOM Paris **SD-TSIA 210**
2019/2020 **G. Staerman, L. Brogat-Motte**

TP n° 1 : Support Vector Machine (SVM)

- Before start -

Useful links for this lab :

⋆⋆⋆ http://scikit-learn.org/stable/modules/svm.html

⋆⋆ http://en.wikipedia.org/wiki/Support_vector_machine

⋆⋆ http://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support

- Introduction and Mathematical Foundations -

The SVMs were introduced by Vapnik's group [3], and are discussed in Chapter 12 of the book [1]. More mathematical details can be be found in Chapter 7 of the book [2]. The popularity of SVM methods, for binary classification in particular, comes from the fact that they rely on the application of algorithms of linear decision rules : we speak of hyperplanes (affine) separators. However, this search takes place in a higher-dimensional *feature space*, that is the image of the original input space transformed by a non-linear transformation Φ.

The aim of this TP is to put into practice this type of classification techniques on actual and simulated data In the `scikit-learn` package (which implements the library in `C` LIBSVM) and to learn to control the parameters guaranteeing their flexibility (hyper-parameters, kernel).

Definitions and Notations

It is recalled that within the framework of the supervised binary classification we use the notations :

— $\mathcal{Y}$ The set of labels, commonly $\mathcal{Y} = \{-1, 1\}$ in the case of binary classification,

— $\mathbf{x} = (x_1, \ldots, x_p) \in \mathcal{X} \subset \mathbb{R}^p$ is an observation (or an example),

— $\mathcal{D}_n = \{(\mathbf{x}_i, y_i), i = 1, \ldots n\}$ a learning set containing n examples and their labels,

— There is a probabilistic model that governs the generation of our observations according to random variables X and Y : $\forall i \in \{1, \ldots, n\}, (\mathbf{x}_i, y_i) \overset{i.i.d}{\sim} (X, Y)$.

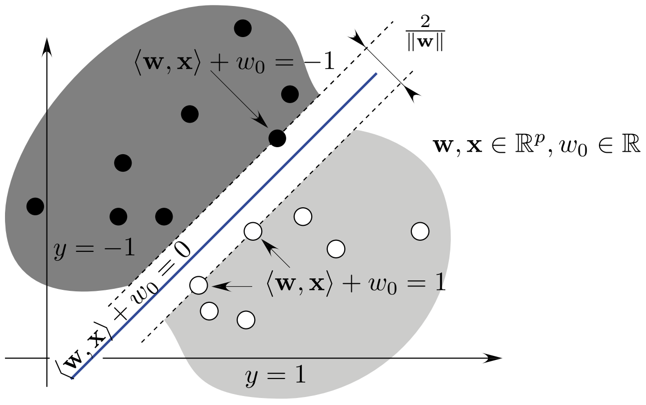
— We try to build from the learning set $\mathcal{D}_n$ a function $\hat{f} : \mathcal{X} \mapsto \{-1, 1\}$ which for an unknown point $\mathbf{x}$ (*i.e.,* which is not present in the learning set) predicts its label : $\hat{f}(\mathbf{x})$. The rule considered here is called linear, in the sense that the space is separated by a hyperplane (affine) : according to the position with respect thereto, then predicting $+1$ or -1.

SVM and kernels for binary classification

SVM (nonlinear) techniques use an implicit function Φ transforming the input space $\mathcal{X} \subset \mathbb{R}^p$ in a Hilbert space $(\mathcal{H}, \langle\cdot,\cdot\rangle)$ of higher dimension. Learning is then carried out using the model $(\Phi(X), Y)$ in the space $\mathcal{H}$, of larger size of course, but in which it is hoped that the data is "more linearly separable". From a practical point of view, It should be noted that the calculation of projections $\Phi(X)$ is not used in the method, only scalar products $\langle\Phi(\mathbf{x}),\ \Phi(\mathbf{x}')\rangle$, $(\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2$, are required. Now these are given by a kernel K, via the relation *kernel trick* :

$$K(\mathbf{x}, \mathbf{x}') = \langle\Phi(\mathbf{x}), \Phi(\mathbf{x}')\rangle.$$

The method therefore requires selecting a kernel (as well as other parameters). Among the possible choices, there are in particular the following kernels



$\frac{2}{\|\mathbf{w}\|}$
$\langle \mathbf{w},\mathbf{x}\rangle + w_0 = -1$
$\mathbf{w},\mathbf{x} \in \mathbb{R}^p, w_0 \in \mathbb{R}$
$y=-1$
$\langle \mathbf{w},\mathbf{x}\rangle + w_0 = 0$
$\langle \mathbf{w},\mathbf{x}\rangle + w_0 = 1$
$y=1$

FIGURE 1 – Margin and separator hyperplane in separable classes (case of a linear kernel)

- Linear Kernel : $K(\mathbf{x},\mathbf{x}') = \langle \mathbf{x},\mathbf{x}'\rangle$ (corresponding to linear SVMs)
- Gaussian Kernel (Gaussian RBF) $K(\mathbf{x},\mathbf{x}') = \exp\left(-\gamma\|\mathbf{x}-\mathbf{x}'\|^2\right)$
- Polynomial Kernel $K(\mathbf{x},\mathbf{x}') = (\alpha + \beta\langle \mathbf{x},\ \mathbf{x}'\rangle)^\delta$ for a $\delta > 0$
- Laplacian Kernel (Laplace RBF) $K(\mathbf{x},\mathbf{x}') = \exp\left(-\gamma\|\mathbf{x}-\mathbf{x}'\|\right)$ for a $\gamma > 0$
- Hyperbolic tangent kernel (Sigmoid) $K(\mathbf{x},\mathbf{x}') = \tanh\left(\alpha + \beta\langle \mathbf{x},\ \mathbf{x}'\rangle\right)$ for a pair $(\alpha,\beta) \in \mathbb{R}^2$

A SVM classifier is of the form

$$\hat{f}_{\mathbf{w},w_0}(\mathbf{x}) = \operatorname{sign}\left(\langle \mathbf{w}, \Phi(\mathbf{x})\rangle + w_0\right), \tag{1}$$

or $\mathbf{w} \in \mathcal{H}$ and $w_0 \in \mathbb{R}$ are parameters adjusted during phase learning from a sample of i.i.d. examples, $\mathcal{D}_n = \{(\mathbf{x}_i, y_i), i = 1, \ldots n\}$. Note : in the case where the funcion Φ is the identity on $\mathbb{R}^p$, we simply find that the decision rule is

$$\hat{f}_{\mathbf{w},w_0}(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{p} w_i x_i + w_0\right).$$

The boundary associated with the decision rule (1) is the set $\{\mathbf{x} : \langle \mathbf{w}, \Phi(\mathbf{x})\rangle + w_0 = 0\}$. It corresponds to a hyperplane in space $\mathcal{H}$, but is much more complex in $\mathcal{X}$ (depending on the shape of the selected kernel).

In $\mathcal{H}$, the hyperplane is obtained by maximizing the margin separating the two classes, which amounts to solving an optimization problem under linear constraints :

$$\begin{cases} (\mathbf{w}^*, w_0^*, \xi^* \in \mathbb{R}^n) \in \underset{\mathbf{w}\in\mathcal{H}, w_0\in\mathbb{R}, \xi\in\mathbb{R}^n}{\arg\min} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i\right) & \\ \text{s.t} \quad \xi_i \geq 0, & \forall i \in \{1,\cdots,n\}, \\ \qquad y_i\left(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + w_0\right) \geq 1 - \xi_i, & \forall i \in \{1,\cdots,n\}. \end{cases} \tag{2}$$

It can be shown that the solution $\mathbf{w}$ can be expressed in the following way :

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \Phi(\mathbf{x}_i).$$

The indices i for which $\alpha_i^* \neq 0$ are those for whom equality is carried out in the constraint, the corresponding points $\mathbf{x}_i$ are called *support vectors* (vecteurs supports en français) of the decision. The coefficients α_i^* denote the solutions of the dual problem which is a quadratic program (QP) under affine constraints :

$$
\begin{cases}
\alpha^* \in \underset{\alpha \in \mathbb{R}^n}{\arg\max} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{1 \le i,\, j \le n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right) \\
\text{s.t} \quad 0 \le \alpha_i \le C, \quad \forall i \in \{1, \cdots, n\}, \\
\qquad \sum_{i=1}^{n} \alpha_i y_i = 0.
\end{cases} \tag{3}
$$

The parameter C controls the complexity of the classifier insofar as it determines the cost of a misclassification : the higher the C value, the more the rule obtained is complex (the number of points for which one wants to minimize the classification error crop). This approach is called C-classification and is used with the object `sklearn.svm.SVC` in the module `scikit-learn`.

Another way to control complexity (*i.e.,* the number of support vectors), called ν-classification, is to consider, in place of the dual problem described above, the following problem :

$$
\begin{cases}
\alpha^* \in \underset{\alpha \in \mathbb{R}^n}{\arg\min} \left(\frac{1}{2} \sum_{1 \le i,\, j \le n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right) \\
\text{s.c} \quad 0 \le \alpha_i \le 1, \quad \forall i \in \{1, \cdots, n\}, \\
\qquad \sum_{i=1}^{n} \alpha_i y_i = 0, \\
\qquad \sum_{i=1}^{n} \alpha_i \ge \nu.
\end{cases} \tag{4}
$$

where $\nu \in [0, 1]$ is a parameter approximating the percentage of support vectors among the training data. This approach is used with the object `sklearn.svm.NuSVC` in the module `scikit-learn`.

Extensions to multi-class case

In case the output variable Y has more than two modalities, there are several ways to extend the methods of the binary case directly.

"One Vs One". In the case where it is desired to predict a label that has $K \ge 3$ classes, we can consider all the pairs of labels (k, l) possible, $1 \le k < l \le K$ (there are $K(K-1)/2$) and learning a classifier $C_{k,l}(X)$ for each of them. The prediction then corresponds to the label that won the most "duels".

"One Vs. All". For each class k, we learn a classifier to discriminate between populations $Y = k$ and $Y \neq k$. From the a posteriori probability estimates, we assign the most probable estimate.

- Implementation -

We will use the object `sklearn.svm.SVC` :

```python
from sklearn.svm import SVC
```

The file `svm_script.py` provides a complete example of classification using the function `SVC`.

Preliminary questions : SVM as a convex relaxation to minimize the empirical classification risk

1) Show that the primal problem solved by the SVM can be rewritten as follows :

$$
\underset{\mathbf{w} \in \mathcal{H}, w_0 \in \mathbb{R}}{\arg\min} \left(\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \left[1 - y_i \left(\langle \mathbf{w},\ \Phi(\mathbf{x}_i) \rangle + w_0\right)\right]_+ \right)
$$

2) Explain the sentence : "an SVM minimizes the classification error using a convex upper bound". The function $x \to [1 - x]_+ = \max(0, 1 - x)$ is called *Hinge* (charnière en français). Explain the difference between the pivotal loss and the loss of binary classification.

Linear SVM

3) Draw a i.i.d. sample from a mixture of two Gaussian distrtibutions : each class is a Gaussian with specific parameters. One could use the function `make_blobs` available in `sklearn.datasets` library. Reserve 75% of the data for training and 25% for the test data.
4) Since the probability distributions are known, numerically estimate the Bayes risk.
5) Draw the decision boundary H induced by SVM as well as the hyperplanes H_1 and H_{-1}. Vary the parameter C to see its impact on the number of support vectors. We can use the example `https://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane.html`.
6) Define the Gaussian classes such that the two distributions overlap. Draw an i.i.d. sample from the joint probability distribution. Apply a 5-fold Cross-Validation to find the optimal parameter $\mathbf{C}^*$ to classify this new dataset using a linear kernel.
7) Show how tuning SVM hyperparameters on training data, for example by taking a Gaussian kernel (the parameters are therefore gamma and C), can lead to overfitting.

Non-Linear SVM

8) Define a new binary nonlinear classification problem : for instance, define one class as a Gaussian surrounded by the other chosen as a circle class, or choose the second class as a mixture of two Gaussian in such way that the separation problem is nonlinear. Generate a non-linearly separable dataset (we could for example use the function `make_blobs` available in `sklearn.datasets`library).
9) Use an SVM with a Gaussian kernel then a polynomial (with well-adapted parameters) then plot the decision boundaries of these algorithms on separate graphs.
10) We wish to compare classification performance between different SVMs based on different kernels (linear, polynomial and Gaussian). Propose a method allowing to compare these three algorithms.

Learning curve

11) Draw the learning curve of the algorithm : with fixed hyper-parameters and fixed test set, calculate the training and test errors by using training sub-sets of training data of various sizes (drawn randomly). Plot the train and test error based on the size of the train set subset. Estimate and display the accuracy of the Bayes predictor on the same graph. Comment.

Error versus complexity

12) Add noise to the dataset by randomly modifying the labels of some training data. Then, draw the complexity curves of the algorithm : with set train and test set, draw the train and test error as a function of the complexity (*i.e.* as a function of the value of the hyper-parameter controlling the complexity, or the number of support vector). Comment.

SVM GUI

13) Start the script `svm_gui.py` available at the link : `http://scikit-learn.org/stable/auto_examples/applications/svm_gui.html`

    This application allows real-time evaluation of the impact the choice of the kernel and the regularization parameter C.
14) Generate a very unbalanced data set with much more points in one class than in the other (at least 90% vs 10%).
15) Using a linear kernel and decreasing the parameter C what do you observe ?

    This phenomenon can be corrected in practice by weighting more errors on the lesser class (parameter `class_weight` de SVC) or by a re-calibration technique (used with `SVC(..., probability=True)`).

To go further : application to face classification

The following example is a face classification problem. The database to be used is available in `sklearn.datasets` library.

By modifying the sample code given in the last part of the file `svm_script.py` :

16) Show the influence of the regularization parameter. For example, the prediction error can be displayed as a function of C on a logarithmic scale between `1e5` and `1e-5`.
17) By adding nuisance variables, thus increasing the number of variables to the number of learning points fixed, show that performance drops.
18) Explain why the features are centered and reduced.
19) What is the effect of choosing a non-linear RBF kernel on prediction ? You will be able to improve the prediction with a reduction of dimension based on the object `sklearn.decomposition.RandomizedPCA`.

Références

[1] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning.* Springer Series in Statistics. Springer, New York, second edition, 2009. `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`. 1

[2] B. Schölkopf and A. J. Smola. *Learning with kernels : Support vector machines, regularization, optimization, and beyond.* MIT press, 2002. 1

[3] Vladimir N Vapnik. *Statistical learning theory.* Wiley, 1998. 1