Support Vector Machine

A Support Vector Machine (abbreviated SVM) is a statistical learning method introduced by Boser, Guyon and Vapnik in the early 90's [**?**], then widely developed by the latter [5] (see [2, chapter 12] and [3, chapter 7] for a detailed description). As a supervised approach, an SVM relies on a learning set $\mathcal{D}_n = \{(\mathbf{x}_i, y_i),\ i \in [\![1, n]\!]\}$ containing n pairs of explanations observations $\mathbf{x}_i$ from a vector space $\mathcal{X} \subset \mathbb{R}^p$ and explaining observations y_i from a discrete (classification) or continuous space (regression) $\mathcal{Y}$.

There are various ways to present SVM, including through the theory of regularization and functional analysis. We will restrict ourselves here to the geometric framework of a binary classifier ($\mathcal{Y} = \{-1, +1\}$)) but this discourse easily extends to regression. We therefore assume that we have a function of redescription of the data $\phi\colon \mathcal{X} \to \mathbb{R}^p$, associating each observation $\mathbf{x}_i$ to a point $\Phi(\mathbf{x}_i)$ in a Euclidean space. A SVM produces an affine estimator f with $f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + w_0$. It is assumed that the data are linearly separable in the redescription space. Geometrically, the f model defines a hyperplane that divides the data space into two parts (figure 1) : the points for which $f(\mathbf{x}) \geq 0$ and those for which $f(\mathbf{x}) < 0$.

The model f is estimated by solving the optimization problem (called primal) :

$$\underset{\mathbf{w}\in\mathbb{R}^p, w_0\in\mathbb{R}}{\arg\min} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \ell(y_i, \langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + w_0). \tag{1}$$

In this formulation, $\ell\colon \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_+$ is a loss function that measures the model's attachment to the data, weighted by the positive scalar C (sometimes called « parameter cost »). The most common loss is the hinge function (*hinge*) : $\ell(y, \tilde{y}) = \max(0, 1 - y\tilde{y})$. Regularization in the 2-norm of $\mathbf{w}$, meanwhile, aims - among other things - at preventing over-fitting. Geometrically, this regularization is related to the inverse of the margin $\frac{1}{2}\|\mathbf{w}\|$ (figure 1) ; thus, an SVM determines a hyperplane maximizing the margin between the two classes to be separated (since one minimizes $\frac{1}{2}\|\mathbf{w}\|^2$).

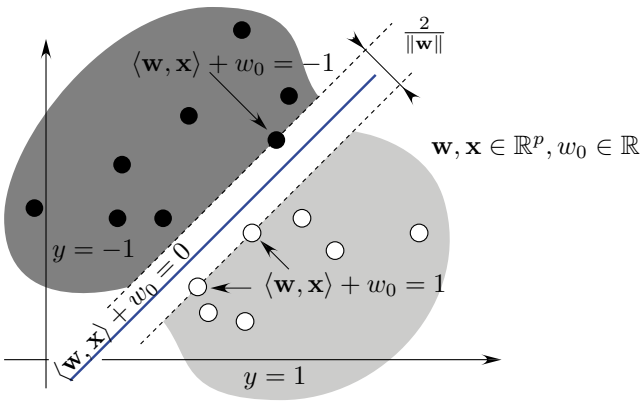

Figure 1 – Margin and hyperplane separator for separable classes (here, Φ is the identity).

It is possible to show that at the optimality of the learning problem, $\mathbf{w}$ is written $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \Phi(\mathbf{x}_i)$, where $\boldsymbol{\alpha}$ is solution of the problem of optimization (said dual) :

$$\underset{\substack{\boldsymbol{\alpha}\in\mathbb{R}^n, 0\leq\alpha_i\leq C \\ \sum_{i=1}^n y_i\alpha_i = 0}}{\arg\min} \frac{1}{2} \sum_{1\leq i,j\leq n} \alpha_i\alpha_j y_i y_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle - \sum_{i=1}^{n} \alpha_i.$$

Learning points $\mathbf{x}_i$ for which α_i is not null are called « support vectors ». In practice, they alone define the function f.

For a new arriving point $\mathbf{x}$, the decision function is written : $\text{sign}(f(\mathbf{x})) = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle + w_0) \in \mathcal{Y}$. Note that the evaluation as well as the learning of the f model only involves the scalar product of the data in the redescription space and not the Φ redescription function itself. We can then replace Φ with a *kernel* $k \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (*i.e.,* a function with the properties of a scalar product) : $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle = k(\mathbf{x}, \mathbf{x}')$. This change is commonly referred to as the « kernel trick » (*kernel trick*). It makes it possible to work in complex and large (even infinite) redescription spaces without calculating data images by the Φ redescription function, or even knowing explicitly Φ. Some examples of kernels are given below :

- linear : $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$ (in this case, Φ is the identity) ;
- gaussian (Gaussian RBF) : $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right)$;
- polynomial : $k(\mathbf{x}, \mathbf{x}') = (\alpha + \beta \langle \mathbf{x},\ \mathbf{x}' \rangle)^{\delta}$ pour un $\delta > 0$;
- laplacian (Laplace RBF) : $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|\right)$ for $\gamma > 0$.

For a linear kernel, we speak of a linear SVM (f is linear with respect to $\mathbf{x}$) ; if not of a non-linear SVM (f is non-linear with respect to $\mathbf{x}$). In the latter case, the boundary between classes in the $\mathcal{X}$ data space is no longer a hyperplane but a surface whose shape depends on the chosen kernel.

Note

The approach presented so far is called C -classification in `scikit-learn` and is accessible through the `sklearn.svm.SVC` object. A different point of view, named ν -classification (`sklearn.svm.NuSVC`) considers the following optimization problem :

$$\underset{\mathbf{w} \in \mathbb{R}^p, w_0 \in \mathbb{R}, \rho \in \mathbb{R}_+}{\arg\min} \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^{n} \tilde{\ell}(\rho, y_i, \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + w_0) - \nu\rho.$$

In this formulation, $\tilde{\ell} \colon \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_+$ is a loss function similar to the hinge function but in which the *margin of safety* can vary : $\tilde{\ell}(\rho, y, \tilde{y}) = \max(0, \rho - y\tilde{y})$. Note that $\ell(y, \tilde{y}) = \tilde{\ell}(1, y, \tilde{y})$.

The corresponding dual problem is then :

$$\underset{\substack{\boldsymbol{\alpha} \in \mathbb{R}^n, 0 \le \alpha_i \le 1 \\ \sum_{i=1}^{n} y_i \alpha_i = 0}}{\arg\min} \frac{1}{2} \sum_{1 \le i,j \le n} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j), \quad \text{constrained by} \sum_{i=1}^{n} \alpha_i \ge \nu.$$

Here, $\nu \in [0, 1]$ is a parameter approaching the ration between the number of support vectors and training data. C and ν -classification are two equivalent approaches.

Multi-class case extensions

There are several ways to extend a binary classifier to several classes, including :

one-vs-one : build binary classifiers on all possible class peers. The final decision is given by the class predicted the most times.

one-vs-all : build binary classifiers for each class (all the others are grouped under the same class labeled -1). The final decision is given by the predicted class with the largest decision $f(\mathbf{x})$.

Crammer and Singer : change the loss function to consider multiple classes [**?**].

Regression

In addition to the classification, the SVMs have also been adapted to the regression ($\mathcal{Y} = \mathbb{R}$, that is, the y_i labels are real). For a fixed $\epsilon \ge 0$ threshold, the estimate of the (f, b) pair then corresponds to the resolution of the optimization problem 1, where $\ell(y, \tilde{y}) = \max(0, |y - \tilde{y}| - \epsilon)$. This technique is very close to the regularized regression but replaces the quadratic loss with a linear loss thresholded at ϵ (that is, when the $\tilde{y}$ prediction is close enough to y, the estimator is not penalized).

The corresponding dual problem is then :

$$\underset{\substack{\boldsymbol{\alpha}\in\mathbb{R}^n,\boldsymbol{\alpha}'\in\mathbb{R}^n,\\ 0\leq\alpha_i\leq C,0\leq\alpha'_i\leq C}}{\arg\min} \frac{1}{2}\sum_{1\leq i,j\leq n}(\alpha_i-\alpha'_i)(\alpha_j-\alpha'_j)k(\mathbf{x}_i,\mathbf{x}_j)+\epsilon\sum_{i=1}^{n}(\alpha_i+\alpha'_i)-\sum_{i=1}^{n}y_i(\alpha_i-\alpha'_i)$$

$$\text{constrained by}\sum_{i=1}^{n}(\alpha_i-\alpha'_i)=0.$$

Questions

Intuitive approach

1. Run the `svm_gui.py` script. It allows to evaluate in real time the impact of the choice of the kernel and the C regularization parameter.Perform some tests (linearly separable or not, unimodal or not, different kernels and parameters) and comment on the observations. In particular, for unimodal data (separable and then with class overlap), how does the classifier (boundary and margin) react according to the choice of C first (linear kernel) and then γ (kernel parameter Gaussian) ? Is the choice of these parameters crucial to obtain good recognition rates ?
2. Estimate a linear classifier on unimodal data with class overlap and then add points one by one. What are the three areas of interest and how does the classifier react when adding a point to one of these areas ? What about the dual variable α_i associated with each point $\mathbf{x}_i$ according to the zones.
3. Generate a very unbalanced dataset (many more points in one class than in the other). With a linear kernel, gradually decrease the value of C. Comment. This phenomenon can be corrected by weighting the attachment more closely to the data on the least present class (parameter `class_weight` of `sklearn.svm.SVC`).

Classification

4. From the documentation : http://scikit-learn.org/stable/modules/svm.html, write a script estimating a SVM on the 1 and 2 classes of the IRIS data set. You will only use the first two variables and a linear kernel. Display the average score and the decision boundary (use the `plot_2d` and `frontier` functions of the `utils.py` file).
5. Leaving half of the data aside, evaluate the performance in generalization of the model. To do this, you will determine C by cross validation in 5 steps (`scores = cross_val_score (clf, X, y, cv = 5)`). Compare the result with a polynomial SVM.
6. Using the example http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html (but not using `MidpointNormalize`), view a Gaussian kernel performance map applied to the learning data (we will use only 13 values in base 2 for C and γ, respectively between $(2^0, 2^{12})$ and $(2^{-8}, 2^4)$).
7. In the following, we are interested in a problem of classification of faces. From the `svm_lfw.py` script, show the influence of the regularization parameter. For example, we can display the prediction error (in test) as a function of C on a logarithmic scale between `1e-6` and `1e3` .
8. The script you use centers and normalizes the data. Describe how and explain the value of this operation.
9. Keeping the data previously created, and for different sizes of the learning set, evaluate the performance in generalization of a linear SVM (to do this, choose C by cross validation and save the score under test). Draw the *learning curve* (*i.e.* Score according to the size of the learning set).The latter is an empirical approach to the consistency of our estimator.
10. The example http://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane.html#example-svm-plot-separating-hyperplane-py explains how to access the estimated parameters when learning : vector of coefficients $\mathbf{w}$ in the attribute `coef_`, w_0 stored in the attribute `intercept` , list of media vectors, coefficients of the dual problem). From this example, write a script that calculates the values of primal and dual functionals. Check that these values are close (be careful, the labels must be -1 or 1). How does the difference between the two values vary when we vary the tolerance on the optimization (parameter `tol` of `SVC`) ?

Regression

11. We are now interested in predicting the activity of a molecule. To do this, we consider a molecule as a labeled graph, represented by a set of relations between its nodes (the atoms of the molecule). By following these relations within a x molecule, we run through a *path* p in the corresponding graph. Let $\mathcal{P}_d$ be the set of possible paths (of length less than d) for the family of graphs considered. We write $\imath(x,p)$ the indicator which is worth 1 if the path p is present in the graph x, and 0 otherwise. For two molecules x and x', we define the measure of similarity : $u(x,x') = \sum_{p\in\mathcal{P}_d} \imath(x,p)\imath(x',p)$. We then call *Tanimoto kernel* the following function :

$$k(x,x') = \frac{u(x,x')}{u(x,x) + u(x',x') - u(x,x')}.$$

From the file `drug_activity.py` [1],give the best possible prediction score on the test dataset using a ϵ support vector regression. Study the role of ϵ. Try instead of ϵ-SVR its variant ν-SVR where the *epsilon* value is integrated into the learning problem and is learned for you. Compare the two approaches.

12. Compare with Kernel Ridge Regression
13. Compare with at least two methods that need a feature vector presentation : k-NN, trees, random forests,... after having transformed the Gram matrix using KernelPCA[4] to get a feature vector representation.

- Useful links -

⋆⋆⋆ 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

Références

[1] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92 : Proceedings of the Fifth Annual Workshop on Computational Learning Theory. New York, NY, USA : ACM Press, pp. 144–152.*

[2] 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

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

[4] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Advances in kernel methods. chapter Kernel Principal Component Analysis, pages 327–352. MIT Press, Cambridge, MA, USA, 1999. 4

[5] Vladimir N. Vapnik. *Statistical Learning Theory.* Wiley. 1

1. Data provided by Markus Heinonen from Aalto University.