Kernel Methods and SVMs

A convex optimization point of view

Gael Lederrey Corentin Tallec

June 10, 2016

Abstract

Kernel methods, and more specifically support vectors machines, are well known, well theorized and efficient machine learning tools. For years, they were presenting state of the art performances in many machine learning fields, before being overtaken by deep learning methods. They have the enormous advantage of being well understood theoretically, and most interestingly to belong to the class of convex optimization problems, which is clearly not the case of the latter.

In this document, we aim at presenting the general framework of Kernel methods, to expose how they relate to convex optimization, and how their optimization can be undertaken in practice. Besides, we intend to give a more precise treatment of the support vectors machines special case.

1 Kernel methods[1]

1.1 Generalities

Kernel methods are tools broadly used in various fields, but most notably in machine learning and pattern discrimination. Among this class of methods, support vector machines are probably the best known kernel method. This class of algorithm is used to find and study some types of relations in datasets, *e.g.* classifications of the data, correlations between them, etc. Many machine learning algorithms rely on the idea of embedding the data at hand into a vector space that makes it easier analyse. Kernel methods provide a quite general framework for doing so, and provide tools to transport data into possibly infinite dimensional spaces. Kernel methods rely on a measure of similarity between input data known as a **kernel**, which has to be selected beforehand.

A typical case of application for a kernel method is the following. Define $\mathcal{X}$ (resp. $\mathcal{Y}$) as a set of datapoints (resp. a set of labels). Given a training set, that is a set of couples $(x_i, y_i)_{i\leq n} \in (\mathcal{X} \times \mathcal{Y})^n$, we want to find an $f : \mathcal{X} \mapsto \mathcal{Y}$ in a particular set of functions $\mathcal{F}$ such that f predicts accurately labels in the training set, and is able to generalize to unseen datapoints. One way of doing so is to solve the following optimization problem:

$$\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda\Omega(f) \tag{1}$$

where L is a cost function, that measure how close $f(x_i)$ is to y_i for each i, λ is a positive real number and Ω is a regularizing term, that aims at describing the complexity of a

[1] The different elements presented in this section comes from the course on Machine Learning [1] as well as the paper explaining how convex optimization is used with Kernel methods[2].

certain function f. In the general case, this optimization problem is not required to be convex. Kernel methods provide a set of function $\mathcal{F}$ and a regularizing term $\Omega(f)$ such that when the cost function is assumed to be convex in its second argument, the entire problem is convex, and can be easily solved.

In the following sections, kernel functions are defined, as well as their relation to the idea of feature mapping, and the general principle of kernel methods is exposed.

1.1.1 Positive Definite Kernel

Denote by $\mathcal{X}$ an arbitrary input set (notably $\mathcal{X}$ is not required to be a vector space). A kernel can be seen as a measure of similarity between two elements of $\mathcal{X}$. Formally

Definition 1 *A positive definite (p.d.) kernel on $\mathcal{X}$ is a function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ that is symmetric and that satisfies for all $N \in \mathbb{N}$, $(x_1, \ldots, x_n) \in \mathcal{X}^N$, $(a_1, \ldots, a_N) \in \mathbb{R}^{\mathbb{N}}$:*

$$\sum_{1 \leq i,j \leq N} a_i a_j K(x_i, x_j) \geq 0. \tag{2}$$

If $\mathcal{X} = R^d$, the simplest kernel that can be thought of is the canonical inner product. It obviously is symmetric, and positivity is easily verified. Under the same assumptions, it can quite easily be shown that $K(x, x') = (\langle x, x' \rangle_{\mathbb{R}^d})^p$ is a kernel too (known as the polynomial kernel). Another well known kernel is the gaussian kernel, defined as $K(x, x') = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$.

1.1.2 Feature mapping

As seen in Subsection 1.1.1, the canonical inner product of a vector space is a kernel. Mercer's theorem provides a kind of reciprocal statement: any kernel can be viewed as an inner product in a certain hilbert space which is a functional space on $\mathcal{X}$.

Theorem 1 (Mercer's theorem) *The function K is a positive definite kernel on $\mathcal{X}$ if and only if there exists a Hilbert space $\mathcal{H}$ with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and a mapping*

$$\phi : \mathcal{X} \mapsto \mathcal{H} \tag{3}$$

such that, for any x, x' in $\mathcal{X}$:

$$K(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}} \tag{4}$$

Using a kernel can thus somehow be viewed as embedding the datapoints in a (larger) space. One benefit of doing so can be, for example, to simplify regression or classification problems, as these problems tend to be easier in high dimensional space than in low dimensional ones.

1.1.3 Kernel methods as convex optimization problems

Recall the kind of problem exposed in Sec 1.1, by eq 1. Now, once a kernel K is defined, it implicitly defines a functional space $\mathcal{H}$ on $\mathcal{X}$, as explained in Sec 1.1.2. This space can be used as the prediction function space, that is $\mathcal{F} = \mathcal{H}$, and the regularization term can be replaced by the squared norm in $\mathcal{H}$.

This gives the following minimization problem:

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2 . \tag{5}$$

When L is a convex function over its second argument, it is quite clear that the resulting minimization problem is a convex optimization problem: the squared norm is a convex function of f, $f \mapsto f(x)$ is a linear function for any $x \in \mathcal{X}$, thus $L(y_i, f(x_i))$ is convex for all i, which lead to the global convexity. However, the space $\mathcal{H}$ considered is, in the general case, infinite, and this makes convex optimization unapplicable.

1.2 From infinite to finite

Happily enough, the representer theorem stated below turns this infinite dimensional convex optimization problem into your gentle everyday n-dimensional convex optimization problem:

Theorem 2 (Representer theorem) *Using the notations previously defined, let $\Psi : \mathbb{R}^{n+1} \mapsto \mathbb{R}$ be a function strictly increasing in the last variable. Then any solution of*

$$\min_{f \in \mathcal{H}} \Psi(f(x_1), \dots, f(x_n), \|f\|_{\mathcal{H}}) \tag{6}$$

admits a representation of the form:

$$\forall x \in \mathcal{X}, f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x). \tag{7}$$

The equation 5 clearly falls in the field of application of the representer theorem. The optimization problem can thus be rewritten in term of α's as:

$$\min_{(\alpha_1,\dots,\alpha_n)\in\mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} L(y_i, \sum_{j=1}^{n} \alpha_j K(x_j, x_i)) + \lambda \sum_{1 \le i,j \le n} \alpha_i \alpha_j K(x_i, x_j) \tag{8}$$

which is a finite dimensional convex optimization problem.

2 Support vector machines: a special case[2]

Support vector machines are kernel methods used in the field of binary classification, and that can be extended (quite painfully, unfortunately) to n-ary classification. Among kernel methods, they are probably the best known. This might be explained by the fact that they enforce sparsity in the best fit representation. Formally, they ensure that in eq 7, a varying number of α's will be exactly zero. Sparsity often brings along a lot of nice properties. For SVMs, most notably, they ensure a better regularization (intuitively, a fit with less term is less complex than a fit with many terms, and the number of term is directly related to the number of non-zero α's), and they induce smaller computational cost. Indeed, the best fit represented in eq 7 has as many terms as there are datapoints in the train dataset. This means that the computational cost at test time will depend directly on the number of datapoints in the training set. This might prove computationally costly. SVMs partly solve this problem by ensuring that some (most) α's are zero, and thus that the best fit will only depend on a few training datapoints.

[2] The different elements presented in this section comes from the course on Machine Learning [1] as well as the article on Wikipédia on SVM[3].

2.1 Generalities

Given an input set $\mathcal{X}$, a p.s.d kernel on $\mathcal{X}$, K and a training dataset, $((x_1, y_1), \dots, (x_n, y_n)) \in (\mathcal{X} \times \{\pm 1\})^n$, SVMs solve the kernel optimization problem eq 5 with the particular choice of loss $L(x, y) = \max(0, 1 - x^\top y)$, which is known as the Hinge Loss.
A few remarks might be of use at this point:

- The Hinge Loss is clearly convex in x. It is a maximum over two affine functions of x, and is thus convex.
- Datapoints are classified simply by looking at the sign of $f(x)$ where f is the fit. If $f(x) > 0$, x is classified as $+1$, else x is classified ad -1. The specificity of the Hinge Loss is to try and drive $f(x)$ towards $+1$ when x is in the $+1$ class, and toward -1 when x is in the -1 class, but not to do any further effort.

Rewritting eq 5 in the SVM case, and introducing additionnal slack variables, the optimization problem we intend to solve takes the form:

$$\min_{f \in \mathcal{H}, \xi \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda \|f\|_{\mathcal{H}}^2 \tag{9}$$

subject to:

$$\begin{cases} \xi_i \geq 1 - y_i f(x_i) \\ \xi_i \geq 0 \end{cases} \tag{10}$$

Using the representer theorem, this can be transformed in

$$\min_{\alpha \in \mathbb{R}^n, \xi \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda \alpha^\top \mathbf{K} \alpha \tag{11}$$

subject to:

$$\begin{cases} y_i \sum\limits_{j=1}^{n} \alpha_j K(x_i, x_j) + \xi_i - 1 \geq 0 \\ \xi_i \geq 0 \end{cases} \tag{12}$$

where $\mathbf{K}$ is the p.s.d. matrix associated to the kernel K on the dataset, that is $\mathbf{K}_{i,j} = K(x_i, x_j)$. This is a quadratic program.
The dual problem can easily be expressed:

$$\max_{0 \leq \mu \leq 1/n} \sum_{i=1}^{n} \mu_i - \frac{1}{4\lambda} \sum_{1 \leq i,j \leq n} y_i y_j \mu_i \mu_j K(x_i, x_j). \tag{13}$$

Now expressing the relation between primal and dual solutions:

$$\alpha = \operatorname{diag}(y) \mu / 2\lambda \tag{14}$$

(this is done by rewriting the optimization condition on the Lagrangian with respect to the α variables) and injecting this relation into the dual, we get the following easier problem:

$$2\alpha^\top y - \alpha^\top \mathbf{K} \alpha \tag{15}$$

subject to:

$$0 \leq y_i \alpha_i \leq \frac{1}{2\lambda n} \tag{16}$$

(The crucial point in deriving this new optimization problem is to notice that $y_i^2 = 1$, since $y_i \in \{\pm 1\}$).
On this very simple optimization problem, the KKT conditions can be easily expressed, and they lead to the following complementary slackness equations:

$$\begin{cases} \alpha_i \left[y_i f(x_i) + \xi_i - 1 \right] = 0 \\ \left[\alpha_i - \frac{y_i}{2\lambda n} \right] \xi_i = 0. \end{cases} \tag{17}$$

With a bit of analysis with these equations and the constraints of the primal, it can be easily shown that the only terms for which $\alpha_i \neq 0$ are the examples "hard to classify" that is for which $y_i f(x_i) \leq 1$ (those examples can be well classified, but their classifiction margin is not large). All in all, this tends to express the fact that only a few datapoints are involved in the best fit expression. This has a tremendous impact on further computations involving the best fit.

2.2 Use case

Let us now dwelve in a commonly presented use case. We decided to use the SVM to perform handwritten digits recognition handwritten digits. The *sklearn*[4] machine learning library was used, to train SVMs on the MNIST database of handwritten digits[5] retrieved from Yann Lecun's website. Each of the images (a handwritten digit between 0 and 9) has a given label. The goal is to train a SVM to recognize these digits. Some examples of the MNIST dataset are given in Figure 1.

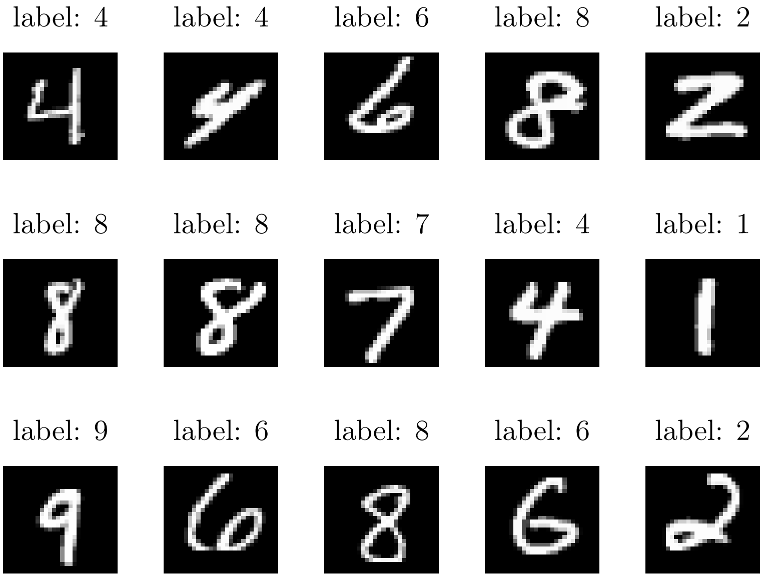

Figure 1: Some examples of the MNIST train set with their label.

The purpose of this script was to train SVMs on 60000 images with their labels. And then, test the model on 10000 images. We could finally compare the predicted labels against the real labels. In order to test the SVM method, we trained the algorithm with different kernel functions. As SVMs naturally perform binary classification, and not n-ary classification, one-vs-all classification was used (that is we train the model n-times, once on each digit).

- Linear kernel: $K(x_i, x_j) = \langle x_i, x_j \rangle$
- Polynomial kernel: $K(x_i, x_j) = (\gamma \langle x_i, x_j \rangle + r)^p$, γ is a parameter that needs to be found and we decided to keep $r = 0$ and we used $p = 4$.
- RBF kernel: $K(x_i, x_j) = \exp(-\gamma |x_i - x_j|^2)$, γ is a parameter that needs to be found.

The most important result is the correctness of the prediction. To do this, we take the number of correct prediction and we divide it by the length of the dataset. This gives us a percentage on the correctness of the model. We give in Table 1 the results for each of the kernels.

Kernels	Correct prediction
Linear	94.62%
Polynomial (degree 4)	97.31%
RBF	98.56 %

Table 1: Percentage of correct prediction for each of the kernels.

As we can see, the best kernel is the RBF one. It has already been shown in the literature that this kernel is the state-of-the-art in SVM. Another important aspect is to have a look at the predictions more precisely. For this, we use a confusion matrix. This matrix shows the labels (vertical axis) in function of the predictions (horizontal axis). It is interesting to have a look at this matrix to see if there is some bias. (For example, 1 can be often predicted as 7). The confusion matrix for the RBF kernel is given in Table 2. As we can see in this table, we don't have any bias. We just show some wrong predictions in Figure 2. This figure shows us that some of the images are difficult to label, even for us.

		Predictions									
		0	1	2	3	4	5	6	7	8	9
	0	974	0	1	0	0	2	0	1	2	0
	1	0	1128	3	1	0	1	0	1	1	0
	2	4	0	1017	0	1	0	0	7	3	0
	3	0	0	2	997	1	2	0	4	3	1
Labels	4	0	0	2	0	968	0	4	0	1	7
	5	2	0	0	5	1	877	3	1	2	1
	6	3	2	0	0	2	2	948	0	1	0
	7	0	3	8	1	1	0	0	1007	1	7
	8	3	0	1	3	1	1	0	2	959	4
	9	2	3	1	6	6	2	1	5	1	982

Table 2: Confusion Matrix for the RBF kernel.

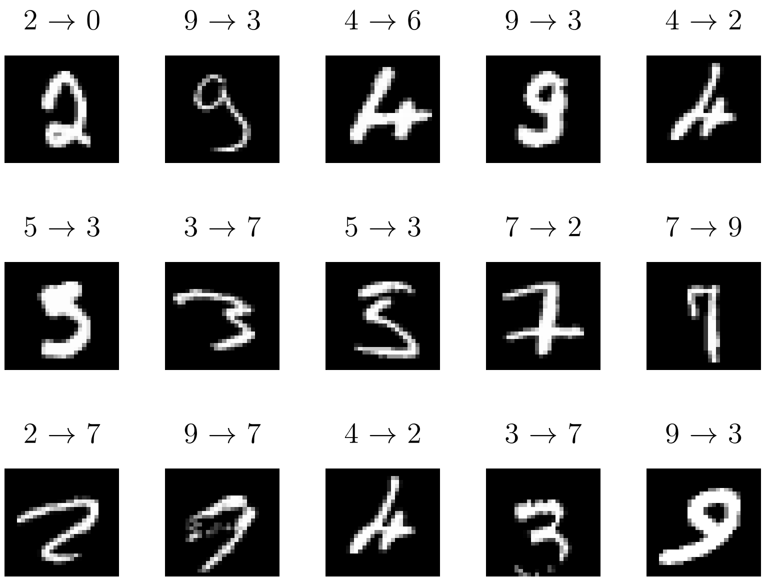

Figure 2: Examples of images that have not been labelled correctly. This number above the pictures are like this: label → prediction.

3 Conclusion

Kernel methods remain methods of interest, most notably in machine learning, since they provide both a satisfactory theoretical background, as well as some good results in a large variety of fields. One thing that might be worth noticing is that they have a quadratic complexity in the number of data-entries in the train dataset (we need to be able to compute the whole kernel, which is of dimension $n \times n$ where n is the number of train datapoints), making them hardly applicable for very large datasets.
This might partly explain the recent arising of deep learning methods, which are much less theoretically grounded, and among other ill-understood properties, perform well while not achieving global optimality, but keep a linear complexity in the number of datapoints, both at train and at test time (even if it might argued that the train time probably doesn't scale linearly in the number of training samples).
Kernel methods remain useful baselines when considering small datasets, as they remain computationnally tractable, and furthermore provide a natural regularization, that tends to reduce efficiently overfitting. Besides, neural networks tend to perform quite poorly in those cases where only few datapoints are available as they tend to overfit rapidly, even though recent researches tend to bridge the gap.

References

[1] Slides of the course on "*Kernel Methods for Machine Learning*", Inria, Grenoble, France

[2] Kim, Seung-Jean, et al. "Learning the kernel via convex optimization." *Acoustics, Speech and Signal Processing, ICASSP.* IEEE, 2008.

[3] Wikipédia - Support Vector Machine

[4] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[5] http://yann.lecun.com/exdb/mnist/