Category Similarity as a Predictor for SVM Learning Performance

Fabio Pardi
Dipartimento di Scienze dell'Informazione, Università La Sapienza
Via Salaria, 113 - 00198 Roma - Italy
fabio.pardi@tiscalinet.it

Abstract

In this paper we propose a method for the prediction of learning performance in Support Vector Machines based on a novel definition of intra- and inter-class similarity. Our measure of category similarity can be easily estimated from the learning data. In the second part of the paper we provide experimental evidence to support the effectiveness of this measure.

Introduction

In (supervised) learning tasks several design choices must be made in order to build a learning system. Among these, it is widely agreed that certain learning parameters affecting the representation of the hypotheses and the instance space, have a crucial impact on the algorithm performance, perhaps even more than the algorithm itself. Determining these parameters is usually based on heuristics. Alternatively, one can use a validation set to estimate the generalization error of the classifier (i.e. the expected error on unseen cases) for a specific set of choices. Automatically changing a parameter and evaluating the generalization performance may eventually lead to the optimal choice (or at least to a good one). Though this approach is quite common (for instance in feature selection) it is both time consuming and resource consuming, since it requires additional manual training for validation.

Therefore, the importance of a numeric estimator to predict learning performance is twofold: from a *theoretical* standpoint, it sheds light on the learning process by showing the factors that make it possible, while from a *practical* standpoint, it can be used for the construction of an optimal classifier.

In the following, we describe a method for constructing a matrix that might be a good source of information for predicting the generalization performance of a specific, but very popular, kind of classifiers: Support Vector Machines (SVMs). SVMs are relatively young learning systems (Cortes and Vapnik 1995) that have shown in many problems better generalization performance than competing methods (Burges 1998). One of their main features is that their performance depends on a single design choice: the choice of a kernel function. Roughly speaking, a kernel is a formal equivalent for the informal idea of similarity. We use this notion of similarity to define a measure of similarity between the various categories[1] the classifier must learn. The intuition behind this method is that high intra-class and low inter-class similarity positively affect the learning process. By verifying this intuition through a series of experiments, we shall provide a method to choose a 'good' kernel in a data dependent way.

Support Vector Machines

In the two-class case, SVMs output functions of the form:

$$h(x) = sign\left(\sum_{i=1}^{m} y_i \alpha_i^{opt} K(x_i, x) - \theta^{opt}\right) =$$

$$= sign\left(\sum_{x_i \in SV} y_i \alpha_i^{opt} K(x_i, x) - \theta^{opt}\right), \text{ where:}$$

$\{(x_i, y_i)\}_{i=1}^{m}$ is the training set ($x_i \in X$ and $y_i \in \{-1,+1\}$), $\alpha_1^{opt},...,\alpha_m^{opt},\theta^{opt}$ are the solutions of an optimization problem (briefly discussed later), SV is the set of training instances of non-zero weight, i.e. the set of x_i s.t. $\alpha_i^{opt} \neq 0$ (called *support vectors*), and $K : X \times X \to \Re$ is a function, called *kernel*, such that, for all $x, y \in X$

$$K(x, y) = < \phi(x), \phi(y) >$$

where: $\phi : X \to F$ is a mapping to a (possibly infinite dimensional) Euclidean space F, called *feature space*[2].

The choice of the kernel function, together with the definition of the optimization problem, completely defines a specific SVM. In most cases one verifies that a chosen function is a kernel through Mercer's condition (Vapnik, 1995) that guarantees the existence of the mapping ϕ (but it does not tell how to construct ϕ or even what F is).

SVMs look for an 'optimal' hyperplane that separates the data in the feature space, where different ways of defining 'optimal' give rise to different SVMs. The simplest approach is that of looking for the separating hyperplane h that maximises the margin with the training points, i.e. the minimum among the distances between h and the varous $\phi(x_i)$. It turns out that it is possible to express the search for such a hyperplane with an optimization problem whose variables are the weights $\__1,\ldots,\__m$ to assign to the training points. In this optimization problem the various $\phi(x_i)$ never appear alone, but always

[1] We shall use terms 'category' and 'class' interchangeably.
[2] $< x, y >$ denotes the inner product between x and y.

inside an inner product $<\phi(x_i),\phi(x_j)>$ that can be computed through *K*. It follows that we can define a kernel without worrying about the mapping ϕ it implicitly defines. This is very important, since usually the feature space is high-dimensional and working directly with $\phi(x)$ would be a problem.

Another advantage of SVMs is that the optimization problem has a unique global solution and it is not subject to the problems deriving from the existence of several local minima.

In the multi-class case a hyperplane is learned for each category and every new instance *x* is assigned to the class whose hyperplane is furthest from $\phi(x)$. This means that the feature space is split into *N* convex regions.

A Similarity Based Performance Predictor

Following what we said in the preceding section, feature space decision surfaces are represented by linear functions. Therefore, we expect that the more categories are linearly separable[3], the easier will be for SVMs to learn such categories. Hence we are interested in finding an efficiently computable measure of such linear separability.

It is clear that a measure of linear separability between classes in the feature space must take into account the positions $\phi(x)$ of the learning points. Yet, as we saw before, the definition of a kernel does not necessarily imply the possibility to construct ϕ. It follows that the distribution of the examples in the feature space must be investigated only through the values of *K* on the various example pairs.

Let us consider those kernels such that $K(x,x) = 1$ for every instance *x*. This restriction is equivalent to considering all kernels such that $K(x,x)$ equals a constant *k*, since every kernel of this kind can be transformed in an equivalent kernel $K' = K / k$, such that $K'(x,x) = 1$ for all *x*. We call these kernels *Hypersphere kernels*, since instances are mapped on the surface of a sphere of radius 1: $\|\phi(x)\| = \sqrt{<\phi(x),\phi(x)>} = \sqrt{K(x,x)} = 1$. Notice that $K(x,y)$ equals the cosine of the angle _ between vectors $\phi(x)$ e $\phi(y)$:

$$\cos\theta = \frac{<\phi(x),\phi(y)>}{\|\phi(x)\|\cdot\|\varphi(y)\|} = K(x,y)$$

Besides, limiting our analysis to Hypersphere kernels is not an excessive restriction, since they include some of the most successful kernels, Gaussian kernels [4]. Finally every kernel *K* can give rise to an *induced* Hypersphere kernel H_K, in this way: $H_K(x,y) \stackrel{def}{=} \frac{K(x,y)}{\sqrt{K(x,x)}\sqrt{K(y,y)}}$ (unless either $K(x,x)$ or $K(y,y)$ equals 0, and in that case $H_K(x,y) \stackrel{def}{=} 0$).

We are now ready to state the thesis of this paper:

> We claim that, for Hypersphere kernels, the matrix $Avg(K)$ defined by:
>
> $$Avg(K)_{i,j} = \mathbf{E}[K(x,y) \mid x \in C_i, y \in C_j]$$
>
> is a good predictor for the performance of a SVM that has *K* as kernel.
>
> Here, by $\mathbf{E}[K(x,y) \mid x \in C_i, y \in C_j]$ we mean the expectation of *K* evaluated on *x* and *y* taken randomly and independently, according to the class-conditional probability distributions (Duda and Hart 1973) respectively of classes C_i and C_j.

This claim has an intuitive justification. $Avg(K)_{i,j}$ is the expected value of the cosine of the angle between an *i*-th class element and one of the *j*-th class. Therefore it is related to the 'confusion' between classes C_i and C_j, if $i \neq j$, or to the 'learnability' of class C_i, if $i=j$.

For example, if $Avg(K) = \begin{pmatrix} 1 & 1-\varepsilon \\ 1-\varepsilon & 1 \end{pmatrix}$, where $0< \varepsilon <2$, since the elements on the diagonal equal 1, the internal cohesion of the two classes is maximal, i.e. the images of the instances in the feature space are concentrated in two points. Note that $Avg(K)_{1,2} \neq 1$ implies that these points are distinct. No matter how close the two classes are (i.e. how close to 1 $Avg(K)_{1,2}$ is), the linear separability between them is guaranteed: the generalization error will be 0 whenever an example for each category will be collected. Unfortunately this is an extreme case, since $Avg(K)$ is seldom so informative.

In the following we provide experimental evidence that shows the existence of a connection between $Avg(K)$ and the error of a SVM that has *K* as kernel.

Notice that $Avg(K)$ only depends on the (unknown but fixed) probability distribution on the set of input/output couples $(x,y) \in X \times \{C_1, ..., C_N\}$. It follows that $Avg(K)$ cannot be 'computed' (since the distribution is always unknown) but only 'estimated' using the training data. We could have defined $Avg(K)$ directly from these data, but we think that it is easier to consider $Avg(K)$ as an intrinsic quality of distributions, independent from the result of any sampling of data (x, y).

In general the estimation of $Avg(K)$ requires computing $K(x,y)$ for every pair of examples in the learning set, i.e. a number of steps proportional to m^2. This is however a highly parallelizable computation: all the $K(x,y)$ can be

[3] One might object that training data either are linearly separable or they are not, without anything in the middle. Here, we implicitly refer to a quantity, that could be formally defined, related to the difficulty to find surfaces that discriminate between the distibutions (in a probabilistic sense) of the various classes in the feature space.

[4] We define Gaussian kernels later.

computed simultaneously in time O(1), while the estimation of $Avg(K)$ requires parallel time[5] O(log m).

Experimental Evidence for Gaussian Kernels

Let us now discuss experimental results on two different implementations of a SVM that uses a Gaussian kernel:

$$K(x, y) = \exp\frac{-\|x - y\|^2}{2\sigma^2}$$

This class of kernels assumes that a notion of distance $\_x - y\_$ is defined in X (usually $X = \Re^f$). Besides, it is parametric in σ. For larger σ, a higher number of training examples will influence the classification of a new instance. For smaller σ, the classification will only depend on the examples in its neighbourhood. When σ is small there is a risk of overfitting, when σ is large the risk is to give the wrong importance to the various training examples. It is therefore important to find the best compromise for the value of σ:

It would be useful to define a parameter, in function of training data and kernel, such that, varying σ, its value approximately follows the generalization error of the SVM, and, even more important, in such a way that its minimum is approximately where the error has its minimum. In this way we would be able to identify the value of σ that minimizes this parameter, as we show later, in a much more efficient way than looking for the value of σ that minimizes the error on a validation set.

Let us start with some definitions:

$W(K) = \mathbf{E}[K(x,y) \mid x \text{ and } y \text{ belong to the same class}]$

$B(K) = \mathbf{E}[K(x,y) \mid x \text{ and } y \text{ belong to different classes}]$

Both $W(K)$ and $B(K)$[6] can be easily derived from $Avg(K)$, or directly estimated from the data in the same way as $Avg(K)_{i,j}$. Since $W(K)$ and $B(K)$ respectively represent the *internal cohesion* of classes and the *external similarity* between different classes, it is clear that the parameter we are defining must decrease with $W(K)$ and increase with $B(K)$. We shall therefore define our parameter, that we call *Class Similarity* (CS), so that it is monotone in $B(K) - W(K)$. Here is one of the many possible definitions:

$$CS = \arccos W(K) - \arccos B(K)$$

The three following figures refer to three experiments on three different data sets, all available in the UCI Repository (Blake and Merz 1998).

In each of the following graphs the solid line on the top indicates an estimate of CS and its values appear on the right axis. The lines at the bottom indicate the generalization errors of two SVMs and they refer to the values appearing on the left axis. The solid line is a leave-one-out estimate of the error of SVM^{light} (Joachims 1998), while the dashed line is a 10-fold cross evaluation of the error of $LIBSVM$ (Chang and Lin 2001). The behaviour of the parameter CS is quite satisfying: in all the experiments, and in others that we have performed, not only the minimum of CS is not far from the minima of the errors, but also the curve closely follows those of generalization performance.

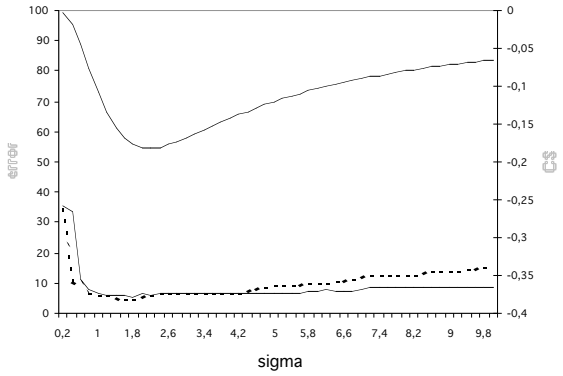

Fig. 1: Experiment on **`ionosphere`**

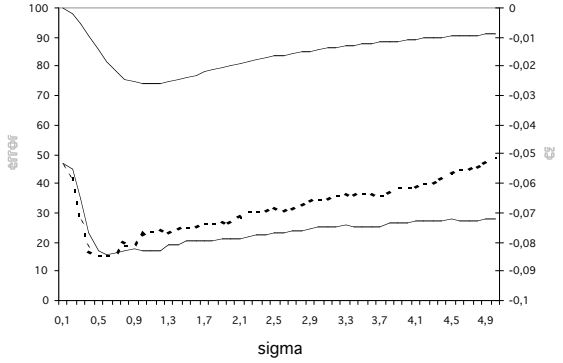

Fig. 2: Experiment on **`sonar`**

Notice that, in the three experiments, SVM^{light} and $LIBSVM$ approximately show the same generalization performance. In the experiments that will follow we therefore report only the results referring to SVM^{light}.

Experimental Evidence for an Aggregation Kernel

The kernels specified by the following definition can be applied to problems whose instances are vectors of symbolic attributes[7]:

[5] Since the sum of n numbers can be executed in time O(log n) by O(n) processors (JàJà 1992).

[6] W stands for 'within', B stands for 'between'.

[7] One of the possible values of each attribute is '?', which represents an unknown value.

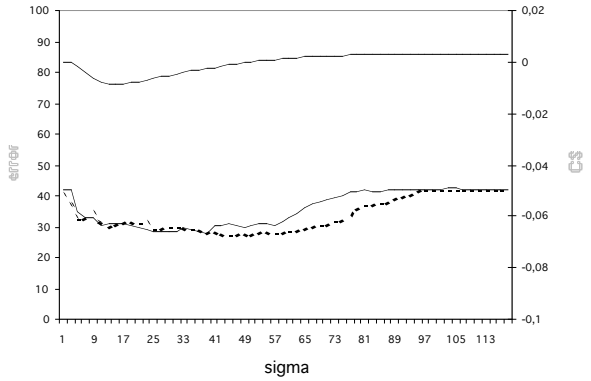

Fig. 3: Experiment on **liver**

$$K(\underline{x},\underline{y}) = \frac{\sum_{k=1}^{f} w_k \cdot \delta(x_k, y_k)}{\sum_{k=1}^{f} w_k}$$

where $\delta(a,b) = 1$ if $a = b \neq$ '?', else $\delta(a,b) = 0$, and $\underline{w}$ is any vector of weights.

$\underline{w}$ is the parameter that indexes this class of kernels. The search for its optimal value is a problem of Feature Weighting. If we succeed in verifying that *CS* strictly follows the performance of the SVM defined by K, we would have found an efficient method for solving this problem.

We shall explore the space of possible weights and plot the graphs of both the generalization error and *CS*. This will be done so that, at every step, the weights of a 'window' of consecutive attributes are 1, while all the other weights are 0. Initially all weights, except the first ($w_1 = 1$), will be set to 0. The window will shift towards the right until it reaches the end of the vector. Then its width will be increased by 1, and the window will continue moving in the opposite direction. If we do not stop this exploration[8], it will continue until all weights will equal 1. For example if $f = 4$, the explored sequence of weights will be: (1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1), (0,0,1,1), (0,1,1,0), (1,1,0,0), (1,1,1,0), (0,1,1,1), (1,1,1,1).

In practice we consider all $\underline{w}$ whose elements are 1 or 0 and such that the '1s' are assigned to consecutive positions. The reason for defining our exploration in this particular way, is that we wanted to obtain each $\underline{w}$ from the preceding one, by modifying at most 2 weights. In this way the graphs we obtain are not too 'zigzagging' and are sufficiently continuous. We finally remark that choosing

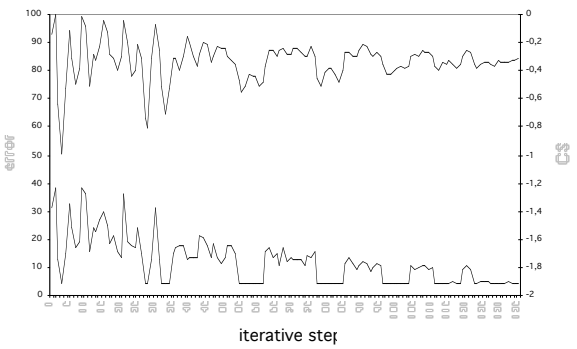

Fig. 4: Experiment on **votes**

weights in {0,1} is motivated only by reasons of simplicity.

In each of the following three figures, the line at the bottom shows the performance of SVM^{light} and its values appear on the left axis. The upper line indicates the value of *CS* and refers to the right axis. Each figure refers to a different dataset.

First of all, let us consider figure 4.

Thanks to the small number of features (16) and to the limited dimension of the dataset, it has been possible to execute the exploration until the end (i.e. until $\underline{w} = \underline{1}$) and to estimate the error through a leave-one-out evaluation. Notice that, despite its similarity with performances, *CS* shows an unjustified growth. Compare its minimum with the value it assumes in the last iteration: although they approximately correspond to the same error (about 5%) they have very different values: –1 versus about –0.25. This suggests that *CS* still needs some modifications, before being readily usable.

The exploration corresponding to figure 5 starts from the vector $\underline{w} = (1,1,0,\ldots,0)$ and stops once all windows of width 6 have been taken into account[9]. The generalization error is estimated on a fixed test set, whose size is approximately one half of the entire dataset.

The results are here very positive: *CS* precisely reflects the error. Also the unjustified growth of *CS* looks less evident, but this is probably due to the early interruption of the exploration.

The experiment on **splice** also required interrupting the execution after a certain number of iterations (180): the number of attributes is in fact high ($f = 60$) and each iteration needs a long time (since $m = 3190$). The error is estimated through a 2-fold evaluation. Even in this case the correspondence between *CS* and generalization error is quite good.

[8] In practice this is often necessary, since the exploration can go on for a long time. This is due to the computational complexity of training and testing the SVM for each weights assignment.

[9] For the entire exploration $f(f-1)/2 + f = 666$ iterations ($f = 36$) would be needed. We thought they were too many.

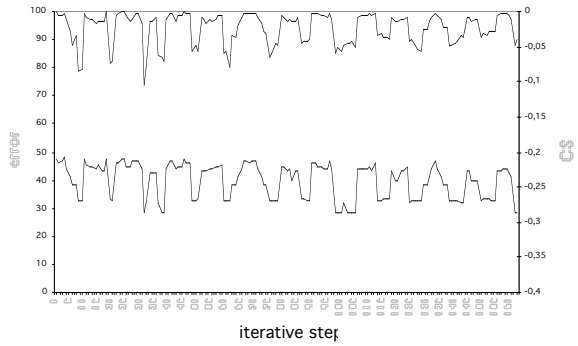

Fig. 5: Experiment on **`kr-vs-kp`**

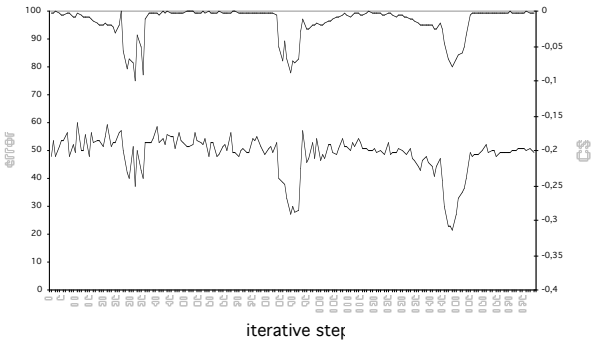

Fig. 6: Experiment on **`splice`**

The Usefulness of *Avg*(*K*)

In the last two sections, we provided strong experimental evidence to suggest a connection between $Avg(K)$ and the error of a SVM that has K as kernel. We also mentioned that a predictor of SVM learning performance can provide a method to select an optimal kernel in a very efficient way.

In order to see this, consider a predictor defined on the basis of $Avg(K)$. Every parameter of this kind can be estimated only looking at values of K on pairs of training examples. Let CS be such an estimate. Notice that, if $\{(x_i, y_i)\}_{i=1}^{m}$ is the training set, CS is function of every $K(x_i,x_j)$, for $i,j = 1,\ldots,m$.

Besides, suppose we are looking for an optimal kernel in a family of kernels K_σ indexed by parameters σ. If K_σ can be differentiated with respect to σ (as for Gaussian kernels), and if CS is a differentiable function of each $K(x_i,x_j)$, it will be possible to express the gradient of CS with respect to σ. There are several techniques for using this gradient to efficiently find the value of σ that optimizes CS and, hopefully, the performance of the resulting SVM.

Current Work

We are currently investigating two possible directions of future research. The first one consists in extending our method so that it can be applied to every kernel. Recall our definition of the induced kernel H_K. Our thesis is that, for every K, $Avg(H_K)$ can be a good predictor of the performance of *unbiased* SVMs, i.e. SVMs that select an optimal hyperplane among those passing through the origin ($b = 0$). In fact $Avg(H_K)_{i,j}$ is the expected value of the cosine of the angle between an i-th class element and one of the j-th class[10]. It is therefore related to the hardness of separating classes C_i and C_j, using a hyperplane such that $b = 0$.

The other aim of current research consists in looking for a formal connection between $Avg(K)$ and the error of a SVM using K. We recently proved an upper bound, expressed in terms of $Avg(K)$, for the optimal error of linear classifiers in the feature space.

Acknowledgments

I would like to thank my graduation thesis supervisor, Paola Velardi, for helpful comments and discussions.

References

Blake, C.L. and Merz, C.J. 1998. *UCI Repository of machine learning databases*. http://www.ics.uci.edu/~mlearn/MLRepository.html Irvine, CA: University of California, Department of Information and Computer Science.

Burges, C. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:1.

Chih-Chung Chang and Chih-Jen Lin 2001. *LIBSVM: a Library for Support Vector Machines (Version 2.31)*. http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Cortes, C. and Vapnik, V. 1995. Support Vector Networks. *Machine Learning*, 20: 273-297.

Duda, R. and Hart, P. 1973. *Pattern Classification and Scene Analysis*, John Wiley and Sons.

JàJà 1992. *An Introduction to Parallel Algorithms*, Addison Wesley.

Joachims, T. 1999. Making large-scale SVM Learning Practical. *Advances in Kernel Methods – Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press.

Vapnik, V. 1995. *The Nature of Statistical Learning Theory*, Springer Verlag.

[10] This is true also for $Avg(K)$ only when K has the hypersphere property.