SUPPORT VECTOR CLASSIFICATION

Submitted to

Dr. Joseph Picone
ECE8990 — Special Topics in ECE
Pattern Recognition

By
Issac Alphonso
May 9, 2001

SUPPORT VECTOR CLASSIFICATION

Issac Alphonso

Critical Review Paper
ECE 8990 - Special Topics in ECE
Pattern Recognition
email: {alphonso}@isip.msstate.edu

ABSTRACT

The foundations of Support Vector Classifiers (SVC) have been developed by Vapnik [1] and are gaining popularity due to many attractive features, and promising empirical performance. The formulation embodies the Structural Risk Minimization (SRM) principle, which has been shown to be superior to traditional Empirical Risk Minimization (ERM), employed by conventional neural networks. SRM minimizes an upper bound on the VC dimension, generalization error, as opposed to ERM that minimizes the error on the training data. It is this difference which equips SVC's with a greater ability to generalize, which is why they are able to perform better that most traditional classifiers.

1. INTRODUCTION

There is a family of bounds governing the relation between the capacity, i.e., the ability of the machine to learn any set without error and its performance. We will take a look at one of these bounds which was first introduced in [1]. Suppose we are given l observations each of which consists of a pair: a vector **x** and the associated label y. Now it is assumed that there exists some unknown probability distribution $P(\mathbf{x_i}, y_i)$ from which these data points are drawn iid. Now suppose we have a machine whose task it is to learn the mapping $\mathbf{x_i} \to y_i$. The machine is actually defined by the set of possible mappings $\mathbf{x_i} \to \boldsymbol{f}(\mathbf{x}_i, \mathbf{k})$, where the functions $\boldsymbol{f}(\mathbf{x_i}, \mathbf{k})$ themselves are labeled by adjustable parameters **k**. The machine is deterministic and a particular choice of **k** generates a trained machine. The expectation of the test error for the trained machine is:

$$R(\boldsymbol{k}) = \int \frac{1}{2} |y - f(\boldsymbol{x}_i, y_i)| \nabla P(\boldsymbol{x}_i, y_i)$$

The quantity R(**k**) is called the expected risk, or just the risk. The empirical risk $R_{emp}(\mathbf{k})$ is defined to be just the measured mean error rate on the training set:

$$R_{emp}(\boldsymbol{k}) = \frac{1}{2l} \sum_{i=1}^{l} |y_i - \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{k})|$$

The quantity $1/2|y - \boldsymbol{f}(\mathbf{x_i}, y_i)|$ id called the loss. For the case described here it can only take on values 0 and 1. We can define a quantity **v** such that it lies in the range [0, 1]. For losses taking these values, with probability 1-**v** the following bound holds [1]:

$$R(\boldsymbol{k}) \le R_{emp}(\boldsymbol{k}) + \sqrt{\left(h\left(\log\left(2\frac{l}{h}\right) + 1\right) - \log\left(\frac{\boldsymbol{v}}{4}\right)\right) / l}$$

where h is a non-negative constant called the Vapnik Chervonenkis (VC) dimension and is a measure of the notion of capacity. The right hand side of the equation is called the risk bound and the second term on the right hand side is the VC confidence.

The VC dimension is a property of a set of functions $\{\boldsymbol{f}(\mathbf{k})\}$. We only consider the two-class pattern recognition case, so that $\boldsymbol{f}(\mathbf{x_i}, \mathbf{k})$ is an element of {-1, 1} for every $\mathbf{x_i}$ and **k**. Now if a given set of l points can be labeled in 2^l ways, and for each labeling, a member of the set $\{\boldsymbol{f}(\mathbf{k})\}$ can be found which correctly assigns those labels, we say that the set of points are classified by that set of functions. The VC dimension is defined as the maximum number of training points that can be classified by $\{f(\mathbf{k})\}$.

The VC confidence depends on the chosen class of functions, whereas the empirical risk depends on the function chosen by the training procedure. Hence, we would like to find that subset of the chosen set of functions, such that the risk bound for that subset is minimized. SRM then consists of finding that subset of functions which minimizes the bound on the actual risk. This can be done by training a series of machines, one for each subset, where for a given subset the goal of training is to minimize the empirical risk. One then takes that trained machine whose sum of empirical risk and VC confidence is minimal.

2. SUPPORT VECTOR MACHINES

The goal of SVC is to devise a computational efficient way of learning good separating hyperplanes in a high dimensional feature space. We see how SVC works by first considering the simple case of linear machines trained on separable data. We assume that the training data is in the form $\{\mathbf{x_i}, y_i\}$ where $\mathbf{x_i}$ is an element of $\mathbf{R}$ and y is an element of the set {-1, 1}. Suppose we have a hyperplane that separates the negative from the positive examples. The points $\mathbf{x_i}$ which lie on the hyperplane satisfy $\mathbf{w}.\ \mathbf{x_i} + b = 0$, where $\mathbf{w}$ is the normal to the hyperplane and $|b|/||\mathbf{w}||$ is the perpendicular distance form the hyperplane to the origin. We define the margin of the hyperplane, i.e., the shortest distance from the hyperplane to the closest positive (negative) example to be:

$$\boldsymbol{x}_i \bullet \boldsymbol{w} + b \geq 1$$
$$\boldsymbol{x}_i \bullet \boldsymbol{w} + b \leq -1$$

These can be combines into one set of inequalities:

$$y_i((\boldsymbol{x}_i \bullet w + b) - 1) \geq 0$$

The distance form the negative and positive hyperplanes to the origin is $1/||\mathbf{w}||$. Hence, the margin is simply $2/||\mathbf{w}||$. Thus we can find the pair or hyperplanes which give maximum margin by minimizing $||\mathbf{w}||^2$. Using a Lagrangian formulation to the problem gives us:

$$L_P \equiv \frac{1}{2}\|\boldsymbol{w}\|^2 - \left(\sum_{i=1}^{l} \alpha_i y_i (x_i \bullet w + b) + \sum_{i=1}^{l} \alpha_i\right)$$

We must now minimize L_P with respect to $\mathbf{w}$, b and simultaneously require that the derivatives of L_P with respect to all the alphas vanish, all subject to the constraint that the alphas be greater that to equal to zero. This is a convex quadratic programming problem. Requiring that the gradient of L_P with respect to $\mathbf{w}$ and b vanish give the conditions:

$$\boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i$$
$$\sum_i \alpha_i y_i = 0$$

Since these are equality constraints in the dual formulation we substitute in L_P to get:

$$L_D = \sum_i \alpha_i - \sum_{ij} \alpha_i \alpha_j y_i y_i \boldsymbol{x}_i \bullet \boldsymbol{x}_j$$

The above technique when applied to non-separable data will find no feasible solution. We can extend the ideas above by relaxing the conditions when necessary by introducing a cost function. This can be done by adding positive slack variables in the constraints:

$$\boldsymbol{x}_i \bullet \boldsymbol{w} + b \geq 1 - \xi_i$$
$$\boldsymbol{x}_i \bullet \boldsymbol{w} + b \leq -1 + \xi_i$$

Thus, for an error to occur the sum of the slack variables must exceed unity. Hence, a natural way to assign an extra cost for the errors is to change the objective function to be minimized from $||\mathbf{w}||^2/2$ to $\|\boldsymbol{w}\|^2 / 2 + C\sum_i \xi_i$

Hence, similar to the linear case we need to maximize L_D subject to the constraints:

$$0 \leq \alpha_i \leq C$$
$$\sum_i \alpha_i y_i = 0$$

The solution is again given by:

$$\boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i$$

Where *i* loops over the support vectors, i.e., the points that lie on the positive (negative) hyper-planes that make up the margins.

3. KERNEL CHARACTERISTICS

The use of a kernel is an attractive computational short-cut. If we wish to use this approach, there appears to be a need to first create a complicated feature space, then work out what the inner products of that space would be, and finally find a direct method of computing that value in terms of the original inputs.

Kernels come into play when we deal with non-linear SVM's. In the solution for the non-separable data we notice that the only way that the data appears in the training problem is in the form of dot products $\mathbf{x}_i.\mathbf{x}_j$. If we first mapped the data to some other space using a mapping Φ then the training algorithm would depend on the data through the dot products in the feature space, i.e., on functions of the form$\Phi(\mathbf{x}_i) . \Phi(x_j)$, we would use **K** in the training algorithm and never need to explicitly even know what Φ is. An example of the use of kernels is in the test phase by computing the dot products of a given test point **x** with **w** and computing the sign:

$$f(\boldsymbol{x}) = \sum_{i=1} \alpha_i y_i \Phi(\boldsymbol{s}_i) \bullet \Phi(\boldsymbol{x}) + b$$

where $\mathbf{s}_i$ are the support vectors. Hence, we avoid computing $\Phi(\mathbf{x})$ explicitly by using the kernel function **K**. A function must satisfy some properties to ensure that is a kernel for some feature space. Two basic conditions are that a kernel must be symmetric and satisfy the Cauchy-Schwartz inequality. These conditions are, however, not sufficient to guarantee the existence of a feature space. To guarantee the existence of a feature space a kernel function must satisfy Mercer's theorem.

Mercer's theorem in short states that if X is a finite input space with $K(\mathbf{x}, \mathbf{z})$ a symmetric function on X. Then $K(\mathbf{x}, \mathbf{y})$ is a kernel function if and only if the kernel matrix is positive semi-definite (has non-negative eigen values). The kernel matrix is simply $\mathbf{K}_{ij}=K(\mathbf{x}_i, \mathbf{x}_j)$.

4. MULTI-CATEGORY CLASSIFICATION

Constructing a classifier to produce a posterior probability P(class|input) is very useful in practical recognition situations. Posterior probabilities are also required when a classifier is making a small part of an overall decision, and the classification outputs must be combined for the overall decision.

However, SVM's produce an uncalibrated value that is not a probability. Let the unthresholded output of the SVM be $f(\mathbf{x}) = h(\mathbf{x}) + b$ where

$$h(\boldsymbol{x}) = \sum_i y_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x})$$

Instead of estimating the class-conditional densities $p(f|y)$, we use a parametric model to fit the posterior $P(y = 1|f)$ directly. The class-conditional densities between the margins are apparently exponential. Bayes' rule onto two exponentials suggests using a parametric form of a sigmoid

$$P((y = 1|f)) = 1/(1 + \exp(Af + B))$$

where A and B are the slope and offset of the sigmoid fit for the model. The way multi-category classification works is that we train one SVM for each category. Then using the models generated by the trained SVM's we generate models to fit the sigmoids. In the testing stage each trained SVM give us a distance measure for a test point. Using the sigmoid models we can determine the posterior probability for each of the distances. Hence, the sigmoid model with the highest posterior probability is the most like category that can be assigned to the test point.

5. EXPERIMENTS

The experiments were carried out on two data sets representing both static and temporal modelling of the data. The first data set, static modelling, is made up of 528 feature vectors for training, 379 feature vectors for development testing and 83 evaluation feature vectors. The feature vectors in the first data set have 10 dimensions and 11 categories. The second data set, temporal modelling, is made up of 925 feature vectors for training, 350 feature vectors for development testing and 225 evaluation feature vectors. The feature vectors in the second data set have 39 dimensions and 5 categories.

Table 1: Error rates for development testing

Dev Set	Avg# Support Vectors	WER
Set 1	89.82	0.57%
Set 2	1257.80	80.0%

The first data set has 11 categories, hence, 11 SVM's were trained on the 11 categories in a typical one verses the rest fashion. The posterior probability for each SVM was then determined using a parametric form of a sigmoid. Similarly, the second data set has 5 categories, hence, 5 SVM's were trained on the 5 categories in a one versus the rest fashion. Again the posterior probability for each SVM was then determined using a parametric form of a sigmoid. With either data set the category with the highest posterior probability was the most likely label for the test point.

The SVM's were trained using the Sequential Minimal Optimization given by J.C. Platt [2]. The parametric model fit for the sigmoid was determined using a model-thrust algorithm based on the Levenberg-Marquardt algorithm [3].

The development test result for the first data set in Table 1 indicates that the SVM's will be able to learn and classify the first evaluation set well. The 0.57% wer result was obtained after a line search to find the optimal number for bins to histogram the data when determining the posterior probability. The main reason for doing this is because the number of data points for training the sigmoid was insufficient to get a proper estimate to the posterior probability distribution. In fact when testing the evaluation set the points for both the training set and the development test set were merged to form a larger pool to train from.

The development test results for the second data set in Table 1 indicates that the SVM's will not be able to learn and classify the second evaluation set well. The main problem was that the posterior probability distribution predicted the second category for each test point in the evaluation set. On closer inspection the sigmoid models for the first three categories and the last two categories were almost identical. Furthermore, the number of support vector for the first three and last two categories were almost equal to the number of training point which indicates a high degree of overlap in the second data set.

6. CONCLUSION

In conclusion as expected SVC's are good static classifiers, however, they are not very good when it comes to modelling temporal data. If the temporal structure of the second data set were removed I don't expect any improvements in the error rate primarily due to the high overlap in the categories as evident from the support vectors.

REFERENCES

[1] V. N. Vapnik, "The Nature of Statistical Learning Theory," Springer-Verlag, New York, 1995.

[2] N. Cristianini, J. Shawe-Taylor, "An Introduction to Support Vector Machines," Cambridge Press, Cambridge, England, 2000.

[3] A. J. Smola, P. L. Barrlett, B. Scholkopf, D. Schuurmans, "Advances in Large Margin Classifiers," MIT Press, Massachusetts, USA , 2000.

[4] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," Available on-line at URL: http://www.research.microsoft.com/