Improving SVM Classifiers Training Using Artificial samples

A. LABED, M. NADIL & D. E. DAOUADI
Department of Computer Science
Polytechnic School of Bordj El Bahri
Bp 17 Bordj El Bahri Algiers
Algeria

Abstract: - Estimating the generalization capability is one of the most important problems in supervised learning. That is why, various generalization error estimators have been proposed in the literature.

In this paper we propose an approach based on randomly generated objects to enhance the quality of training step of a standard SVM multi-class classifier and consequently try to reduce its generalization error. The idea is to generate artificial test samples which help automatic classifiers learn from their mistakes by reintroducing the misclassified examples in training set. But adding misclassified examples to the training set will induce a more complex quadratic program on which the decision rule is based. To overcome this complexity, while additional learning vectors are introduced, we integrated the idea of incremental training to our method.

Key-Words: - SVM, Retraining, Generalization error, Artificial samples, Incremental training.

1 Introduction

A classifier is considered to be good or not accordingly to its ability to generalize. In practice we can obtain classifiers that have very low empirical errors, but may give unacceptable errors when used to classify new data (examples not seen during the training phase).

The SVM (binary case) technique is based on the structural risk minimization using the Vapnik and Chervonenkis dimension. Theoretically speaking, SVM classifiers present some robustness due to the margin maximization. This margin allows classifying a new object with more confidence.

It has become common to say that the popular support vector machines (SVM) classifiers have good generalization when applied to any kind of dataset [10]. But, in many cases SVMs do not give the best performances especially when applied to datasets with thousands of objects or data sets with few training examples.

The SVM (binary case) classifiers are based on the structural risk minimization using the Vapnik and Chervonenkis dimension. They rely on margin maximization [8] and hence present some robustness.

Despite its theoretical robustness, in practice an SVM classifier does not always give low generalization errors. This is particularly true, when the training sample size is small compared to the number of features (sub-training) or when training is based on a set of objects that does not contain some important individuals (hidden information). So, if they are further submitted to the classifier they will be misclassified.

It is clear that the performances of classifiers depend crucially on the ability of the training sample to represent the entire population of objects.

The criterion that is often used to appreciate the quality of a classifier is the generalization error. Unfortunately it is very hard to find an exact analytical expression for this error, given the training data, and use standard optimization techniques to minimize it.

Researchers proposed solutions to the weak generalization problem of classifiers for which the training is based on samples of small size, by techniques like regularization and noise injection [9]. Our heuristic is similar in spirit to the combination of two ideas:

a- The idea of noise injection in the sense that we randomly generate data to be tested from a wider distribution;

b- We focus on misclassified objects by reintroducing them in the training dataset.

But these two ideas were exploited in a slightly different way in our work. Explicitly, we use the available sample to train our classifier, and then

progressively generate samples to test it. The misclassified objects are added to the previous training set and training step is repeated. The goal of this retraining is to enlarge the variation specter of the training data. So, a larger number of practical situations will be taken into account in the learning phase. We have called the '*training-test-retraining'* process, *'progressive learning'*. It is in some way similar to the natural learning of a child.

Other researchers gave estimates for upper bound on the generalization error to give some confidence to the future users of such classifiers [3]. So they are sure that in the worst cases they risk to have very low errors with high probabilities. Others tried to conduct statistical studies on the evolution of this error and constructed the receiver operator curves (ROC) to show the robustness of classifiers.
In our work we focus on training enrichment to overcome under-training and hidden information problems. More explicitly, we intend to allow training become as general as possible, so that our SVM classifier will be able to minimize effects of causes leading to a weak generalization.

After the selection of a good model accordingly to some criterion, we begin a retraining step. It is based on artificial data (randomly generated samples). The idea is that a larger number of practical situations will be taken into account in the learning phase.
Before giving more details about this approach, we will give the main sources of bad generalization.

2 Sources of weak generalization

SVM classifiers are based on the construction of a hyperplane that separates the objects of two classes: negative objects labelled by $y_i = -1$ and positive objects labelled by $y_i = +1$. The equation of the hyperplane is: $wx + b = 0$. (1)
The vector w and the constant b are chosen such that they maximize the margin.
In the general case we have to solve the quadratic program bellow:

$$Max\, L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \quad (2)$$

Subject to $0 \le \alpha_i \le C$ (3)

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad (4)$$

The αi's are Lagrange multipliers, C is the regularization parameter and K(.,.) is a kernel function (linear, polynomial or radial basis function: RBF).

The decision function for the classifier is then:

$$f(x) = \mathrm{sgn}\left(\sum_{i=1}^{N} y_i \alpha_i K(x_i, x) + b \right) \quad (5)$$

2.1 Model selection

The choice of parameter C in all cases (linear, polynomial and RBF kernels) and the type of kernel with its own parameters is a crucial step in the construction of an SVM classifier. The performance of SVM classifiers is very sensitive to the model's parameters. Indeed, not only changing the kernel will influence the result but simply changing the kernel's parameters will induce great variations in the results. So, in our work, before using the classifier to assign new data, we have to be sure that model parameters have been chosen in a way that assure the lowest generalization error.

2.2 Undertraining

If the training set is a sample of relatively small size, compared to the number of features, it can be a source of high generalization error. Training SVM classifier using this sample can lead to what is called under training.

2.3 Hidden information

The database used for the construction of a classifier may not contain some important data that should have significantly influenced its decision function. If these data where present, they should have been taken as support vectors. These data are called hidden information. If their number is sufficiently high, we can say that our sample is information poor. Consequently, the classifier will have a bad generalization.

To reduce the generalization error, solutions must be found for solving these three problems.
Our approach focuses on the two last problems.

3 Solutions for weak generalization

3.1 Model selection

Until now there is no method that helps to systematically choose the parameters of the model. Indeed many researchers in the field of automatic training are still working on choice of parameters that allow the classifier to well behave when facing new individuals. The best way to choose these parameters is based on an estimation of the generalization error given the training data. The most common techniques for estimating this error are based on cross-validation.

However, even if this error is relatively high, no satisfactory solution is proposed for undertraining and hidden information.

The cross-validation can be described as fellows:

Given a training set

$S=\{(x_1,y_1),(x_2,y_2),.....,(x_N,y_N)\}$ $x_i \in R^l$,

$i=1,\ldots,N$ and $y_i \in \{\pm 1\}$.

Subdivide this set into *M* subsets of approximately equal sizes.

For every subset

Perform training using (*M*-1) subsets.

Estimate the error with the remaining subset.

End for

Calculate the average error on the *M* estimations.

End.

If *M* is chosen such that *M = N*, the cross-validation is called Leave one out (Loo) technique.

Estimating the generalization error this way, is time consuming because we would need to construct N classifiers. Jaakkola & Hausler [1] proved a theorem that gives an upper bound for the estimate of the generalization error by only constructing one SVM classifier.

For the SVM classifier, the generalization error estimated by N *cross-validations* (loo) is bounded by:

$$\frac{1}{N}\sum_{i=1}^{N} \mathrm{STEP}\left(- y_i \left(\sum_{j\neq i} \alpha_j y_j\, k(x_i,x_j) \right) + b \right) \quad (1)$$

The functions STEP count the number of objects that are misclassified using the prediction model.

One interesting method that helps selecting good parameters for SVMs is described by Chih-Jen Lin [2]. It consists in first, choosing the kernel (linear, polynomial or RBF) then defining intervals of variation for its parameters. For the polynomial kernel the degree is often taken in the set

A = {1, 2, 3, 4, 5, 6, 7}.

For the RBF, the values of the set:

{0.00025, 0.0005, 0.001, 0.002, 0.004, 0.008, 0.016, 0.032, 0.064, 0.128, 0.256, 0.512, 1.024, 2.048} were tried for the parameter $\gamma = \frac{1}{2\sigma^2}$.

The parameter C (for all the kernels) was taken in the set:

{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024}.

For every model, we estimated the upper bound on the generalization error (UGE) using formula (1) and selected the model that produced the lowest bound.

3.2 Undertraining and hidden information

The first difficulty in classification is to find a representative database to be used to train a machine (classifier). Once these data are collected, they are subdivided into two parts: one to extract the parameters of the classifier and the other to test its performances.

The main idea of our work is to, first augment the training set by adding the misclassified objects of the test sample and then retrain the classifier. So we make the machine learns from its mistakes. Unfortunately, in practice, data are often rare and expensive and the available database does not suffice to repeat the training-test process many times. To obtain a classifier that generalizes well even though based, in its training, on a sample of small size, we generate artificial samples to enrich the training. Hence, the data used to train the machine will cover a larger range of variation and take account of more practical situations.

We assume that a tolerated error (Tol_Err) is given. So after training, we perform some tests and consider the training to be satisfactory if the error produced by these tests is lower than Tol_Err.

The main steps of our approach are given bellow:

Step 1: Subdivide the data into three parts: One for training, the second, to have an idea its generalization, and the third as a reference for measuring the improvement induced by our approach.

Step 2: Train the SVM and estimate the UGE.

Step 3: Calculate the rate of misclassified objects of the second test sample.

Step 4: a) Classify the objects of the first test sample. If the rate of misclassified objects is greater than the Tol_err, add these objects to the SVMs of the previous step and repeat step 2.

b) Recalculate the rate of misclassified objects of the second test sample. If this rate is lower than the tolerated error, stop.

c) Else, go to step 5.

Step 5: a) Generate artificial data and classify them (according to the model obtained for the last training).

b) If their classification error is greater than the Tol_Err, then add misclassified data to the SVMs of previous step and repeat step 2.

c) Recalculate the classification error for reference sample.

If the new error is lower than Tol_Err, stop training.

d) Else, generate a new sample and repeat sub steps (b) and (c).

But before implementing our idea, we had to answer two questions:

1. According to which distribution we can generate the artificial samples?
2. It may happen that after a few failures, the training sample size becomes very high and the associated quadratic program difficult to solve. So, how to speed the construction of SVMs?

For the first, we try to fit the data to appropriate theoretical distributions and generate samples from these distributions. In our work we only generated samples from Gaussian distributions.

To overcome the second problem we exploited the idea of incremental learning. So, at each step, the new data are learned recursively by only augmenting them with the support vectors of the previous step [11]. The underlying idea is that the support vectors are able to represent all the relevant information contained in the whole sample.

The main steps of incremental learning algorithm can be summarized as fellows:

1- Subdivide the entire input set ***TR*** into ***k*** subsets $\boldsymbol{DS_1}$, $\boldsymbol{DS_2}$,…,$\boldsymbol{DS_k}$.

2- Initialize the learning set ***TS*** to $\boldsymbol{DS_1}$.

3- Train the SVM on ***TS*** and save the support vectors $\boldsymbol{SV_1}$.

4- For $\boldsymbol{i = 1,\ldots k-1}$, repeat:

(a) $\boldsymbol{TS = SV_i \cup DS_{i+1}}$

(b) Train the SVM on ***TS*** and save $\boldsymbol{SV_{i+1}}$.

4 Experiments

To show the effectiveness of our heuristic, we used three datasets:

a. Quality of wines: these 178 objects subdivided (at random) into: 98 instances for training, 40 for a first test and 40 for a second test sample.

b. Iris dataset: the 150 individuals subdivided into: 90 instances (30 for each class) for training, 30 (10 for each class) for the first test and 30 for the second.

c. Fingerprints dataset: 68 fingerprints taken from four different classes were obtained by applying a Radon transform followed by a PCA transform on the original pixels. We obtained a dataset of 68 objects described by 67 features for training, 100 for the first test and 100 for the second.

The second test sample was used as a reference sample to compare the error of a standard SVM and that that the retrained SVM.

Our implementation in visual C^{++} was based on the sequential minimal optimization (SMO) algorithm. For the multi-class case the one against all strategy has been used.

For the first database, we succeeded to correcly classify the entire training individuals using linear kernels with the parameters: C = 512 for the first hyperplane, C = 4 for the second and C = 8 for the third. The UGE was 11.22%. This model is then used to estimate the class of the first test sample individuals. All the objects were assigned to the correct classes. But the same model gave an error of 2.50% (one misclassified individual) on the second test sample. So, we generated artificial samples of 100 individuals (34 from the first class, 33 from the second and 33 from the third). For the first artificial sample, we obtained two misclassified individuals (2% error). These two objects were added to the training set and the SVM is retrained using the same model (C = 512 for the first hyperplane, C = 4 for the second and C = 8 for the third). UGE estimate, for the new model, is 15% and applying the new SVM to the second test sample gave a 0% error. This shows that using our approach, we were able to reduce the error (from 2.5% to 0%).

For the second data set, linear kernels (with any value of C) allowed to separate all the training objects of class1 from the other training objects (from class2 and class3). For all the tried models (all the values of parameter C), the UGE estimate is 2.22%. So we chose the model with the lowest C (C=1). But linear and polynomial kernels were inappropriate to separate the second and the third hyperplanes. So, RBF kernels were used. For the second hyperplane, the minimal UGE was 4% and obtained with C = 2 and γ =1.024, while a minimal UGE of 6.88% was obtained for the third with the parameters C =128 and γ = 0.256. The tests, with this model, produced a classification error of 6.66% (two misclassified objects) on the reference sample and 3.33% (one misclassified object) on the first test sample. So, we reintroduced the misclassified object of the first sample in the training set and retrained the SVM with augmented sample, using the same parameters of the previous model.

UGE estimate after retraining was 6.95% and the classification error over the second test sample 6.66% (two misclassified data). We can then notice that retraining the SVM by adding the misclassified data of the first test sample did not reduce the

classification error over the second test sample. That is why artificial data were generated. The first sample (of 100 objects) gave five misclassified objects.
The resulting model gave 3.33% error (one misclassified object) on the reference sample, which is lower than that obtained after the previous retraining (6.66%). A second artificial sample has been generated, for which we obtained three misclassified objects. With this last retraining we succeeded to classify correctly all the elements of the reference sample. Again, this shows that, with our retraining process, we reduced the error for the iris data set.

The third experiment illustrates the case of training dataset of small size (68 objects) compared to the number of features (67 features). Using our features, we were able to linearly separate the four classes. The lowest errors were obtained with the same parameter C which was 100 for the four hyperplanes. The empirical error for the initial training was zero. The generalization error estimates were: 32% for the first Hyperplane, 36% for the second, 36% for the third and 30% for the fourth. In this case the SVM classifier gave a bad generalization. This result was also true for the reference sample on which we initially obtained a 20% error. Applying our heuristic to this data, we succeeded to decrease this error to about 10% after having generated 9 samples of 100 objects (25 objects from each class).

5 Conclusions

The tests performed on the three data sets, have shown that we were able to ameliorate the scores of standard SVM multi-class classifiers. Indeed, this has been proven for two different kinds of data: linearly separable data (first and third data sets) and not linearly separable data (second data set).

We selected optimal models (parameters that gave the lowest UGE); to be sure that improvement did not result from randomness of the SMO algorithm, but from training enhancement based on artificial data.

Finally, we can then assert that training a standard SVM, using our approach, reinforces training and helps this classifier generalize better.

References:

[1]T. S. Jaakkola, D. Haussler, "Probabilistic Kernel Regression Models ", Tech. Report, Department of Computer Science, University of California 1999.

[2] J. H. Lee and C.J. Lin., "Automatic Model Selection for SVMs ", Tech. Report, Department of Computer Science and Information Engineering, National Taiwan University, 2000.

[3] M. Sugiyama, Y. Okabe and H. Ogawa, "Perturbation Analysis of a Generalization Error Estimator", Neural Information Processing - Letters and Reviews, Vol.2, No.2, February 2004.

[4] R. Meir and T. Zhang, "Generalization Error Bounds for Bayesian Mixture Algorithms", JMLR, Vol. 4, 2003, pp. 839-860.

[5] S. Mannor and N. Shimkin, "A Geometric Approach to Multi-Criterion Reinforcement Learning", JMLR 5 (2004) 325-360.

[6] M. Anthony, "Generalization Error Bounds for Threshold Decision Lists", JMLR, vol 5, 2004, pp 189–217.

[7] J.C. Platt, ``Fast Training of Support Vector Machines using Sequential Minimal Optimization," Advances in Kernel Methods - Support Vector Learning, MIT Press, 1999, pp. 185-208.

[8] V.N. Vapnik, ``The Nature of Statistical Learning Theory " 2nd edition, Springer, 2000.

[9] M. Skurichina " stabilizing weak Classifiers ", PhD Thesis, TUDelft, 2001.

[10] L. Wolf & I. Martin, " Robust Boosting For Learning from Few Examples ", Proceedings of CVPR'05, 2005.

[11] J. Langford and D. McAllester, J. Langford and D. McAllester, "Computable Shell Decomposition Bounds", JMLR, vol 5, 2004, pp 529-547.