ON THE STABILITY OF SUPPORT VECTOR MACHINES FOR FACE DETECTION

I. Buciu   C. Kotropoulos   and   I. Pitas

Department of Informatics, Aristotle University of Thessaloniki
GR-540 06, Thessaloniki, Box 451, Greece, {nelu,costas,pitas}@zeus.csd.auth.gr

ABSTRACT

In this paper we study the stability of support vector machines in face detection by decomposing their average prediction error into the bias, variance, and aggregation effect terms. Such an analysis indicates whether bagging, a method for generating multiple versions of a classifier from bootstrap samples of a training set, and combining their outcomes by majority voting, is expected to improve the accuracy of the classifier. We estimate the bias, variance, and aggregation effect by using bootstrap smoothing techniques when support vector machines are applied to face detection in the AT & T face database and we demonstrate that support vector machines are stable classifiers. Accordingly, bagging is not expected to improve their face detection accuracy.

1. INTRODUCTION

Pattern recognition has numerous applications in digital image analysis and computer vision, notably in region segmentation. In this case, image pixels can be assigned to a given homogenous region according to a homogeneity criterion on a feature vector. A class is a set of identically labelled regions in the image domain. A classifier is a system developed to automatically assign pixels to predefined classes. In such a task, the classifier takes a new image as input and segments it into clearly identified regions, if properly trained. The objectives of classifier design are: to find the proper feature vector, to employ the proper parameterized classifier structure that captures the classification rules, to define the cost function to be minimized, and to select the optimization algorithm, so that, finally, the resulting data classification is as close as possible to the available ground truth.

A performance measure of a classifier is its *accuracy*, which is defined as the ratio of correct classifications. Improving the accuracy is equivalent with a decrease in the *prediction error* which is defined as $1 - accuracy$. A well known method for estimating the prediction error is the so-called *bootstrap*, where sub-samples of the original data set are analyzed repeatedly [1]. Bagging is a variant of the bootstrap technique, where each sub-sample is a random sample created with replacement from the full data set [2]. Bagging has produced superior performance in many applications is found to improve the performance of unstable classifiers [2]. However, there are several classifiers for which bagging has either a little effect or may slightly degrade the performance of classifier (e.g. k-nearest neighbor, linear discriminant analysis) [3]. The classifiers for which bagging does not work are considered as stable classifiers. If small changes of the training set lead to a varying classifier performance after bagging, the classifier is considered as an unstable one. The unstable classifiers are characterized by a high variance although they can have a low bias. On the contrary, stable classifiers have a low variance but can have a high bias.

Support Vector Machines (SVMs) [4, 5] among others have been successfully applied for face detection [6, 7]. Motivated by the studies in [3] we address in this paper the question whether support vector machines are stable classifiers or not. To do so we decompose the average prediction error in terms of the bias, variance, and aggregation effect and we estimate the aforementioned terms using bootstrap techniques. We provide numerical evidence that SVMs are indeed stable classifiers. Accordingly, bagging will not improve their classification accuracy in the framework considered. To the best of the authors' knowledge, such a result has not been presented elsewhere previously.

2. DECOMPOSITION OF THE AVERAGE PREDICTION ERROR

Let $\mathbf{x}$ be a vector of l attributes $\mathbf{x} = (x_1, x_2, \ldots, x_l)$ and y the class label associated with $\mathbf{x}$. A labeled instance or *training pattern* is a pair $\mathbf{z} = (\mathbf{x}, y)$ where $\mathbf{x}$ is an element from space $\mathbf{X}$ and y is an element from space Y. Without loss of generality, we consider the two-class problem, i.e. $y_i \in \{-1, +1\}$. The probability distribution over the space of labeled instances is denoted with $\mathcal{F}$. A sample set is a set of training samples $\mathcal{L} = \{\mathbf{z}_i \mid i = 1, \ldots, n\}$. We call this set the training set. We assume that the training samples are independent and identically distributed, that is, $\mathbf{Z}_1, \ldots, \mathbf{Z}_n \sim \mathcal{F}(\mathbf{x}, y)$.

In a classification problem we construct a classification rule $C(\mathbf{x}, \mathcal{L})$ on the basis of $\mathcal{L}$. The output of the classifier will be then $c_i \in \{-1, +1\}$. Let $Q[y, c]$ indicate the loss function denoting an error measure between the predicted class label c and the actual class label y. A plausible choice is $Q[y, c] = I_{\{y \neq c\}}$, that is, $Q[y, c] = 1$ if $y \neq c$ and 0 otherwise.

This work was supported by the European Union Research Training Network "Multi-modal Human-Computer Interaction (HPRN-CT-2000-00111).

Let $\mathbf{Z}_o = (\mathbf{X}_o, Y_o)$ be a another independent draw from $\mathcal{F}$ called the test pattern with value $\mathbf{z}_o = (\mathbf{x}_o, y_o)$. The *average prediction error* for the rule $C(\mathbf{X}_o, \mathcal{L})$ is defined as

$$err(Y_o, C) = E_{\mathcal{F}}\{E_{O\mathcal{F}}\{Q[Y_o, C(\mathbf{X}_o, \mathcal{L})]\}\} \tag{1}$$

where $E_{\mathcal{F}}$ indicates expectation over the training set $\mathcal{L}$ and $E_{O\mathcal{F}}$ refers to expectation over the test pattern $\mathbf{Z}_o \sim \mathcal{F}$. The average prediction error can be decomposed in components to allow a further investigation. Several approaches to decompose the prediction error into its bias and variance have been suggested. However, due to the fact we would like to express the decomposition in terms of the loss function, we are motivated to follow the approach proposed in [8].

Let $P(y_j \mid \mathbf{x})$ denote the a posteriori probabilities. It is well known that the Bayes decision rule yields the minimum prediction error, that is:

$$err(Y, C_{opt}) = 1 - \int_{\mathbf{X}} \max_{y_j}(P(y_j \mid \mathbf{x}))p(\mathbf{x})d\mathbf{x}, \tag{2}$$

where

$$C_{opt}(\mathbf{x}) = \arg\max_{y_j} P(y_j \mid \mathbf{x}). \tag{3}$$

Unfortunately, in real life, it is very difficult to have an exact knowledge of $C_{opt}(\mathbf{x})$ as long as neither the a priori probabilities $P(y_j)$ nor the class conditional densities $p(\mathbf{x} \mid y_j)$ are known. However, it has been shown that, as the size of the training set goes to infinity, the nearest neighbor prediction error is bounded below by the Bayes minimum prediction error and above by the bound given by [9]:

$$err(C_{opt}) \leq err_{NN} \leq err(C_{opt})\left(2 - \frac{n}{n-1} err(C_{opt})\right), \tag{4}$$

i.e., the prediction error for the nearest neighbor rule is bounded above by twice the Bayes prediction error.

Let us form B quasi-replicas of the training set $\mathcal{L}_1, \ldots, \mathcal{L}_B$, each consisting of n instances, drawn randomly, but with replacement. An instance $(\mathbf{x}, y)$ may not appear in a replica set, while others appear more than once. The learning system then generates the classifiers C_b, $b = 1, \ldots, B$, from the bootstrap samples and the final classifier C_A is formed by aggregating the B classifiers. C_A is called the *aggregated classifier*. In order to classify a test sample $\mathbf{x}_o$, a voting between the class labels y_{ob} derived from each classifier $C_b(\mathbf{x}_o, \mathcal{L}) = y_{ob}$ is performed and $C_A(\mathbf{x}_o)$ is then the class with the most votes. In other words

$$C_A(\mathbf{x}_o) \equiv E_{\mathcal{F}}\{C(\mathbf{x}_o, \mathcal{L}^*)\}. \tag{5}$$

where $\mathcal{L}^* = \{\mathcal{L}_1, \ldots, \mathcal{L}_B\}$. In the following, we define the bias and variance of a classifier. Let us define the *bias* and the *variance* of a classifier C as, $bias(C) \equiv err(C_{opt}, C_A)$ and $var(C) \equiv err(C, C_A)$, respectively. The bias can be written further as:

$$bias(C) = err(Y, C_A) - err(Y, C_{opt}). \tag{6}$$

Another quantity of interest is defined by the *aggregation effect*

$$ae(C) = err(Y, C) - err(Y, C_A) = (\delta - 1) \cdot err(Y, C_A) \tag{7}$$

where:

$$\delta \equiv \frac{err(Y, C)}{err(Y, C_A)}. \tag{8}$$

We expressed here the aggregation effect slightly different than in [8] by introducing the term δ, which is useful because it allows us to compute the aggregation effect regardless of the method used to calculate the prediction errors.

Having defined the bias, the variance and the aggregation effect of a classifier, it can be shown that the following decomposition is valid [8]:

$$err(Y, C) = err(Y, C_{opt}) + bias(C) + ae(C). \tag{9}$$

In the following, we will develop bootstrap estimates for the aggregated classifier and the terms in (9).

3. BOOTSTRAP ESTIMATES OF THE AGGREGATED CLASSIFIER AND THE AVERAGE PREDICTION ERROR COMPONENTS

Using the leave-one-out strategy, a sample-based estimate of the prediction error decomposition can be derived according to [8]. Following this way, we can draw the numerical evidence to demonstrate if a classifier is stable or not.

We can estimate the aggregated predictor expressed in (5) by

$$\widehat{C}_A(\mathbf{x}) \equiv E_{\widehat{\mathcal{F}}}\{C(\mathbf{x}, \mathcal{L}^*)\} \tag{10}$$

where $\widehat{\mathcal{F}}$ is the empirical distribution. The computation of the above expression is performed according to the following steps:

1. Create ordinary bootstrap samples $\mathcal{L}^* = \{\mathbf{z}_1^*, \mathbf{z}_2^* \ldots, \mathbf{z}_n^*\}$ with replacement from $\{\mathbf{z}_1, \mathbf{z}_2 \ldots, \mathbf{z}_n\}$;
2. Repeat B times the step 1, the total bootstrap samples being $\mathcal{L}_b^*$, $b = 1, 2, \ldots, B$;
3. Let N_i^b the number of times $\mathbf{z}_i$ appears in the b-th bootstrap sample and let us define:

$$I_i^b = \begin{cases} 1 & \text{if} \quad N_i^b = 0 \\ 0 & \text{if} \quad N_i^b > 0. \end{cases} \tag{11}$$

If $\widehat{\mathcal{F}}_{(i)}$ is the distribution having probabilities $1/(n-1)$ on all training observations except for $\mathbf{x}_i$ where it has zero probability, then the aggregated classifier can be estimated as:

$$\begin{aligned} \widehat{C}_A(\mathbf{x}) &= \frac{1}{n}\sum_{i=1}^{n} E_{\widehat{\mathcal{F}}_{(i)}}\{C(\mathbf{x}_i, \widehat{F}_{(i)})\} \\ &= \frac{1}{n}\sum_{i=1}^{n}\left[\frac{\sum_b I_i^b C(\mathbf{x}_i, \mathcal{L}_b^*)}{\sum_b I_i^b}\right]. \end{aligned} \tag{12}$$

The estimate of the variance is:

$$\begin{aligned} \widehat{var}(C) &= E_{\widehat{\mathcal{F}}} E_{\widehat{\mathcal{F}}_{(i)}}\{Q[C(\mathbf{x}, \mathcal{L}_b^*), \widehat{C}_A(\mathbf{x}, \widehat{\mathcal{F}}_{(i)})]\} \\ &= \frac{1}{n}\sum_{i=1}^{n} E_{\widehat{\mathcal{F}}_{(i)}}\{Q[C(\mathbf{x}_i, \mathcal{L}_b^*), \widehat{C}_A(\mathbf{x}_i, \widehat{\mathcal{F}}_{(i)})]\}. \end{aligned} \tag{13}$$

Here $\widehat{C}_A(\mathbf{x}_i, \widehat{\mathcal{F}}_{(i)})$ is the bootstrap estimate of the aggregated classifier built for each i and can be expressed by:

$$\widehat{C}_A(\mathbf{x}_i, \widehat{\mathcal{F}}_{(i)}) = \frac{\sum_b I_i^b C(\mathbf{x}_i, \mathcal{L}_b^*)}{\sum_b I_i^b}. \tag{14}$$

Then the bootstrap estimate of the variance of classifier becomes

$$\widehat{var}(C) = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{\sum_b I_i^b Q[C(\mathbf{x}_i,\mathcal{L}_b^*),\widehat{C}_A(\mathbf{x}_i,\widehat{\mathcal{F}}_{(i)})]}{\sum_b I_i^b}\right\}. \quad (15)$$

Subsequently, we determine the estimate of the prediction error for classifier C. Using the leave-one-out cross validation technique, the average prediction error (1) can be estimated in the following manner

$$\widehat{err}(Y_o, C) = E_{\widehat{\mathcal{F}}}\{E_{\widehat{\mathcal{F}}_{(i)}}\{Q[y, C(\mathbf{x},\mathcal{L}_{(i)})]\}\}, \quad (16)$$

where the set $\mathcal{L}_{(i)}$ contains samples drawn from $\mathcal{F}_{(i)}$ that never contain $\mathbf{x}_i$, that is, $\mathcal{L}_{(i)} = \mathcal{L} - \{(\mathbf{x}_i, y_i)\}$. Repeating the steps required for estimating the aggregated predictor, the final expression for the leave-one-out bootstrap estimation of the prediction error for C and $\widehat{C}_A$ is:

$$\widehat{err}(Y_o, C) = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{\sum_b I_i^b Q[y_i, C(\mathbf{x}_i,\mathcal{L}_b^*)]}{\sum_b I_i^b}\right\} \quad (17)$$

and

$$\widehat{err}(Y_o, \widehat{C}_A) = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{\sum_b I_i^b Q[y_i, \widehat{C}_A(\mathbf{x}_i,\widehat{\mathcal{F}}_{(i)})]}{\sum_b I_i^b}\right\}, \quad (18)$$

respectively. Now we can estimate the aggregation effect as:

$$\widehat{ae}(C) = (\widehat{\delta} - 1)\cdot\widehat{err}(Y_o, \widehat{C}_A). \quad (19)$$

The minimum (optimal) prediction error can be estimated as suggested in [9]

$$\widehat{err}(Y_o, C_{opt}) \geq \alpha - [\alpha(\alpha - err_{NN})]^{1/2} \quad (20)$$

where err_{NN} is the prediction error of the 1 - nearest neighbor classifier and $\alpha = 1/2$ in the case of a two-class problem. Then, the bias estimate is bounded by:

$$\widehat{bias}(C) \leq \widehat{err}(Y, \widehat{C}_A) - [\alpha - [\alpha(\alpha - err_{NN})]^{1/2}]. \quad (21)$$

Finally, the bootstrap estimate of prediction error is obtained by

$$\widehat{err}(Y_o, C) = err(Y_o, C_{opt}) + \widehat{bias}(C) + \widehat{ae}(C). \quad (22)$$

4. EXPERIMENTAL RESULTS

4.1. Data description

The AT&T (former Olivetti) database[1] was used in our experiments. This database of faces contains 10 different images for 40 distinct persons. The images have dimensions of 92 × 112 pixels. The images have been recorded at different times, with variations in the lighting, facial expression and facial details (glasses / nonglasses). Each face image includes fiducial points such as the eyebrow, eyes, nose, mouth and chin. Each image has been downsampled several times, yielding finally an image of 17 × 14 pixels. This preprocessing step was used to reduce the dimension of input patterns. The ground truth, that is, the true class label $y_i = +1$ was appended to each face pattern. Non-face patterns have been collected from images depicting wheels, bubbles, trees, etc. in the manner described in [10]. Therefore, the set has 306 face patterns, chosen randomly from this database, and 294 non-face patterns.

4.2. Support vector machines

Support vector machine (SVM) maps the input vectors $\mathbf{x}$ onto a high-dimensional feature space through a nonlinear mapping and seeks to construct the optimal separating hyperplane in the feature space [4, 5]. The algorithm relies on the computation of inner products between the so-called *support vectors*, i.e., input vectors whose associated Langrage multipliers in the underlying quadratic optimization problem solved during the training phase are nonzero as well as between the support vectors and the test vectors. If we deal with Hilbert feature spaces, there is no need to calculate the inner product in the feature space, because the inner product can be computed through a kernel function, $K(\mathbf{x}, \mathbf{x}_i)$, in the original space. In our experiments SVMs were based on the following kernels: (1) Linear kernel; (2) Polynomial of degree $q = 2$; (3) Exponential Radial Basis Function (ERBF) having σ equal to 10. These kernels have the analytical form listed in Table 1, where $||\cdot||_p$ denotes the vector p-norm.

Table 1: Kernel functions used in SVMs.

SVM type	Kernel function $K(\mathbf{x}, \mathbf{x}_i)$
Linear	$\mathbf{x}^T\mathbf{x}_i$
Polynomial with degree q	$(\mathbf{x}^T\mathbf{x}_i + 1)^q$
ERBF	$\exp(-\frac{\lVert\mathbf{x}-\mathbf{x}_i\rVert_1}{2\sigma^2})$

The purpose of SVM is to detect faces among the images. Our experiments had two phases:

4.2.1. Training phase

In this first phase we randomly chose a set of 50 input patterns from the entire data set, thus building the training set. Since bagging can be useful, especially when the available amount of data for training is small, we intentionally kept only 50 patterns from the entire set for training. We calculated the empirical distribution $\widehat{\mathcal{F}}$ and we computed the leave-one-out bootstrap estimate of the prediction error of SVMs, the leave-one-out bootstrap estimate of the prediction error for aggregated classifier, the bias, and the variance. The number of bootstrap replicas was 21. We repeated the experiment 10 times by forming other replicas. By increasing the number of bootstrap replicas up to 40, we ran again the experiment 10 times. For comparison, we experimented with a 5-nearest neighbor classifier and the results are depicted in Table 2 for 21 bootstrap replicas. The values tabulated in Table 2 are averages over 10 runs. The standard deviations are given inside parentheses. Note that we took the upper bound of expression (21) to estimate the bias. From Table 2 it can be seen that

[1]ftp://ftp.uk.research.att.com/pub/data/att_faces.zip

Table 2: Estimated components of the prediction error.

	SVM	kNN
$\widehat{err}(Y_o, C)$	0.5200 (0.0000)	0.0000 (0.0000)
$\widehat{var}(C)$	0.0000 (0.0000)	0.0257 (0.0053)
$\widehat{bias}(C)$	0.5200 (0.0000)	0.0044 (0.005)
$\widehat{err}(Y_o, \widehat{C}_A)$	0.5200 (0.0000)	0.0043 (0.005)
$\widehat{ae}(C)$	0.0000 (0.0000)	-0.0043 (0.005)
$\widehat{\delta}$	1	0

the prediction error of SVM after bagging does not change from the value it had before bagging. Due to the fact that err_{NN} is zero, the bias equals $\widehat{err}(\widehat{C}_A)$. A zero value of variance is a characteristic of a stable classifier. In the case of 5-nearest neighbor, bagging degrades the performance of this classifier.

4.2.2. Test phase

In the test phase we are only concerned with the prediction error before and after bagging. The following steps were executed:

1. Divide the initial set of images randomly into a training set of 50 images and a large test set comprised of remaining images. Train the SVM with this training set and then apply the trained SVM on the test set.
2. Build $B = 21$ bootstrap replicas from the initial training set. Train the SVM on each replica, thus obtaining B classifiers.
3. Apply each of the B classifier on the test set and aggregate those B classifiers for a final decision.
4. Repeat steps 1 - 3 for $m = 60$ times. Name this experiment $E1$.
5. Repeat steps 1 - 3 for $m = 20$ times with a number of bootstrap samples of $B = 61$. Name this experiment $E2$.

By averaging over the m iterations, we have $\overline{err}(C)$ and $\overline{err}(C_A)$. The results for the two experiments $E1$ and $E2$ are summarized in Table 3 for three kernel functions. Notice that the results are rates expressed in percentage.

Table 3: Average prediction errors (%) in the test phase for face detection in AT & T face database.

	Kernel	$\overline{err}(C)$	$\overline{err}(C_A)$	$\widehat{\delta}$
$E1$	linear	4.87	4.48	1.09
	polynomial	4.86	5.78	0.84
	ERBF	2.93	3.04	0.96
$E2$	linear	4.72	4.52	1.05
	polynomial	5.03	5.67	0.89
	ERBF	2.86	2.93	0.98

From Table 3, one can see that, after many trials, on average, $\widehat{\delta}$ is less than unity for the polynomial and ERBF kernel functions, and exceeds unity by a small amount for the linear kernel.

5. CONCLUSIONS

We have investigated the stability of SVMs in face detection conducted on the AT & T database by decomposing its average prediction error into the bias, variance, and aggregation effect terms and estimating the latter quantities by means of leave-one-out bootstrap smoothing techniques. Our effort was focused on the aggregation effect of SVM that measures the excess of the average prediction error of the standard SVM over the aggregated SVM (i.e., the classifier that results by training several SVMs on bootstrap samples of the training set and combining their outputs by uniform voting). If there were a significant gain when bagging was applied, then the aggregation affect would be a large positive quantity and the parameter $\widehat{\delta}$ would be much greater than one. We have demonstrated that this is not the case, accordingly SVM is a stable classifier that is not sensitive to variations of the training set. Since bagging is a computationally intensive technique, even with small accuracy improvement, this approach is not suitable for face detection where real time processing is needed.

6. REFERENCES

[1] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*, Chapman & Hall, New York, 1993.

[2] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.

[3] L. Breiman, "Bias, variance and arcing classifiers," *Technical Report 460*, 1996.

[4] V.N. Vapnik, *Statistical Learning Theory*, J. Wiley, N.Y., 1998.

[5] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, Cambridge, U.K., 2000.

[6] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," *in Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, pp. 130–136, 1997.

[7] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-based object detection in images by components," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 4, pp. 349–361, April 2001.

[8] R. Tibshirani, "Bias, variance and prediction error for classification rules," *Technical Report*, 1996.

[9] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inform. Theory*, pp. 21–27, 1967.

[10] J.K. Kung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 39–51, January 1998.