# Fingerprint Classification with Combinations of Support Vector Machines

Yuan Yao[1], Paolo Frasconi[2], and Massimiliano Pontil[3,1]

[1] Department of Mathematics, City University of Hong Kong, Hong Kong
[2] Department of Systems and Computer Science, University of Firenze, Firenze, Italy
[3] Department of Information Engineering, University of Siena, Siena, Italy

**Abstract.** We report about some experiments on the fingerprint database NIST-4 using different combinations of Support Vector Machine (SVM) classifiers. Images have been preprocessed using the feature extraction technique as in [10]. Our best classification accuracy is 89.3 percent (with 1.8 percent rejection due to the feature extraction process) and is obtained by an error-correction scheme of SVM classifiers. Our current system does not outperform previously proposed classification methods, but the focus here is on the development of novel algorithmic ideas. In particular, as far as we know, SVM have not been applied before in this area and our preliminary findings clearly suggest that they are an effective and promising approach for fingerprint classification.

## 1 Introduction

The pattern recognition problem studied in this paper consists of classifying fingerprint images into one out of five categories: whorl (W), right loop (R), left loop (L), arch (A), and tented arch (T). These categories were defined during early investigations about fingerprint structure [9] and have been used extensively since then. The task is interesting because classification can be employed as a preliminary step for reducing complexity of database search in the problem of automatic fingerprint matching [7,10]. Basically, if a query image can be classified with high accuracy, the subsequent matching algorithm only needs to compare stored images belonging to the same class.

Several pattern recognition algorithms have been proposed for fingerprint classification, including early syntactic approaches [12], methods based on detection of singular points [11], connectionist algorithms such as self-organizing feature maps [8], neural networks [13], and structural methods based on (dynamic) graph matching [1]. The current highest accuracy (92.2 percent on the NIST Database 4) was obtained with a method based on multi-space principal component analysis [2]. In spite of these efforts, the problem has not beed solved satisfactorily and there is undoubtedly room for further improvements in terms of classification accuracy, rejection threshold, and simplicity of design.

In this paper, we propose an fingerprint classifier based on Support Vector Machines (SVM), a relatively new technique to train classifiers that is well-founded in statistical learning theory [17]. One of the main attractions of using SVMs is that they are capable of learning in *sparse, high-dimensional spaces*

J. Bigun and F. Smeraldi (Eds.): AVBPA 2001, LNCS 2091, pp. 253–258, 2001.

with very few training examples. SVMs have been successfully applied to various classification problems (see [3] and references therein).

Since basic SVM are formulated for solving binary classification tasks, we designed a multiclass strategy based on a new error correcting code (ECC) strategy.

Our system is validated on FingerCode [10] preprocessed images from the NIST database 4 [18]. The best SVM combination achieves 89.3 percent accuracy with 1.8 percent rejection, due to failures of FingerCode in reliably locating the fingerprint core. This result is only 0.7 percent worse than the accuracy obtained in [10] using the same features and a two stages $k$-NN/MLP classifier. Interestingly, SVM's accuracy is much better than separate accuracies of both $k$-NN and MLP, but slightly worse than the cascading of the two. Hence, although preliminary, we believe our results are very promising and might yield state-of-the-art improvements if further refined.

## 2 Support Vector Machines

Support vector machines (SVMs) [17] perform pattern recognition for two-class problems by determining the separating hyperplane[1] with maximum distance to the closest points of the training set. These points are called *support vectors*. If the data is not linearly separable in the input space, a non-linear transformation $\Phi(\cdot)$ can be applied which maps the data points $\mathbf{x} \in \mathbb{R}^n$ into a high (possibly infinite) dimensional space $\mathcal{H}$ which is called feature space. The data in the feature space is then separated by the optimal hyperplane as described above.

The mapping $\Phi(\cdot)$ is represented in the SVM classifier by a kernel function $K(\cdot,\cdot)$ which defines an inner product in $\mathcal{H}$, i.e. $K(\mathbf{x},\mathbf{t}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{t})$. The decision function of the SVM has the form:

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}), \tag{1}$$

where $\ell$ is the number of data points, and $y_i \in \{-1, 1\}$ is the class label of training point $\mathbf{x}_j$. Coefficients $\alpha_i$ in Eq. (1) can be found by solving a quadratic programming problem with linear constraints [17]. The support vectors are the nearest points to the separating boundary and are the only ones for which $\alpha_i$ in Eq. (1) can be nonzero.

An important family of admissible kernel functions are the Gaussian kernel, $K(\mathbf{x},\mathbf{y}) = \exp\left(-\|\mathbf{x}-\mathbf{y}\|/2\sigma^2\right)$, with $\sigma$ the variance of the gaussian, and the polynomial kernels, $K(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}\cdot\mathbf{y})^d$, with $d$ the degree of the polynomial. For other important examples of kernel functions used in practice see [5,17].

Let $M$ be the distance of the support vectors to the hyperplane. This quantity is called *margin* and it is related to the cofficients in Eq. (1),

$$M = \left(\sum_{i=1}^{\ell} \alpha_i\right)^{\frac{1}{2}}. \tag{2}$$

[1] SVM theory also includes the case of non-separable data, see [17].

The margin is an indicator of the separability of the data. In fact, the expected error probability of an SVM is bounded by the average (with respect to the training set) of $\frac{R^2}{\ell M^2}$, with $R$ the radius of the smallest sphere containing the data points in the feature space.

### 2.1 Multi-class Classification

Many real-world classification problems involve more than two classes. Attempts to solve $q$-class problems with SVMs have involved training $q$ SVMs, each of which separates a single class from all remaining classes [17], or training $q^2$ machines, each of which separates a pair of classes [14,6,15]. The first type of classifiers are usually called *one-vs-all*, while classifiers of the second type are called *pairwise* classifiers. For the one-vs-all a test point is classified into the class whose associated classifier has the highest score among all classifiers. In the pairwise classifier, a test point is classified in the class which gets most votes among all the possible classifiers [6].

Classification schemes based on training one-vs-all and pairwise classifiers are two extreme approaches: the first uses all the data, the second the smallest portion of the data. In practice, it can be more effective to use intermediate classification strategies in the style of error-correcting codes [4,16]. In this case, the number of classifiers grows linearly with the number of classes. Each classifier is trained to separate a subset of classes from another disjoint subset of classes (the union of these two subsets does not need to cover all the classes). For example the first set could be classes A and T and the second classes R,L and W. By doing so, we associate each class with a row of the "coding matrix" $M \in \{-1, 0, 1\}^{q \times s}$, where $s$ denotes the number of classifiers. $M_{ij} = -1$ or 1 means that points in class $i$ are regarded as negative or positive examples for training the classifier $j$. $M_{ij} = 0$ says that points in class $i$ are not used for training classifier $j$. A test point is classified in the class whose row in the coding matrix has minimum distance to the output raw of the classifiers. The simplest and most commonly used distance is the hamming distance. We will discuss other distance measures in Section 3.

## 3 Experimental Results

### 3.1 Dataset

Our system was validated on FingerCode preprocessed fingerprints from the NIST Database 4 [18]. FingerCode is a representation scheme described in [10] and consists of a vector of 192 real features computed in three steps. First, the fingerprint core and center are located. Then the algorithm separates the number of ridges present in four directions (0°, 45°, 90°, and 135°) by filtering the central part of a fingerprint with a bank of Gabor filters. Finally, standard deviations of grayscale values are computed on 48 disc sectors, for each of the four directions. The NIST Database 4 consists of 4000 images analyzed by a human expert and labeled with one *or more* of the five structural classes W, R, L, A, and T (more

than one class is assigned in cases where ambiguity could not be resolved by the human expert).

Previous works on the same dataset either rejected ambiguous examples in the training set (loosing in this way part of the training data), or used the first label as a target (potentially introducing output noise). The error correcting code developed in this paper allows a more accurate use of ambiguous examples, since each SVM is only in charge of generating one codebit, whose value discriminates between two disjoint sets of classes. If a fingerprint has labels all belonging to the same set for a particular codebit, then clearly we can keep this example in the training set without introducing any labeling noise. As far as testing is concerned, we followed the customary convention of counting as errors only those predictions that do not agree with any of the labels assigned to the fingerprint.

Before discussing our results, we briefly summarize the results in [10]. Three different experiments were performed there: (a) A $k$-NN classifier with $k = 10$, (b) A MLP classifier, (c) A hybrid combination of (a) and (b): given a test point, first the k-NN classifier is used to compute the two most frequent labels. Then, the MLP corresponding to these two labels is used for the final classification. The accuracies obtained were of 85.4, 86.4, and 90.0 percent, respectively.

### 3.2 Results with SVMs

We used the three types of multi-class classification schemes discussed in section 2.1 which are based on the combination of binary SVMs. SVMs have been trained using the SVMFu code[2] on a 550MHz Pentium-II PC. Training on 2000 examples takes about 10s for pairwise classifiers and 20s for one-vs-all classifiers.

**One-vs-All SVMs.** We trained five one-vs-all SVM classifiers using both Gaussian kernels and polynomials of degree between 2 and 6. The best result was obtained with the Gaussian kernel ($\sigma = 1$): 88.0% (see the confusion matrix in Table 2a). The best polynomial SVM was of degree 3 and achieved a performance of 84.5%. Rejection can be decided by examining the margin (see Eq. (2)). Table 1 shows the accuracy-rejection tradeoff obtained in our experiments.

**Table 1.** Accuracy vs. rejection rate for the one-vs-all SVMs combination.

| Rejection Rate: | 1.8% | 3.0% | 6.1% | 10.9% | 17.3% | 24.2% | 30.2% |
|---|---|---|---|---|---|---|---|
| Accuracy: | 88.0% | 89.0% | 89.7% | 90.9% | 92.4% | 93.4% | 94.8% |

Finally, in the four classes task (classes A and T merged together) a Gaussian SVM with $\sigma = 1$ and $C = 10$ obtains an accuracy of 93.1%. For comparison, the accuracy reported in [10] for the sole MPL's is 92.1%, but the cascade of k-NN and MLP yields 94.8%.

[2] This software can be downloaded at `http://five-percent-nation.mit.edu/SvmFu`.

**Table 2.** Confusion matrices: (a) One-vs-all SVMs; (b) Pairwise SVMs; (c) ECC SVMs with margin weighted Euclidean decoding). Raws denote the true class, columns the assigned class.

| | W | R | L | A | T |
|---|---|---|---|---|---|
| W | 356 | 23 | 14 | 3 | 1 |
| R | 4 | 344 | 1 | 7 | 33 |
| L | 4 | 2 | 356 | 6 | 13 |
| A | 0 | 2 | 5 | 371 | 55 |
| T | 0 | 7 | 7 | 48 | 303 |

(a)

| | W | R | L | A | T |
|---|---|---|---|---|---|
| W | 359 | 24 | 15 | 3 | 1 |
| R | 4 | 341 | 1 | 6 | 36 |
| L | 5 | 0 | 356 | 6 | 15 |
| A | 0 | 2 | 4 | 363 | 58 |
| T | 0 | 7 | 9 | 38 | 318 |

(b)

| | W | R | L | A | T |
|---|---|---|---|---|---|
| W | 362 | 22 | 11 | 3 | 0 |
| R | 4 | 350 | 3 | 8 | 27 |
| L | 7 | 2 | 357 | 5 | 11 |
| A | 0 | 3 | 3 | 398 | 32 |
| T | 0 | 10 | 9 | 51 | 287 |

(c)

**Pairwise SVMs.** We trained the ten pairwise SVMs using always a Gaussian kernels with $\sigma = 1$. The test set accuracy increases to 88.4%, improving of 2% the MLP accuracy reported in [10]. The confusion matrix is reported in Table 2b.

**Error-Correction SVM Scheme.** Three sets of SVM classifiers were used to construct the coding matrix: 5 one-vs-all

classifiers, 10 two-vs-three classifiers and 10 pairwise classifiers. Three kinds of decoding distances were compared: (i) Hamming decoding: 88.0%, (ii) The loss-based decoding proposed in [16]: 88.8%; (iii) Margin weighted decoding (the distance function is defined as the Euclidean distance weighted by the margin): 89.3%. The confusion matrix for the last case is reported in Table 2c.

We have measured the margin as in Eq. (2) and the number of support vectors of each SVM classifier used in our experiments (the training error of each individual classifier was always zero). The number of support vector ranges between 1/5 and 1/2 of the number of training points. As expected the margin decreases for those classifiers which involve difficult pairs of classes. Among the pairwise classifiers, the A-T classifier has the smallest margin. The margin of the T-vs-all and A-vs-all is also small. However the margin of the AT-vs-RLW classifier increases, which might explain why our error correcting strategy works well.

## 4 Conclusions

We have presented experiments for fingerprint classification using different combinations of SVM classifiers. Images have been preprocessed using the features extraction technique as in [10]. Our best classification accuracy is obtained by an error-correction schemes of SVM classifiers. It improves separate accuracies of both $k$-NN and MLP of 3.9 and 2.9 percent respectively, while is only 0.7 percent worse than the best performance obtained with the same features [10].

As far as we know, this is the first experimental study of SVMs in the area of fingerprint classification. Therefore, we believe that our preliminary findings are promising and might yield state-of-the-art improvements if further refined. In particular this might be obtained by reweighting the features used by each SVM classifier using the technique in [19].

**Acknowledgment:** We wish to thank Anil Jain for providing us the dataset of preprocessed NIST-4 fingerprints.

## References

1. R. Cappelli, A. Lumini, D. Maio, and D. Maltoni. Fingerprint classification by directional image partitioning. *Transactions on Pattern Analysis Machine Intelligence*, 21(5):402–421, 1999.
2. R. Cappelli, D. Maio, and D. Maltoni. Fingerprint classification based on multi-space KL. In *Proceedings Workshop on Automatic Identification Advances Technologies (AutoID'99)*, pages 117–120, 1999.
3. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and other kernel-based learning methods)*. Cambridge University Press, 2000.
4. T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 1995.
5. T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.
6. Jerome H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1997.
7. R.S. Germain, A. Califano, and S. Colville. Fingerprint matching using transformation parameter clustering. *IEEE Computational Science and Engineering*, 4(4):42–49, 1997.
8. U. Halici and G. Ongun. Fingerprint classification through self-organizing feature maps modified to treat uncertainty. *Proceedings of the IEEE*, 84(10):1497–1512, 1996.
9. E.R. Henry. *Classification and Uses of Finger Prints*. Routledge, London, 1900.
10. A.K. Jain, S. Prabhakar, and L. Hong. A multichannel approach to fingerprint classification. *PAMI*, 21 (4):348–359, 1999.
11. K. Karu and A.K. Jain. Fingerprint classification. *Pattern Recognition*, 29(3):389–404, 1996.
12. B. Moayer and K.S. Fu. A syntactic approach to fingerprint pattern recognition. *Pattern Recognition*, 7:1–23, 1975.
13. K. Moscinska and G. Tyma. Neural network based fingerprint classification. In *Third International Conference on Neural Networks*, pages 229–232, 1993.
14. J. Platt, N. Cristianini, and J. Shawe-Taylor. Lrge margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, Denver, Colorado, 2000.
15. M. Pontil and A. Verri. Support vector machines for 3-d object recognition. *IEEE Trans. PAMI*, pages 637–646, 1998.
16. Robert E. Schapire, Yoram Singer, and Erin Lee Young. Reducing multiclass to binary: A unifying approach for margin classifiers. Technical report, AT&T Research, 2000.
17. V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.
18. C.I. Watson and C.L. Wilson. National Institute of Standards and Technology, March 1992.
19. J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for support vector machines. In *NIPS-13*, 2001. To Appear.