A New Machine Learning Approach to Fingerprint Classification

Yuan Yao[1], Gian Luca Marcialis[2], Massibmiliano Pontil[1,3], Paolo Frasconi[4], and Fabio Roli[2]

[1] Dept. of Mathematics, City University of Hong Kong mayyao@cityu.edu.hk
[2] DIEE University of Cagliari, Italy {marcialis,roli}@diee.unica.it
[3] DII, University of Siena, Italy mapontil@cityu.edu.hk
[4] DSI, University of Florence, Italy paolo@dsi.unifi.it

Abstract. We present new fingerprint classification algorithms based on two machine learning approaches: support vector machines (SVMs), and recursive neural networks (RNNs). RNNs are trained on a structured representation of the fingerprint image. They are also used to extract a set of distributed features which can be integrated in the SVMs. SVMs are combined with a new error correcting code scheme which, unlike previous systems, can also exploit information contained in ambiguous fingerprint images. Experimental results indicate the benefit of integrating global and structured representations and suggest that SVMs are a 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 [4] and have been used extensively since then. The task is important because classification can be employed as a preliminary step for reducing complexity of database search in the problem of automatic fingerprint matching [5]: 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.

In this paper, we propose new fingerprint classification algorithms based on two machine learning approaches: support vector machines (SVMs), and recursive neural networks (RNNs). SVMs are a relatively new technique for pattern classification and regression that is well-founded in statistical learning theory [8]. One of the main attractions of using SVMs is that they are capable of learning in *sparse, high-dimensional spaces* with very few training examples. RNNs are connectionist architectures designed for solving the supervised learning problem when the instance space is comprised of labeled graphs [2]. This architecture can exploit structural information in the data, which improves the discrimination of certain pairs of classes. RNNs are also used to extract a distributed vectorial representation of the relational graph associated with a fingerprint. This vector is regarded as an additional set of features subsequently used as inputs for the SVM classifier. An important issue in fingerprint classification is the problem of ambiguous examples: some fingerprints are assigned to two classes simultaneously, i.e. they have double labels. In order to address this issue, we designed

an error correcting code [1] scheme of SVM classifiers based on a new type of decoding distance. This approach has two main advantages: (a) It can tolerate the presence of ambiguous fingerprint images in the training set, and (b) It can effectively identify the most difficult fingerprint images in the test set. By rejecting these images the accuracy of the system improves significantly.

The paper is organized as follows: In Section 2 we briefly describe SVM's theory and discuss how to combine SVM classifiers for multi-class classification tasks. Section 3 briefly presents RNNs. In Section 4 we report the experimental results. Section 5 concludes the paper.

2 Support Vector Machines

SVMs [8] perform pattern recognition for two-class problems by determining the separating hyperplane with maximum distance or *margin* 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 Hilbert space $\mathcal{H}$. The mapping $\Phi(\cdot)$ is represented 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: $f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i c_i K(\mathbf{x}_i, \mathbf{x})$, where ℓ is the number of data points, and $c_i \in \{-1, 1\}$ is the class label of training point $\mathbf{x}_i$. Coefficients α_i can be found by solving a quadratic programming problem with linear constraints. An important kernel function is the Gaussian kernel: $K(\mathbf{x},\mathbf{y}) = \exp\left(-\|\mathbf{x}-\mathbf{y}\|/2\sigma^2\right)$, with σ the variance of the gaussian. See [8] for more information on SVMs.

2.1 Multi-class Classification with Error Correcting Codes

Standard approaches to solve q-class problems with SVMs are: (a) One-vs-all: We train q SVMs, each of which separates a single class from all remaining classes [8]; (b) Pairwise: We train $\frac{q(q-1)}{2}$ machines, each of which separates a pair of classes. These two classification schemes 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 (ECCs) [1]. In this case, 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 consist of classes A and T and the second of 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 the $j-$th classifier. ECCs allows a more accurate use of ambiguous examples, since each SVM is in charge of generating one codebit only, 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 an example consider fingerprints with double labels A and T. These examples are discarded by the one-vs-all classifiers

of class A and T, and the pairwise classifier A-vs-T. However they are used to train the classifier AT-vs-RLW. A test point is classified in the class whose row in the coding matrix has minimum distance to the output raw of the classifiers. Let $\mathbf{m}$ be a row of the coding matrix, $\mathbf{f}$ the real output vector of the classifiers, and γ_i the margin of the $i-$th classifier. The simplest and most commonly used distance is the Hamming distance $d(\mathbf{f}, \mathbf{m}) = \sum_{i=1}^{s} 1-\text{sign}(f_i m_i)$. We will also use two other distance measures which take in account the margin of the classifiers: The margin weighted Euclidean distance, $d(\mathbf{m}, \mathbf{f}) = \sum_{i=1}^{s} \gamma_i(1 - f_i m_i)$, and the soft margin distance $d(\mathbf{f}, \mathbf{m}) = \sum_{j=1}^{s} |1 - f_i m_i|_+$. In the later expression, the function $|x|_+$ is equal to x, when $x > 0$, and zero otherwise.

3 Recursive Neural Networks

A RNN [2] is a connectionist model designed to solve supervised learning problems such as classification or regression when the output prediction is conditioned on a hierarchical data structure, like the structural representation of fingerprints of [6]. The input to the network is a labeled direct positional acyclic graph (DPAG) $\boldsymbol{U}$ (see [2] for a definition), where the label $U(v)$ at each vertex v is a real-value feature vector associated with a fingerprint region, as described above. A hidden state vector $X(v) \in R^n$ is associated with each node v. This vector contains a distributed representation of the subgraph dominated by v (i.e. all the vertices that can be reached starting a directed path from v). The state vector is computed by a state transition function which combines the state vectors of v's children with a vector encoding of the label of v:

$$X(v) = f(X(w_1), \cdots, X(w_k), U(v)),$$

where $\{w_1, \cdots, w_k\}$ is the ordered set of v's children. Computation proceeds recursively from the frontier to the supersource (the vertex dominating all other vertices). Transition function f is computed by a multilayer perceptron, which is replicated at each node in the DPAG, sharing weights among replicas. Classification with RNNs is performed by adding an output function g that maps the hidden state vector $X(s)$ associated with the supersource to the class label. The state vector $X(s)$ is a distributed representation of the entire input DPAG and encodes features of the input DPAG deemed to be relevant for discrimination amongst classes. In the subsequent experiments, the components of the state vector at the supersource are thus used as an additional set of features.

4 Experimental Results

Our system was validated the NIST Database 4 [9]. This Database 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). Fingerprints were represented with the structured representation of [6] (22 real value features) as well as with FingerCode features [5] (192 real features).

4.1 Stacked Integration of Flat and Structural Classifiers

In a first set of experiments we investigated the potentialities of the combination of flat and structural methods. We coupled our structural approach with the flat neural network proposed in [5]. In these experiments, we used a "stacked" approach for combining classifiers, with a k-nearest neighbor as the additional classifier for combination [3].

Comparison between Vector-based and Structural Classification. We have trained a multi-layer perceptron (MLP) using the FingerCode feature vector as input. The best test set performance was obtained with 28 hidden units. The overall accuracy is 86% at 1.8% rejection rate. For comparison, the structural approach described in [6] achieves the performance of 71.5% only. This low accuracy is mainly due to the large degree of confusion among L, R and T, while A and W are well discriminated. Afterwards, we analyzed the degree of complementarity between the two above classifiers. To this end, we assessed the performance of an ideal "oracle" that, for each input fingerprint, always selects the best of the two classifiers. Such an oracle provides an overall accuracy of 92.5% at 1.8% rejection rate. This value obviously represents a very tight upper bound for any combination method applied to the two classifiers. However, it points out the potential benefit of combining the flat and structural classifiers.

Combined Flat and Structural Classification. A k-nearest neighbor classifier (with a value of $k = 113$) was used for combining the flat and structural classifiers. Such metaclassifier takes the outputs of the two above classifiers as inputs and provides the final fingerprint classification as output. Such combination attains 87.9% accuracy at 1.8% rejection rate, outperforming the MLP classifier discussed above. This result indicates that accuracy can be improved by exploiting structural information. In particular, we observed that such combination improves the performances related to A and W classes.

4.2 Results with SVMs

We compared the three types of multi-class classification schemes discussed in section 2.1. SVMs were trained using the SVMFu code (`http://five-percent-nation.mit.edu/SvmFu`) on a 550MHz Pentium-II PC. Training on 2000 examples takes about 10s for pairwise classifiers and about 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 with $\sigma = 1$: 88.0% at 1.8% rejection rate. The best polynomial SVM was of degree 3 and achieved a performance of 84.5% only. Then, in the remaining experiments we used only the Gaussian kernel.

Pairwise SVMs. We trained the ten pairwise SVMs. The test set accuracy increases to 88.4%, improving of 2.4% the MLP accuracy reported above.

Error-correction SVM scheme. Three sets of SVM classifiers were used to construct the coding matrix: 5 one-vs-al classifiers, 10 two-vs-three classifiers and 10 pairwise classifiers. The three kinds of decoding distances discussed in

Section 2.1 were compared: (i) Hamming distance: 88.0%, (ii) Margin weighted Euclidean distance: 89.1%, (iii) Soft margin distance: 88.8% (all results are at 1.8% rejection rate). This results confirm the advantage of incorporating the margin of the classifiers inside the decoding distance. We have also trained the ECC of SVMs for the four classes task (classes A and T merged together) using the margin weighted Euclidean distance. The obtained accuracy is of 93.7% at 1.8% rejection rate. For comparison, the accuracy reported in [5] for the sole MPL's is 92.1%, while the cascade of $k-$NN and MLP yields 94.8%. This series of experiments indicate the advantage of the ECC scheme over the first two to the better exploiting information contained in multiple labeled examples.

Analysis of the Margin. We measured the margin and the number of support vectors of each SVM classifier used in our experiments (the training error of each individual classifier was always equal to zero). The number of support vectors 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.

Rejection versus Accuracy. Let d_1 be the minimum distance of the output vector of the classifiers from the coding row, and d_2 the second minimum distance. Rejection can be decided by looking at the difference $\Delta = d_2 - d_1$. A large value of Δ indicates high confidence in classification; when Δ is smaller than a given threshold we reject the data. The rejection rate is controlled by this threshold. Table 1 shows the accuracy-rejection tradeoff obtained in our experiments. Notice that the accuracy of the system increases sharply with the rejection rate. At 20% and 32.5% rejection, the system shows a moderate improvement over the best results in [5].

Table 1. Accuracy vs. rejection rate for the ECC scheme of SVMs (Margin weighted Euclidean decoding) trained on FingerCode features.

Rejection Rate:	1.8%	8.5%	20%	32.5%
5 Classes:	89.1%	90.6%	93.9%	96.2%
4 Classes:	93.7%	95.4%	97.1%	98.4%

4.3 Combining Flat and Structural Features with SVM and ECC

We have trained SVMs on both FingerCode and RNN-extracted features and used the ECC scheme with margin weighted Euclidean decoding. The confusion matrix is summarized in Table 2a. The performance improves to 90.0% at 1.8% rejection rate. If we compare this performance to the performance obtained with FingerCode features only (89.1%), we observe the benefit of integrating global and structural representations. This effect is especially clear in the accuracy vs. rejection rate results. As shown in Table 2b, the accuracy sharply increases with the rejection rate, improving significantly over the results obtained with FingerCode features only (see Table 1).

Table 2. (a): Confusion matrix for the ECC of SVMs (Margin weighted Euclidean decoding) trained on both FingerCode and RNN-extracted features. (b): Accuracy vs. rejection rate for the ECC of SVM classifiers trained on the union of FingerCode and RNN- extracted features.

	W	R	L	A	T
W	366	18	8	2	0
R	5	354	0	7	29
L	6	1	357	2	13
A	0	2	2	396	33
T	1	8	12	48	294

(a)

Rejection Rate:	1.8%	8.5%	20.0%	32.5%
5 Classes:	90.0%	92.2%	95.6%	97.6%
4 Classes:	94.7%	96.6%	98.4%	99.2%

(b)

5 Conclusions

In this paper have studied the combination of flat and structured representations for fingerprint classification. RNNs were used for process this structural representation and to extract a distributed vectorial representation of the fingerprint. This vectorial representation was integrated with other global representations, showing significant improvement over global features only. Experiment were performed on the NIST Database 4. The best performance was obtained with an error correcting code of SVMs. This method can tolerate the presence of ambiguous examples in the training set and shown to be precise to identify difficult test images, then sharply improving the accuracy of the system at a higher rejection rate.

Acknowledgments: We wish to thank R. Cappelli and D. Maltoni for useful discussions and A. Jain for providing us the dataset of preprocessed NIST-4 fingerprints. This work was partially supported by CERG grant No. 9040457.

References

1. T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 1995.
2. P.Frasconi, M.Gori, and A.Sperduti. A general framework for adaptive processing of data structures. IEEE Trans on Neural Networks, 9(5), pp 768-786, 1998.
3. G.Giacinto, F.Roli, L.Bruzzone. Combination of Neural and Statistical Algorithms for Supervised Classification of Remote-Sensing Images. Pattern Recognition Letters, May 2000, 21 (5) pp. 385-397.
4. E.R. Henry. *Classification and Uses of Finger Prints*. Routledge, London, 1900.
5. A.K. Jain, S. Prabhakar, and L. Hong. A multichannel approach to fingerprint classification. *PAMI*, 21 (4):348–359, 1999.
6. G. Marcialis, F Roli, and P. Frasconi. Fingerprint classification by combination of flat and structural approaches. Proc. of AVBPA 2001, June 2001, pp. 241- 246.
7. M. Pontil and A. Verri. Support vector machines for 3-d object recognition. *IEEE Trans. PAMI*, pages 637–646, 1998.
8. V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.
9. C.I. Watson and C.L. Wilson. National Inst. of Standards and Technology, 1992.