ICANN 98: Proceedings of the 8th International Conference on Artificial Neural Networks, Skövde, Sweden, 2–4 September 1998. Perspectives in Neural Computing. Springer, 1998. 755-760.

DOI: http://dx.doi.org/10.1007/978-1-4471-1599-1_116

Automatic neural generalized font identification

A.M. González, J. Dorronsoro, C. Santa Cruz *
Department of Computer Engineering and
Instituto de Ingeniería del Conocimiento
Universidad Autónoma de Madrid, 28049 Madrid, Spain

1 Introduction

Neural methods are gaining a steady acceptance as powerful tools in a variety of pattern detection problems, OCR certainly being one of them. The concrete implementation of these neural OCR systems is of course a well guarded corporate secret, but in broad terms it can be said that in most of the cases, multilayer perceptrons (MLPs) are used. There are several reasons for the MLPs' success. To begin with, they are based in well understood mathematical and statistical principles and there are efficient tools and methodologies for their training and evaluation. Furthermore they have good generalization properties.

Nevertheless, MLPs have also some drawbacks. For instance, their correctness rates over individual characters, while very good from a broad point of view, are not usually good enough for what may be termed massive OCR tasks, involving processing jobs of hundreds of thousands or even millions of documents. Notice that a simple combinatorial argument shows that a fairly small error rate of 0.5 % per character translates into an unacceptable error rate of about 45 % in a ten field document with about 10 caracters per field. Another drawback of MLPs is their relatively long training times, and more so in OCR, where a trainining set for recognition of large alphabets involving capital and lowercase letters, digits and some punctuation marks may well run into one million examples. Moreover, all this training effort can be partially undone if new samples are to be introduced for a better recognition rate.

These considerations would tend to suggest that MLP recognizers have to complemented with other tools for an effective use in massive OCR. A simple way to try to improve individual character recognition rates can be derived from the fact that very often massive OCR deals with printed data. Thus, the characters to be recognized can be assumed to concentrate in a relatively small number of fonts. Of course, to exploit this, a rather precise knowledge of the concrete font set involved is required. Howewer such an a priori knowledge does not usually exist, and the sheer sample sizes in massive OCR make nearly

*With partial support of grant TIC 95–965 of Spain's CICyT.

impossible any manual font labelling of individual characters. In the following section we will briefly describe a general automatic approach based on radial basis function networks to what we may term "generalized font" detection. A strategy for the selection of the correct number of basis functions is discussed next, together with an illustration over a specific example.

2 RBF networks and generalized fonts

We will describe here an unsupervised approach for the identification of the fonts present on a sample of printed versions of a certain character, which is based on the estimation of its probability density. This task falls within the scope of both neural network methods and also classical statistical theory. A common ground between both approaches can be found if gaussian RBF networks are applied. Their very well known transfer function has the form $\sum_1^N w_i g(X, C_i, \Sigma_i)$, where $g(X, C, \Sigma)$ is a general multidimensional gaussian with a certain mean C and covariance matrix Σ. If such a function is to represent a density $p(X, W)$, the normalizing conditions $w_i \geq 0$, $\sum_1^N w_i = 1$, Σ_i positive definite have to be imposed. When doing so, $p(X, W)$ becomes what is called a *finite mixture distribution*. In our case we will use simpler, homogeneous gaussians, assuming that the covariance matrix is of the form $\Sigma = \sigma I$, with I the identity matrix.

These networks have been extensively studied [1]. We will train them using the well known "Expectation–Maximization" (EM) algorithm, which seeks to maximize the log likelihood per single data of a M character sample $\mathcal{L}(W) = \left(\log \Pi_{m=1}^M p(X_m, W)\right)/M = \left(\sum_{m=1}^M \log p(X_m, W)\right)/M$ with respect to the current weight set W (see [2] for more concrete details of EM implementation and [4] for a thorough up to date analysis of EM convergence properties). Let us now discuss how to use this set of ideas to automatic font detection.

In our illustration, images are scanned at a 200 dpi black and white resolution, and once segmented, characters are normalized to a 32 by 32 bit matrix. Data space consists then of the first 40 Discrete Cosine Transform (DCT) coefficients derived from that matrix. Once a particular sample density function has been approximated by a gaussian mixture, each one of its components defines in a natural way an hyperspherical influence region. We thus have an automatically constructed clustering of data space, which in our case can be naturally identified as *generalized fonts*.

Let's briefly explain what we mean. Ordinary typing fonts come in well defined families (courier, helvetica, times roman and so on), characterized by the precise shape and size of each character. However several factors (print quality, scanning effects, noise of various kinds) produce random variations of the originally defined font. In any case, if a gaussian RBF network approximates the sample density, each gaussian "attracts" a certain subset of the sample. It can be thus seen as capturing a general "font" around which randomly varying samples cluster. This approach has several clear advantages in OCR problems. First, classification is straightforward: individual samples are assigned to a con-

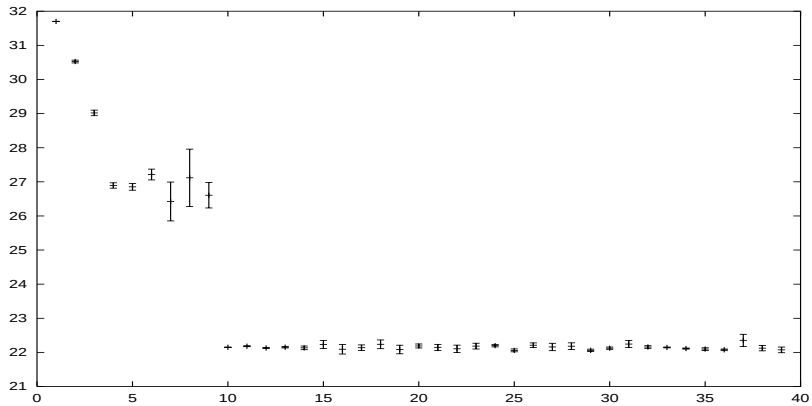

Figure 1: Approximate evolution of $N^2\Phi'(N)$ sample values derived from a 10 center gaussian mixture (the means and error bars given correspond to 10 experimental values).

crete character applying for instance likelihood ratio rules. Second, concrete a priori shape and size knowledge of font parameters is irrelevant: fonts are instead *defined* after training. Third, training times are much shorter since we deal with samples of individual characters and not with whole alphabets. Fourth, the addition of new samples does not imply a complete retraining: their own densities can be computed separately and merged with the previous ones after appropriate normalizations are performed. Fifth, "font removal" can be done in just the same way. In any case, a big question remains open: how to decide the number of such "generalized fonts" to be used or, equivalently, what is the appropriate number of gaussians in the RBF network. We will deal with this issue next, while considering a concrete example of such a generalized font identification.

3 Generalized font detection

Suppose we have a certain number M of samples of a single character C. If all of them come from a well defined number N_0 of fonts, that sample would consist in gaussian noise perturbations of the corresponding "ideal characters" $C_1, \ldots, C_{N_0}$. If a RBF network with more than N_0 gaussians is used to cluster the sample, the network training procedure would ideally concentrate the sample characters around the "real" N_0 fonts, making negligible the contribution of the other $N - N_0$ gaussians. In other words, the sample likelihoods would be constant when $N \geq N_0$ gaussians are used. On the other hand, that likelihood would decrease rather sharply when the number N of gaussians is well below N_0. This is precisely the situation depicted in figure 1. It shows the evolution of a numerical approximation to $N^2\Phi'(N)$ on a sample made up of gaussian noise

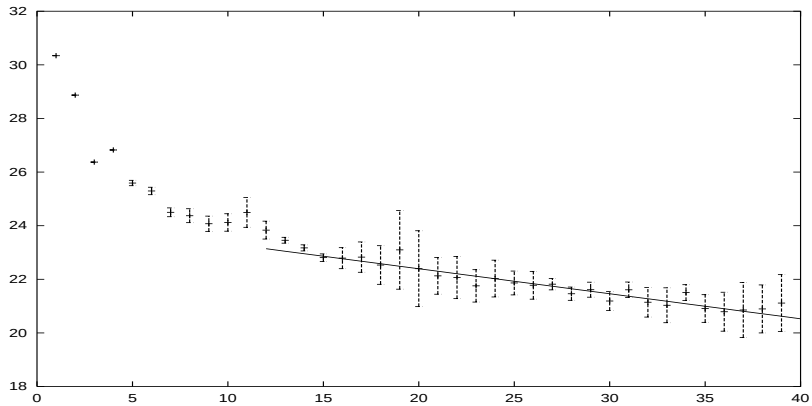

Figure 2: Approximate evolution of $N^2\Phi'(N)$ values obtained from a 6.000 character 3 sample (means and error bars given correspond to 10 experimental values). The straight line gives the best MSE fit in the interval [14,40].

added to 10 "ideal" characters 3, where $\Phi(N) = \mathcal{L}(N)/N$ is what we may call the "normalized" log–likelihood.

$\Phi(N)$ tries to capture the average "contribution" of the gaussians being used to the overall value of $\mathcal{L}(N)$. It also makes the dependence on N of the values of $\mathcal{L}(N)$ more explicit. Here $\mathcal{L}(N)$ denotes the log–likelihood per single data after training of a N gaussian RBF network. $\mathcal{L}$ is computed by the EM algorithm, starting with the centers obtained after the K–means clustering method is applied to randomly chosen initial centers. This seeks faster convergence of the subsequent EM iterations. As it can be seen in the figure, $N^2\Phi'(N)$ remains constantly equal to a value λ while $N \geq N_0 = 10$, but this is no longer true once N is below 10. This is what can be expected in the ideal situation. In fact, if $\mathcal{L}(N)$ is to be constantly equal to a given value $-\lambda = \mathcal{L}(N_0)$ for $N \geq N_0$, $\Phi(N)$ should then be essentially equal to $-\lambda/N$. We would have then $N^2\Phi'(N) = \lambda$.

However, when an actual character sample is used, things are different. Using a sample of 6.000 printed characters 3 obtained in a large scale OCR project [3], the corresponding log likelihoods $\mathcal{L}(N)$ show a slow decrease for large values of N that accelerates for N small. This is not surprising at all since a large number of gaussians may cause models to overfit. This may not happen when all the characters come from random sampling a certain gaussian mixture, as it was the case before, but overfitting is almost certain with the sample considered now. Figure 2 shows the corresponding evolution of $N^2\Phi'(N)$. Instead of constant values up to a given number of gaussians, $N^2\Phi'(N)$ increases first at a relatively constant rate, but this pattern breaks down for small values of N. A natural idea is to try to use these facts to fit a model to the values of $\mathcal{L}(N)$ above N_0. More precisely, we can consider then that $\mathcal{L}(N)$ has the form $\mathcal{L}(N) = -\lambda + \ell(N)$. ℓ is a positive, slowly increasing function with $\ell(N_0) = 0$ that captures the small "improvement" on $\mathcal{L}$ caused by model overfitting for $N \geq N_0$. For $N < N_0$ this

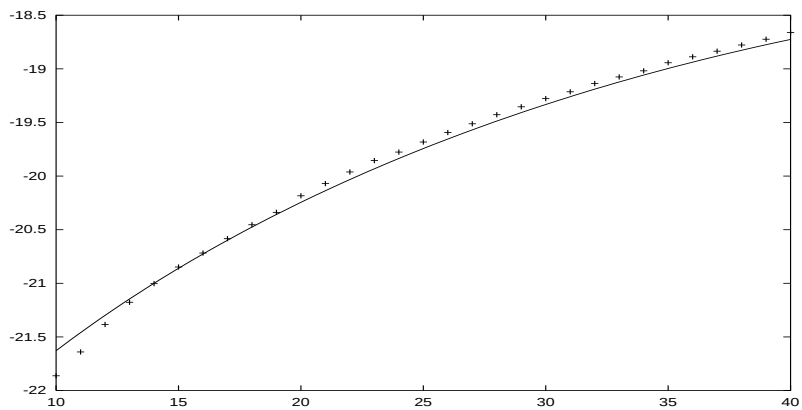

Figure 3: Actual values of $\mathcal{L}(N)$ for a 6.000 character 3 sample and values obtained from a model based on a 14 center gaussian mixture (continuous line).

model is not longer valid and $\mathcal{L}(N)$ decreases markedly faster. Thus, N_0 could be chosen then as the point where the models for $\ell(N)$ fail to apply.

Figure 2 gives support to this approach. It shows that the values $N^2\Phi'(N)$ have a linear structure for N inside intervals starting around 10 and ending at 40, structure that is rapidly being lost for $N < 10$. This suggests to estimate the right number of gaussians (or the number of "generalized fonts") by choosing the best linear fit for $N^2\Phi'(N)$ and deriving a model of $\ell(N)$ and hence of $\mathcal{L}(N)$. The mathematical derivation is quite simple. We assume for the time being that $N \geq N_0$. The model $\mathcal{L}(N) = -\lambda + \ell(N)$ implies then that $\Phi'(N) = [N\mathcal{L}'(N) - \mathcal{L}(N)]/N^2$. Hence, $N^2\Phi'(N) = -\mathcal{L}(N) + N\mathcal{L}'(N) = \lambda - \ell(N) + N\ell'(N)$, which we approximate by a straight line with negative slope $\kappa - \gamma(N - N_0)$. It follows that

$$\Phi(N) = -\frac{\kappa + \gamma N_0}{N} - \gamma \log N + C,$$

where the value of C is obtained making $N = N_0$ in this expression and recalling that $\Phi(N_0) = -\lambda/N_0$. Therefore, the final expression for Φ is

$$\Phi(N) = (\kappa + \gamma N_0)\left(\frac{1}{N_0} - \frac{1}{N}\right) - \gamma \log \frac{N}{N_0} - \frac{\lambda}{N_0},$$

and it follows that the fit for the sample values of $\mathcal{L}(N)$ when $N \geq N_0$ is

$$\hat{\mathcal{L}}(N) = -\lambda + \frac{\kappa + \gamma N_0 - \lambda}{N_0}(N - N_0) - \gamma N \log \frac{N}{N_0}.$$

For our character 3 sample, the best MSE linear model corresponds to $N_0 = 14$, giving a model for $\mathcal{L}$ in the range [14, 40]. The corresponding values of λ, κ and γ are then $\lambda = 21$, $\kappa = 23$ and $\gamma = 0.09$. Figure 3 shows a very close

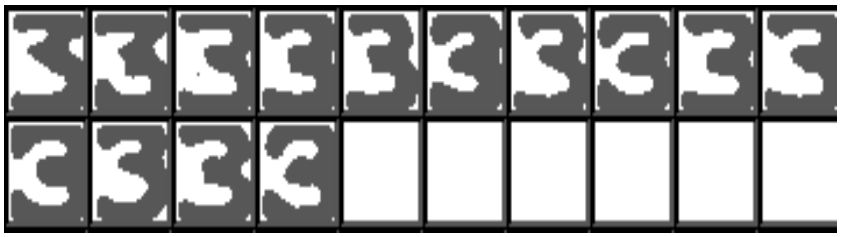

Figure 4: Generalized character 3 fonts deduced from a 6.000 character 3 from a 14 center RBF model.

fit between the model associated to these values and actual sample values of $\mathcal{L}(N)$. Finally, figure 4 shows the characters given by the inverse DCT of the 14 centers of the resulting gaussians. These are the "generalized fonts" associated to the final gaussian mixture.

4 Conclusion

Character recognition by RBF networks is an attractive and natural approach to be used in printed character OCR problems, provided that the right number N_0 of gaussian units (that is, of "generalized fonts") is chosen. We have briefly discussed its advantages and have proposed a simple method to estimate N_0 based on fitting the sample log–likelihoods $\mathcal{L}(N)$ for $N \geq N_0$. Of course, some open questions remain, such as the true nature of these generalized fonts and, more importantly, their recognition advantages against other RBF classifiers. These are currently under study. If succesful, they could offer a complementary and competitive alternative to pure MLP recognizers.

References

[1] F. Girosi, M. Jones and T. Poggio, "Regularization Theory and Neural Networks Architectures", Neural Computation **7** (1995), 219–269.

[2] R.A. Redner, H.F. Walker, "Mixture densities, maximum likelihood and the EM algorithm", SIAM Review **26** (1984), 195–239.

[3] A. Sierra, C. Santa Cruz, V. López, G. Fractman, J. Dorronsoro, C. Aguirre, J.M. Soto, A. Medina, R. López, "Neural networks in large scale bank effect recognition", Procs. SNN 1995 Symposium on Neural Networks, B. Kappen, S. Gielen (eds), 393–396, Springer Verlag, 1995.

[4] L. Xu, M.I. Jordan, "On convergence properties of the EM algorithm for gausian mixtures", Neural Computation **7** (1995), 129–151.