

Applying Probabilistic Neural Networks to the Multifont Recognition Problem with Large Training Set

Marcin Paprzycki, Sean Bowers, Aaron Costeines

Department of Computer Science and Statistics

University of Southern Mississippi

Hattiesburg, MS 39406-5106

{m.paprzycki, sean.bowers, aaron.costeines}@usm.edu

Abstract

Neural networks are very often applied to the pattern recognition problem. In 1990 D. Specht introduced a special class of Probabilistic Neural Networks which were unnoticed in the computational practice due to their extremely large computer memory requirement. In this note we present and discuss results of experiments assessing the usability of Probabilistic Neural Networks to the multifont recognition problem for large size of the training set.

Introduction

Since their re-introduction into the computational practice (Rumelhardt et. al. 1989) Neural Networks (NNs) have been successfully applied to a number of pattern recognition problems (see for instance (Looney 1997) and the references collected there). In 1989-90 D. Specht introduced a new class of NNs based on a different activation function modeling Bayesian strategy for estimation of non-parametric estimators for probability density functions (Specht 1989, 1990a, 1990b). Until recently, Probabilistic Neural Networks (PNNs) were not popular as they require one node in the hidden layer for each input element in the training set and thus consume very large amounts of memory. Computers with memory this large were not easily available in the early 1990's. (It is worth mentioning that Specht recognized this problem and proposed a simplified PNN (Specht 1990b) that required substantially reduced computer resources.) Only now, with computer memory being very cheap and easily available, we observe a resurgence of interest in the PNNs. Major vendors of NN software either already have a PNN module in their NN packages (Ward Systems Inc. in the NeuroShell 2.0 (Ward 1998) and Trajan Software Inc. in the Trajan 3.0 (Trajan 1998)) or plan to add such a module shortly (Mathworks in the Matlab Neural Networks Package (Mathworks 1998)). This led us to investigate the potential of PNNs applied to pattern recognition. In earlier work we have reported the results of experiments devoted to investigate the performance of PNNs applied to the recognition of computer generated characters (Fairchild 1998). We also compared the performance of PNNs with that of Ward-nets (a sophisticated proprietary version of the backpropagation network) (Paprzycki 1998). The results indicated that PNNs have a definite potential for

becoming an effective tool for solving the printed character recognition problem and that their performance was quite favorable in comparison with Ward-nets. At the same time, the discussions during both conferences led us to believe that a much broader investigation was necessary to properly assess the potential of PNNs. The primary direction of such an investigation was to involve much larger training sets.

The aim of this note is to report on such an attempt. We have applied the NeuroShell based implementation of PNNs to the multifont pattern recognition problem for a relatively large size of the training set. The problem studied was a slight simplification of a general problem of recognition of computer-printed characters (only the 26 upper case Latin letters are used). It was assumed that a computer generated printed character is to be scanned and digitized and presented to a computer to be recognized. It was also assumed that one letter is presented at a time and that each image presented does represent a letter.

The remaining part of the paper is organized as follows. In the next section we briefly sketch the theory behind Probabilistic Neural Networks. We follow it with a description of how the data was generated. Finally, we describe the experiments performed and discuss their results.

Probabilistic Neural Networks

We only briefly present the ideas behind the PNNs. The sketch is based on (Specht 1989). For more details one should consult all three original papers by Specht. PNNs are based on an application of a classification theory based on the minimization of the "expected risk" function (such strategies are typically called "Bayesian" and can be applied to problems with multiple categories as long as the decision boundaries remain relatively smooth). When considering a state of nature characterized by a two-category situation one may need to decide which of the situations occurs and this decision is to be based on a series of measurements. The decision can be expressed in terms of Bayesian decision rules and will depend on two probability density functions $f_A(X)$ and $f_B(X)$ (where X is the measurement vector) corresponding to the two situations. The key to the classification process is the ability to estimate the probability density functions (PDFs).

For NN applications this means that the PDFs have to be estimated on the basis of the training patterns (which are the only source of available information about the data). In 1962, Parzen introduced a class of estimators which asymptotically approach the real density as long as it is smooth and continuous (Parzen 1962). The estimator used in his study was:

$$f_A(x) = \frac{1}{(2\pi)^{p/2} \sigma^p} \frac{1}{m} \sum_{i=1}^m \exp\left[-\frac{(X - X_{ai})^t (X - X_{ai})}{2\sigma^2}\right]$$

where i = pattern number and X_{ai} = i th training pattern from category A and σ = smoothing factor. This is also the estimator that, according to Specht, should be used in the PNN design.

Observe that the above equation requires that all information from the training set must be stored and used during testing. In addition, the time necessary to process the data when the network is already trained is also directly proportional to the size of the training set.

Even though the above limitations were considered severe in the early 1990's they became less so nowadays. We were able to experiment with training sets of size 2574 on a 400 MHz Pentium II based system.

Data generation

In the earlier papers (Fairchild 1989, Paprzycki 1989) we presented results based on 5 fonts represented as 10x10 bit-maps including shifted and fuzzy data and on 14 fonts represented as 14x14 bitmaps and their fuzzy representations, respectively. Here, as suggested above, we further expanded the data set used for experimentation. Using a Delphi program, provided by Daniel Hellsson of Unseen Technology, we have generated a set of bit-maps of size 20x20 for the capital letters from 124 different fonts available from a collection of fonts prepared by Digital Systems Research (Digital Systems Research 1993). Even though it is suggested that the CD contained more than 1000 fonts we were able only to find 124 base-fonts that were different (excluding bold, italic and fonts that varied only in name). In all cases letters were centered in the bit-map and 0's filled the "borders" of the map (so the effective letter image was at most 18x18). The point size was selected (by the Delphi environment) in such a way that all fonts fit into the 18x18 bitmap without any information loss. In the process of generating the data we have found that this may not have been the best choice as this led to very different letter sizes. Letters of some fonts filled almost the entire space, while letters of some fonts were rather small and left a lot of "white-space" around them. This has two counteracting effects. This adds variability to the training and testing data that mirrors the realistic situation that we try to model. At the same time this introduces a new variable to the experimental setup,

one over which we do not exert a proper control. We decided to complete our experiments with the data as generated and investigate the effects of size separately.

This data has been preprocessed to match the requirements of the NeuroShell environment (using software developed specifically for this project). In the PNN setup we used (20*20 = 400) input nodes (each node corresponding to one element of the input vector) and 26 output nodes (corresponding to the 26 letters of the alphabet). The number of nodes in the hidden layer varied depending on the size of the input data (there is one node in the hidden layer for each letter in the input data set).

The results of our earlier experiments suggested that the capability of recognizing letters from an unknown font improved as the number of different fonts used in the training process increases. This result naturally follows the intuition that the more difference in the training data, the better the ability of the network to generalize and thus deal with the unknown test data. The major aim of the present study was to follow this intuition and to investigate the performance characteristics of the PNNs for the increasing number of fonts in the training set. To achieve this goal we have divided our 124 fonts into four groups; three groups of 33 fonts (to be used mainly for training purposes) and one group of 25 fonts (to be used mainly for testing). We have divided the data into groups alphabetically (based on the font names), but this should not have any effect on the results.

Experimental Results

In the first series of experiments we have used each of the larger font-groups as the training set (a PNN with 33x26=858 nodes in the hidden layer). First we applied the network to the training set itself and then to the remaining font-groups. The results are summarized in Table 1.

Training	Testing	Correct Answers
1	1	857
1	2	421
1	3	273
1	4	316
2	1	339
2	2	839
2	3	284
2	4	315
3	1	338
3	2	441
3	3	856
3	4	440

Table 1. Performance for single-group training

The results are consistent. When the training set itself is presented to the network a very high reliability of the recognition occurs (this is in agreement with the earlier

results reported in Paprzycki (1998)). When a different group of fonts is presented the recognition is not good at all. It varies from 31% for group 3 presented to the network trained with group 1 to 51% for group 2 presented to the network trained with group 3. For the test group (group 4 with 25 fonts) the recognition varies from 48% for the network trained with group 2 to 67% for the network trained with group 3. Overall, the results indicate that group 3 contains a collection of fonts that are better trainers than the fonts combined into the remaining groups.

It should be pointed out that the quality of recognition depends on the value of the smoothing factor σ . In the earlier results (for smaller number of smaller bitmaps) a relatively large interval (e.g. 1.2-2.5) of potential smoothing factors led to the same outcome (Paprzycki 1998). In the current experiments, the results are highly sensitive to the smoothing factor. A change of order 0.01 may result in one of the letters being recognized incorrectly or not recognized at all. In addition, for each network and for each testing set the optimal value differed. While a “good” result was typically obtained for $\sigma=1.4$, the best result could occur for σ anywhere between 1.25 and 1.55. Due to the timing constraints (each test took about 1 minute to complete) we have reported only approximately the best results (it is thus possible that slightly better outcomes of the experiments exist). This observation has a rather peculiar effect on the potential of applying PNNs to the practical problems. Typically it is not possible to know the correct value of the smoothing factor a priori and thus calibrate the network to deal correctly with all possible input patterns.

In the second series of experiments we combined the larger groups and used them as the training set (PNN with $66 \times 26 = 1716$ nodes in the hidden layer) and applied them to themselves and to the remaining fonts. The results are summarized in Table 2. As discussed above, only the “approximately-best” results are reported.

Training	Testing	Correct Answers
12	12	1679
12	3	311
12	4	340
13	13	1695
13	2	484
13	4	340
23	1	370
23	23	1675
23	4	332

Table 2. Performance for double-group training

The results are very similar to these reported in Table 1, and thus rather disappointing. When the training set itself is shown to the network its recognition is approximately 97%. However the recognition of the letters belonging to the unknown fonts is only between 36% and 50%. It can

be postulated that an increase in variability of the training data has no positive effect on the capabilities of the PNN. The data seems to suggest that we have reached a limit of what a PNN can recognize and addition of further elements into the training set does not help.

To investigate this effect further, in the next series of experiments, we have combined the data into training sets consisting of 3 groups (PNNs with $99 \times 26 = 2574$ or $91 \times 26 = 2366$ nodes in the hidden layer). The results are summarized in Table 3.

Training	Testing	Correct Answers
123	123	2507
123	4	354
124	124	2322
124	3	318
134	134	2333
134	2	485
234	234	2309
234	1	388

Table 3. Performance for triple-group training

The results remain consistent with the previously reported. When the training set itself is shown to the network the recognition level remains at about 97%. However, when the letters representing unseen fonts are presented to the network the recognition level remains consistent, between 37% and 54%. As for the training sets based on two groups of fonts, the added variability of the training data did not lead to an improvement in the ability of the network to generalize.

It is also worth mentioning that the initial unpopularity of the PNNs due to their hunger for computer resources should not be a problem with modern PCs. A typical training time for the largest networks used was not longer than 10 minutes and the time of testing the fully trained network for 2574 letters was also not longer than a few minutes. In addition, large part of this time was used in loading data to memory, a process that likely can be implemented in a more efficient way by taking a full advantage of the underlying hardware.

Concluding Remarks

In this note we have discussed the performance of Probabilistic Neural Networks applied to the multifont recognition problem for relatively large training sets. We have found that the performance is rather disappointing. Each time PNNs were able to recognize the set used for training purposes very well. However, when confronted with unknown data they reached only up to 54% accuracy. Furthermore, the accuracy did not improve as the size of the training set increased. There can be a number of potential reasons for such behavior. First, the way that the data was prepared with varying sizes of letters in the

bitmap may have influenced the performance. Second, PNNs may not be very well suited to the multifont recognition problem (e.g. due to the fact that the probability densities are not smooth and continuous, or the decision surfaces are too complex). Third, a different estimator function could possibly provide a better match for the probability space spanned by the letters. Fourth, (even though we have no reason to suspect this) there could be an error in the implementation of PNN's available from Ward Systems. This could be an error that manifests itself only for the very large training sets used in our experiments and for that reason remained undetected.

We plan to continue our experiments to eliminate at least some of the possibilities described above. We will generate a new data set where the font-size will be selected in such a way to fill in the available 18x18 box to the maximum and rerun our experiments. We will also apply the PNN module available in the Trajan environment to the problem and compare the results. We will then turn our attention to other NN architectures available in the NeuroShell and Trajan environments. We will start our investigations with the Ward Nets and the Kohonen self-organizing network.

References

- Digital Systems Research and Page Tech. 1993. CD with 1500 fonts for Windows based computers.
- Fairchild A.N. and. Paprzycki, M. 1998. Performance Evaluation of Neural Networks Applied to the Pattern Recognition. In Proceedings of the 14th CAM, 161-172. Edmond: University of Central Oklahoma Press.
- Looney, C.G. 1997. *Pattern Recognition Using Neural Networks*. New York: Oxford University Press.
- Mathworks. 1998.
<http://www.mathworks.com/products/neuralnet/>
- Paprzycki, M. and Bowers, S. 1998. Applying Probabilistic Neural Networks to the Multifont Recognition Problem (submitted for publication)
- Parzen, E. 1962 On Estimation of a Probability Density Function and Mode. *Ann. Math. Stat.* 33:1065-1076
- Rumelhardt, D.E., McClelland D.L., and the PDCP Research Group. 1986. *Parallel Distributed Processing*. Cambridge: MIT Press.
- Specht, D. 1989. Probabilistic Neural Networks for Classification, Mapping, or Associative Memory. In Proceedings IEEE Conference on Neural Networks, vol. 1, 525-532. Los Alamitos: IEEE Press.
- Specht, D. 1990a. Probabilistic Neural Networks. *Neural Networks* 3:109-118.
- Specht, D. 1990b. Probabilistic Neural Networks and the Polynomial Adaline as Complementary Techniques for Classification. *IEEE Transactions on Neural Networks* 1(1):111-121.
- Trajan. 1998. <http://www.trajan-software.deman.co.uk>
- Ward. 1998. <http://www.wardsystems.com>