False Alarm Reduction Techniques for ASR and OCR

Yaakov Stein
Efrat Future Technology Ltd.
110 Yigal Alon St.
Tel Aviv 67891, Israel

Abstract

Techniques which may reduce the false alarm rate of *sliding networks* employed in ASR (automatic speech recognition) and OCR (optical character recognition) problems are reviewed. Such techniques include the use of hypersphere classifiers, training out of negatives, addition of garbage units, and exploitation of multiple networks. The combination of several techniques may reduce the false alarm rate to tolerable levels.

1 Introduction

A pattern classifier inputs N *features*, and outputs a classification of the input as belonging to one of C *classes*. The most common implementation of a neural network classifier consists of N input neurons, and C outputs encoded via the so called *selector* representation, wherein there are C output neurons, each trained to be active (+1) if the input corresponds to its class and inactive (0) otherwise. Most of the neural network pattern recognition literature deals with 'closed-set' problems, ie. problems wherein the input to the network always belongs to one of the classes to be recognized. Many real-world problems are of the 'open-set' variety, for which inputs may not belong to any of the classes. In such problems we have two input types, which we shall call *positives* and *negatives*, and three types of errors, deletions (classifying a positive as a negative), misclassifications (classifying a positive as a incorrect positive) and false alarms (classifying a negative as a positive).

Two examples of real-world open-set problems are provided by continuous speech recognition and cursive script reading. When classifying phonemes in continuous speech or characters in cursive script, we have no general method of faultless a priori segmentation, and may elect to require the network to perform this function. Such a network slides a window along the continuous input attempting to simultaneously locate and classify positives. An identification is signalled when some output unit's activity passes a threshold, and the classification is according to the output unit with the highest activity. When such a network identifies a region *between* two phonemes or characters, a false alarm results. As the network slides over a properly classified positive input, the appropriate output unit's activity rises from close to zero to some maximum value, and then tends back to zero. This *tuning curve* will have some characteristic width, and the optimal identification locality is at its maximum.

Classifiers with adjustable thresholds can trade off different types of error, eg. decreasing the threshold may decrease the number of deletions at the expense of increased false alarm rate. Since one can never recover from deletions, while misclassifications and false alarms can possibly be corrected by appropriate post-processing, often one sets the threshold such that deletions are relatively improbable. The present paper reviews techniques for reducing *network's* false alarm rate in such cases. The *system's* false alarm rate may be further reduced by application-specific post-processing, which is beyond the scope of the present review.

Specification of the performance of a sliding network requires more than a simple probability of error figure. In the *word spotting* literature, the false alarm rates at several reference misclassification probabilities are usually given. Often the entire false alarm vs. misclassification curve is displayed. With such specifications it is not in general possible to uniquely declare one classifier to be superior to another. A single reference index is given by the Levenshtein distance, defined as the minimum number of insertions, deletions and substitutions required to correct the network output. This index, which is prevalent in the OCR literature, can be easily generalized to take application-specific cost functions into account, but requires dynamic programming to evaluate.

2 Hypersphere Classifiers

Hypersphere classifiers, including RBF and RCE architectures, have modest training and classification times, excellent false alarm rejection rates and can attain Bayes error with large training sets. Hyperplane classifiers, such as the multilayer perceptron (MLP), may require longer training times but classify faster. They also can attain minimal classification error, but have virtually no inherent false alarm rejection capabilities [8]. While hypersphere classifiers endeavor to capture the class probability distributions, hyperplane classifiers only try to find inter-region boundaries.

Hyperplane classifiers have proven to be more popular than hypersphere ones in practice, for several reasons. The simplest hyperplane classifier, the perceptron, can be trained in a finite number of steps, at least when a separating hyperplane exists. The multilayer perceptron can create arbitrary decision regions, and tends to have somewhat lower misclassification rate than hypersphere classifiers for small training sets, due to the more efficient use of examples during training. The most popular MLP training algorithms are variants of backpropagation, which do not usually converge to a global optimum, but are straightforward to implement. RBF training methods either call for a clustering stage, or arbitrarily chose a small number of input examples as bases.

If a large enough training set is available hypersphere classifiers may indeed be preferable for ASR and OCR applications; several commercially available OCR engines use hypersphere classifiers. Hybrid systems, which can select to use hyperplanar or hyperspherical boundaries [8], may prove to be the best alternative. In the next sections we will discuss the reduction of false alarm rate of hyperplanar classifiers.

3 Training Out Negatives

Multilayer perceptrons do not efficiently reject negatives, simply because they have not been trained to do so. One popular remedy calls for adding negatives to the training set, requiring all the output neurons to be inactive (see eg. [2]). The question arises as to which negative examples to use — there are often an infinite number of possible ones. The inclusion of too large a percentage of negatives in the training set, even if this percentage is below the percentage in actual input, is detrimental. This is due to the fact that the network quickly learns that it is usually advantageous to remain inactive, and falls into a local minimum corresponding to the all zero output.

One method of overcoming this defect is to include a controlled percentage of random negatives in the training set. However, it usually turns out that a large number of these negatives do not significantly influence the output units in any case, and do not contribute to the efficient rejection of other false alarms.

We have found a two-pass technique which we call *training on errors* to be most effective. First one trains on positive examples only, until some (weak) convergence criterion is reached. One then reparses the input with the newly trained network and notes the activity of negatives. A controlled percentage of the most significant false alarms is added to the training set, and training is continued. This process can be repeated, once the training negatives are sufficiently well rejected.

One often finds convergence problems while training out negatives in continuous-input problems. Networks with sufficient degrees of freedom do not converge due to contradictory demands of the training set. The training out of negatives described so far is further observed to decrease the width of the tuning curves of positives. The two phenomena are due to the suppression of the positive outputs which surround the undesired negative. For example, let us assume that a strong X false alarm was detected between the positives A and B, such that the network responds AXB. When this X is added as a negative example, all positives, including A and B are penalized, resulting in sharpening of A and B's tuning curves. In addition, this desired rejection of X contradicts the desired shift invariance of the network to A and B.

In order to avoid the aforementioned maladies, we have employed a technique which we call *getting along with neighbors*. The idea is that when a negative is presented during training, all positives are suppressed *except* for those surrounding the negative. These positives are not strengthened, but not penalized either. We have found that this technique ameliorates the situation, accelerates convergence and strengthens translation invariance.

4 Garbage Units

Although training-out of negatives often significantly reduces the false alarm rate (from our experience often by a factor of two to three), in many cases this is still insufficient. An idea which comes to mind is to add a *garbage unit* to the output layer, which is trained to fire when any invalid input is presented. There are thus $C + 1$ output units, and the output activity threshold can now be set lower, since the criterion for rejection is high activity of this special unit, rather than low activities of C outputs.

We have found severe convergence problems in attempting to naïvely apply this technique. Were the negative inputs to be concentrated in a small number of areas in input space, then the complexity of training this augmented network would be similar to that of the regular problem. However, if the negatives are more multimodal in structure than a typical positive input, then this special unit needs to learn a more complex task, which may be too difficult for a single output unit. We are thus led to the idea of employing multiple garbage units.

The obvious question is how to separate the different types of negatives for training. One could, before supervised training, apply some clustering algorithm, classical or neural. However it seems more natural to allow the network to self-organize. One can also envision dynamic addition and deletion of garbage units during such training. We have utilized a simple combined supervised/unsupervised algorithm. For each training negative presented, all C regular outputs are depressed according to backprop, and among the garbage units we strengthen only that unit which responds best to the input. This method is reminiscent of many *follow the leader* clustering algorithms, such Kohonen's self-organizing map.

It often occurs that such networks, even when they do converge, do not significantly capture the structure of the false alarms in input space. In order to study this structure it is useful to employ Sammon's *nonlinear mapping* [7] or NLM. This maps the N dimensional training set onto a two dimensional plane in such way that the matrix of distances between pairs of training examples is minimally disturbed. The required transformation can be found by gradient descent of an energy function proportional to the total distortion. If it turns out that the negatives are homogeneously distributed among the positives, then garbage units are of no avail, and hyperspherical classifiers should be considered.

5 Combining Multiple Networks

Let us train several MLP networks on the same training set. These networks may even be instantiations of the same basic architecture, but are trained differently, eg. using different initial couplings, different order of pattern presentation, different training algorithms, etc. Assuming the training process succeeds, these networks agree on the training set, and should behave similarly on patterns similar to those in the training set. However, the networks will probably disagree as to patterns unlike those in the training set, since there were no constraints placed on these during the training process. Thus it would seem that by combining the opinions of several networks the false alarm rate may be drastically reduced without significantly reducing the classification rate, (perhaps even improving it).

Geometrically speaking, the training set imposes restrictions on the placing of each classifier's hyperplanes. The negative examples will natural fall randomly into domains corresponding to some class, but unless they are sufficiently similar to positive examples, there should be decorrelation between the domains into which they fall. When comparing identifications, the different networks should respond similarly to positive examples, but will tend to disagree regarding the negative ones. This behavior should allow one to differentiate between negatives and positives, thus effectively rejecting false alarms.

The fact that different networks can be combined in a general fashion is a major advantage of the neural network approach. In contrast expert systems, which induce tree structured decision mechanisms, can not in general be combined. Thus the expertice of several experts can not be used simultaneously. The application of multiple networks has been presented before [3, 5, 6, 10], but for the purpose of decreasing the misclassification fraction on closed-set problems.

There are several possible strategies for combining K networks. We may require a simple majority vote, discarding as not belonging to any class any point not similarly classified by at least $K/2$ of the networks. We may require more generally that at least βK networks agree ($\frac{1}{2} \leq \beta \leq 1$), with $\beta = \frac{1}{2}$ corresponding to simple majority and $\beta = 1$ to unanimous vote. We will refer to this strategy as *generalized majority*. Another method of combining the individual networks is to sum (or average) the responses of the corresponding output units, and require this sum to pass some threshold. We call this method the *average* decision mechanism. It is equivalent to building a single network with K times as many hidden units and an additional additive layer; however the training procedure is different from that of a single large network. There are yet further possibilities, such as deciding on the basis of that single network which responds most strongly, or training a network to decide among competing networks (the *expert network* strategy) [4].

Over the past two years we have found empirically that the false alarm rate of cursive script and continuous speech systems can be significantly reduced by combining the outputs of several multilayer perceptrons. In order to better understand the basic mechanisms involved, we have studied several idealized problems. First for simple cases which can be solved analytically we have determined the conditions under which the combination of several networks improves classification, and/or reduces false alarm rate [9]. Secondly, for the simple perceptron, we have calculated the difference between the average and majority strategies. Finally, we have performed simulations on an archetypal classification problem (which will be reported elsewhere). The conclusion from these studies is that the use of multiple networks may reduce the false alarm rate of MLP networks to hypersphere levels.

In real world ASR and OCR problems, we have found that combining networks significantly reduces the false alarm rate, even when the individual networks have had negatives trained out using the getting along with neighbors technique. This further reduction can be a factor of two to three compared with the separate networks. The final system will have longer computation times (linear in the number of networks employed) but may have satisfactory performance.

Acknowledgements The techniques reviewed here were developed and tested over the past three years, by the neural network group at Efrat Future Technology, Ltd. This group consists of the author, Ruthi Aloni-Lavi, Coby Metzger, Adir Even and Gaddy Getz.

References

[1] Aloni-Lavi R., Metzger Y. & Stein Y. 1992. *A BP Variant with Improved Convergence and Generalization.* IJCNN 1992 (Baltimore) I-932 - I-937.

[2] Bromley J. and Denker J.S. 1993. *Improving Rejection Performance on Handwritten Digits by Training with Rubbish.* Neural Computation **5**, 367-370.

[3] Hansen L.K. & Salamon P. 1990. *Neural Network Ensembles.* IEEE Trans. on Pattern Analysis and Machine Intelligence **12**, 993-1001.

[4] Jacobs R.A., Jordan M.I., Nowlan S.J. and Hinton G.E. 1991. *Adaptive Mixtures of Local Experts.* Neural Computation **3**, 79-87.

[5] Lincoln W.P. and Skrzypek J. 1990. *Synergy of Clustering Multiple Back Propagation Networks.* Advances in Neural Information Processing Systems **2**, 650-657.

[6] Pearlmutter B. and Rosenfeld R. 1991. *Chaitin-Kolmogorov Complexity and Generalization in Neural Networks.* Advances in Neural Information Processing Systems **3**, 925-931.

[7] Sammon J.W. (1969) *A Nonlinear Mapping for Data Structure Analysis.* IEEE Trans. on Computers **C-18**, 401-409.

[8] Stein Y. 1993. *Hyperplane Training of a Hypersphere Classifier.* these proceedings.

[9] Stein Y. and Aloni-Lavi R. 1993. *Improving Performance via Multiple Classifier Voting.* to appear.

[10] Watkin T. 1993. *Optimal Learning with a Neural Network.* Europhysics Letters **21**, 871-875.