Selecting a Critical Subset of Given Examples during Learning

Byoung-Tak Zhang
German National Research Center for Computer Science (GMD)
D-53757 Sankt Augustin, Germany
E-mail: zhang@gmd.de

1 Introduction

The quality of training examples is one of the most important factors for effective and efficient training of neural networks, provided the network architecture and the weight modification rule are fixed. Earlier work [4] shows that the perceptron learning can be significantly accelerated by utilizing specific instances. For classification problems, these examples are the border patterns, i.e. the patterns that lie closest to the separating hyperplane. For real applications, however, it is difficult to estimate *in advance* the exact utility of examples, since the ultimate criticality of examples depends not only on the task, but also on the network configuration.

In this paper we describe an example selection method that finds a critical subset of given examples which achieves as good a performance as the whole set. It is efficient because the selection is performed in one trial *while* the learning proceeds. We select examples incrementally, an idea introduced first in [8] and developed further in [5]. The incremental selection is robust against possible errors since the inclusion of some non-critical examples at some stage can be automatically corrected at subsequent stages.

The method differs from [1, 2] in that we are using a computationally inexpensive squared error selection criterion instead of computing second derivatives. Our method is applicable to both classification and function approximation problems of high-dimensional inputs and outputs.

In the remainder of this paper we summarize the algorithm and show some results to demonstrate the effectiveness and generality of the incremental selection method.

2 Algorithm

The selection proceeds incrementally during learning. Given a data set B, the training set D is initialized to contain a small number of examples randomly chosen from B. The rest of B is used as a candidate set C. During learning, D is increased by selecting examples from C. We use a superscript s, as in $D^{(s)}$, to denote the sth training set. The weights of the network are initialized randomly with values from the interval $-\omega \leq w_{ij} \leq +\omega$.

The initial network is trained with the training set $D^{(0)}$. The trained network is used to expand the training set for the next generation $D^{(1)}$ which is again used to find and train the network. Training and selection is repeated until an acceptable generalization is achieved on the candidate data set. Notice that for each s the following conditions are always satisfied

$$D^{(s)} \cup C^{(s)} = B, \quad D^{(s)} \cap C^{(s)} = \emptyset, \quad \text{and} \quad D^{(s)} \subset D^{(s+1)}. \tag{1}$$

In the training phase, the weights are updated using the examples in the training set. If we denote by $\mathbf{w}^{(s,t)}$ the weight vector of the network for the t-th sweep through the training set $D^{(s)}$, the weights are modified by

$$\mathbf{w}^{(s,t+1)} \;=\; \mathbf{w}^{(s,t)} - \epsilon \nabla E_s|_{\mathbf{w}=\mathbf{w}^{(s,t)}} + \eta \Delta \mathbf{w}^{(s,t-1)} \tag{2}$$

where E_s is the total sum of the errors for $D^{(s)}$

$$E_s = E(D^{(s)}|\mathbf{w}^{(s,t)}) = \sum_{m=1}^{N_s} \left(\mathbf{y}_m - f(\mathbf{x}_m; \mathbf{w}^{(s,t)})\right)^2 \tag{3}$$

and the error gradient $\nabla E_s|_{\mathbf{w}=\mathbf{w}^{(s,t)}}$ is approximated by a back-propagation procedure [3] or possibly by any other weight modification rule. ϵ and η are the step size and the momentum factor, respectively.

At every Δt epochs we check the convergence of the error minimization. If the total error for the current training set is reduced to a specified error tolerance level,

$$E(D^{(s)}|\mathbf{w}^{(s,t)}) \;\leq \frac{1}{\tau}\{(I+1)\cdot H + (H+1)\cdot O\} \tag{4}$$

the training phase terminates and the example selection phase follows. Here I, O and H_g are the number of input, output and hidden units of the network. The constant τ determines the error sensitivity per connection.

In the selection phase, the generalization accuracy of the current network is tested on the original data, $B = D^{(s)} \cup C^{(s)}$:

$$G_s = \frac{1}{N} \sum_{(\mathbf{x}_q, \mathbf{y}_q) \in B} \Theta\left(\mathbf{y}_q, f\left(\mathbf{x}_q; \mathbf{w}^{(s,t)}\right)\right) \tag{5}$$

where the function $\Theta(\cdot,\cdot)$ is some measure of correctness. If G_s satisfies the desired performance level ℓ, say 99%, then the entire algorithm stops. If $C^{(s)}$ is empty, the algorithm also stops. Notice that halting with a nonempty $C^{(s)}$ means the network has generalized correctly to the candidate data set. Otherwise, the criticality with respect to the current model $\mathbf{w}$ is computed.

The criticality $(\mathbf{x}_c, \mathbf{y}_c)$ of an example is defined as

$$e_{\mathbf{w}}(\mathbf{x}_c) = \frac{1}{\dim(\mathbf{y}_c)} \left(\mathbf{y}_m - f(\mathbf{x}_m; \mathbf{w}^{(s,t)})\right)^2 \tag{6}$$

which has a value $0 \leq e_{\mathbf{w}}(\mathbf{x}_c) \leq 1$ if the sigmoid activation function is used at the output layer. Then the training set is increased by selecting λ candidate examples, $(\mathbf{x}_c, \mathbf{y}_c)$, which are most critical:

$$D^{(s+1)} = D^{(s)} \cup \{(\mathbf{x}_c, \mathbf{y}_c)\}, \quad \text{and} \quad C^{(s+1)} = C^{(s)} - \{(\mathbf{x}_c, \mathbf{y}_c)\}. \tag{7}$$

In case of $|C^{(s)}| < \lambda$, all the remaining candidate examples are selected into $D^{(s+1)}$. Using the expanded training set, the next cycle of training and selection is done.

3 Performance

The performance of the algorithm is demonstrated on the following function:

$$y(x_1, x_2) = \begin{cases} 1 & \text{if } (x_1 < 0.5 \text{ and } x_2 > 0.5) \text{ or } (x_1 > 0.5 \text{ and } x_2 < 0.5) \\ 0 & \text{otherwise,} \end{cases}$$

where $0.1 \leq x_1, x_2 \leq 0.9$. This function, called four-quadrant, is a real-input variant of the XOR problem. Due to its graphical characteristics the problem allows an easy analysis of the learning results.

We used a feed-forward network with four hidden units and a total of 400 (20×20 resolution) examples. The training set was initialized with four seed examples at the four corners. In each selection step, additional four examples ($\lambda = 4$) were added to the existing training set. Figure 1 shows the graphes of the approximated function and the corresponding training points used at $s = 0$, 5, and 10.

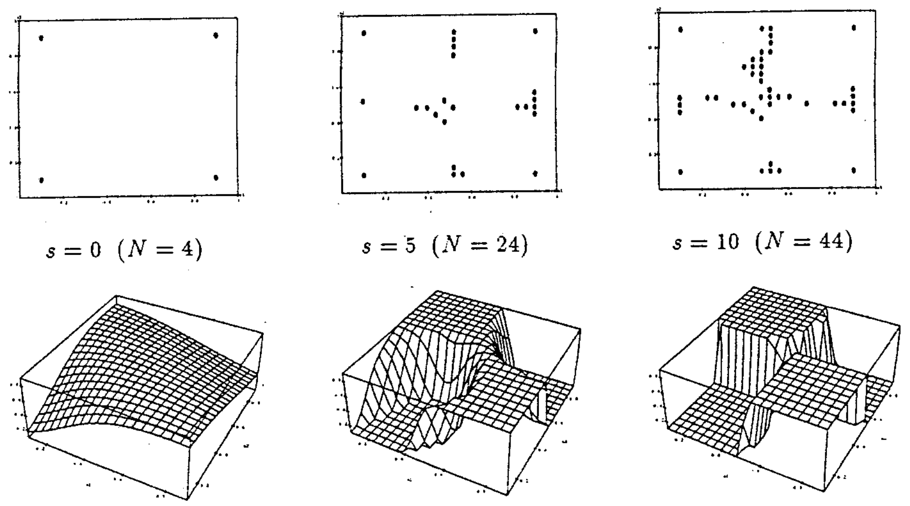

Figure 1: Selective incremental learning of the four-quadrant function

Notice that preferred examples lie on the separating lines of output 0 and 1. These are the border patterns and thus critical to solving this problem. Notice also that 44 selected examples, corresponding to 11% of the whole data, are sufficient for learning the function.

Next we applied the method to digit recognition in which training patterns consisted of 15×10 bitmap scanned from 20 sheets written by 10 persons. Starting from 10 examples and increasing the training set by 50 examples in each selection step, the method found about 1400 examples on average whose performance achieved the same generalization performance as the whole training set of 3400 examples.

The method has also been tested on several discrete problems, including majority and contiguity, and continuous functions such as the forward and inverse kinematics of a robot arm [5]. In most applications a moderate amount of training examples was sufficient to achieve a reasonable performance.

4 Concluding Remarks

We presented a method for selecting a critical subset of given examples during training the network. The method starts with a small training set which is increased incrementally after training. This incremental approach is based on the observation of close interdependence between the network performance and the training set: the performance of the network can be best improved by being trained on a good training set, whereas the quality of the training set can be maximally enhanced by taking account of the current state of the network.

In effect, the selective incremental learning performs an adaptive global-to-local search not only on the weight space but on the example space as well. This is contrasted with the plain backpropagation training procedure that "pushes" the network to load all the data at the same time, leading frequently to a delayed convergence. Our experimental results suggest the incremental learning generalizes much better and thus converges faster to good solutions than a non-incremental one. The improvement is proportional to the data reduction ratio by example selection.

References

[1] MacKay, D. J. C. (1992) "Information-based objective functions for active data selection," *Neural Computation*, vol. 4, pp. 590–604.

[2] Plutowski, M. and White, H. (1993) "Selecting concise training sets from clean data," *IEEE Trans. Neural Networks*, vol. 4, 305–318.

[3] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986) "Learning internal representations by error propagation," in *Parallel Distributed Processing*, Rumelhart, D. E. *et al.* (eds.), MIT Press, pp. 318–362.

[4] Volper, D. J. and Hampson, S. E. (1987) "Learning and using specific instances," *Biological Cybernetics*, vol. 57, pp. 57–71.

[5] Zhang, B. T. (1992) *Learning by Genetic Neural Evolution*, Ph.D. thesis in German, Universty of Bonn, ISBN 3-929037-16-5, Infix-Verlag.

[6] Zhang, B. T. (1993) "Accelerated learning by active example selection," to appear in *International Journal of Neural Systems.*

[7] Zhang, B. T. (1993) "Self-development learning: Constructing optimal size neural networks via incremental data selection," *GMD-Arbeitspapiere* No. 768, German Nat'l Res. Ctr. for Comp. Sci., Sankt Augustin.

[8] Zhang, B. T. and Veenker, G. (1991) "Focused incremental learning for improved generalization with reduced training sets," in *Artificial Neural Networks*, Kohonen, T. *et al.* (eds.), North-Holland, vol. I, pp. 227-232.