TR－A－0060

Objective Functions for Improved Pattern Classification with Back-propagation Networks

－BPネットワーク における誤差測度の改良－

Ronald CHRISLEY, Erik MCDERMOTT and Shigeru KATAGIRI

ロナルド・クリスリ　エリック・マクダモット　片桐　滋

1989. 8. 22

ATR視聴覚機構研究所

〒619-02 京都府相楽郡精華町乾谷　☎07749-5-1411

ATR Auditory and Visual Perception Research Laboratories

Inuidani, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1411
Facsimile: +81-7749-5-1408
Telex: 5452-516 ATR J

Objective Functions for Improved Pattern Classification with Back-propagation Networks

R. Chrisley, E. McDermott, S. Katagiri

ATR Auditory and Visual Perception
Research Laboratories, Kyoto, Japan

ABSTRACT

A discrepancy is noted between the error measure implied by standard objective functions used for the training of back-propagation networks and their actual error in performance. Specifically, if one uses such a network for pattern classification, with one output node per class, and the most active output node indicating the network's classification of the input, then standard objective functions will 1) ascribe non-zero error to network states that are classifying correctly and 2) modify the network more than is necessary to account for incorrectly classified input, thus violating the "minimal disturbance principle." It is hypothesized that objective functions that lack these two characteristics will more closely reflect the actual recognition error and thus their use will result in better performance (i.e., fewer classification errors). Several such functions are presented, and a few are benchmarked against standard error functions on phoneme recognition tasks. Two of the methods show a consistent improvement in performance on a small (BDG) task, but result in worse performance for a large (all consonants) task.

1. Introduction

It has already been suggested (Kohonen, Chrisley, Barna, '88, *inter alia*) that the performance of back-propagation (BP) networks as pattern classifiers would be enhanced if the well-known discrepancy between minimization of an error function and minimization of classification errors could be somehow explicitly addressed. That is, BP uses an objective function, which determines an ideal response to the current input, and then compares the actual response to the ideal response in order to calculate error. Change in the weights of the network is then related to the error thus calculated. It is our hypothesis that if the objective function used results in an error value that more closely models actual recognition error, then minimization of that error will result in better actual performance (i.e., fewer classification errors).

Specifically, we first propose that an objective function that always yields a zero error value for correct classifications will improve overall performance by minimizing classification errors further than could be done with a standard objective function. Our second proposition is that the target output for incorrect classifications should be chosen so that the minimum necessary change in the network is made. Both proposals can be seen as applications of Widrow, Winter, and Baxter's "minimal disturbance principle" (MDP):

> The idea is to adapt the network to properly respond to the newest input pattern while minimally disturbing the responses already trained in for the previous input patterns. Unless this principle is practiced, it is difficult for the network to simultaneously store all of the required pattern responses...
>
> When training the network to respond correctly to various input patterns, the "golden rule" is: give the responsibility to the neuron or neurons that can most easily assume it. In other words, don't rock the boat any more than necessary to achieve the desired training objective. This minimal disturbance principle has been tested extensively, and appears to converge and behave robustly in all cases.[1]

But they go on to say that "a great deal of effort will be required to derive its [the MDP's] mathematical properties." It is our idea to analyse the principle in terms of which objective functions to use, and to conduct this analysis in two cases: the case where the network's current classification is correct, and the case where the current classification is incorrect.

2. The standard approach

We will take the following to be the standard means of using BP for pattern classification. There are a number of input units, n, equal to the dimensionality of the patterns to be classified. Then there are 1 or more layers of hidden units, and last there is an output layer with a number of units m equal to the number of categories into which the patterns are to be classified. Recognition involves the fixing of the input units to the values of one of the input patterns to be classified. This activation is forward-propagated through the network, until the values for the output units are determined. The input is classified as being in the class corresponding to the output unit with the highest value (perhaps requiring some minimal margin of error difference between the greatest and second greatest output values).

Note that this itself is an improvement of the standard BP recognition algorithm, which was developed principally for deterministic (i.e., non-statistical, where input patterns may belong to more than one category with different probabilities) pattern recognition, where it is not uncommon to be able to train an input to produce an ideal output, i.e, one that is almost a unit-vector (e.g., [0.1, 0.1, 0.1, 0.9, 0.1]). In such cases, equality of the output with one of the m ideal vectors is checked, and if it is not equal to any of them, then it is considered a misclassification. Classes are associated with whole patterns of activity across the output units, not just with a particular unit. This recognition rule is obviously too stringent, especially in the case of statistical pattern classification, since it might be impossible to get all of the inputs to produce a near-ideal output due to the fact that the same input is mapped to different classes, and therefore ideal outputs, with different probabilities. (Dahl '87) has pointed

[1] (Widrow, Winter, and Baxter '87), pp. 417, 419-20.

out that the information in the network will be put to better use if the recognition rule is to choose the class whose ideal pattern is closest to the current output. It is clear to see that in the case of near-unit-vector ideal outputs, this is equivalent to the rule we first described, above: choose the class whose output unit is greatest.

3. McClelland's logarithmic error

The use of a standard error function, in combination with a standard sigmoid activation function, results in near-zero weight changes for output units when they are maximally incorrect (i.e., when the difference between desired and actual outputs is close to one). This means the network will learn very little in precisely those cases where it should learn the most. This is especially a problem in learning tasks that involve a large number of classes, since a network will learn to drive all the outputs to zero, unless each output node learns a substantial amount in those few cases when its target is one (note that this problem could be solved by using non-unit targets). McClelland and Franzini (Franzini '88) have suggested an error function that solves this problem. They define the error to be the sum of the log of the squared difference between desired and actual output. This results in weight changes that go to zero as the difference goes to zero, but go to infinity as the difference approaches 1. (Fahlman '88) suggests a hyperbolic arctangent error function for the same reason. We used a variation on McClelland's error (Haffner '88) as a standard against which to benchmark our proposed error functions when we were considering problems with many classes, such as the all consonants task in phoneme recognition.

4. Our proposals

4.1. Don't learn when correct

Although learning when correct may be helpful at early stages of learning, one of our central hypotheses is that at least in the fine tuning stage, performance will be increased if you don't try to make the network conform to some ideal response, since it may be impossible to do so without unlearning other patterns (cf the discussion of statistical pattern recognition in section 2. above). Part of LVQ2's improved performance over LVQ, for example, is its adherence to this principle. The principle is easy enough to implement: if, on a particular input pattern, the recognition rule determines a correct classification, then set the desired outputs to be the actual outputs. This will result in a zero error value, and thus no change in the weights. Some results of using this error function are given below, under experiments. Although we have been assuming that the goal of any new error function is to improve performance on test data, even perhaps at the expense of greater learning time, this method might actually result in a shorter learning time, since the samples that don't require more learning are skipped.

4.2. Use the closest correct output as the target

Our second proposal is one idea of how to make the objective function minimally disturb the network in the case of an incorrect classification. The initial insight is that once one abandons the restriction of ideal (i.e., near-unit-vector) target output vectors, then one has an infinite number of target output vectors to choose from. That is, there are an infinite number of vectors whose largest coördinate is the one corresponding to the correct category. Keeping the MDP in mind, and remembering that the recognition rule involves a margin of error ∂ (the classification is that of the most active output unit, if it is at least ∂ greater than the second most active unit), we can then choose the target output in a principled manner: out of the subspace of vectors that 1) have as their greatest coördinate the one that corresponds to the correct category and 2) meet the condition that their greatest coördinate is at least ∂ greater than the second greatest, choose the vector closest to the actual output vector. This subspace has the nice property of having a convex boundary (where it is bounded), and thus, for any vector not in the subspace, there will be a unique closest vector in the subspace. Although the general n-dimensional solution for the closest correct vector, given an output vector and a category, is rather complex, the 3-dimensional solution is rather easy to derive, and thus we could apply this method to the 3-category BDG task. The following equations are used to calculate a proposed target vector for each of the possible two (in our case; in general, N-1) cases: when the closest target falls on the x_t-x_a-∂=0 plane, and when it falls on the x_t-x_b-∂=0, where x_t is the output unit corresponding to the correct class, and x_a and x_b are the other outputs. Then, the distances between each of these proposed targets and the actual output of the network are calculated; the target that is closest is used for error calculation and weight change. The equations:

$$d_t = \frac{o_t + o_c + o_i + 2\partial + \mu}{3}, \quad d_c = \frac{o_t + o_c + o_i - \partial + \mu}{3}, \quad d_i = \frac{o_t + o_c + o_i - \partial - 2\mu}{3}$$

where:

d_t, o_t are the desired and actual values of the output node that corresponds to the correct class;

d_c, o_c are the desired and actual values of the output node of the current case (of the two possible cases) we are considering;

∂ is the margin of error term used in recognition, mentioned above;

$\mu = \left[\frac{o_t + o_c - 2o_i - \partial}{2}\right]^+$ (i.e., μ is equal to the bracketed term when it is greater than zero, otherwise μ is equal to zero).

Notice that if μ is greater than zero, the targets are simply:

d_t = the average of o_t and o_c, plus $\partial/2$;

d_c = the average of o_t and o_c, minus $\partial/2$;

$d_i = o_i$.

Initial results on the BDG task of this idea under a few parameter choices, however, were not competitive with standard methods.

Possible variations on this idea include using a metric other than the Euclidean in order to determine closeness (see 4.3. below), and using a value for ∂ different than the one used in recognition. We also tried a slightly related idea: when incorrect, set the target to one for the unit that should have been greatest, zero for the unit that (incorrectly) was greatest, and set all the other targets to be equal to the actual outputs. Again, initial results on the BDG task of this idea under a few parameter choices, however, were not competitive with standard methods.

4.3. Use weight commitment to find closest output

A different way of construing the MDP in the incorrect case is to minimally disturb the weights that have already been committed and are thus storing information about previous samples. That is, choose a target output that might not be closest in Euclidean distance, but that will assign most of the blame to uncommitted weights (those whose absolute values are near zero), and proportionately less blame to weights with higher absolute values. One could use the same method as described above, except that instead of using the unweighted Euclidean metric, one uses the weighted metric defined by:

$$d(\bar{o},\bar{t}) = \sqrt{\sum_{i=1}^{L_N} w_i(\bar{o}_i - \bar{t}_i)^2}$$

where:
o is the actual output vector;
t is the target vector;
d(x, y) is the distance between vectors x and y;
L_i is the number of nodes in the ith layer of the network;
N is the number of layers in the network;
and the weights w are defined by:

$$w_i = 1 - \frac{A_i^N}{\sum_{j=1}^{L_N} A_j^N}$$

where the A's are defined recursively by:

$$A_j^n = \sum_{k=1}^{L_{n-1}} A_k^{n-1} C_{kj}; \quad A_j^1 = 1;$$

and where C_{kj} are the standard interconnection weights from unit k of layer n –1 to unit j of layer n.

(Widrow, Winter, and Baxter '87) modify weights based on which units have *outputs* close to zero, but do not look at the values of the weights. Also, they only use the outputs to determine priority of change, not amount. It is interesting to note that several connectionist researchers have been spending effort on devising schemes that do *not* recruit more weights, in order to improve generalization (see below). In this light, perhaps it would be best for the weights should have less and less influence as learning proceeds. Information is initially distributed about the network, but as performance improves, the network constrains itself to fine tune the solution it has, instead of recruiting more weights to learn an arbitrary solution. This may require the network to have sophisticated abilities to monitor its own performance.

4.4. Use average response to find closest output

Another idea is to use the average correct response of the network to determine the target vectors. This proposal adheres to the MDP, but unlike the above methods, and like the standard method, it advocates a single target for the entire class (although this target will change slowly over time), which might prove necessary if the other methods have difficulty in converging because of the dynamic nature of the target outputs. One proposal would be to start out with near-unit-vectors, but then at regular intervals sample the correct outputs for each of the classes, and average them, thus providing a single target output for that class, until the next sample is drawn. This way the targets will be guaranteed to be consistent (i.e., they will meet the criterion of the greatest unit corresponding to the correct vector), but they will also better conform to the actual outputs, so that correct outputs will be result in a smaller change in weight than would a standard, near-unit target.

There is also an "on-the-fly" version of the adaptive target scheme that is related to the method described above in a way analogous to the difference between learning after every sample and only learning after an entire epoch of samples. The targets are intialised to some value (the unit vectors, for example). Then, after each sample that is classified correctly, the targets are modified to be closer to the actual outputs:

$$d_i(t+1) = d_i(t) + \pi[o_i(t) - d_i(t)],$$

where d is the target output vector, o is the actual output vector, and π is a constant that determines what proportion of the distance the target will move toward the actual output. A value of 0.5 for π will average the two vectors. We have generally found that the reciprocal of the number of samples per epoch is a good first guess for the best value of π. Some results of using this on-the-fly method are described under Experiments, below.

If there is a wide variance in the correct responses for a particular class, it might be advantageous to have two or more targets for one class, to account for the clusters in the correct output vectors. This would be much better than trying to use one vector for all the clusters. Either one could do clustering once, at the beginning of learning, clustering samples by their input values, or one could cluster periodically during learning, using output values produced by the samples. One could even use LVQ or LVQ2 as a means of doing this.

5. Experiments

5.1 The BDG task

We benchmarked the performance of three error function methods on a phoneme recognition task involving 3600 samples of the stop consonants /b/, /d/, and /g/. The architecture for the network was the TDNN as described in (Haffner '88) and originally mentioned in (Waibel). Although the reader should consult those works for details of our network, we will summarize here the connectivity:

	Input	Layer 1	Layer 2	Output	Total
Units	241	104	27	3	375
Physical units	16	8	3	3	
Fan-in	0	49	41	10	
Connections	0	5096	1107	30	6233
Physical connections	0	392	123	6	521

Table 1. Network architecture figures for the BDG task. Taken from (Haffner, Waibel, Sawai, and Shikano '88; page 8).

The learning rate was 0.1, the momentum value was 0.9. The network employed momentum scaling, learning rate overshoot control, and skipping of learned samples as described in (Haffner '88). The results are given below:

	EPOCHS				
	50	55	60	65	70
Standard	2.97	2.64	2.48	2.49	2.41
DLC	2.63	2.23	2.74	2.55	2.11
ADT	2.30	2.26	2.30	2.54	2.02

Table 2: Test data error rates (averaged over 10 trials) on the BDG task as determined by error function and epoch in the learning phase. DLC = "don't learn when correct" method, ∂ = 0.5. ADT = adaptive target method, $\pi = 0.01$.

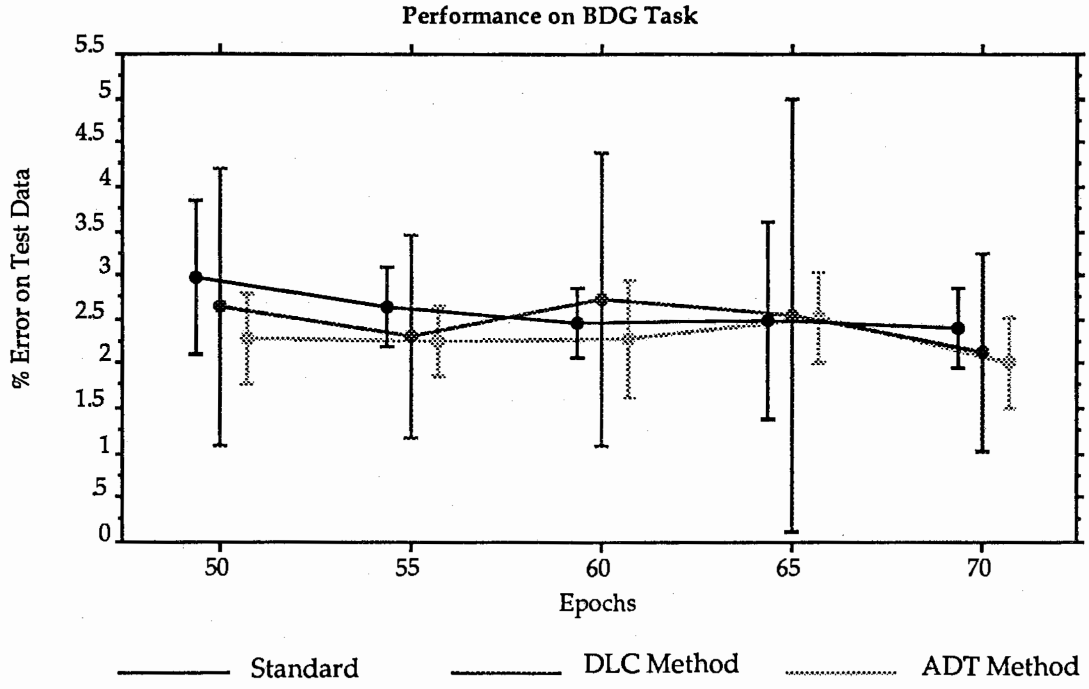

Figure 1. Error percentages (number of misclassifications of test data samples divided by the number of samples tested on [3600]) at different stages of learning for the three different error functions on the BDG task, ∂ = 0.5, π = 0.01. Error bars indicate one standard deviation. Percentages were averaged over 10 trials.

As one can see from Table 2, both the DLC ("don't learn when correct") and the ADT ("adaptive target") methods show a modest improvement in performance on the BDG task over the standard error function. Also, one can deduce from Figure 1 that the DLC method generally found the best solutions overall, although it frequently found poor solutions as well. It seems that if one 1) desires the best performance; 2) is willing to perform several learning trials in order to achieve it; and 3) has a means of comparing the results of different trials (i.e., one has some test data comparable to the data on which one desires the network to perform), then the DLC method will be the most appropriate. But if one can only afford one or a few trials, then the small variance of the ADT results indicate that ADT would be a safe bet.

5.1 The all consonants task

We also compared the performance of these methods on an all consonant phoneme recognition task (9000 samples, 18 classes). Although the methods clearly performed better than the standard error function, this is not a notable accomplishment, given that the standard function does so poorly because of the large number of categories. For a fair comparison, we benchmarked them against a network using McClelland's logarithmic error function (see section 3, above). The two methods sometimes came within 1-2% of the performance of the McClelland network, but were always at least 5% below its performance on test data. Perhaps this is dependent on the parameters used.

6. Summary

Several alternatives to the standard error function for back propagation were presented. These were designed to obey, to varying degrees, the minimal disturbance principle. The main ideas were 1) don't learn when correct; 2) use the closest target vector; 3) use the closest target vector weighted by weight commitment; 4) use an averaged target vector. Ideas 1) and 4) showed modest improvements over the standard error function method on a small (3600 samples, 3 classes) phoneme recognition task. They performed significantly worse than standard methods on a large (9000 samples, 18 classes) phoneme recognition task. The success of the methods in the BDG case indicates that future exploration of parameters might result in an error function that can out-perform standard methods on large tasks as well.

7. References

Dahl, E. D. (1987) "Accelerated learning using the generalized delta rule", *Proceedings of the First Annual IEEE International Conference on Neural Networks, San Diego, CA..*

Fahlman, S. E. (1988) "An empirical study of learning speed in back-propagation networks," Carnegie Mellon technical report CMU-CS-88-162.

Franzini, M. A. (1987) "Speech recognition with back-propagation," *Proceedings of the Ninth Annual Conference of IEEE Engineering in Medicine and Biology Society.*

Haffner, P. (1988) "Dynet, a fast program for learning in neural networks," ATR Interpreting Telephony Research Laboratories technical report TR-1-0059.

Haffner, P, Waibel, A., Sawai, H., and Shikano, K. (1988) "Fast back-propagation learning methods for neural networks in speech," ATR Interpreting Telephony Research Laboratories technical report TR-1-0058.

Hampshire, J. B. and Waibel, A. H. (1989) "A novel objective function for improved phoneme recognition using time-delay neural networks", in (?).

Hinton, G. E. (1987) "Connectionist learning procedures", Revised version of Technical Report CMU-CS-87-115, to appear (appeared ?) in Artificial Intelligence.

Kohonen, T., Barna, G., and Chrisley, R. (1988) "Statistical pattern recognition with neural networks: benchmarking studies," *Proceedings of the Second Annual IEEE International Conference on Neural Networks, San Diego, CA.*.

Solla, S. A., Levin, E., and Fleisher, M. (1988) "Accelerated Learning in Layered Neural Networks" in *Complex Systems* 2.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. (1988) "Phoneme recognition using time-delay neural networks," ATR Interpreting Telephony Research Laboratories technical report TR-1-0006.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. (1988) "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing.*

Widrow, B., Winter, R. G., and Baxter, R. A. (1987) "Learning phenomena in layered neural networks ", in the *Proceedings of the First Annual IEEE International Conference on Neural Networks, San Diego, CA.*.