2001-07-19

Lazy Training: Improving Backpropagation Learning through Network Interaction

Timothy L. Andersen

Tony R. Martinez
martinez@cs.byu.edu

Michael E. Rimer

Original Publication Citation

Rimer, M., Andersen, T., and Martinez, T. R., "Lazy Training: Improving Backpropagation Learning through Network Interaction", Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'1, pp. 27-212, 21.

Lazy Training: Improving Backpropagation Learning through Network Interaction

Michael E. Rimer, Timothy L. Andersen and Tony R. Martinez

Brigham Young University
Computer Science Department
Provo, UT 84602, USA
{mrimer, tim}@axon.cs.byu.edu, martinez@cs.byu.edu

Abstract

Backpropagation, similar to most high-order learning algorithms, is prone to overfitting. We address this issue by introducing interactive training *(IT), a logical extension to backpropagation training that employs interaction among multiple networks. This method is based on the theory that centralized control is more effective for learning in deep problem spaces in a multi-agent paradigm [25]. IT methods allow networks to work together to form more complex systems while not restraining their individual ability to specialize.* Lazy training, *an implementation of IT that minimizes misclassification error, is presented. Lazy training discourages overfitting and is conducive to higher accuracy in multiclass problems than standard backpropagation. Experiments on a large, real world OCR data set have shown interactive training to significantly increase generalization accuracy, from 97.86% to 99.11%. These results are supported by theoretical and conceptual extensions from algorithmic to interactive training models.*

1 Introduction

Artificial neural networks have received substantial attention as robust learning models for tasks including classification [18]. Much research has gone into improving their ability to generalize beyond the training data. Many factors play a role in their ability to learn, including network topology, learning algorithm, and the nature of the problem being learned. Often, overfitting the training data, caused through the use of an inappropriate objective function, is detrimental to generalization. In this work we introduce *interactive training* (IT), a novel environment wherein multiple networks can be trained simultaneously. We present *lazy training*, an implementation of IT with an objective function that directly minimizes classification error while discouraging overfitting. Lazy training performs markedly better than optimized standard backpropagation training on a large OCR database, increasing accuracy from 97.86% to 99.11%.

An overview of related work and discussion of objective functions is provided in section 2. Interactive training and the lazy training algorithm are presented in section 3. Experiments and results are given in section 4. Analysis and discussion are in section 5. Further work and conclusions are presented in section 6.

2 Related work

Over the last decade, much effort has been put into developing optimized backpropagation learning models and algorithms. Techniques, such as Quickprop [10] and Rprop [17], seek to speed up learning by dynamically adjusting update parameters. Other models are classified as adaptive learning algorithms, which seek to generate network topologies that are more suited to learning a given problem [1, 2, 11]. Non fully-connected static architectures are also considered in [9]. These networks have fewer parameters and are therefore simpler and more efficient than fully-connected networks yet are able to perform equally well.

2.1 Critique of current training techniques

To generalize well, a learner must have a proper objective function. Most learning techniques incorporate an objective function of minimizing SSE. The validity of using SSE as an objective function to minimize error relies on the assumption that sample outputs are offset by inherent gaussian noise, being normally distributed about a cluster mean. For learning function approximation of an arbitrary signal, this presumption often holds. However, this assumption is invalid for classification problems, where the target vectors are class codings (i.e., arbitrary nominal or boolean values representing designated classes).

Cross-entropy (CE) assumes *idealized* class outputs (i.e., target values of zero or one for a sigmoid activation) [16] and is therefore more appropriate to classification problems. However, error values using SSE and cross-entropy have been shown [12] to be inconsistent with ultimate sample classification accuracy. That is, minimizing CE or SSE is not necessarily correlated to high recognition rates.

0-7803-7044-9/01/$10.00 ©2001 IEEE

Numerous experiments in the literature provide examples of networks that achieve little error on the training set but fail to achieve the best possible accuracy on test data [1, 20]. This is due to a variety of reasons, such as *overfitting* the data or having an incomplete representation of the data distribution in the training set. There is an inherent tradeoff between fitting the (limited) data sample perfectly and generalizing accurately over the entire population.

2.2 Shortcomings of search methodologies

More fundamentally, the above objective functions provide mechanisms that do not reflect the true goal of learning, which is to achieve high recognition rates on unseen data. In [12], a new objective function, the classification figure-of-merit (CFM) is introduced for which minimizing error remains consistent with increasing classification accuracy. Networks that use the CFM as their criterion function are introduced in [12] and further considered in [6]. They are, however, also susceptible to overfitting. The question of how to prevent overfitting is a subtle one. When a network has many free parameters, not only is learning fast, but local minima can often be avoided. On the other hand, networks with few free parameters tend to exhibit better generalization performance. Determining the appropriate size network remains an open problem [8].

The problem of overfitting has received much attention in the literature. Methods of addressing this problem include using a holdout set to stop training early [22], cross-validation [3], node pruning [7, 8], and weight decay [26], among others. These techniques approach optimal solutions given the inductive bias of the standard learning model, but do not consider possible enhancements to the inductive bias itself. Node pruning seeks to improve accuracy by simplifying net topology, rather than alleviating the problems common to larger topologies, for example. Methods for overcoming problems in the inductive bias inherent to training with backpropagation generally involve forming network ensembles. Ensemble techniques, such as bagging and boosting [15], or wagging [4], are more robust than single networks when the errors among the networks are not positively correlated.

In [5], there is evidence that the size of the weights in a network plays a more important role to generalization than the number of nodes. A simpler method of preventing overfitting is to provide a maximum error tolerance threshold, d_{max}, which is the smallest absolute output error to be back-propagated. In other words, no weight update occurs for a given d_{max}, target value, t_j, and network output, o_j, if the absolute error $| t_j - o_j | < d_{max}$. This threshold is arbitrarily chosen to represent a point at which a sample has been sufficiently approximated. With an error threshold, the network is permitted to converge with much smaller weights [21].

Classical methods of designing ensembles involve a two-step process, where the networks are first generated independently, and then combined. It has been proposed in [23, 24] that learning models that interact with an external environment (e.g., another learner) have a greater theoretical power of expression than non-interactive models. To support this, [25] shows that coupled agents perform much more efficiently than independent agents at complex learning tasks. The paradigm shift from optimized, but isolated, algorithms to interactive models reflects the current evolution in the philosophy of the field of computer science from procedure-oriented to object-oriented languages and single mainframes to networks of personal computers. Interactive neural network models are proposed to be superior over independent models. An introduction to combining neural networks can be found in [19]. In [13, 14], networks in an ensemble are trained simultaneously with the inclusion of an additional error term that encourages negative error correlation among the networks. This generally provides some improvement. However, the field of simultaneous learning with neural nets is largely unexplored. The interactive method proposed in section 3 is an original contribution to the budding field of multi-agent neural network learning.

3 Interactive training method

Common ensemble training methods provide for training of networks separately. Independent training of domain-specific experts is only marginally beneficial to an ensemble as a whole. The aspect of simultaneous training has been addressed in part in [13, 14], but does not provide for overfitting. This work addresses overfitting by applying *lazy training*, a conservative form of training, to the learning process. Its philosophy is similar to CFM. However, CFM does not prevent weight saturation, which is often detrimental to accuracy [5]. Lazy training simultaneously trains all networks in an ensemble, updating only the weights of nets that endanger the classification accuracy of the ensemble. This approach allows the model to relax more conservatively into a solution and discourages overfitting.

Interactive training (IT) considers the output candidate of the entire network ensemble during training. For each sample considered, only those networks that are credited with classification errors are updated through backpropagation. The result is training without idealized outputs of 0 and 1, providing a training mechanism that is reminiscent of constraint satisfaction and reinforcement learning, where the network learns to interact with its (changing) environment. As this forces networks to learn only when explicit evidence is presented that their state is a detriment to classification accuracy, we have dubbed this technique *lazy training* (not to be confused with lazy learning approaches). Backpropagation training often uses

an objective function that tends to a *saturation* of the weights. That is, it tends to encourage larger weights in an attempt to output a value approaching 1 or 0. The ramifications of this will be discussed further in section 5. Lazy training is biased toward simpler solutions, meaning smaller weights, even approaching zero, can provide an acceptable solution.

When networks are trained concurrently, rather than sequentially as in standard ensembles, they can take advantage of greater expressive power through interaction during the training process. Two or more networks can collaborate together to decide how learning is to proceed at any given point. More specifically, interaction among networks allows a *dynamic error threshold* to be implemented. That is, when one network presents a sufficient solution in an area of the problem space, other networks do not need to work at redundantly modeling the same local data. Consequently, they are able to specialize and break a complex problem up into smaller, simpler ones. This provides for a more conservative form of training that converges with smaller network weights, hence with less overfitting and greater generalization accuracy.

3.1 Interactive training topology

Interactive training provides a logical extension of the standard multi-layer perceptron (MLP). In an MLP, an input vector fans out to a layer of hidden nodes. The output of the hidden nodes is consolidated into one or more output nodes, generally with sigmoid activation functions. With IT, a similar topology is utilized (see Figure 1). However, there are three crucial differences in the implementation.

First, rather than being comprised of perceptrons, the hidden layer of an IT ensemble can be comprised of MLPs. Each MLP serves to specialize in learning a certain area of the problem space. For instance, in learning a multi-class data distribution, each MLP can be responsible for learning one of the classes.

Second, the output node employs a winner-take-all (WTA) activation function rather than a sigmoid function. For a given sample, the feature vector is presented to all MLPs in the network's hidden layer. Each MLP sends an output to the WTA layer. The classification of the sample is considered correct if an MLP corresponding to the target output class propagates the highest value to the WTA node (i.e., the IT network outputs the target classification).

Third, network weights are updated exclusively to minimize classification error. When the network misclassifies a sample, credit for the error is assigned to two sources. The first is the set of MLP nodes that fired higher outputs than the target MLP (resulting in the WTA node outputting the wrong class value). The second is the target MLP itself, which fired too low to have its classification selected. In an

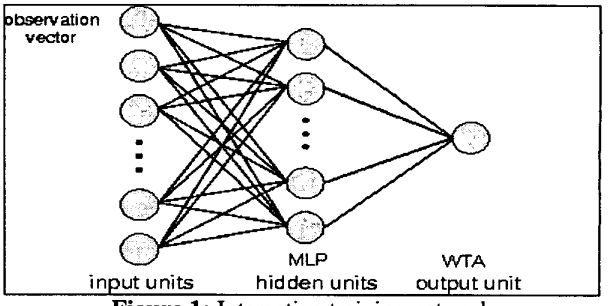

Figure 1: Interactive training network.

MLP, weights are updated based on how much each node contributes to the output error. Analogously, the IT model updates only the weights of nets that contribute to hindering the classification accuracy of the ensemble. Incorporation of a novel objective function minimizing the classification error directly is made possible through the increased interactive expressiveness of the IT network. This approach is formalized in the following algorithm.

3.2 Interactive training algorithm (lazy training)

Let n be the number of MLPs in an IT network. Let o_i be the output of the i^{th} MLP in the network ($1 \le i \le n$, $0 \le o \le 1$). Let T designate the target MLP for a given sample, where the sample's target category corresponds to the output class of the T^{th} MLP. Let o_T be the output of the target net. The error, ε_i, back-propagated to the output node of the i^{th} MLP in the network is defined as

$$\varepsilon_i = \begin{cases} \tau_U - o_i & \text{if } i = T \wedge \exists j (o_j > o_i) \quad \text{where } o_i + \delta \le \tau_U \le 1 \\ \tau_L - o_i & \text{if } i \ne T \wedge (o_i > o_T) \quad \text{where } 0 \le \tau_L \le o_i - \delta \\ 0 & \text{otherwise} \end{cases}$$

where τ_U and τ_L are upper and lower target values and δ is a small constant. The rate of convergence is partly dependent on the values used for τ_U and τ_L. A τ closer to comparable output values in the other nets implies less error and will result in slower, but steadier, convergence than values closer to 0 or 1.

Training of the IT ensemble proceeds at a much different pace than with standard backpropagation. Training only the nodes that directly contribute to classification error allows the model to relax more gradually into a solution, learning only as much as it needs to and thereby discouraging overfitting. This approach is reminiscent of training with an error threshold; however whereas a fixed error threshold causes training to stop at a pre-specified point, IT dynamically halts at the first possible point for a given sample at a given point in time. Weights are updated only through necessity. After all, a sample can be considered "learned" with any output value, providing opposing nets output lower values.

Additionally, overfitting is minimized in an IT network because outliers (noisy samples) have minimal detrimental impact to the decision surface's accuracy. This is because the target output is only required to output a value that is negligibly higher than the output representing the neighboring class (see Figure 2b). This is in contrast to classical gradient descent training, where hard target values of 0 and 1 are required (translating to pushing the decision surface as far away as possible) even for outliers (see Figure 2a). Hence, in testing, samples close to the outlier belonging to the competing class (represented by the question mark) have a much better chance of being correctly classified.

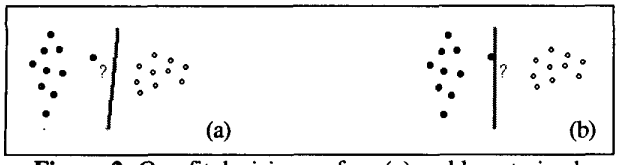

Figure 2: Overfit decision surface (a) and lazy-trained decision surface (b).

3.3 Enlarging the margin

Most often, the highest outputting net, n_1, outputs a value only slightly higher than the second highest-firing network, n_2 (see Figure 3). This is true for correctly classified samples (above 0 in Figure 3), and also for incorrect ones (below 0). An error margin, ϵ, can be introduced during the training process that serves as a confidence buffer between the outputs of target and non- target networks. This measure requires $o_T > (\epsilon + o_{n_2})$ for no error to be back-propagated. During the training process, ϵ can be increased gradually and might even be negative to begin with, not expressly requiring correct classification at first. This gives the networks time to configure their parameters in an even more relaxed way. Then ϵ is increased to an interval sufficient to account for the variance that appears in the test data, allowing for robust generalization. The value of ϵ can also remain negative to account for noisy outliers. At the extreme of ϵ equal to 1, lazy training becomes standard backpropagation training, with target values of 1.0 and 0.0 for positive and negative samples, respectively.

4 Experiments

The performance of independent versus interactive training models has been evaluated on an OCR data corpus consisting of over 495,000 alphanumeric character samples, partitioned into roughly 415,000 training samples and 80,000 test samples.

4.1 Parameters

We compared fully connected feed-forward MLPs trained through standard on-line backpropagation minimizing SSE against lazy-trained networks. In all experiments presented,

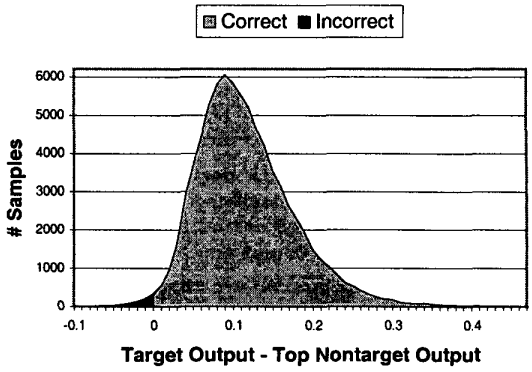

Figure 3: Network output error margin after lazy training.

each MLP contained a single hidden layer comprised of 32 hidden nodes. Weights were initialized to uniform random values in the range [-0.3,0.3]. The same initial weights were used for each training method. Learning rate was 0.2 and momentum was 0.5. Separately trained networks used an error threshold (d_{max}) of 0.1. In these tests a τ_U of 1 and τ_L of 0 were used for faster lazy training; δ was 0 and ϵ was 0.05. Training was halted after 500 epochs.

4.2 Results

Table 1 displays the results of standard SSE back-propagation versus lazy training. *Train* and *Test* are the final training and test set accuracy (above 0.5 for positive samples, below 0.5 for negative samples for standard backpropagation, or the target network outputting the highest value for lazy training) in percent. *Train MSE* and *Test* MSE are the mean squared errors for the training and test sets at convergence (values in parentheses denote the mean-squared difference between the top two net outputs).

Table 1: Results on *OCR* data set.

Method	Train	Train MSE	Test	Test MSE
Standard BP	99.28	.0047	97.86	.0092
Lazy Train	99.27	.203 (.108)	99.11	.241 (.121)

5 Analysis and discussion

The results show that networks generated through interactive training have the capability of significantly improving accuracy from 97.86% for standard backpropagation training to 99.11% for lazy training. These tests show that, although SSE increased, the amount of overfitting is sharply reduced (see section 5.3).

5.1 Standard approach

Following a training run on a standard set of networks without lazy training, winning net outputs on the test set were distributed as shown in Figure 4. The network fires

very close to 1.0 on the majority of the samples. Only 3-5% of the samples lie close to where the decision surface is located. The weights have been enlarged to the point that the dividing sigmoidal surface becomes very sharp. Whereas networks learning separately perform at 99.28% on the training set, together they only score 97.86% on the test set.

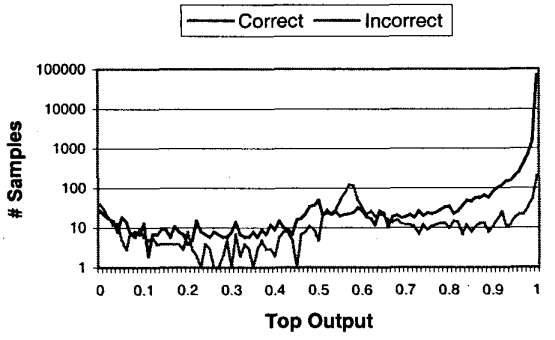

Figure 4: Network outputs on test set for standard training.

5.2 Lazy training approach

Lazy training produces a distribution quite unlike that seen in Figure 4. When networks only perform weight updates to prevent misclassification, the distribution in Figure 5 appears.

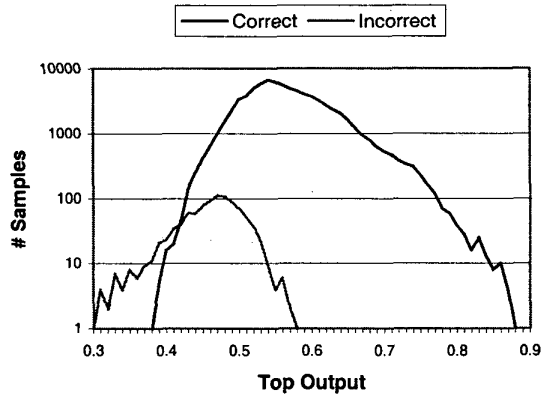

Figure 5: Network outputs on test set after lazy training.

Instead of pushing the samples to one end of the output range or the other, the vast majority remains spread out just slightly above the decision boundary. Sample output distribution is roughly gaussian, reflecting an actual gaussian data distribution, with a larger variance than appears from standard backpropagation, but only a fraction of the error. This suggests that the decision surface follows a much gentler slope. Misclassified samples usually have outputs below 0.5 and are lower than the output for correctly classified samples in the majority of cases.

Observe that training set accuracy is largely preserved on the test set. Since the networks learn together, their solutions are highly correlated and their solution transfers well to unseen data. Error is 58.4% less than with decoupled networks, presenting a strong case for centralized training on large, complex data sets.

5.3 Network complexity

At first, it seems counter-intuitive that networks firing only around 0.5 will generalize so well. Ordinarily, training networks together allows a classifier to become more complex, prone to overfitting. According to Occam's razor, adding parameters to a network, beyond the smallest correct solution for a given problem, can be a detriment to the generalization ability of the network. This is similar to the claim that a network with higher learning capacity tends to "memorize" noise in the data, an undesirable trait.

Recently, however, it has been illustrated how the number of nodes in a network is not as influential as the *magnitude* of the weights [5]. The topology, rather, serves more as a mechanism that lends itself to solving of certain problems, while the weights represent how tightly the network has fit itself to the (admittedly incomplete) training data distribution. Network complexity is further defined in [22] as the number of parameters and the *capacity to which they are used in learning* (i.e., their magnitude). In light of this, it is understandable why complex networks and lazy training, which allows networks to have small weights, perform so well together. Although the IT network has a high number of parameters, lazy training prevents further weight updates once samples are correctly classified and results in low complexity. Hence, the possibility of overfitting is reduced in the training process.

The networks used in our experiments had 64 inputs, 32 hidden nodes and 1 output node, with 2080 weight parameters. The rows of Table 2 list the average magnitude of the weights in a network initialized with uniform random weights in the range [-0.3,0.3], after standard training, and after lazy training, respectively. The columns denote the average of the bias weight on the hidden nodes, bias on the output node, average weight from input to hidden node, and from hidden to output node, respectively. The lazy-trained network has weights that are roughly two to four times larger than the initial random values, while standard training produces weights from ten to twenty times larger. The lazy-trained network is a simpler solution than the network produced by standard backpropagation training.

Table 2: Average network weights.

Method	Hid Bias	Out Bias	Hid Wgt	Out Wgt
Initial	0.16	0.15	0.15	0.15
Standard	2.21	4.66	1.27	6.25
Lazy	0.56	0.02	0.31	0.74

6 Further work and conclusion

More research on modifying the error margin and its effect on training will be performed. Also, further studies of the effect of network size on lazy training and classification accuracy will be done. We will investigate the effect of more sophisticated output functions than WTA on IT networks, such as adding a perceptron or MLP to perform final classification. We will explore the advantages of interactive training techniques over independent learning on different network models, such as single-output networks on two-class problems, and multi-output networks (one output per class). Various styles of network interaction will be considered, including: (1) between dual (complementary) networks for each class (two per class), (2) among redundant networks (many per class), (3) within a lazy-trained multi-output network (one output per class).

Lazy training reduces overfitting in gradient descent backpropagation training, increasing the probability of discovering better solutions. Its advantages over standard backpropagation have been demonstrated on a large real world data set.

8 References

[1] Andersen, Tim and Tony R. Martinez, "A Provably Convergent Dynamic Training Method for Multilayer Perceptron Networks", *Proceedings of the 2nd International Symposium on Neuroinformatics and Neurocomputers*, 1995, pp. 77-84.

[2] Andersen, Tim and Tony R. Martinez, "Using Multiple Node Types to Improve the Performance of DMP (Dynamic Multilayer Perceptron)", *Proceedings of the IASTED International Conference on Artificial Intelligence, Expert Systems and Neural Networks*, 1996, pp. 249-252.

[3] Andersen, Tim and Tony R. Martinez, "Cross Validation and MLP Architecture Selection", *Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'99*, CD Paper #192, 1999.

[4] Andersen, Tim and Martinez, Tony, "Wagging: A learning approach which allows single layer perceptrons to outperform more complex learning algorithms", *Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'99*, CD Paper #191, 1999.

[5] Bartlett, Peter L., "The Sample Complexity of Pattern Classification with Neural Networks: The Size of the Weights is More Important than the Size of the Network", *IEEE Trans. Inf. Theory*, 44(2), 1998, pp. 525-536.

[6] Barnard, Etienne. "Performance and Generalization of the Classification Figure of Merit Criterion Function", *IEEE Transactions on Neural Networks*, 2(2), March 1991, pp. 322-325.

[7] Castellano, G., A. M. Fanelli and M. Pelillo, "An empirical comparison of node pruning methods for layered feed-forward neural networks", *Proc. IJCNN'93-1993 Int. J. Conf. on Neural Networks*, Nagoya, Japan, 1993, pp. 321-326.

[8] Castellano, G., A. M. Fanelli, and M. Pelillo, "An iterative pruning algorithm for feed-forward neural networks", *IEEE Transactions on Neural Networks*, Vol. 8 (3), 1997, pp. 519-531.

[9] Chakraborty B., Y. Sawada and G. Chakraborty, "Layered fractal neural net: computational performance as a classifier", *Knowledge-based Systems*, Vol. 10, 1997, pp. 177-182.

[10] Fahlman, S.E., "Faster-learning Variations on Back-propagation: An Empirical Study", *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann.

[11] Fahlman, Scott E. and Lebiere, Christian, "The Cascade-Correlation Learning Architecture", *Advances in Neural Information Processing Systems 2*, David S. Touretsky, Morgan Kaufmann, San Mateo, California, 1990, pp. 524-532.

[12] Hampshire II, John B., "A Novel Objective Function for Improved Phoneme Recognition Using Time-Delay Neural Networks", *IEEE Transactions on Neural Networks*, Vol. 1, No. 2, June 1990.

[13] Liu, Yong and Yao, Xin, "Ensemble Learning via Negative Correlation", *Neural Networks*, 12(10), Dec. 1999, pp. 1399-1404.

[14] Liu, Yong and Yao, Xin, "Simultaneous Training of Negatively Correlated Neural Networks in an Ensemble", *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 29, no. 6, December 1999, pp. 716-725.

[15] Maclin, R and Opitz, D, "An empirical evaluation of bagging and boosting", *The Fourteenth National Conference on Artificial Intelligence*, 1997.

[16] Mitchell, Tom. *Machine Learning.* McGraw-Hill Companies, Inc., Boston, 1997.

[17] Riedmiller, Martin and Braun, Heinrich, "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm", *Proceedings of the IEEE Conference on Neural Networks*, San Francisco, 1993.

[18] Rumelhart, David E., Hinton, Geoffrey E. and Williams, Ronald J., "Learning Internal Representations by Error Propagation", Institute for Cognitive Science, University of California, San Diego; La Jolla, CA, 1985.

[19] Sharkey, A., "On Combining Artificial Neural Nets", *Connection Science*, vol. 8(3-4), 1996, pp. 299-313.

[20] Schiffmann, W., Joost, M. and Werner, R., "Comparison of Optimized Backpropagation Algorithms", *Artificial Neural Networks*, European Symposium, Brussels, 1993.

[21] Schiffmann, W., Joost, M. and Werner, R., "Optimization of the Backpropagation Algorithm for Training Multilayer Perceptions", University of Koblenz: Institute of Physics, 1994.

[22] Wang, C., Venkatesh, S. S., and Judd, J. S. "Optimal stopping and effective machine complexity in learning", in Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems*, vol. 6, Morgan Kaufmann, San Francisco, 1994, pp. 303-310.

[23] Wegner, Peter, "Why Interaction is more powerful than algorithms", *Communications of the ACM*, May 1997.

[24] Wegner, Peter and Goldin, Dina, "Interaction, Computability and Church's Thesis", 1999. Accepted to the *British Computer Journal*.

[25] Gerhard Weiβ, editor. *Multi-agent Systems, A Modern Approach to Distributed Artificial Intelligence*. 1999. MIT Press, Cambridge, Massachusetts.

[26] Werbos, P., "Backpropagation: Past and future", *Proceedings of the IEEE International Conference on Neural Networks*, IEEE Press, 1988, pp. 343-353.