01 Jan 1999

# Efficient Training Techniques for Classification with Vast Input Space

Donald C. Wunsch
*Missouri University of Science and Technology*, dwunsch@mst.edu

Emad W. Saad

J. J. Choi

J. L. Vian

## Recommended Citation

D. C. Wunsch et al., "Efficient Training Techniques for Classification with Vast Input Space," *Proceedings of the International Joint Conference on Neural Networks, 1999. IJCNN '99*, Institute of Electrical and Electronics Engineers (IEEE), Jan 1999.
The definitive version is available at https://doi.org/10.1109/IJCNN.1999.831156

# Efficient Training Techniques for Classification with Vast Input Space

E. W. Saad[1], J. J. Choi[2], J. L. Vian[3], and D. C.Wunsch II[1]
[1]Applied Computational Intelligence Laboratory, Dept. of Electrical Engineering,
Texas Tech University, Lubbock, TX 79409-3102
[2]The Boeing Company, PO Box 3707, MS 7L-66, Seattle, WA 98124-2207
[3]The Boeing Company, PO Box 3707, MS 45-89, Seattle, WA 98124-2207
saade@ttu.edu, jai.j.choi@boeing.com, john.l.vian@boeing.com, and dwunsch@coe.ttu.edu

## Abstract

*Strategies to efficiently train a neural network for an aerospace problem with a large multi-dimensional input space are developed and demonstrated. The neural network provides classification for over 100,000,000 data points. A query-based strategy is used that initiates training using a small input set, and then augments the set in multiple stages to include important data around the network decision boundary. Neural network inversion and oracle query are used to generate the additional data, jitter is added to the query data to improve the results, and an extended Kalman filter algorithm is used for training. A causality index is discussed as a means to reduce the dimensionality of the problem based on the relative importance of the inputs.*

## Introduction

Neural network training can be based on data generated by an oracle, sometimes referred to as a teacher. The oracle can be a physical system, a simulation model, a set of mathematical equations, or any information source that can be queried about a set of inputs, and give an answer about the corresponding output response.

Query-based learning (QBL) [1], [2], [3], [4], [5] is commonly proposed to efficiently train neural networks when the data generation is expensive with regard to economics or time. Query-based learning is an active technique where the learner participates in selecting its training data, as opposed to passive learning where data is randomly used in training according to the *a priori* joint probability distribution of the input and output.

In query-based learning, a typical scenario starts by roughly training a neural network using a relatively small training data set. An inversion algorithm is then used to generate input patterns lying on the decision boundary. The oracle is then queried to give the correct output for each input pattern. The new set of input/true output pairs (the query data), is then used to further train the neural network. Various techniques of generating the query data, and using it to train a neural network are discussed in the following sections. The query methods are then applied to an aerospace problem involving classification of flight conditions.

## Query Based Learning

### *A. Neural Network Inversion*

The goal of query learning is to acquire set of input data that has high information content. This can be achieved by an inverse mapping. This inversion can be accomplished by an network inversion method [6]. For a given target value the network inversion generates the corresponding input data.

In the forward path through the network, the output of a neuron unit is given as

$$u_i = \sum_{j=0}^{p} w_{ij} x_j \, , \tag{1}$$

and

$$x_j = f(u_j) \tag{2}$$

where $p$ is the number of neurons in the previous layer, $w_{i0}$ is the threshold, $f(.)$ is an activation function, and $x$ represents the output of a neural unit.

The reverse activity of producing an input vector corresponding to a pre-determined output value in a feed forward network is referred to as network inversion [6]. The idea is similar to the backpropagation (BP) algorithm, where the error signals are propagated back to tell the weights the manner in which to change in order to decrease the output error

$$E = \frac{1}{2}\sum_{k=1}^{N_L}(t_k - y_k)^2 , \quad (3)$$

where $t_k$ is the target output, $y_k$ is the actual neural network output, and $N_L$ is the number of neurons in the output layer.

The inversion algorithm backpropagates the error signals to the input layer to update the activation values of input units so that the output error is minimized. Of course in the network inversion process, the network weights are frozen. The final input activation is obtained by $x_i^0 = f(u_i^0)$, which guarantees that the network input will be in the range of the activation function.

### *B. Query Based Learning with Jitter*

Consider a two-class classification problem where the network decision boundary is very close to the real decision boundary in some region of the input space, but deviates from it in another region. The objective is to teach the network to correct its decision boundary only where it deviates from the true one. When queried, the oracle will find that along the correct region of the network decision boundary, some of the inverted inputs lie on one side of the true boundary, and some on the other side. These points when used in retraining will result in an irregular boundary and inferior generalization. Along the deviating part, all inputs will lie in the same class, and the oracle will find the true output for these points.

Adding jitter (noise) to the input data without correcting the target output has shown the effect of smoothing the decision boundary by minimizing the gradient term at the training points [6], [7], [8], [9], [10], [11]. Depending on the variance of the noise, jitter has only effect on data points close to the decision boundary. Data points lying deep inside the class region are not affected. Therefore, training with jitter can be particularly beneficial when applied on query data since many of these points may already lie very close to the target boundary.

## Aerospace Application: Safe Escape System

In this section we present the application of query-based learning to a system that determines the safety envelope for ejection seats in military airplanes. The safe escape envelope is determined as a function of airplane velocity, attitude and other parameters. It is a classic two-category problem where the parameters fall in either a safe or unsafe classification. The system provides this ejection safety information to crewmembers or other systems in real-time. Ejection safety is determined via high fidelity ejection seat simulation using EASY5® software. The processing time that it takes to complete one ejection scenario precludes implementation of the simulation in a real-time system as a means to determine the safe outcome. The alternative approach used by the system is to generate safe escape data using high fidelity simulation, and train a neural network to predict the ejection safety based on the simulation data.

The safe escape criteria are based on achieving a specified recovery velocity at a specified altitude. Eight flight parameters which affect ejection safety were used for the example shown herein. These include airplane pitch and roll angle, flight path angle (FPA), the angular rates p, q, and r, ejection altitude, and airplane speed.

Due to the high dimensionality of the input space, sampling with a reasonable resolution would give at least 302,330,880 points. Generating this amount of data was estimated to take approximately 35 years. Training a neural network using an advanced training algorithm like the extended Kalman filter takes about one hour per 10,000 points using a Pentium Pro processor. Thus it would be very difficult to train using a uniformly sampled the input space with a reasonable resolution. Since only a small fraction of the total data is generated, it is vital to intelligently select, generate, and use the data. Our approach is to generate data around the decision boundary. Learning to correctly classify these points guarantees the correct classification in the rest of the space. Query-based learning was used in an iterative manner, in order to generate data on the neural network decision boundary, and then migrate it toward the real decision boundary.

## Data Analysis: Causality Index

The relevance of the different input variables was investigated as a means to reduce the input dimensionality. A Causality Index (CI) [12], [13], [14] measures the dependence of a neural network output on each of its inputs. It is a heuristic approximation of detailed sensitivity analysis. It depends solely on the connection weights obtained after training, and is calculated as

$$CI_{ki} = \sum_{j=1}^{N_1} w_{kj} \cdot w_{ji} . \quad (4)$$

The normalized causality index is defined as

$$CI'_{ki} = \frac{\sum_{j=1}^{N_1} w_{kj} \cdot w_{ji}}{\sqrt{\frac{1}{N_L N_0} \sum_{m=1}^{N_L} \sum_{i=1}^{N_0} \left( \sum_{j=1}^{N_1} w_{mj} \cdot w_{ji} \right)^2}} \qquad (5)$$

where $N_L$ and $N_0$ are the number of output and input neurons respectively.

Several comments that can be made regarding this causality index include the following:

1. The causality index defined in (4) measures the average dependence of output node $k$ on input node $i$.

2. A positive causality index means that an increase in the input causes an increase in the output and vice versa. A negative causality index means that an increase in the input causes a decrease in the output and vice versa. In our decision problem, a positive causality index means that increasing the corresponding input makes the system safer, while decreasing that input would decrease the system safety.

3. While a large causality index indicates that the corresponding input has big effect on the output, a small one does not necessarily mean the irrelevance of the corresponding input. An input variable may have a small causality index if the relation between the output and the input is symmetric, (e.g. the roll angle at zero angular rates). In this case, the causality index is positive in one half space, and negative in the other half space. The net average is zero, (or very close to zero, since practically, networks never exactly models the real relationship).

4. Though this index depends on the neural network training, our results using different networks were consistent.

## Experimental Results

A feed-forward multi-layer perceptron with a bipolar sigmoid activation function was used in all layers. The extended Kalman filter method was used for training due to its speed and capability of escaping local minima. Details of the Extended Kalman Filtering algorithm and example of its application to recurrent neural networks can be found in [15] and [16] respectively. Three data sets are used, one for training, one for validation (during training), and one for testing. The optimum architecture was determined to be a single layer network with 5 hidden neurons.

### *A. Experimental Results of the Causality Index*

The causality index was calculated for different neural network architectures trained on the safe escape data using different training sets. The Causality Indices averaged over 4 cases were as follows: pitch: 0.129, roll:-0.022, FPA:2.333, p:-0.032, q:-0.205, r:-0.069, altitude:1.33, velocity:-0.552. The conclusion from this is is that the FPA and the altitude have large positive influence on the output, and the velocity has a large negative influence on the output. The rest of the inputs having smaller causality indices with inconsistent signs indicate. either a small effect on the output and/or their non-monotonic relationship with the output. The three inputs with the largest causality indices were plotted in order to visualize the data as shown in Figure 1. The plot presents the projection of the 8D space into the 3D space. Plotting the parameters with the largest causality indices assists in reducing the overlap shown between the two classes. The causality index can also assist in establishing the sampling resolution for the different input parameters. Usually a higher sampling resolution is chosen for the parameters with higher causality index, as demonstrated in the experimental results.

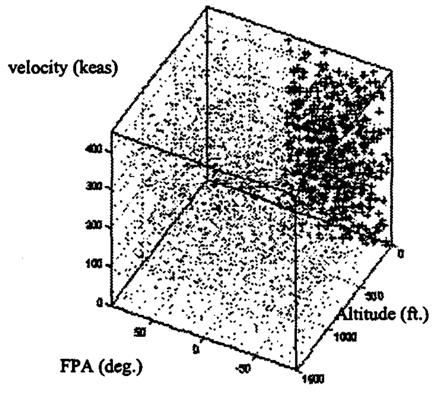

Figure 1 Classes Shown in 3D Projection

For the purpose of comparison, the performance of QBL was initially assessed using a problem where the input space could be scanned with sufficient resolution, and still have a training data set of manageable size. The angular rates were fixed to zero, and thus reduced the input space to 5 dimensions. Two cases have been compared. In the first case, a neural network has been trained on a high-resolution data set without QBL. In the second case, the network has been trained on a sparse data set, followed by QBL. In higher resolution case, the number of samples used for pitch angle, roll angle, FPA, altitude and velocity were 6, 6, 9, 30, and 9 respectively. This resulted in training, validation, and testing each containing about 86,000 points. In the lower resolution case, 3, 3, 5, 15, and 5 samples were used respectively. Thus we started with about 3300 points in each data set. A higher resolution has been chosen for the FPA, altitude and velocity because of their higher causality index, compared with the other parameters, and also because of the availability of high resolution measurements of these parameters. In both cases, the test set was generated by scanning the input parameters using a uniform step. In the validation and training sets, uniform noise has been added to the step size with amplitude equal to ¼ and ½ the step size respectively. Table 1 summarizes the performance with and without query. In each case, the training was stopped at the epoch with minimum number of classification errors on the validation set (see Figure 2). For case 3 (QBL), shown in Table 1, the query data was added to the original data, then, training was continued, starting with the set of weights obtained from the first training stage. QBL is most efficient when the network is originally only roughly trained. Therefore, we had to stop the training using the sparse data early enough, before inverting the network. The error rate shown in Table 1 is the total number of classification errors divided by the number of data points in the test set. The best result was obtained after network inversion and using the query data.

Table 1. Five Input Case Results

| Case number | Size of Training Set | | | Error Rate |
|---|---|---|---|---|
| | Uniform sampling | Inversion data | Total | |
| 1 | 86046 | 0 | 86046 | 2.21% |
| 2 | 3317 | 0 | 3317 | 1.69% |
| 3 | 3317 | 551 | 3868 | 1.63% |

## B. *The Full Dimension Problem*

In the full dimensional problem all 8 input parameters are allowed to change. Again, the network is trained using the validation technique described above, to determine the optimum number of training epochs. The best network architecture was found to contain one hidden layer with 5 neurons.

QBL was found effective when we started with only a small and sparse data set as shown in Table 2. A larger training set of 2869 samples was also generated using the sampling rate of 2 along the pitch and roll angles, and 3 samples along the other parameters. Here the sampling rates are not related to the causality index. The angular rates have a higher sampling rate than the attitude parameters (pitch and roll angles) because 3 was the minimum number of samples such that the network could learn the roll symmetry at zero angular rates. We used the above validation technique as a stopping criterion. Table 3 shows a slightly larger decrease in the error of the validation set after query, compared with the test set in the second case. Comparing the second and third case shows that combining the query data with the original data has better result than just using the query data alone.

Table 2 Two-Stage Query-Based Learning Results

| Case number | Size of Training Set | | | Error Rate |
|---|---|---|---|---|
| | Uniform sampling | Inversion data | Total | |
| 1 | 100 | 0 | 100 | 11.7% |
| 2 | 0 | 485 | 485 | 3.89% |
| 3 | 0 | 461 | 461 | 3.36% |

Table 3 Results of Training With Larger Data Set

| # | Size of Training Set | | | Test Set Error Rate | Valid. Set Error Rate |
|---|---|---|---|---|---|
| | Uniform sampling | Inversion data | Total | | |
| 1 | 2869 | 0 | 2869 | 2.69% | 2.59% |
| 2 | 2869 | 595 | 3464 | 2.49% | 2.15% |
| 3 | 0 | 595 | 595 | 2.88% | 2.25% |

## C. *Training with Jitter*

As previously discussed, adding a small amount of jitter to the query data can increase the benefit of the query based-learning. In three cases, the network has 8 inputs. In the last two, the angular rates parameters are fixed to zero. In two cases the noise has uniform distribution with range 0.01 and 0.001 respectively and in the third case, the noise has normal distribution with 0.0001 standard deviation. In all cases training with only query data has actually worsened the results, but adding a very small amount of jitter had a slight beneficial effect. In one case, the validation error was decreased from 2.2% to 2.1%, and in another case, the total error was decreased from 2.41% to 2.38%.

### *D. Minimizing Misclassifications*

Misclassifications occur when the neural network declares safe ejection, while the ejection is actually unsafe. These cases are worse than the false alarms where the network predicts an unsafe prediction when actual ejection is safe. It is desired to minimize misclassification rate, even on the expense of a reasonable increase in the false alarm rate. One way to bias the network decision toward more conservative safe decisions is to present more unsafe examples in the training set. In Table 3 case 2, 595 inversion data points were generated, and only the unsafe instances (234 points) were added to the original data set of case 1. A considerable decrease in the number of misclassification is seen (23 to 15), but at the expense a slight increase in the total number of errors.

## Conclusion

Query-based learning allows the training data to be utilized more efficiently, which is necessary for problems where data generation is expensive. The effectiveness of QBL has been shown for the aircraft safe escape classification problem. For this classification problem, the neural network was inverted to find inputs near the network decision boundary. It was shown that after querying, the oracle data obtained from only one class could be used to *minimize misclassifications, which is an import* consideration for many applications. In utilizing newly acquired query data, both the original and the query data can be used to re-train the network, or one can retrain the network with the query data only. Additionally, the network can be incrementally retrained (i.e. re-training the network starting with the previous weights), or can be reinitialized before retraining.

The causality index has been shown to provide valuable insight into the sensitivity of the input variables to the safe escape status. It can be used for guidance in determining the sampling resolution of the chosen input variables or selecting which parameters to use for visualizing the data.

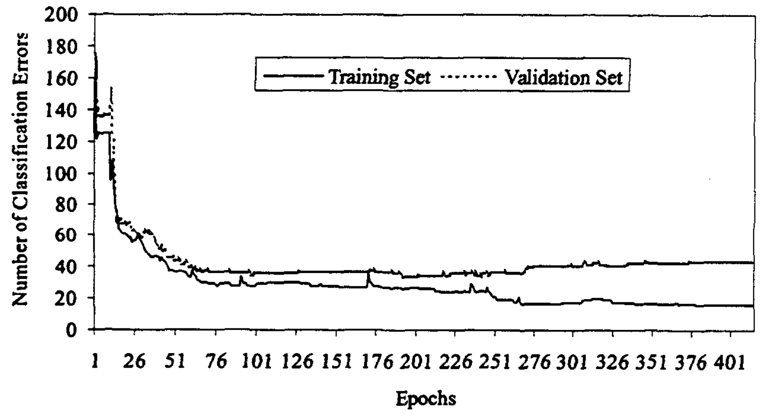

Figure 2 Classification Errors for Table 1 Case 2

## References

[1] J. Hwang, J. Choi, S. Oh, and R. Marks II, "Query-based learning applied to partially trained multilayer perceptrons," *IEEE Trans. on Neural Networks*, vol. 2, no. 1, pp. 131-136, Jan. 1991.

[2] J. Choi, "Efficient learning in artificial neural network," Ph.D. diss., Dept. of Electrical Engineering, Univ. of Washington, 1990.

[3] J. Hwang, J. Choi, S. Oh, and R. Marks II, "Query learning based on boundary search and gradient computation of trained multilayer perceptrons," in *Proc. International Joint Conf. on Neural Networks*, San Diego, 1990, pp. 57-62.

[4] S. Oh, R. Marks II, and M. El-Sharkawi, "Query based learning in a multilayered perceptron in the presence of data jitter," in *Proc. First International Forum on Applications of Neural Networks to Power Systems*, pp. 57-62.

[5] J. Hwang, H. Li, "Interactive query learning for isolated speech recognition," in *Proc. IEEE-SP Workshop on Neural Networks for Signal Processing II*, pp. 93-102.

[6] A. Linden, and J. Kindermann, "Inversion of multilayer nets," in *Proc. Int. Joint Conf. Neural Networks,* Washington D.C., vol. II, 1989, pp. 425-430.

[7] R. Reed, R. J. Marks II, and S. Oh, "Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter," *IEEE Trans. Neural Networks*, vol. 6, no. 3, pp. 529-538, 1995

[8] R. Reed, S. Oh, and R. J. Marks II, "An equivalence between sigmoidal gain scaling and training with noisy (jittered) input data," in *Proc. RNNS/IEEE Symp. Neuroinformatics Neurocomputing,* Rostov-on-Don, Russia, 1992.

[9] S. Oh, R. J. Marks, II, and M. A. El-Sharkawi, "Query based learning in a multilayered perceptron in the presence of data jitter," in *Applications of Neural Networks to Power Systems,* M. S. El-Sharkawi and R. J. Marks, II, Eds. New York: IEEE Press, 1991, pp. 72-75.

[10] C. H. Séquin and R. D. Clay, "Fault tolerance in feed-forward artificial neural networks," in *Neural Networks: Concepts, Applications, and Implementations,* P. Antognetti and V. Milutinovié, Eds. Englewood Cliffs, NJ: Prentice-Hall, vol. IV, 1991, pp. 111-141.

[11] J. I. Minnix, "Fault tolerance of the backpropagation neural network trained on noisy inputs," in *Proc. Int. Joint Conf. Neural Networks,* Baltimore, vol. I, 1992, pp. 847-852.

[12] K. Baba, I. Enbutu, and M. Yoda, "Explicit representation of knowledge acquired from plant historical data using neural network," *Proc. Int. Joint Conf. on Neural Networks,* Washington, vol. II, 1992, pp. 579-583.

[13] Z. Boger, "Knowledge extraction from artificial neural networks models," *Proc. IEEE Int. Conf. on Systems, Man & Cybernetics,* Orlando, 1997, pp. 3030-3035.

[14] B. Mak and R. Blanning, "An Empirical Measure of Element Contribution in Neural Networks," *IEEE Trans. Systems, Man and Cybernetics*, vol. 28, no. 4, pp. 561-564, 1998.

[15] G. Puskorius, and L. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks," *IEEE Trans. on Neural Networks*, vol. 5, no. 2, pp. 279-297, 1994.

[16] E. W. Saad, D. V. Prokhorov and D. C. Wunsch II, "Comparative Study of Stock Trend Prediction Using Time Delay, Recurrent and Probabilistic Neural Networks," *IEEE Trans. Neural Networks*, vol. 9, no. 6, 1998.