Application of Artificial Neural Networks in Classification of Autism Diagnosis Based on Gene Expression Signatures

Kathleen T Quach
Department of Neuroscience
University of California, San Diego
San Diego, CA 92093
ktquach@ucsd.edu

Abstract

Clinical genetic tests of autism spectrum disorder (ASD) have largely been difficult to develop due to the complex and heterogeneous nature of autism's dependence on genetic factors. No single genetic component accounts for all cases of autism, and large genetic variation is observed across individuals. Thus, genetic data must be analyzed on a genomic scale in the creation of a generalized autism diagnosis method based on genetics. Artificial neural networks, when presented with enough examples, can learn to correctly classify large sets of input. This project utilizes multilayer perceptron network architecture with a backpropagation training method in order to develop a trained network that can distinguish toddlers as having ASD or not when fed large microarray genetic expression data. Successful identification of ASD toddlers was able to be accomplished at a 73% rate.

1 Introduction

Long term prognosis of individuals with autism spectrum disorder (ASD) can improve dramatically with early intervention [1][2], thus making early detection critical to the treatment of ASD. Unfortunately, the current most reliable test for early autism diagnosis, the Autism Diagnostic Observation Schedule-Toddler Module [3], is based on behavioral observations, which limits age of detection to when clinically defined behavioral symptoms begin to manifest. Rather than relying on behavioral data, diagnostic approaches based on genetics may further decrease the age at which ASD can be detected. Numerous studies have indicated that autism has a strong genetic basis, although the genetic mechanisms that lead to the highly variable ASD phenotypes are complex and poorly understood.

Supervised machine learning, such as artificial neural networks (ANN), has been successfully implemented to predict cancer classifications that typically elude routine histology [4] [5]. The goal of this project is to train an ANN to return output of correct ASD diagnoses when given input microarray data collected from toddlers with ASD or typically developing (TD) toddlers. The ANN in this project will be built using Matlab's Neural Network Toolbox [6] with the specific approach of applying the feedforward resilient backpropagation learning algorithm to a multilayer perceptron. Multilayer perceptrons were chosen on the assumption that subjects with ASD and those without ASD are not linearly separable. A major goal of this project is to minimize the size of the input set for optimal classification performance.

2 Methods

2.1 Microarray Input

Microarray genome-wide gene expression levels were obtained using the Illumina Humanht 12 v4 Expression BeadChip ((Illumina, San Diego, CA, USA). Gene expression levels were collected from 222 toddlers (137 ASD, 85 TD) aged 18-36 months. Of the 222 subjects, 150 were used for training and 72 were used for independent testing. Differential expression levels were logged to give equal weight to ratios less than 1 and ratios greater than one. 835 differentially expressed genes were identified. To reduce the dimensionality of the data, principal component analysis was done and the top 10 principal components were selected such that at least 80% of the variance is captured. These data manipulations resulted in a x 10 matrix that can then be fed into the multilayer perceptron for training.

2.2 Cross-Validation and Committee Voting

As part of the training process, a portion of the training set is set aside for validation such that the network will know when to stop training to prevent overtraining and maintain ability to generalize. Training is stopped when performance on the validation set fails to improve after 6 iterations of training. 10-fold cross-validation groups were randomly selected such that 1 group will be used for validation and the other 9 groups will be used for learning. Subsequent networks will cycle through each of the 10 cross-validation groups to use as the validation set. Thus, 10 networks will be trained per instance of cross-validation. After training, an independent sample of 72 test subjects was fed into the network to test the network. To prevent effects of uneven random partitioning of data, cross-validation will be repeated many times and the outputs from networks will be averaged across these groups to determine a committee vote for classification. A committee size of 100 was used, for a total of 1000 networks trained per committee vote.

2.3 Network Architecture

Networks with multiple layers are capable of solving non-linearly separable classification problems. The feedforward multilayer perceptron architecture used in this project only had one hidden layer between the input and output layer, for a total of 2 layers of neurons (Figure 1). The first layer had 5 neurons for receiving input, and the second layer had 2 neurons for producing output. One output neuron classifies TD subjects as 0 and ASD subjects as 1, while the other neuron classifies ASD subjects as 0 and TD subjects as 1. Even the ASD/TD classification is 2-bit information and can be explained with only one neuron, 2 output neurons were used to observe training discrepancies between the 2 output neurons, since they did not always perform equally.

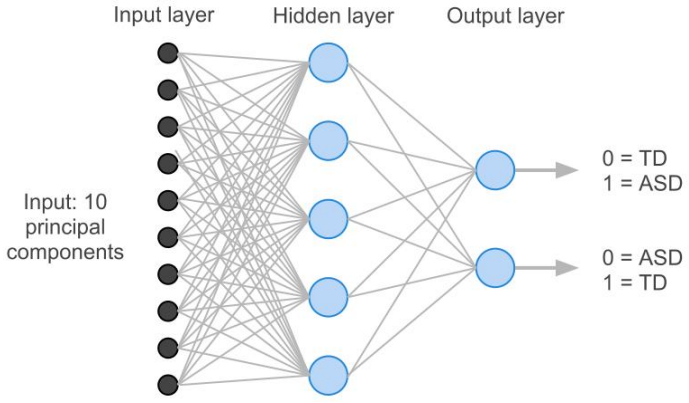

Figure 1: Network architecture

Information flow from input to output first occurs by multiplying each input into each

neuron of the hidden layer by a weight value. The output (a_1) of the hidden layer is calculated by multiplying a matrix of weights (W_1) from each input to each neuron in the hidden layer by the matrix of input values (p) for one subject from plus a matrix of bias values (b_1) for each neuron in the hidden layer:

$$a_1 = tansig(W_1 p + b_1),$$
$$tansig(n) = \frac{e^n - e^{-n}}{e^n + e^{-n}}$$

The output (a_2) of the output layer is calculated by multiplying a matrix of weights (W_2) from each output of the hidden layer to each neuron in the output layer by the matrix of output values from the hidden layer (a_1) for one subject from plus a matrix of bias values (b_2) for each neuron in the output layer:

$$a_2 = purelin(W_2 a_1 + b_2),$$
$$purelin(n) = n$$

The transfer function $tansig$ was chosen for the hidden layer because it is differentiable and has inputs and outputs that can be any real number.

2.4 Training

Each network will be trained using the resilient backpropagation algorithm [7][8] in the Matlab Neural Network Toolbox. Backpropagation is a method of updating weights and biases after each example of input is fed into the network. First, the approximate performance index is calculated as:

$$\hat{F}(x) = e^T(k)\, e(k) = (t(k) - a(k))^T.$$

where t is the target value, a is the final output value, k is the input expression value for one gene, and T is a superscript indicating the connection.

The derivative of the performance index is used to calculate the sensitivity of the performance index to changes in each element of the net input (n^m) at neuron layer m.

$$s^m = \frac{\partial \hat{F}}{\partial n^m}$$

$$s^M = -2\hat{F}^M(n^M)(t - a)$$

$$s^m = \hat{F}^m(n^m)(W^{m+1})^T s^{m+1}, for\ m = M-1, \ldots, 2, 1.$$

The term “resilient” in resilient backpropagation refers to the method for updating weights and biases. The sigmoid transfer function used in this network is characterized by the fact that their slope must approach zero as the input becomes large. This results in a gradient that can have a very small magnitude and thus result in small changes in weights and biases. To circumvent these problems, only the sign, but not the magnitude, of the derivative is used to determine a weight change. Each weight and bias update values increases by a factor of a

specified $\delta_{increase}$ whenever the derivative of the performance function with respect to that weight has the same sign for 2 successive iterations. Similarly, each weight and bias update values decreases by a factor of a specified $\delta_{decrease}$ whenever the derivative of the performance function with respect to that weight changes sign from the previous iteration. For the training of the networks in this project, $\delta_0 = 0.7$, $\delta_{increase} = 1.2$ and $\delta_{decrease} = 0.5$, and $\delta_{max} = 50$.

2.5 Gene Minimization

It was unknown a priori how many and which of the total 835 genes were relevant to ASD diagnosis, so the initial dataset was noisy. To reduce the noise of the dataset, a gene minimization algorithm [7] was implemented after training 1000 networks (10-fold cross-validation x 100 committee size) with the full 835 genes. Gene minimization entailed ranking each gene based on the output's sensitivity to each gene. Sensitivity (S) of output (o) with respect to each of the 835 gene variables (x_k) were calculated as:

$$S_k = \frac{1}{N_s}\frac{1}{N_o}\sum_{s=1}^{N_s}\sum_{i=1}^{N_o}\left|\frac{\partial o_i}{\partial x_k}\right|$$

where N_s is the number of subjects (150) and N_o is the number of outputs (2). Genes were ranked based on their sensitivity values, and then a subset of the highest ranked genes was used to train a new committee of networks.

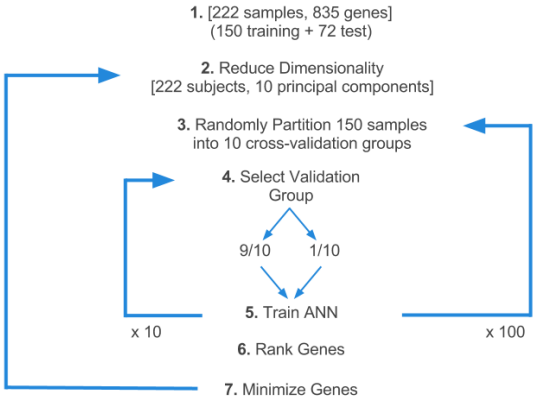

Figure 2: Methods overview

3 Results

3.1 Principal Component Analysis

Using principal component analysis, the 835 genes were reduced to 10 principal components that collectively captured 84.64% of the variance (Figure 3).

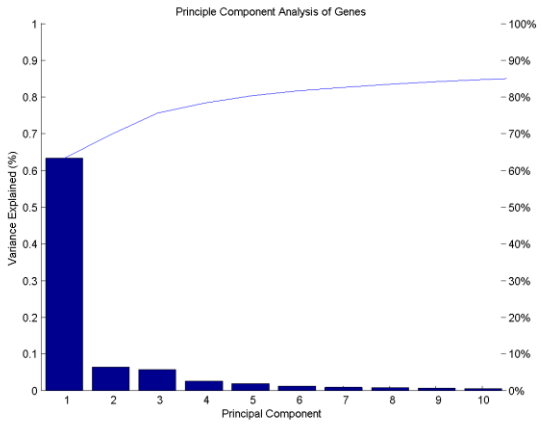

Figure 3: Principal component analysis

3.2 Training with 835 Genes

Training with 835 genes produced a committee voting that produced 68.0556% correct output from Output Neuron 1 (Figure 4) and 68.0556% correct output from Output Neuron 2 (Figure 5).

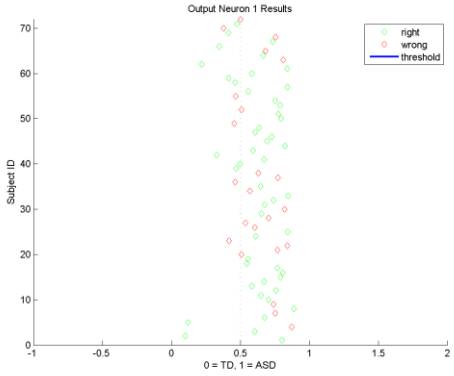

Figure 4: Output Neuron 1 results using 835 genes

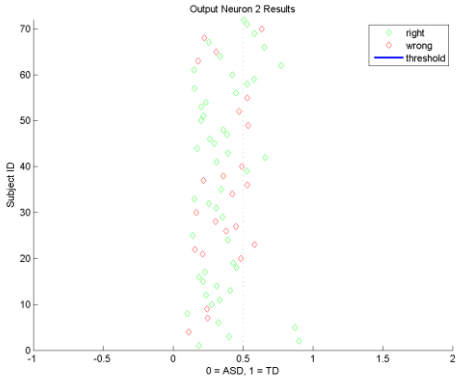

Figure 5: Output Neuron 2 results using 835 genes

3.2 Gene Minimization

Sensitivities were calculated for each gene, and various subsets of the highest ranking genes were used for training the networks. Performance peaked when 821 genes were used and declined quickly with decreasing number of genes (Figure 6). Training with 821 genes produced a committee voting that produced 72.2222% correct output from Output Neuron 1 (Figure 7) and 70.8333% correct output from Output Neuron 2 (Figure 8).

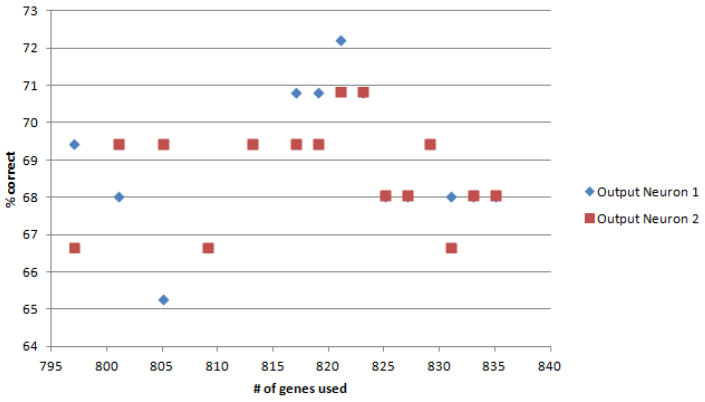

Figure 6: Gene Minimization

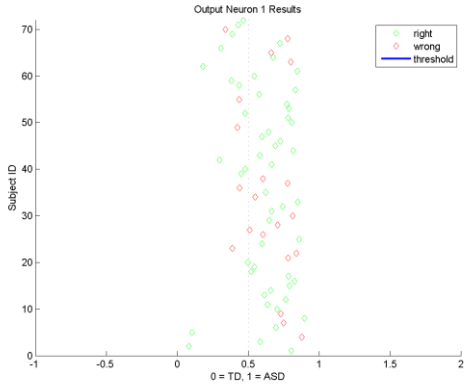

Figure 7: Output Neuron 1 results using 821 genes

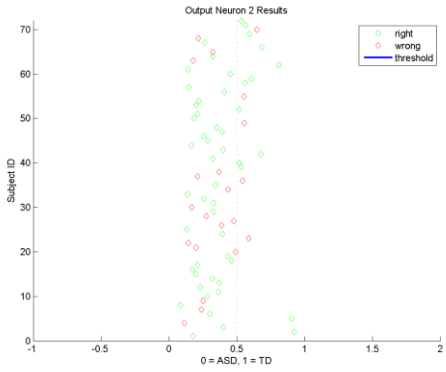

Figure 8: Output Neuron 2 results using 821 genes

4 Discussion

While this project's artificial neural network architecture was able to increase classification above random change (50%), the highest success rate achieved (72.222%) is not high enough to be of substantial clinical application. Even though some improvement (4.1666%) in correct classification was obtained through gene minimization, the gain was not substantial. The likely problem with the classification problem is that ASD is a very heterogeneous disorder that may have subgroups with drastically different genetic expression signatures. To improve classification, it may be useful to stratify the ASD class into subgroups and enrich the input set with clinical measures.

References

[1] Dawson, G., & Osterling, J. (1997). "Early intervention in autism." The effectiveness of early intervention, 307-326.

[2] Rogers, S. J. (1998). "Empirically supported comprehensive treatments for young children with autism." Journal of Clinical Child Psychology, 27(2), 168-179.

[3] Luyster, R., Guthrie, W., Gotham, K., Risi, S., DiLavore, P., & Lord, C. (2009). "The Autism Diagnostic Observation Schedule-Toddler module: Preliminary findings using a modified version of the ADOS."

[4] Greer, B. T., & Khan, J. (2004). "Diagnostic classification of cancer using DNA microarrays and artificial intelligence." Annals of the New York Academy of Sciences, 1020(1), 49-66.

[5] Greer, B., & Khan, J. (2007). Online analysis of microarray data using artificial neural networks. *METHODS IN MOLECULAR BIOLOGY-CLIFTON THEN TOTOWA-*, *377*, 61

[6] Demuth, H., & Beale, M. (1993). Neural network toolbox for use with MATLAB.

[7] Khan, J.,Wei, J. S., Ringner, M., et al. (2001) Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nat.Med.* **7,** 673–679

[8] Demuth, H., Beale, M., & Hagan, M. (1992). Neural Network Toolbox™ 6. *User Guide, COPYRIGHT, 2008.*