Neural Networks Approaches for Discovering the Learnable Correlation between Gene Function and Gene Expression in Mouse

Emad A. M. Andrews, Quaid Morris, Anthony J. Bonner

Department of Computer Science
University of Toronto
Toronto, ON.
emad@cs.toronto.edu

Abstract. Identifying gene function has many useful applications. Identifying gene function based on gene expression data is much easier in prokaryotes than eukaryotes due to the relatively simple structure of prokaryotes. Recent studies have shown that there is a strong learnable correlation between gene function and gene expression. In previous work, we presented novel clustering and Neural Network (NN) approaches for predicting mouse gene functions from gene expression. In this paper, we build on that work to significantly improve the clustering distribution and the network prediction error by using a different clustering algorithm along with a new NN training technique. Our results show that neural networks can be extremely useful in this area. We present the improved results along with comparisons.

Keywords: Gene function prediction; Self organizing maps (SOM); Multilayer perceptrons (MLP), Gene expression; Neural networks

1 Introduction

1.1. Gene function prediction

Gene function prediction is one of the primary goals of Bioinformatics. Identifying gene function can be extremely useful in many ways, especially in Gene Therapy [18]. Identifying gene function in prokaryotes is much easier than eukaryotes due to their lower structural complexity and number of genes. Tissue-specific gene expression is the most widely used predictor for gene function in mammals; for example, genes expressed in tongue are most probably involved in tasting. However, this method has not been scientifically justified. Recent studies have shown that there is a strong learnable correlation between gene function and gene expression [22].

According to Zhang et al, predicting gene function based on its expression is more effective than using tissue-specific function as a guide [22]. In previous work, we presented novel clustering and Neural Network (NN) approaches for predicting mouse gene functions from gene expression [2] .In this paper, we improve on that work.

1.2. Neural Networks as a machine learning approach

The machine learning technique used in this work is a NN for both prediction and clustering. As we will describe in the next section, Support Vector Machines (SVM) and graphical models have been used in related work. In spite of their scalability and huge learning and classification capabilities, NN have not, to our knowledge, been used to predict gene function in higher order organisms than yeast. According to Baldi, NN are superior classification machines, and in theory they can approximate *any* reasonable function to *any* degree of required precision [3]. In the same source, the authors showed that NN, with the right modeling, can be extremely useful in various Bioinformatics problems, such as sequence encoding and correlation, prediction of protein secondary structure, and prediction of signal peptides and their cleavage sites.

In this work, we used the Back-Propagation algorithm (BP) to predict gene functions from gene expression. The earliest description of BP was presented by Werbos in 1974 [21]; then, it was independently rediscovered by Le Cun et al [12]. Multilayer perceptrons (MLP) is perhaps the most famous implementation of BP. It has been very successfully used in various domains, such as prediction, function approximation and classification. For classification, MLP is considered a super-regression machine that can draw complicated decision boundaries between nonlinearly separable patterns [7]. The nonlinearity power of MLP is due to the fact that all its neurons use a nonlinear activation function to calculate their outputs.

Unlike SVM, MLP can be implemented in two versions with respect to the output: first is the Softmax version, which has only one output neuron to be activated with a certain input, while the other has more than one output activated at a time, which is a very powerful feature of MLP as it correlates all the functions a gene might be involved in. In this work we used MLP in three different ways: Firstly, the 'binary method', where we used MLP as a binary classifier, in which, the input is gene expression measurements and the desired output is a single Gene Ontology 'Biological Process' (GO-BP) category. Having 922 different GO-BP categories, this will result in 922 binary classifiers. In the second approach, the 'multi-output method', we unleashed the MLP power by training it with the same input as before but with all 922 GO-BP categories as the desired output. This approach not only predicts gene function from gene expression but also ties the expression to all possible functions simultaneously. By using this method, the prediction that a gene is being involved in a certain function is not independent any more, but it depends also on the prediction of the rest of the functions the NN knows about; which is very useful and desired results from a biological point of view. Finally, we introduced the 'clustering method', in which we clustered genes annotations into distinct groups and used those groups as the desired output for the network.

1.3. Relation to previous work and results summary

In [2], we presented promising results for the three methods with some drawbacks for the multi-output and the clustering methods. In this paper, we used a highly optimized MLP implementation and different clustering algorithm to overcome these drawbacks. As for the multi-output method, we used Generalized Feed Forward (GFF) MLP with two-phase training mechanism which resulted in about 10 times more accurate prediction in less than half the training time. For the clustering method, we replaced the simple K-means algorithm [13, 17] with Kohonen Self Organizing Maps (SOM) [9, 10] which resulted in much better populated clusters and hence a more accurate MLP prediction on those clusters.

Our results show that NN is a good candidate for predicting gene function from gene expression. As a binary classifier, MLP is very successful and has the advantage of eliminating the need for the kernel function needed by SVM. As a multi-output regression machine, using the two-phase training algorithm, GFF MLP is able to predict the whole set of 922 possible functions for each gene with a remarkable accuracy. Finally, the highly adaptive and competitive nature of SOM resulted in a more uniform clustering distribution which, in turn, helped the MLP to give better generalization in predicting genes groups. Results comparison will be presented in the subsequent results sections.

1.4. Literature review

Using gene expression for predicting gene functions has been proven to be effective in prokaryotes. For eukaryotes, tissue-specific expression is widely used for predicting gene functions [22]. However, the correlation between gene expression, gene sequence and gene functions has attracted the attention of many bioinformaticians. In [4], the authors used a probabilistic approach to correlate gene sequence and expression. In [18], there is an investigation of gene function by identifying interactions between a protein and other macromolecules. Some bioinformaticians tried to predict gene functions only from the pattern of the GO-BP annotation categories vocabulary mentioned at the Gene Ontology (GO) database without considering genes expression. In [8], King et al used Bayesian networks and decision trees to model the relationship among different GO-BP categories vocabulary, only 41 genes out of 100 manually assessed were judged to be true.

SVMs are the most widely used machine learning approach to predict genes functions from genes sequences and expressions. In [5], the authors used SVM to functionally classify genes by using gene expression data from DNA microarray. They used gene expression of 2,467 genes with known function from the budding yeast *Saccharomyce cerevisiae* measured in 79 different DNA microarrays. Still, this work used yeast genes not higher order organisms; also, the classification is binary and the performance is assessed against other basic classification algorithms like Fisher's linear discriminate and decision trees.

In 2004, Vinayagam et al used SVM to predict gene function based on the nucleotide sequences mentioned in the GO database. Although there is no gene expression involved, this research is relevant because it uses the same method to

predict gene function from tissues taken from different species ranging from as simple as yeast to as complicated as mouse [19].

The first attempt to use NN to predict gene function class from gene expression was in 2002 by Mateos et al [15]. In that work, the classes used to train the MLP were taken from the Yeast Genome Database at the Munich Information Center for Protein Sequences (MIPS) functional catalog.

The most related work is by Zhang et al in [22], where gene expression is used as SVM input to predict gene function category. This work is very important in many ways. First of all, it is for mouse genes which are homologous to human genes. Also, the authors showed the shortcomings of depending on tissue-specific information to predict gene function and the crucial need for another method. The dataset of the microarray generated for this work is a public resource for mammalian functional genomics. The authors stated four reasons that support the integrity and validity of their data. Finally, it uses specific 922 GO-BP categories mentioned in [25]. The authors provided a mechanism to measure the accuracy of their predictions through a recall value. The reader is directed to [24] and [22] for full details about this work.

Instead of using SVM, we used MLP for prediction and SOM for clustering. Our work goes beyond binary classification by predicting all possible functions the gene might be involved in simultaneously, which expresses the relation between the expression level and all possible functions. To decrease the required training time, we used SOM to group genes based on their annotations into distinct annotation groups and then trained the MLP to predict gene group number instead of predicting the whole 922 vector of annotations.

2 Data preparation

Data used for this paper is provided by Zhang et al [24]. The reader is directed to [22] for a complete explanation of the data gathering process. The authors designed their own microarray to contain nearly 40,000 genes from 55 different mouse tissues. From those 40,000 only 21,266 confidently detected transcripts were extracted. Those are the ones that exceeded the 99^{th} percentile of intensities from the negative controls. From those 21,266 genes, only 7,388 genes have at least one specific GO-BP category; the rest were considered negative genes because their annotations were too general. We will refer to these 7,388 genes as the *annotated genes*.

Our data preparation process has three stages: first, extracting the expression microarray data of the annotated genes; second, constructing a binary matrix that describes the annotations associated with each of those genes, and finally clustering those binary vectors.

As an input to this data preparation process, two matrices were downloaded from [24]: the (21,266 X 55) matrix of the microarray data normalized and centered by subtracting the mean, and the two-column annotation matrix which has genes in one column and their annotations in the other column. Then, the two-column annotation file is parsed to extract all distinct genes and all distinct annotations. Finally, a two dimensional binary matrix (7,388 X 922) is created; in which, each annotated gene

has a binary vector denoting all of its 922 annotations. If the gene is annotated in a GO-BP category, its corresponding bit is set to 1 otherwise to 0.

This binary matrix is the desired output of the MLP. If we chose only one GO-BP category as a desired output, the MLP will work as a binary classifier; if we considered more than one column as an output, the MLP will have number of output neurons equal to the number of GO-BP categories we are considering.

3 Neural network topology selection

Selecting an appropriate network topology is one of the main difficulties of using NN in classification; that is why Wang et al. used Extreme Learning Machine (ELM) algorithm instead of NN in spite of the potential promising results of NN in classifying protein sequence [20]. In addition to network parameters like learning rate and momentum, NN topology is also determined by its size, synaptic weight connections, and the hidden-units activation function. By network size we mean the number of hidden layers and number of hidden units in each layer. Network size is a measure of system complexity and is directly proportional to the training time required. The less complex the network is, the less its tendency to memorize the training set. That is why MLP size reduction is always recommended when possible [7, 23]. In general, the most common way to reduce network size is weight pruning as mentioned by Zurada and Haykin [7, 23], which suggests removing the ineffective weights during training. Instead of pruning only the weights, we started with number of hidden units equal to twice the number of the input features. Then, without affecting the cross validation (CV) error, we kept dividing this number by 2. The number of input neurons is constant in all our networks and equals to the number of the microarray tissues which is 55.

Concerning the synaptic weight connections, there are two types of MLPs: the standard MLP in which each layer is fully connected to its next layer only; the second type is the Generalized Feed Forward GFF network which is a generalization of the MLP such that connections can jump over one or more layers. Connections that jump from the input layer directly to the output layer are usually called *shortcut weights*. In theory, an MLP can solve any problem that a generalized feedfoward network can solve [7]. In practice, however, generalized feed forward networks often solve the problem more efficiently. In our experiments, in the case of group number prediction, GFF required fewer training epochs than the MLP and showed fewer tendencies to memorize. Also, the GFF structure enabled us to use a two-phase training algorithm which resulted in a much faster training in the case of the multi-output method.

The main idea behind GFF is that both the shortcut weights and the output neurons try to learn the linear relations of the search space providing a starting point for the nonlinear hidden units to model the more complex areas of the same search space. Without shortcut weights, the hidden units will try equally hard for both the linear and non-linear relations of the search space which decreases the efficiency. It is wroth mentioning that GFF does not provide any improvement over the standard MLP in many cases. We used the sigmoidal activation function in all our work because no

negative output is desired. Also, the sigmoidal activation function is superior in classification [1,16, 20].

3.1 Regularization, convergence criteria and error measurement

Cross Validation CV is our main method for regularization and stopping. Before training starts, the data set is divided into 3 subsets: validation set, 5% of the whole dataset; testing set, 15% of the dataset and the remaining examples are used for training. Once the network starts, its initial state after 50 epochs is recorded and the whole training stops if the CV Mean Square Error (MSE) did not improve for 100 consecutive epochs. The best weights are saved once the CV error starts to increase and used for testing. The CV set changes with every epoch, and the test set is randomly chosen and kept hidden during training.

For error measurement, we based our comparison on the (MSE) because the actual error rate of classification sometimes becomes misleading depending on the output values [16]. The MSE is twice the average cost.

4 Neural networks as a binary classifier

In this approach we used NN as a binary classifier. The input is the gene microarray expression levels and the output is either 1 or 0 depending on whether the gene is predicted to be involved in the desired GO-BP category or not, respectively. A separate NN is needed for each gene function. So, 922 binary classifiers are needed to predict all GO-BP categories using this approach. The minimum topology size was 2 hidden layers with 100, and 50 hidden units (from input to output). Instead of using only one output for the network, we used two binary output neurons to better assess the network accuracy through a difference threshold. The ordered pair (0, 1) means the same as 0 and (1, 0) means the same as 1.

Due to space limitations, we present only prediction results for the GO-BP categories with maximum gene participation. Those categories are the best examples of network performance in this type of problem as they require the maximum generalization and hence are the hardest to predict. Table 1 summarizes the prediction results for the selected GO-BP categories. With some statistical analysis of the data, we found most of the categories have gene participation of less than 20; the maximum participation is 455 genes in the *"lipid metabolism [GO:0006629]"* category. It is clear from the way we prepared our data that the minimum participation will be 1 and never 0. Figure 1 shows the overall gene participation distribution. This highly skewed and non-uniform distribution of the desired output might make the network performance questionable because the maximum gene participation in any category is 6.15%. So, in many cases the network might tend to sacrifice the positive output for the sake of negative output and still the MSE will be low.

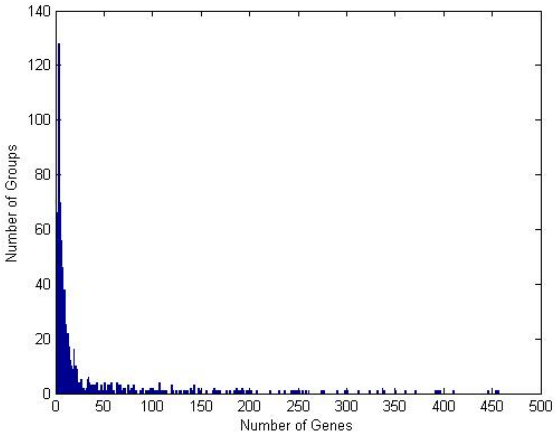

Fig. 1. Overall gene participation distribution for all categories

4.1 More expressive dataset

To prove our hypothesis and provide a fair assessment of the NN performance in this problem without getting affected by the skewed data distribution mentioned above, we prepared a balanced subset from our original data. The number of examples in this dataset is 990, containing nearly equal numbers of positive and negative genes for the *"lipid metabolism [GO:0006629]"* category. The network recorded a much lower training MSE because of the fewer training examples. The actual output is in Table 3.

4.2 Results

We performed binary classification for GO-BP categories with a gene participation of at least 350. That gave us 10 categories to predict. Table 1 summarizes the results for the 10 selected GO-BP categories.

Table 1. Binary prediction results for the selected GO-BP categories. Training MSE, CV MSE and Test MSE are divided by 10^{-3}

Function Name	GO-BP category	# Positive Genes	Training MSE	CV MSE	Test MSE	Training time in Sec.
lipid metabolism	GO:0006629	455	47.726	74.236	43.3793	195
Intracellular protein transport	GO:0006886	454	46.539	35.539	42.299	1054
Carbohydrate metabolism	GO:0005975	446	45.141	34.291	51.039	19281
Cation transport	GO:0006812	410	42.293	42.405	42.220	944
Response to abiotic stimulus	GO:0009628	396	38.608	27.702	58.423	38148

Response to pest/pathogen/parasite	GO:0009613	394	42.712	65.791	32.425	480
Cytoskeleton org. and Biogenesis	GO:0007010	392	41.041	29.260	41.503	1212
Neurogenesis	GO:0007399	371	46.374	46.374	33.157	384
Cell-cell signaling	GO:0007267	361	38.951	45.093	32.468	695
Mitotic cell cycle	GO:0000278	350	36.708	21.205	50.843	42360

Each row in table 1 represents an independent assessment of MLP as a binary classifier over different training and test sets, which eliminates the possibility that a certain choice of a test set led to above noted acceptable performance. To further confirm that the MLP performance is independent on a specific test set, the whole operation is repeated 10 times for the *lipid metabolism* category; each time a different test and CV sets are chosen. Test set size for this experiment is set to be 10% of the original dataset. Results are summarized in table 2. Mean, variance and standard deviation show that the MLP performance is not affected by choosing different test and CV sets.

Table 2. 10 different training trials for predicting the *Lipid Metabolism* category with different CV and testing sets every time. Mean, variance and standard deviation shows that the performance is independent on the test set selection.

	Training MSE	**CV MSE**	**Test MSE**	**Training time in sec**
Training trial 1	0.04759	0.060595	0.037616	600
Training trial 2	0.04601	0.048333	0.063285	801
Training trial 3	0.04757	0.047356	0.039626	660
Training trial 4	0.04741	0.048257	0.049865	900
Training trial 5	0.04862	0.047795	0.050367	780
Training trial 6	0.04659	0.047757	0.052177	880
Training trial 7	0.04597	0.047756	0.057675	806
Training trial 8	0.04884	0.048400	0.038508	816
Training trial 9	0.04743	0.047468	0.041631	836
Training trial 10	0.04778	0.047780	0.040702	912
Average	0.047381	0.0491497	0.0471452	799.1
Variance	9.335434 e-007	1.6295514 e-005	7.8703624 e-005	10058
Standard Deviation	0.0009662004	0.0040367	0.008871	100.29

Tables 1 and 2 reflect an acceptable performance for the MLP for predicting gene function from its expression level. The network successfully predicted genes functions in the 10 categories with test MSE accuracy in the range of [0.032425, 0.051039]. But we used a balanced dataset from the lipid metabolism [GO:0006629] category to assess the MLP performance in case of a harder problem in which a clear distinction between equal numbers of positive and negative genes must be made.
In this dataset, the NN achieved a training MSE of 10^{-7} in approximately 10 hours of training. The testing MSE was *0.247147*. The actual output of the testing results is summarized in the table 2 where D and A stand for the desired and actual outputs respectively. These results provide a fair assessment of the NN performance in this problem without getting affected by the skewed data distribution showed in Figure 4. Consequently, it proves our hypothesis that NN can be a promising tool to successfully predict gene function from expression level.

Table 3. Actual NN test results for the balanced sub-dataset. Each row is the desired and actual output for three different test cases. Bold lines separate different test cases.

D0	D1	A 0	A 1	D0	D1	A 0	A 1	D0	D1	A 0	A 1
1	0	0.99921	0.00078	1	0	0.42998	0.57001	1	0	0.99916	0.00083
1	0	1.00126	0.00127	0	1	0.39137	0.60863	1	0	0.00031	1.00030
0	1	0.0012	1.00120	1	0	0.99938	0.00061	0	1	0.99413	0.00586
0	1	0.00018	1.00017	0	1	0.99937	0.00062	1	0	0.99897	0.00103
1	0	1.00007	7.4E-05	0	1	0.09959	0.90041	0	1	0.99767	0.00232
0	1	8E-06	1.00000	1	0	0.0014	1.00139	1	0	0.00956	1.00956
1	0	0.77768	0.22231	0	1	0.99963	0.00036	0	1	0.99855	0.00144
1	0	0.98661	0.01338	1	0	0.99904	0.00095	0	1	0.25183	0.74817
0	1	0.99954	0.00045	0	1	0.00318	1.00318	1	0	0.99917	0.00082
1	0	1.00188	0.00188	0	1	0.31028	0.68972	0	1	0.96651	0.03348
1	0	0.00018	1.00017	0	1	1.00339	0.0034	1	0	0.12562	0.87437
1	0	0.99915	0.00084	1	0	0.99911	0.00088	0	1	0.00019	0.99980
1	0	0.03331	0.96668	0	1	0.99929	0.00070	1	0	0.98230	0.01769
1	0	0.98329	0.01670	0	1	0.00094	1.00093	1	0	0.99913	0.00086
0	1	0.00022	0.99977	0	1	0.00033	0.99966	1	0	0.99905	0.00094
1	0	0.99926	0.00073	1	0	1.00254	0.00255	0	1	0.99929	0.00070
1	0	1.00058	0.00059	1	0	1.00043	0.00044	1	0	0.00013	0.99986
0	1	1.0005	0.0005	0	1	0.00024	0.99975	1	0	0.99921	0.00078
1	0	0.99846	0.00153	0	1	0.99450	0.00549	1	0	0.99892	0.00107
0	1	0.00455	1.00454	1	0	1.00054	0.00055	1	0	0.99972	0.00027
1	0	0.99928	0.00071	0	1	0.99947	0.00052	1	0	0.99853	0.00147
0	1	0.00019	1.00019	1	0	0.99940	0.00059	1	0	0.99921	0.00078
1	0	0.97760	0.02239	1	0	0.99844	0.00156	1	0	0.23115	0.76885
1	0	0.99915	0.00084	1	0	0.99944	0.00055	1	0	0.11786	0.88213
1	0	0.99945	0.00054	0	1	0.97167	0.02832	0	1	1.00252	0.00252
0	1	0.00021	0.99978	1	0	0.99912	0.00087	1	0	1.00111	0.00112
0	1	0.00044	0.99955	1	0	1.00061	0.00062	0	1	0.99449	0.00550
0	1	0.99962	0.00037	1	0	0.98305	0.01694	0	1	0.00215	1.00215

5 Neural network with 922 outputs

In this approach we tried to use the full capability of NN to simultaneously predict all functions the gene might be involved in as a binary vector. The same input matrix is used but the desired output is a binary vector of 922 bits. We will briefly summarize our previous standard MLP results followed by the improved method and comparisons.

5.1. Standard MLP

Although the results were promising, the training time required was huge, exact training time required and number epochs are in table 4. The minimum topology size was 3 hidden layers with 200, 100, and 50 hidden units (from input to output), and 922 output units. Results and performance summary can be found in table 4. The 2 hidden layers topology used in case of binary classifier in section 4 was too simple to provide and acceptable performance in this case, even when we increased the number

of hidden units. An additional hidden layer boosted the network ability to draw more complex decision boundaries for the 922 different outputs and hence improved the results to a remarkable extend as shown in the results section. Adding a fourth hidden layer might further improve the results, but that will further increase the system complexity and training time required which in turn might compromise the method feasibility. Number of hidden units in each layer is determined using same technique mentioned in section 3 and in [1].

We used batch learning to reduce the training time. The network results along with the CV error progress showed that this method is effective. The next section outlines how we overcame the huge training time difficulty.

5.2. GFF network with two-phase training

The main idea behind GFF is dividing the network units into two sets with respect to their learning capabilities: the shortcut weights along with the output units will learn the linear (the easier) part of the search space, while the hidden units will model the more complex areas to reach the global minima or a satisfactory approximation. However, letting the two sets learn simultaneously will create kind of learning competition as an error update by one set might cancel part or all of the update done by the other set, which delays convergence.

The main idea behind our method is learning the problem search space in two phases. Phase one is freezing the hidden units and allowing only the shortcut weight along with the output units to learn. Without hidden layers, the search space will be convex and both the shortcut weights and the output units, with appropriate parameters, are guaranteed to converge relatively quickly. Phase two starts once phase one is completed by freezing the shortcut weights and allowing all other units to learn. Freezing the shortcut weights during this phase is just for the sake of speed because they reached their minimum energy already and cannot compete any more with the hidden units. Because the hidden weights are randomly initialized at this point, there is a spike increase in MSE which diminishes quickly, in one or two epochs, to use the point reached by phase one as a starting point.

Using this training algorithm, we achieved more than 10 times accuracy in less than half the training time. Table 4 summarizes this method's results along with standard MLP results for comparison.

Table 4. Comparison with previous results

	Training MSE	**CV MSE**	**Test MSE**	**Training time in Sec**	**Epochs**
Standard MLP	0.061335	0.061616	0.061778	211673	1059
GFF with 2 phases training	0.00523	0.00484	0.00455	133200	13118

6 Neural network for gene cluster prediction

In some cases, predicting all gene functions is very difficult, especially if the microarray data is noisy or the function itself did not appear in many training examples. As a result, clustering a gene among the most similar genes in terms of function is still an important Bioinformatics application. In this approach, we used clustering algorithms to group the binary vectors, denoting the genes possible annotations, into k distinct groups. Instead of predicting the whole binary vector, the MLP tries to predict the group number the gene belongs to based on its microarray expression level. Unlike many other studies, we do not cluster gene expression profiles, but we cluster binary vectors of functions annotations without any guidance from the expression levels values. The expression level value will be used afterwards by the NN to predict the cluster the gene should belong to. A suggested future work is to compare the resultant clusters in this work with those resulting from clustering genes expression levels to be able to conclude possible relations between the two clustering techniques.

In general, cluster analysis involves two main tasks: 1) determining the appropriate number of clusters and 2) assigning each data point to one and only one cluster. In this work we focused mainly on the appropriate assignments to clusters to generate as uniform membership distribution as possible. This uniform membership distribution among clusters is necessary to be able to fairly judge the NN ability to solve such a problem. We achieved this uniform distribution by trying different clustering algorithms and fine tuning their parameters as we will see.

As a result, there is no biological preference towards number of clusters used in this study. We did not focus on biologically inferring the number of clusters prior to clustering process because of two main reasons: 1) there is no statistical way to prove that a given number of clusters is the right one [14]. 2) We are not clustering the genes expressions, but rather we are clustering the binary vectors of functions annotations. The next subsection summarizes our previous work followed by how we improved it using SOM and a results comparison.

6.1. K-means for clustering

In [2] we used a sequential version of K-means to cluster genes annotation vectors into k groups; then, based on the expression level, the network should predict the most suitable group for a given gene. Training was fast; however, the network initial state, in terms of MSE and error percentage, was worse. By monitoring the CV error progress, it was easy to see that the network started to memorize the training set after the first few epochs. Table 5 summarizes the results for 4 different groupings (4 different values of k).

Table 5: MLP results for 4 different clusters generated by K-means

#of Groups(k)	Training MSE	CV MSE	Testing MSE	Training time in sec
500	0.096455	7.350921	7.081350	1660
300	0.110561	0.179318	0.179431	7560

200	0.121474	0.294703	0.252912	790
100	0.096305	6.743814	5.059804	24240

This poor performance can be explained by the grouping distribution. As in figures 2 and 3, the first group always dominates and possesses nearly 50% of the total number of vectors regardless of the number of means specified. Some other groups have as few as 1 member only. The next section will illustrate how SOM clustering improved the clustering distribution, which in turn improved the NN ability to generalize.

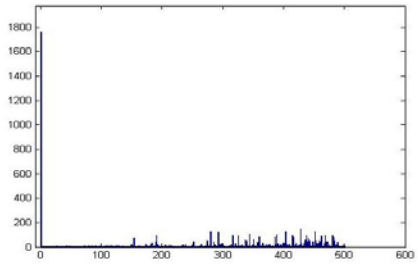

Fig. 2. K-Means Genes distribution (500 groups)

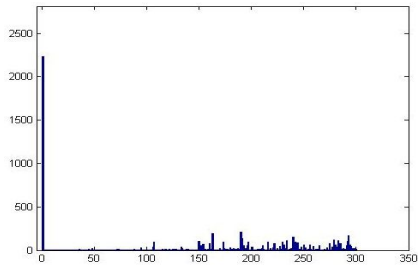

Fig. 3. K-means Genes distribution (300 groups)

6.2. Kohonen SOM for clustering

Another clustering algorithm like Kohonen SOM is suggested because the group centers along with its neighbors adapt and learn with every member they acquire which gives more chance for a more balanced distribution [9, 10]. In this work, we used SOM with hexagonal lattice type, sheet shape and Gaussian neighborhood function defined by:

$$h_{ci}(t) = e^{-\frac{\|r_c - r_i\|^2}{2\sigma_t^2}} \quad (1)$$

Where: t is time, c and i are output units, $\|r_c - r_i\|$ is the Euclidian distance between the output units, and σ is Gaussian neighborhood radius, initialized to 5 and ending with 1. Learning rate α is initialized to 0.5 in the rough training phase and to 0.05 in the fine tuning training phase. The learning rate update function is defined by:

$$\alpha(t) = \frac{\alpha_0}{\left(1 + \frac{100t}{T}\right)} \quad (2)$$

Where: α_0 is the initial learning rate and T is the training length.

Figures 4 and 5 show the histograms for the resultant distribution for 500 and 300 groups respectively. Comparing with figures 2 and 3, it should be clear that we have a more balanced distribution than K-means. It is worth mentioning that Kohonen SOM

outperformed other clustering algorithms like hierarchical clustering and K-means when used to cluster genes expression profiles [6,14]

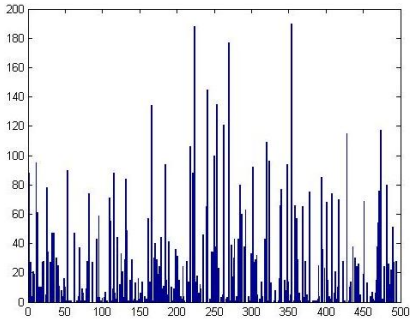

Fig. 4. SOM Genes distribution (500 groups)

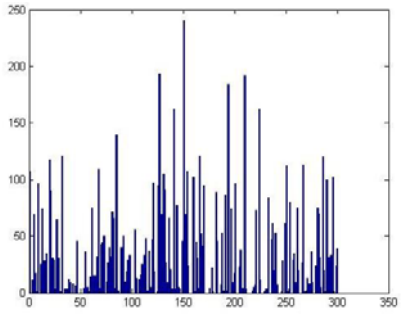

Fig. 5. SOM Genes distribution (300 groups)

Table 6 shows the NN performance results using this more balanced distribution. Table 6 is to be compared with same results of K-means in Table 5. In addition to the improvement over the results obtained by K-means, table 6 shows that Kohonen SOM is a good candidate for clustering binary vectors. Also, this experiment shows the effect of the dataset distribution on NN performance. With a more uniform distribution of classes' memberships, NN can perform well regardless of the number of classes.

Table 6. MLP results for 4 different clusters generated by Kohonen SOM

#SOM Groups	Training MSE	CV MSE	Testing MSE	Training time in sec
500	0.056891	0.054700	0.057854	8345
300	0.059087	0.052453	0.060239	3625
200	0.059664	0.06319	0.06603	8548
100	0.061224	0.056372	0.065313	1251

7 Results Comparison

The closest to this work is Mateos et al [15] and Zhang et al [22]. Although Mateos et al work is for predicting yeast gene functions and among only 100 predefined classes, the prediction methods are similar to ours. Taking the dataset size into consideration, results in our work present a more promising performance for NN. The authors have a consistently poor performance (<60%) in terms of the number of False Positive (FPs). They defined three biological factors, in addition to the high noise to data ratio of their dataset, that cause this poor performance: class size, heterogeneity and the high degree of intersection among functional classes. Their conclusion is that when it comes to NN, FPs and False Negative (FNs) computationally do not necessarily mean the same biologically [15].

A quick comparison with Zhang et al work in [22] shows that NN outperforms SVM in this problem. However, a detailed comparison is not easy to perform for

many reasons: First of all, NN is used in three different ways in our work while only binary classification is performed in [22]. Also, the authors presented prediction results for both annotated and unannotated genes, while we used only the annotated genes for this work. Finally, Zhang et al did not use the SVM discriminate value to asses the SVM performance, they rather processed it to obtain an estimate of the probability of correct prediction of each gene in each category *(the recall value)*.

8 Concluding remarks and future work

In this work we confirmed our previous results that Neural Networks NN can effectively predict mouse gene function from gene expression levels. The learnable correlation between gene expression levels and their function categories in eukaryotes have been confirmed by our results. We presented three different ways to use NN to solve this problem. NN solved the binary classification version of the problem effectively. The actual results for a balanced, and hence harder, data set have been presented. Generalized Feed Forward networks GFF with two-phase training showed superiority over the standard MLP in predicting all GO-BP categories a gene might be involved in, with remarkable accuracy and in a more reasonable training time. Finally, Kohonen SOM is proven to be far more effective than K-means to group similar gene annotations as a preprocessing step for the NN. This research, especially the multi-output method, shows that NN can be a very effective Bioinformatics tool and should give NN a greater impact in gene function prediction in the future.

From a biological point of view, this work is very important because it defines another method that can be used to accurately predict gene function from gene expression in mammals. Results of this method can be used in conjunction with other machine learning methods to narrow down the list of genes for biologists to perform the actual biological experiments. Those biological experiments are usually very expensive and can only be done on a handful number of genes. Consequently, it is very common and always desirable to combine and cross validate results from different computational methods and end up with the most trusted predictions for biological experiments. An example of that is work by Hui Lan et al. in which, the authors cross validated the prediction results from multiple machine learning methods to end up with and accurate prediction of gene functions in *Arabidopsis* [11].

References

[1] E.A.M. Andrews Shenouda, A Quantitative Comparison of Different MLP Activation Functions in Classification, In: Proc. ISNN ‘06, Lecture Notes In Computer Science, Vol.3971 (Springer, Berlin, 2006) 849-857.

[2] E.A.M. Andrews Shenouda, Q. Morris, A.J. Bonner, Connectionist Approaches for Predicting Mouse Gene Function from Gene Expression, In Proc. ICONIP ’06, Lecture Notes in Computer Science, Vol. 4232 (Springer, Berlin ,2006) 280-289.

[3] P. Baldi, S. Brunak, Bioinformatics the Machine Learning Approach, Second ed., (MIT Press, Cambridge, London, 2001).

[4] M.A. Beer, S. Tavazoie, Predicting Gene Expression from Sequence, Cell 117(2) (2004)185-98.
[5] M. P. S. Brown, W. N. Grundy , D. Lin, N. Cristianini,C.W. Sugnet, T.S. Furey, M. Ares Jr, D. Haussler, Knowledge-Based Analysis of Microarray Gene Expression
[6] E. R. Dougherty, J. Barrera, M. Brun, S. Kim, R M. Cesar, Y. Chen, M. Bitter, J. M. Trent, Inference from Clustering with Application to Gene-Expression Microarrays, Journal of Computational Biology 9(1)(2002) 105-126.
[7] S. Haykin, Neural Networks: A Comprehensive Foundation, second ed., (Prentice Hall, New Jersey, 1999).
[8] O.D. King, R.E. Foulger, S.S. Dwight, J.V. White, F.P Roth, Predicting Gene Function from Patterns of Annotation, Genome Research 13(5) (2003) 896-904.
[9] T. Kohonen, Self-Organizing Maps, third ed., (Springer-Verlag New York, New Jersey, 2001).
[10] T. Kohonen, Self-Organizing and Associative Memory, third ed., (Springer-Verlag, New York, 1998).
[11] H. Lan, R. Carson, N. J. Provart, A. J. Bonner, Combining classifiers to predict gene function in Arabidopsis thaliana using large-scale gene expression measurements, BMC Bioinformatics, 8(358)(2007).
[12] Y. Le Cun, A Theoretical Framework for Back-propagation, In: D. Touresky, G. Hinton, T. Sejnowski (Eds), In Proc. The Connectionist Models Summer School (Morgan Kaufman, California, 1988) 21-28.
[13] J.B. MacQueen, Some Methods for classification and Analysis of Multivariate Observations, In Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability Vol.1 (University of California Press, Berkeley, 1967)281-297.
[14] P. Mangiameli, S. K. Chen, D. West, A comparison of SOM neural network and hierarchical clustering methods, European Journal of Operational Research 93(2) (1996) 402-417.
[15] A. Mateos, J. Dopazo, R. Jansen, Y. Tu, M. Gerstein, G. Stolovitzky, Systematic Learning of Gene Functional Classes From DNA Array Expression Data by Using Multilayer Perceptrons, Genome Research 12 (2002) 1703-1715.
[16] D. Michie, D.J. Spiegelhalter , C.C. Taylor, Machine Learning, Neural and Statistical Classification (Elis Horwood, London, 1994)
[17] T. M. Mitchell, Machine Learning (McGraw-Hill Inc., USA, 1997).
[18] T. Strachan, A. P. Read, Human Molecular Genetics, second ed. (Wiley-Liss, New York, 1999).
[19] A. Vinayagam, R. König, J. Moormann, F. Schubert, R. Eils, K. Glatting, S. Suhai, Applying Support Vector Machines for Gene Ontology Based Gene Function Prediction, BMC Bioinformatics 5(116) (2004).
[20] D. Wang, G. Huang, Protein Sequence Classification Using Extreme Learning Machine, In Proc. IJCNN '05, Vol. 3. (2005) 1406-1411
[21] P. J. Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, Ph.D. Thesis, Applied Mathematics, Harvard University, Boston, 1974.
[22] W. Zhang, Q.D. Morris, R. Chang, O. Shai, M.A. Bakowski, N. Mitsakakis, N. Mohammad, M.D. Robinson, R. Zirngibl, E. Somogyi, N. Laurin, E. Eftekharpour, E. Sat, J. Grigull, Q. Pan, W.T. Peng, N. Krogan, J. Greenblatt, M. Fehlings, D.V. Kooy, J. Aubin, B.G. Bruneau, J. Rossant, B.J. Blencowe, B.J. Frey, T.R. Hughes, The Functional Landscape of Mouse Gene Expression, Journal of Biology Vol. 3 Article 21 (2004).
[23] J. M. Zurada, Introduction to Artificial Neural Systems, (PWS Publishing, Boston, 1999).
[24] <http://hugheslab.med.utoronto.ca/Zhang>

[25] <http://ncbi.nlm.nih.gov/COG/>