Connectionist Approaches for Predicting Mouse Gene Function from Gene Expression

Emad A.M. Shenouda, 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 especially in Gene Therapy. Identifying gene function based on gene expression data is much easier in prokaryotes than eukaryotes due to the relatively simple structure of prokaryotes. That is why tissue-specific expression is the primary tool for identifying gene function in eukaryotes. However, recent studies have shown that there is a strong learnable correlation between gene function and gene expression. This paper outlines a new approach for gene function prediction in mouse. The prediction mechanism depends on using Artificial Neural Networks (NN) to predict gene function based on quantitative analysis of gene co-expression. Our results show that (NN) can be extremely useful in this area. Also, we explore clustering of gene functions as a preprocessing step for predicting gene function.

1 Introduction

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 [5]. Identifying gene function in prokaryotes is much easier than eukaryotes due 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 showed that there is a strong learnable correlation between gene function and gene expression [18]. According to Zhang et al, predicting gene function their expression is more effective than using tissue-specific function as a guide [18]. In this paper, we explore a new approach for gene function prediction in mouse based on a quantitative analysis of gene co-expression. We will try to learn this correlation using machine learning techniques to predict gene function category from gene expression.

The machine learning technique used in this work is Neural Networks (NN). As we 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) has 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 [2]. In the same source, 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. Introducing (BP) was a landmark in Neural Networks (NN) [6]. The earliest description of (BP) was presented by Werbos in his PhD dissertation in 1974 [15]; however, it did not gain much publicity until it was independently rediscovered by Le Cun, Parker, Hinton and Williams [7]. Multilayer perceptrons (MLP) is perhaps the most famous implementation for (BP). It has been very successfully used in many applications in various domains, such as prediction, function approximation and classification. For classification, MLP is considered a super-regression machine that can draw complicated decision borders between nonlinearly separable patterns [6]. The nonlinearity power of MLP is due to the fact that all its neurons use a nonlinear activation function to calculate their outputs. If a linear activation function is used, the whole MLP becomes a simple linear regression machine. Different activation functions and our justification for the one used in our own work will be mentioned in more detail in the network topology section later.

Unlike (SVM), MLP does not need additional projection for the input space. Also, MLP can be implemented in two versions: first is the Soft-Max version, which has only one output neuron to be activated with a certain input, while the other has more than one output activated, which is a very powerful feature of MLP as it allows the possibility that a gene might be involved in more than one function. In this work we used MLP in three different ways: Firstly, we used it as a binary classifier, in which the input is gene expression measurements and the desired output is a single Go-category. Having 922 different Go-categories, this will result in about 1000 binary classifiers. In the second approach, we unleashed the power of the MLP as a super regression machine by training it with the same input as before but with all 922 Go-categories as the desired output. This approach not only discovers the relation among co-expressed genes but also ties those expressions to many possible functions. Finally, we tried clustering gene annotations into distinct groups using the K-means algorithm; then, we used those groups as the desired output for the network. The last method aimed to make the problem easier for the network and to reduce the huge training time required by the second method. For full explanation about K-means the reader is directed to Michell [9]

Our results show that NN is a good candidate for predicting gene function from expression. As a binary classifier, MLP is very successful and posses the advantage of eliminating the need for the kernel function needed by SVM. As a super regression machine, MLP was able to predict the whole set of 922 possible functions for each gene but with the drawback of lots of training time needed. Due to the nature of the data, as we will see later, grouping genes annotations and using those groups as the desired output for the network did not produce as good results as the first two methods.

The paper is organized as follows: The next section relates this work to previous work in Bioinformatics. Then, we briefly describe the source of the data used, the reasons for its integrity, and the necessary preparation steps to make it ready for our

NN implementation. The NN topology, activation function selection and justification for our decisions are also presented. Following that, the setup and results for each of our three NN approaches are sketched. Finally, concluding remarks and future work are discussed.

1.1 Related work

Using gene expression for predicting gene functions has been proven to be so effective in prokaryotes. For eukaryote, tissue-specific expression is widely used for predicting genes functions [18]. However, the correlation between gene expression, sequence and gene functions has attracted the attention of many bioinformaticians. In [3], the authors used a probabilistic approach to correlate gene sequence and expression. In [5], 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-annotation categories vocabulary mentioned at the Gene Ontology (GO) database without considering the gene expression. In [11], Oliver et al used Bayesian networks and decision trees to model the relationship among different Go-categories vocabulary, only 41 genes out of 100 manually assessed were judged to be true. Although the success rate is not high in [11] research, it revealed the possible relationship among different GO-categories which can be logically reflected to the annotated genes themselves.

SVMs are the most widely used machine learning approach to predict genes functions from genes sequences and expressions. In [8], 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. To be able to use SVMs, authors converted the unsupervised learning problem into a supervised one by masking the actual function to be predicted into a binary classification problem, whether the gene might be involved in this function or not. Still, this work used yeast and not higher order organisms; also, the classification is binary and the performance is assessed against other basic classification algorithms like Fisher's linear discriminant and decision trees.

In 2004, Arunachalam 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 ranged from as simple as yeast to as complicated as mouse [1].The authors provided a mechanism to measure confidence estimates for their predictions.

The first attempt to use NN to predict gene function class from gene expression was in 2002 by Mateos et al [20]. 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 [18], 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, 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-categories mentioned at [10]. Again, authors provided a mechanism to measure the accuracy of their predictions through a recall value. The reader is directed to [18] and [13] for full details about this work.

Instead of using SVM, we used MLP for prediction and K-means 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 K-means 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.

In our project, we used the same dataset as in [13]; the next section describes what portion of the data is used and how it is preprocessed.

2 Data preparation

Data used for this paper is provided by Zhang et al [13]. The reader is directed to [18] 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-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 input to this data preparation process, two matrices were downloaded from [13]: the 21,266 X 55 matrix of the microarray data and the two-column annotation matrix, the microarray data is normalized and centered by subtracting the mean. Then, the two-column annotation file is parsed to extract all distinct genes and all distinct annotations. Following that, a two dimensional binary matrix 7,383 X 922 is created; in which, each annotated gene has a binary vector denoting all of the 922 annotations. If the gene is annotated in a Go-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-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-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 and Huang used Extreme Learning Machine (ELM) algorithm instead of NN in spite of the potential promising results of NN in classifying protein sequence [14]. In addition to network parameters like learning rate and momentum, NN topology is also determined by its size, synaptic weights 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 measurement of the system complexity and it 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 [19] [6]. Based on the type of network, we used different techniques to reduce the network size as we will elaborate in subsequent sections. In general, the most common way to reduce network size is weight pruning as mentioned by Zurada and Haykin [19] [6], 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 equal to the number of tissues which is 55.

Concerning the synaptic weights 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. In theory, an MLP can solve any problem that a generalized feedfoward network can solve [6]. In practice, however, generalized feedfoward 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 it showed fewer tendencies to memorize. Because there is no theoretical foundation of the possible performance discrepancy between MLP and GFF, we did a lot of experimentations to determine which network to use. The following section justifies our selection for the MLP activation function.

3.1 Activation function selection

The nonlinearity power of MLP is due to the fact that all its neurons use a nonlinear activation function to calculate their outputs. If a linear activation function is used, the whole MLP will become a simple linear regression machine. Not only determining the decision borders, but the value of the activation function also determines the total signal strength the neuron will produce and receive. In turn, that will affect almost all aspects of solving the problem in hand, such as the quality of the network initial state, speed of conversion and the efficiency of the synaptic weights updates.

As a result, a careful selection of the activation function has a huge impact on the MLP classification performance. In theory, the (BP) is universal in this matter, so any activation function can be used as long as it has a first derivative. In practice,

activation functions can be categorized into three basic types [16]: linear, logistic (e.g. Sigmoidal) and Radial Basis Functions (RBF) (e.g. Gaussians).

In our work we used the sigmoidal activation function. The Statlog report in [4] showed that the sigmoidal activation function has very good performance in classification. Also, in [14] Wang and Huang showed that the sigmoidal function outperformed the RBF in the ELM algorithm for protein classification. A thorough comparison in [12] showed that the sigmoidal function constantly outperformed other activation functions. In addition, from a mathematical perspective, negative output is not desired in our problem because the expected output is either always binary or a group number which is never negative.

The Sigmoid function is sometimes called the *nonsymmetric logistic function* [6] because its output is between 0 and 1. Using this function the unit output will be:

$$o_i = \frac{1}{1+e^{-n_i}} \qquad \text{Derivative is : } o_i(1-o_i) \qquad (\mathbf{1})$$

Where n_i is the net input for unit (i), that is: $n_i = \sum_j w_{ij} x_j + \Theta_i$. The reader is directed to [17] where there are more net input calculations and activation functions.

3.2 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 starts to increase and used for testing.

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 [4]. That is why in Statlog report experiments, they did not use the error rate most of the time and used other measurement like average cost [4]. The (MSE) is twice the average cost.

4 Neural networks as a binary classifier

In this approach we used NN as a binary classifier machine. The input is the gene microarray expression level and the output is 1 or 0 depending on whether the gene is predicted to be in the desired Go-Category or not respectively. A separate NN is needed for each function. So nearly 1K binary classifiers are needed to predict all GO-Categories using this approach. Instead of using only one output for the network, we used two binary output neurons to well assess the network accuracy. 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-Categories with maximum gene participation. Those categories are the best example of the network performance in this type of problem as they provide the maximum generalization and hence are the hardest to predict. Table 1 summarizes the prediction results for the selected GO-Categories. With some statistical data analysis from the above prepared 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.

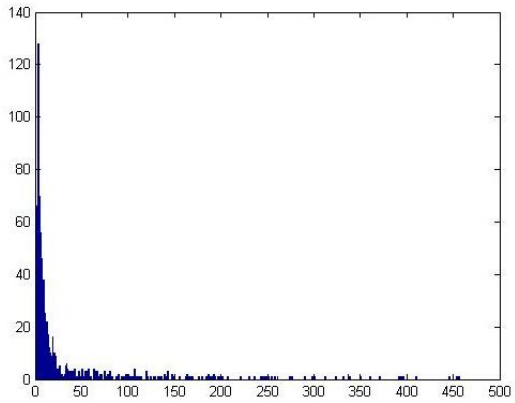

Fig. 1. Overall gene participation distribution for all categories

4.1 More expressive dataset

To prove our hypothesis without getting affected by the non-uniform data distribution mentioned above, we prepared a balanced dataset as a subset of the originally prepared 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. As we will see from the results, the network recorded a much lower training MSE than the previous experiment in spite of the fewer training examples. The actual output of the network test examples is presented in table 2.

4.2 Results

We did binary classification for GO-Categories with a gene participation of at least 350. That gave us 10 categories to predict. The following table summarizes the

results for the 10 selected Go-Categories: where, A is the number of positive genes, B is the training MSE, C is the CV MSE, D is the testing MSE and E is the training time in seconds. (Note: cells B, C, D are divided by 10^{-3})

Table 1. Binary prediction results for the selected GO-Categories

Function Name	Go-category	A	B	C	D	E
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

The above table reflects an acceptable performance for the MLP to predict gene function from its expression level. But, we used a balanced dataset from the lipid metabolism [GO:0006629] category to asses the MLP performance in case of a balanced dataset. 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 following table where D and A stand for the desired and actual outputs respectively.

Table 2. Actual NN results for the balanced sub-dataset

D 0	D 1	A 0	A 1	D 0	D 1	A 0	A 1	D 0	D 1	A 0	A 1
1	0	0.999215	0.000785	1	0	0.429985	0.570019	1	0	0.999167	0.000833
1	0	1.001267	0.00127	0	1	0.391373	0.60863	1	0	0.00031	1.000306
0	1	0.0012	1.001202	1	0	0.999388	0.000613	0	1	0.994135	0.005865
0	1	0.00018	1.000176	0	1	0.999378	0.000622	1	0	0.998971	0.00103
1	0	1.000075	7.4E-05	0	1	0.09959	0.900412	0	1	0.997675	0.002326
0	1	8E-06	1.000009	1	0	0.0014	1.001399	1	0	0.00956	1.009562
1	0	0.777683	0.222318	0	1	0.999638	0.000363	0	1	0.998558	0.001442
1	0	0.986617	0.013384	1	0	0.999047	0.000953	0	1	0.251832	0.748172
0	1	0.999549	0.000451	0	1	0.00318	1.003182	1	0	0.999172	0.000828
1	0	1.001881	0.00188	0	1	0.310283	0.689721	0	1	0.966519	0.033481
1	0	0.00018	1.000177	0	1	1.003398	0.0034	1	0	0.125626	0.874377
1	0	0.999156	0.000844	1	0	0.999119	0.000882	0	1	0.000196	0.999805
1	0	0.033314	0.966688	0	1	0.999297	0.000703	1	0	0.982307	0.017694
1	0	0.983298	0.016702	0	1	0.00094	1.000937	1	0	0.999131	0.000869
0	1	0.000224	0.999776	0	1	0.000332	0.999668	1	0	0.999056	0.000945

1	0	0.999263	0.000738	1	0	1.002547	0.00255	0	1	0.999296	0.000705
1	0	1.000588	0.00059	1	0	1.000438	0.00044	1	0	0.000139	0.999862
0	1	1.0005	0.0005	0	1	0.000245	0.999756	1	0	0.999218	0.000782
1	0	0.998465	0.001536	0	1	0.994506	0.005494	1	0	0.998925	0.001076
0	1	0.00455	1.004549	1	0	1.000548	0.00055	1	0	0.999724	0.000276
1	0	0.999284	0.000716	0	1	0.999474	0.000527	1	0	0.998531	0.00147
0	1	0.00019	1.000193	1	0	0.999405	0.000596	1	0	0.999211	0.000789
1	0	0.977609	0.022392	1	0	0.99844	0.001561	1	0	0.231151	0.768853
1	0	0.999154	0.000846	1	0	0.999442	0.000559	1	0	0.117864	0.882139
1	0	0.999457	0.000542	0	1	0.971673	0.028326	0	1	1.00252	0.00252
0	1	0.000218	0.999782	1	0	0.999129	0.000871	1	0	1.001117	0.00112
0	1	0.000442	0.999559	1	0	1.000617	0.00062	0	1	0.994494	0.005506
0	1	0.999625	0.000375	1	0	0.983058	0.016942	0	1	0.00215	1.002151

5 Neural network with 1000 output *(Non-Softmax)*.

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. Although the results are promising, the training time was huge. We concentrated on reducing the network size to the lowest we can without affecting either the network initial state or the CV MSE. The minimum topology was 3 hidden layers with 200, 100, and 50 hidden units (from input to output), and there were 922 output units. After nearly 60 hours of training, we obtained training MSE of *0.061335*, CV MSE of *0.061616* and testing MSE of *0.061778.*

We used batch learning to reduce the training time; the network then consumed 1059 epochs in 211673 seconds which means about 199 sec/epoch. Those results along with the CV error progress show that this method is effective but with a large sized network and huge training time which needs specialized hardware to produce the best network results in a reasonable amount of time. In the next approach we will try to overcome these issues by grouping the annotating vectors of the genes.

6 Neural network with K-means as a classifier

In this approach we tried to overcome the main drawback of the previous approach which is the huge training time required by the NN. A sequential version of k-means is used to group the binary vectors denoting the genes possible annotations into k groups, while k is a configurable number passed in as an input parameter to the K-means algorithm. 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.

6.1 Results

As expected, the problem became an easier one for the NN to solve and the training time required was far less than the Non-Softmax version. 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. The best results were obtained by using a 2-hidden layer GFF version, which delayed the CV error increase for many more epochs with a considerable improvement in terms of test MSE. Both test MSE and CV MSE were higher than the training MSE. The following table summarizes the results for 4 different groupings (4 different values of k).

Table 3: Results for 4 different GO-Categories groups

# of Groups	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 we see from the following histograms (figures 2 to 5) , the first group always dominates and has 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. A careful review of [18] should explain this grouping behavior because, as stated, about 4475 genes are detected in one sample. Another clustering algorithm like Kohonen Self-Organizing Maps (SOM) is suggested because the group centers adapt and learn with every member they acquire which gives more chance for a more balanced distribution [21].

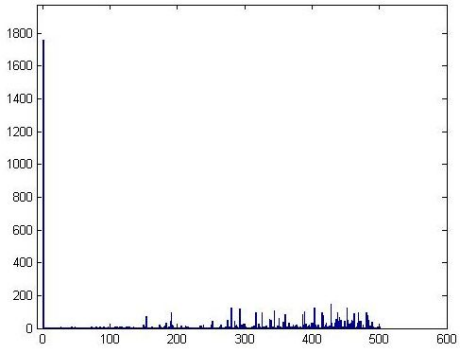

Fig. 2. Genes distribution (500 groups)

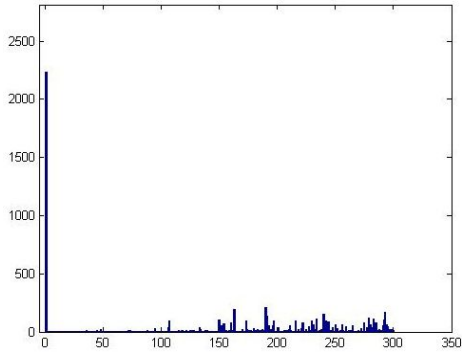

Fig. 3. Genes distribution (300 groups)

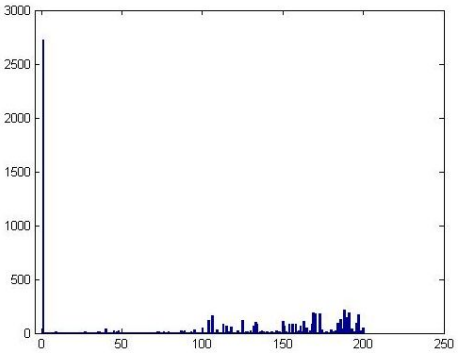

Fig. 4. Genes distribution (200 groups)

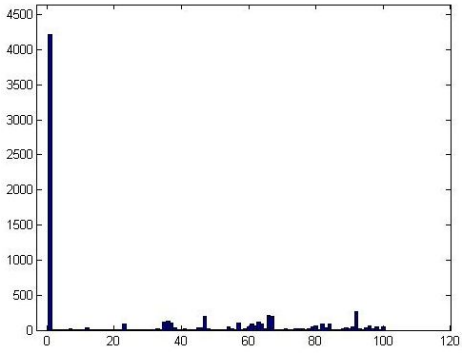

Fig. 5. Genes distribution (100 groups)

7 Concluding remarks and future work

In this work we showed that NN can effectively predict mouse gene function from gene expression levels. The learnable correlation between genes 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. Using MLP to its maximum strength to predict all the available categories for a certain gene showed very promising results; however, a huge NN size and hence lots of training time are needed. Finally, K-means as a clustering algorithm has been used to group similar genes into the same category as a preprocessing step for the NN. We showed that K-means is not the best candidate for this problem because it has fixed means that got readjusted only at the end of the training process. Another clustering algorithm that performs online learning with each example introduced, like Kohonen SOM, might improve this technique. Also, optimizing the MLP parameters using an evolutionary algorithm as in Simulated Annealing or Genetic Algorithms is also likely to improve the NN results mentioned in this paper.

References

1. Vinayagam, A., König, R., Moormann, J., Schubert, F., Eils, R., Glatting, K., Suhai, S.: Applying Support Vector Machines for Gene ontology based gene function prediction BMC Bioinformatics. 5(116) (2004)
2. Baldi, P., Brunak, S.: Bioinformatics the Machine Learning Approach. 2nd edn. MIT Press, Cambridge, London (2001)
3. Beer, M.A., Tavazoie, S.: Predicting Gene Expression from Sequence. Cell 16 117(2) Apr. (2004)185-98

4. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: Machine Learning, Neural and Statistical Classification. Elis Horwood, London (1994)
5. Morrison, K.E., Daniels, R. J., Campbell, L., McPherson, J., Davies, K.E.: (1993). Human Molecular Genetics 2. (1993).
6. Haykin, S.: Neural Networks A Comprehensive Introduction. Prentice Hall, New Jersey (1999)
7. Le Cun, Y., Touresky, D., Hinton G., Sejnowski, T.: A Theoretical Framework for Backpropagation. In: The Connectionist Models Summer School. (1988) 21-28
8. Brown, M. P. S., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C., Furey, T. S., Ares, M., Jr., Haussler, D.: Knowledge-based Analysis of Microarray Gene Expression Data Using Support Vector Machines. In: Proceedings of the National Academy of Sciences 97(1) (1997) 262-267
9. Mitchell, T. M.: Machine Learning. McGraw-Hill Inc USA (1997).
10. NCBI COGs database [http://ncbi.nlm.nih.gov/COG/]
11. King, O.D., Foulger, R.E., Dwight, S.S., White, J.V., Roth F.P.: Predicting Gene Function from Patterns of Annotation. Genome Research 13 (5) (2003) 896-904
12. Shenouda, E.: A Quantitative Comparison of Different MLP Activation Functions in Classification". In: Proceedings of the International Symposium of Neural Networks ISNN06 (2006)
13. The functional landscape of mouse gene expression. [http://hugheslab.med.utoronto.ca/Zhang]
14. Wang, D., Huang, G.: Protein Sequence Classification Using Extreme Learning Machine. In: IJCNN05, Vol. 3. Montréal (2005) 1406-1411
15. Werbos, P. J.: Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Doctoral Thesis, Applied Mathematics, Harvard University. Boston (1974)
16. Duch, W., Jankowski, N.: Transfer functions: Hidden Possibilities for Better Neural Networks. In: *9th* European Symposium on Artificial Neural Network. Bruges (2001) 81-94
17. Hu, Y., Hwang, J.: Handbook of Neural Network Signal Processing. 3rd edn. CRC-Press, Florida (2002)
18. Zhang W., Morris Q.D., Chang, R., Shai, O., Bakowski, M.A., Mitsakakis, N., Mohammad, N., Robinson, M.D., Zirngibl, R., Somogyi, E., Laurin, N., Eftekharpour, E., Sat, E., Grigull, J., Pan, Q., Peng, W.T., Krogan, N., Greenblatt, J., Fehlings, M., Kooy D. V., Aubin J., Bruneau B.G., Rossant, J., Blencowe, B.J., Frey, B.J., Hughes T.R.: The functional landscape of mouse gene expression. Journal of Biology 3 Article 21 (2004)
19. Zurada, J. M.: Introduction to Artificial Neural Systems. PWS Publishing, Boston (1999)
20. Mateos, A., Dopazo, J., Jansen, R., Tu, Y.,Gerstein, M.,Stolovitzky, G.: Systematic Learning of Gene Functional Classes From DNA Array Expression Data by Using Multilayer Perceptrons. Genome Research 12 (2002) 1703-1715
21. Kohonen, T.: Self-Organizing and Associative Memory. 3rd edn Springer-Verlag, New York (1998)