Chapter 11

CLASSIFICATION OF EXPRESSION PATTERNS USING ARTIFICIAL NEURAL NETWORKS

Markus Ringnér[1,2], Patrik Edén[2] and Peter Johansson[2]

[1] *Cancer Genetics Branch, National Human Research Institute, National Institutes of Health, Bethesda, Maryland 20892, USA*

[2] *Complex Systems Division, Department of Theoretical Physics, Lund University, Lund, Sweden*, e-mail: {markus,patrik,peterj}@thep.lu.se

1. Introduction

Artificial neural networks in the form of feed-forward networks (ANNs) have emerged as a practical technology for classification with applications in many fields. ANNs have in particular been used in applications for many biological systems (see (Almeida, 2002) for a review). For a general introduction to ANNs and their applications we refer the reader to the book by Bishop (Bishop, 1995). In this chapter we will show how ANNs can be used for classification of microarray experiments. To this aim, we will go through in detail a classification procedure shown to give good results and use the publicly available data set of small round blue-cell tumors (SRBCTs) (Khan *et al.*, 2001) as an example[1].

The ANNs described in this chapter perform *supervised learning*, which means that the ANNs are calibrated to classify samples using a training set for which the desired target value of each sample is known and specified. The aim of this learning procedure is to find a mapping from input patterns to targets, in this case a mapping from gene expression patterns to classes or continuous values associated with samples. *Unsupervised learning* is another form of learning that does not require the specification of target data. In unsupervised learning the goal may instead be to discover clusters or other structures in the data. Unsupervised methods have been used extensively to analyze array data and are described in other chapters. The main reasons for choosing a supervised method are to obtain a classifier or predictor, and to extract the genes important for the classification. Here we will exemplify this by describing an ANN-based classification of expression profiles of SRBCT samples into four distinct diagnostic categories: neuroblastoma (NB), rhabdomyosarcoma (RMS), Burkitt's lymphoma (BL) and Ewing's sarcoma (EWS). This data set consists of 63 training samples each belonging to one of the four categories and 25 test samples.

There are many other supervised methods (discussed in other chapters) that have been used to classify array data, spanning from simple linear single gene discriminators to machine learning approaches similar to ANNs, in particular *support vector machines* (SVMs). A major advantage of using a machine learning approach such as ANNs is that one gains flexibility. Using an ANN framework, it is for example straightforward to modify the number of classes, or to construct both linear and non-linear classifiers. Sometimes this flexibility is gained at the expense of an intuitive understanding of how and why classification of a particular problem gives good results. We hope this chapter will provide the reader with an understanding of ANNs, such that some transparency is regained and the reader will feel confident in using a machine learning approach to analyze array data.

We begin with a discussion on how to reduce high-dimensional array data to make the search for good ANNs more efficient. This is followed by section 3 on ANNs. Section 3 is split into 5 subsections as follows. We describe how to design an ANN for classification, how to train the ANN to give small classification errors, how a *cross-validation* scheme can be used to obtain classifiers with good predictive ability, how *random permutation tests* can be used to assess the significance of classification results, and finally how one can extract genes important for the classification from ANNs. This chapter ends with a short section on implementation followed by a summary.

2. Dimensional reduction

For each sample in a typical study, the expression levels of several thousand genes are measured. Thus, each sample can be considered a point in "gene-space", where the dimensionality is very high. The number of considered samples, N, is usually of the order of 100, which is much smaller than the number of genes. As discussed below, an ANN using more inputs than available samples tend to become a poor predictor, and in microarray analyses it is therefore important to reduce the dimensionality before starting to train the ANN. Dimensional reduction and input selection in connection with supervised learning have attracted a lot of interest (Nguyen and Rocke, 2002).

The simplest way to reduce dimensionality is to select a few genes, expected to be relevant, and ignore the others. However, if the selection procedure involves tools less flexible than ANNs, the full potential of the ANN approach is lost. It is therefore preferable to combine the genes into a smaller set of components, and then chose among these for the ANN inputs.

For classification, we only need the *relative* positions of the samples in gene-space. This makes it possible to significantly reduce the dimensionality of microarray data without any loss of information relevant for classification. Consider the simple case of only two samples. We can then define one component as a *linear combination* of genes, corresponding to the line in gene-space going through the two sample points. This single component then fully specifies the distance in gene-space between the two samples. In the case of three samples, we can define as components two different linear combinations of genes, which together define a plane in gene-space which is going through the three data points. These two components then fully specify the relative location of the samples. This generalises to N samples, whose relative locations in gene-space can be fully specified in an $N-1$ dimensional subspace. Thus, with N samples, all information relevant for classification can be contained in $N-1$ components, which is significantly less than the number of genes.

Reducing a large set of components (in our case, the genes) into a smaller set, where each new component is a linear combination of the original ones, is called a *linear projection*. In connection with ANNs, *principal component analysis*, PCA, is a suitable form of linear projection. PCA is described in detail in Chapter 5. In brief, it ranks the components according to the amount of *variance* along them, and maximizes the variance in the first components. Thus, the first component is along the direction which maximizes the variance of data points. The second component is chosen to maximize the variance, subject to the condition of orthogonality to the first component. Each new component must be orthogonal to all previous ones, and is pointing in the direction of maximal variance, subject to these constraints.

As an example, Figure 11.1 shows how much of the variance of the SRBCT gene expression data matrix is included in the different principal components. Using the 10 first principal components will in this case include more than 60% of the variance.

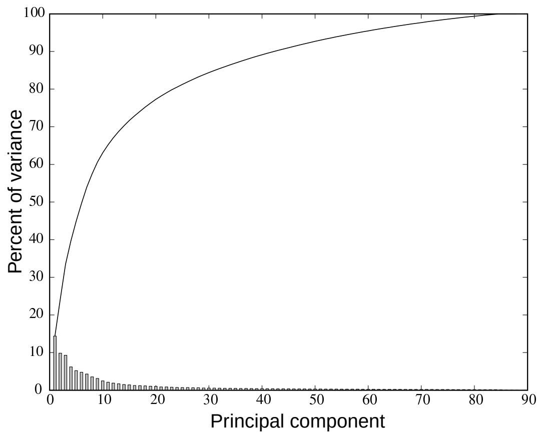

Figure 11.1 The percent of variance in the SRBCT gene expression data matrix (all 88 samples) contained by each principal component (bars). The cumulative contained variance is also shown (solid line). The 10 first principal components contain more than 60% of the variance in the data matrix for this example.

In our example, we used mean centered values for the PCA and did not perform any rescaling. In principle, however, any rescaling approach is possible, since the selection of principal components as ANN inputs can be done in a supervised manner.

Our preference for PCA is based on the following arguments:

1 To allow for a biological interpretation of the ANN results, it is important that the connection between genes and ANN inputs can be easily reconstructed. Linear projections, *e.g.* PCA, fulfills this demand, in contrast to *e.g.* multi-dimensional scaling (Khan *et al.*, 1998).

2 ANN analyses should involve cross-validation, described in section 3.3. This implies that the ANN is trained several times, on different subsets of data, giving different parameter settings. An unsupervised dimensional reduction, *e.g.* PCA, can use all available samples without introducing any bias. The result of this reduction can then be reused for every new training set. In constrast, a supervised reduction scheme can only rely on training samples to avoid bias, and must be redone as the training set changes.

3 In general, the $N - 1$ components containing all information are still too many for good ANN analyses. The ranking of principal components according to the variance gives information about inputs for which the separation of data points is robust against random fluctuations. Thus, PCA gives a hint on how to reduce the dimensionality further, without losing essential information.

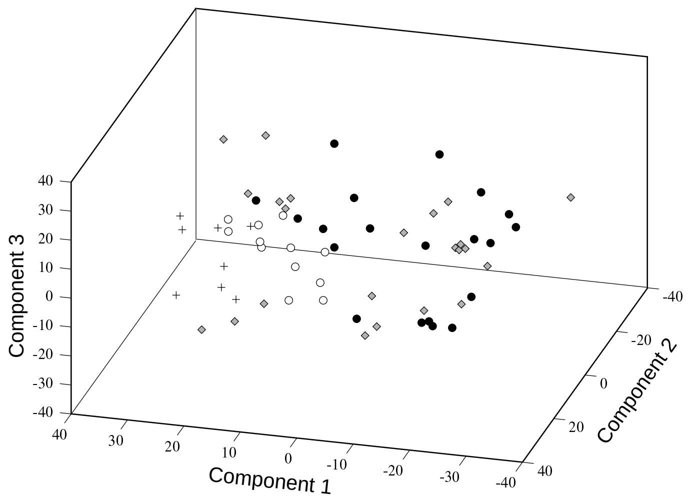

Figure 11.2. Projection of the 63 SRBCT training samples onto the 3 first principal components. The samples belong to four diagnostic categories, NB (circles), RMS (filled circles), BL (pluses) and EWS (diamonds). There is a tendency of separation between the categories. Of note, the first principal component essentially separates tumor samples (on the right) from cell lines (on the left).

4 The ANN inputs may need to be carefully selected, using a supervised evaluation of the components. The more correlated different input candidates are, the more difficult it is to select an optimal set. As principal components have no linear correlations, the PCA facilitates an efficient supervised selection of ANN inputs.

In Figure 11.2, the 63 SRBCT training samples, projected onto the first three principal components, are shown. Along component one there is a clear separation, poorly related to the four classes. This separation distinguishes tissue samples from cell-lines. This illustrates that the major principal components do not need to be the most relevant for the classification of interest. Thus, it can be useful to select ANN inputs using a supervised method. In doing so, we are helped by point 4 above. Since a central idea in the ANN approach is not to presume linear solutions, it is reasonable to use ANNs also for supervised input ranking. A simple method is to train a network using quite many inputs (maybe even all). This network will most likely be heavily overfitted and is of little interest for blind testing, but can be used to investigate how the network performance on the training set is affected as one input is excluded. Doing so for each input gives information for input selection. In the SRBCT example, there was no need to do a supervised input selection, as the classification was successful using the 10 first principal components.

3. Classification using ANNs

3.1 ANN Architecture

The simplest ANN-model is called a *perceptron.* It consists of an input layer and a single output (Figure 11.3). Associated with each input is a weight that decides how important that input is for the output. An input pattern can be fed into the perceptron and the responding output can be computed. The perceptron is trained by minimizing the error of this output. The perceptron is a linear classifier since the weights define a hyperplane that divides the input space into two parts.

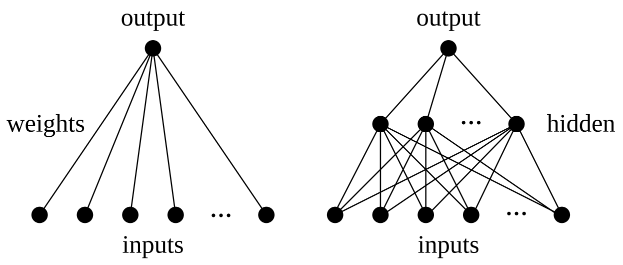

Figure 11.3. In a linear perceptron (left), the input data is fed into the input layer and triggers a response in the output layer. The weights are tuned such that the output ideally should correspond to the target value. In a multi-layer perceptron (right), a hidden layer is added in between the input and output layers.

In our example we have four classes and a single perceptron does not suffice. Instead we use a system of four parallel perceptrons. Each perceptron is trained to separate one class from the rest, and as a classification the class with the largest output is chosen.

It is recommended to first try a linear network. However, for more complicated problems a linear hyperplane is not good enough as a separator. Instead it is advantageous to have a nonlinear surface separating the classes. This can be achieved by using a *multi-layer perceptron*, in which several perceptrons are connected in a series (Figure 11.3). Besides having an input and output layer, one also has one (or several) hidden layer(s) in between. The nodes in the hidden layer are computed from the inputs

$$h_j = f\left(\sum_i w_{ji}^{(1)} x_i\right), \tag{11.1}$$

where x_i denote the ith input and $w^{(1)}$ denote the weights between the input and hidden layers. The hidden nodes are used as input for the output (y) in the same manner

$$y = g\left(\sum_j w_j^{(2)} h_j\right) = g\left(\sum_j w_j^{(2)} f\left(\sum_i w_{ji}^{(1)} x_i\right)\right) , \tag{11.2}$$

where $w^{(2)}$ denote the weights between the hidden and output layers,

$$g(x) = \frac{1}{1 + e^{-x}}, \tag{11.3}$$

is the logistic sigmoid activation function, and

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{11.4}$$

is the "tanh" activation function. A way to view this is that the input space is mapped into a hidden space, where a linear separation is done as for the linear perceptron. This mapping is not fixed, but is also included in the training. This means that the number of parameters, *i.e.* the number of weights, is much larger. When not having a lot of training samples this might cause overfitting. How many parameters one should use varies from problem to problem, but one should avoid using more than the number of training samples.

The ANN is probabilistic in the sense that the output may easily be interpreted as a probability. In other words, we are modelling the probability that, given a certain information (the inputs), a sample belongs to a certain class (Hampshire and Pearlmutter, 1990).

3.2 Training the ANN

Training, or calibrating, the network means finding the weights that give us the smallest possible classification error. There are several ways of measuring the error. A frequently used measure and the one we used in our example is the mean squared error (MSE)

$$E = \frac{1}{N} \sum_k^N \sum_l (y_{kl} - t_{kl})^2 , \tag{11.5}$$

where N is the number of training samples and y_{kl} and t_{kl} are the output and target for sample k and output node l, respectively. Since the outputs in classification are restricted (see equations 11.2 and 11.3), the

MSE is relatively insensitive to outliers, which typically results in a robust training and a good performance on a validation set. Additionally, the MSE is computationally inexpensive to use in the training.

There is a countless number of training algorithms. In each algorithm there are a few training parameters that must be tuned by the user in order to get good and efficient training. Here, we will briefly describe the parameters in the *gradient descent algorithm* used in our example.

Given a number of input patterns and corresponding targets, the classification error can be computed. The error will depend on the weights and in the training we are looking for its minimum. The idea of the gradient descent can be illustrated by a man wandering around in the Alps, looking for the lowest situated valley. In every step he walks in the steepest direction and hopefully he will end up in the lowest valley. Below, we describe four parameters: *epochs, step size, momentum coefficient* and *weight decay*, and how they can be tuned.

The number of steps, epochs, is set by the user. It can be tuned using a plot of how the classification error change during the calibration (see Figure 11.4). Using too few epochs, the minimum is not reached,

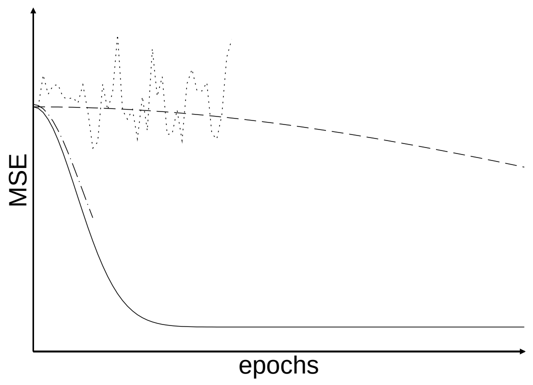

Figure 11.4 The MSE is plotted as a function of the number of training epochs. Using too many epochs (solid) or a too small step size (dashed) is time-consuming. Using too few epochs (dash-dotted) or a too large step size (dotted), the minimum is not reached.

yielding a training plot in which the error is not flattening out but still decreasing in the end. Using too many epochs is time consuming, since the network does not change once a minimum is reached.

The step size is normally proportional to the steepness, and the proportionality constant is given by the user. How large the step size should be depends on the typical scale of the error landscape. A too large step size results in a coarse-grained algorithm that never finds a minimum. A fluctuating classification error indicates a too large step size. Using a too small step is time consuming. An illustration of how weights are updated, depending on step size, is shown in Figure 11.5. The corresponding development of the MSE is illustrated in Figure 11.4.

The gradient descent method can be improved by adding a momentum term. Each new step is then a sum of the step according to the pure gradient descent plus a contribution from the previous step. In general, this gives a faster learning and reduces the risk of getting stuck in a local minimum. How much of the last step that is taken into account is often defined by the user in a momentum coefficient, between 0 and 1, where 0 corresponds to pure gradient descent. Having a momentum coefficient of 1 should be avoided since each step then depends on all previous positions. The gradient descent method with momentum is illustrated in Figure 11.5.

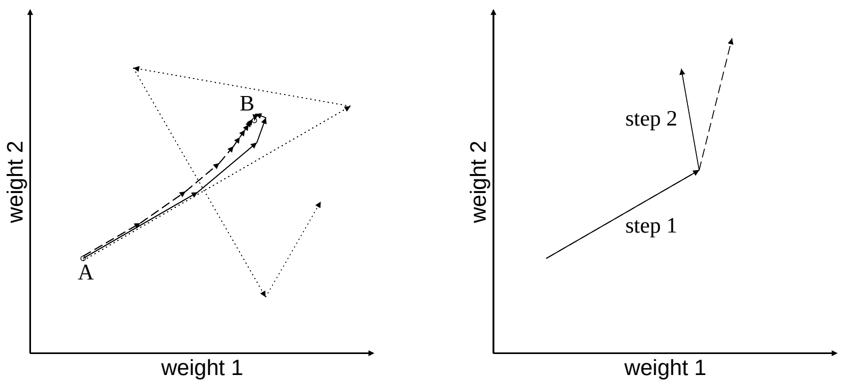

Figure 11.5. In the gradient descent (GD) algorithm the weights of an ANN are tuned to minimize the classification error. An ANN can during training be viewed as moving around in an error landscape defined by its weights. This is illustrated in two dimensions with an ANN that starts out in point A and has the lowest classification error in point B (left). With a suitable choice of the step size parameter, the minimum is reached (solid). It is time-consuming to reach the minimum with a small step size (dashed), whereas with a too large step size the algorithm fails to find the minimum (dotted). In GD with momentum (right), each new step (dashed step) is the sum of the step according to pure GD (step 2) plus a contribution from the previous step (step 1). In the figure a momentum coefficient of 1/3 was used.

The predictive ability of the network can be improved by adding a term that punishes large weights to the error measure. This so-called weight decay yields a smoother decision surface and helps avoiding overfitting. However, too large weight decay results in too simple networks. Therefore, it is important to tune the size of this term in a cross-validation scheme.

3.3 Cross-validation and tuning of ANNs

In the case of array data, where the number of samples typically is much smaller than the number of measured genes, there is a large risk of overfitting. That is, among the many genes, we may always find those that perfectly classify the samples, but have poor predictive ability on additional samples. Here, we describe how a supervised learning process can be carefully monitored using a cross-validation scheme to avoid overfitting. Of note, overfitting is not specific to ANN classifiers but is a potential problem with all supervised methods.

To obtain a classifier with good predictive power, it is often fruitful to take the variability of the training samples into account. One appealing way to do this is to construct a set of classifiers, each trained on a different subset of samples, and use them in a committee such that predictions for test samples are given by the average output of the classifiers. Thus, another advantage with using a cross-validation scheme is that it results in a set of ANNs that can be used as a committee for predictions on independent test samples in a robust way. In a cross-validation scheme there is a competition between having large training sets, needed for constructing good committee members, and obtaining different training sets, to increase the spread of predictions of the committee members. The latter results in a decrease in the committee error (Krogh and Vedelsby, 1995).

In general, 3-fold cross-validation is appropriate and it is the choice in the SRBCT example. In 3-fold cross-validation, the samples are randomly split into three groups. Two groups are used to train an ANN, and the third group is used for validation. This is repeated three times using each of the three groups for validation such that every sample is in a validation set once. To obtain a large committee, the random separation into a training and a validation set can be redone many times so that a set of ANNs are calibrated. The calibration of each ANN is then monitored by plotting both the classification error of the training samples and the validation samples as a function of training epochs (see Figure 11.6). A decrease in the training and the validation error with increasing epochs demonstrates the ability of the ANN to classify the experiments.

There are no general values for the learning parameters of ANNs, instead they can be optimized by trial and error using a cross-validation scheme. Overfitting results in an increase of the error for the validation samples at the point where the models begin to learn features in the training set that are not present in the validation set. In our example

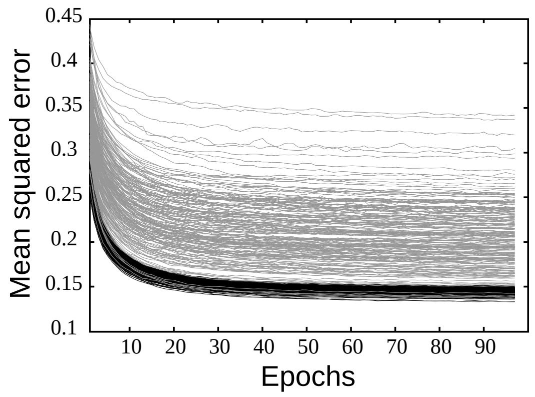

Figure 11.6 The mean squared error is plotted during the training iterations (epochs). A pair of lines, black (training) and gray (validation) represents one model (each corresponding to a random partitioning of the data into a training and validation set). Reproduced with permission (Khan *et al.*, 2001). ©2001, Nature Publishing Group.

there was no sign of overfitting (Figure 11.6). Overfitting can for example be avoided by early stopping, which means that one sets the maximal number of training iterations to be less than where overfitting begins or by tuning the weight decay. In addition, by monitoring the cross-validation performance one can also optimize the architecture of the ANN, for example the number of inputs to use. When tuning ANNs in this way, it is important to choose a cross-validation scheme that does not give a too small number of samples left in each validation set.

Even though cross-validation can be used to assess the quality of supervised classifiers, it is always important to evaluate the prediction performance using an independent test set that has not been used when the inputs or the parameters of the classifier were optimized. The importance of independent test sets should be appreciated for all supervised methods.

3.4 Random permutation tests

It is often stated that 'black boxes' such as ANNs can learn to classify anything. Though this is not the case, it is instructive to evaluate if the classification results are significant. This can be accomplished using random permutation tests (Pesarin, 2001). In microarray analysis, random permutation tests have mostly been used to investigate the significance of genes with expression patterns that discriminate between disease categories of interest (Golub *et al.*, 1999, Bittner *et al.*, 2000). In our example, we randomly permute the target values for the samples and ANNs are trained to classify these randomly labeled samples. This random permutation of target values is performed many times to generate

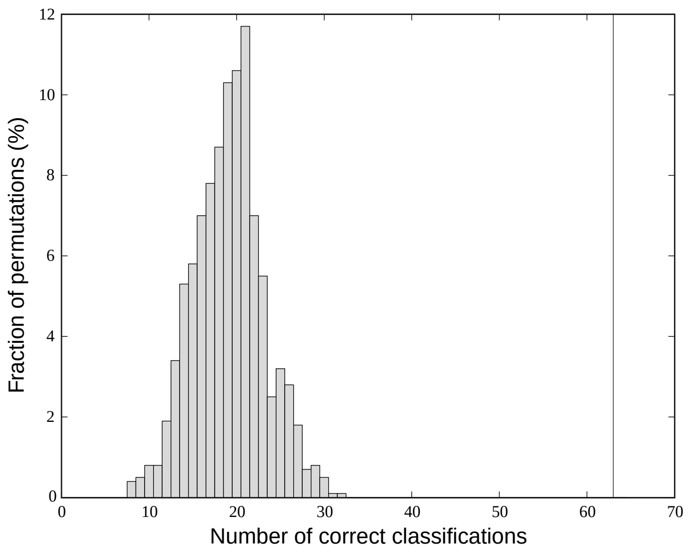

Figure 11.7. Validation results for randomly permuted sample labels (target values) using a committee of ANNs from a 3-fold cross-validation scheme. The number of correctly classified samples is histogrammed for the random permutations. Typically 20 samples are correctly classified for a random permutation, whereas all 63 samples are correct for the diagnostic categories (vertical line).

a distribution of the number of correctly classified samples that could be expected under the hypothesis of random gene expression. This distribution is shown for the 63 SRBCT samples using 3-fold cross-validation to classify the samples in Figure 11.7. The classification of the diagnostic categories of interest resulted in all 63 samples being correctly classified in the validation, whereas a random classification typically resulted in only 20 correctly classified samples. This illustrates the significance of the classification results for the diagnostic categories of the SRBCT samples.

3.5 Finding the important genes

A common way to estimate the importance of an input to an ANN is to exclude it and see how much the mean squared error increases. This is a possible way to rank principal components, but the approach is ill suited for gene ranking. The large number of genes, many of which are correlated, implies that any individual gene can be omitted without any noticeable change of the ANN performance.

Instead, we may rank genes according to how much the ANN output is affected by a variation in the expression level of a gene, keeping all other gene expression levels and all ANN weights fixed. Since the ANN output is a continuous function of the inputs (cf. section 3.1), this measure of the gene's importance is simply the *partial derivative* of the output, with respect to the gene input. To get the derivative, it is important that the dimensional reduction is mathematically simple, so the gene's influence on the ANN inputs can be easily calculated.

For well classified samples, the output is insensitive to all its inputs, as it stays close to 0 or 1 also for significant changes in the output function argument. Not to reduce the influence of well classified samples on the gene ranking, one may instead consider as sensitivity measure the derivative, with respect to the gene input, of the output function argument.

There are problems where neither single input exclusion, nor sensitivity measures as above, identifies the importance of an input (Sarle, 1998). Therefore, the gene list obtained with the sensitivity measure should be checked for consistency, by redoing the analysis using only a few top-ranked genes. If the performance is equally good as before or better, the sensitivity measure has identified important genes. If the performance is worse, the sensitivity measure may be misleading, but it could also be that too few top genes are selected, giving too little information to the ANN.

In Figure 11.8, the ANN performance of the SRBCT classification, as a function of included top genes, is shown. The good result when selecting the top 96 genes shows that these genes indeed are relevant for the problem. One can also see that selecting only the six top genes excludes too much information. The performance measure in this example is the average number of validation sample misclassifications of an ANN. One can also consider less restrictive measures, like the number of misclassifications by the combined ANN committee. If so, it may be that less than 96 top genes can be selected without any noticeable reduction in the performance. When more than 96 genes were selected the average performance of the committee members went down. This increase in the number of misclassifications is likely due to overfitting introduced by noise from genes with lower rank. However, the combined ANN committee still classified all the samples correctly when more than 96 genes were used.

Once genes are ranked, it is possible to further study gene expression differences between classes. For example, one can check how many top genes can be removed without significantly reducing the performance.

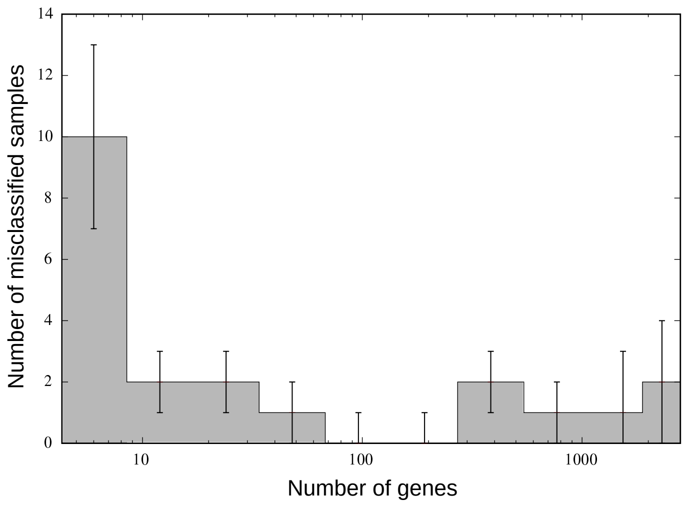

Figure 11.8. The number of misclassified samples for each ANN model is averaged over all models and plotted against increasing number of used genes. As can be seen, using the 96 highest ranked genes results in having cross-validation models that all on average give zero misclassifications for this example. Reproduced with permission (Khan *et al.*, 2001). ©2001, Nature Publishing Group.

Alternatively, several analyses can be made, where the same number of genes have been selected, but from different regions of the ranked gene-list. This approach was used when classifying breast cancers according to estrogen receptor status (Gruvberger *et al.*, 2001). Given the gene-list, 100 genes were selected according to rank in the ranges 1 to 100, 51 to 150, 101 to 200, etc. For each selection of 100 genes, the analysis was redone, and the ANN classification was shown to remain good using genes in the range 301 to 400, giving the conclusion that the gene expression patterns associated with estrogen receptor status are remarkably distinct.

4. Implementation

The principal component analysis can be formulated in terms of a *singular value decomposition* (see chapter 5) of the expression data matrix. It is straightforward to implement the complete analysis procedure outlined in this chapter in MATLAB with the Neural Network application toolbox, both available from The MathWorks (Natick, Massachusetts).

5. Summary

We have presented an ANN-based method for classification of gene expression data. This method was successfully applied to the example of classifying SRBCTs into distinct diagnostic categories. The key components of this classification procedure are PCA for dimensional reduction and cross-validation to optimize the training of the classifiers. In addition, we described a way to rank the genes according to their importance for the classification. Random permutation tests were introduced to assess the significance of the classification results. There are other ANN methods that have been used to classify gene expression data (Selaru *et al.*, 2002).

Acknowledgments

The authors want to thank C. Peterson, P.S. Meltzer, J. Khan and S. Gruvberger for their encouragement and support. MR and PE were supported by postdoctoral fellowships from the Swedish Research Council.

Notes

1. The data is available at http://www.nhgri.nih.gov/DIR/Microarray/Supplement/.

References

Almeida J.S. Predictive non-linear modeling of complex data by artificial neural networks. Curr Opin Biotechnol 2002; 13:72-6.

Bishop C.M. *Neural networks for pattern recognition.* Oxford: Oxford University Press, 1995.

Bittner M., Meltzer P., Chen Y., Jiang Y., Seftor E., Hendrix M., Radmacher M., Simon R., Yakhini Z., Ben-Dor A., Sampas N., Dougherty E., Wang E., Marincola F., Gooden C., Lueders J., Glatfelter A., Pollock P., Carpten J., Gillanders E., Leja D., Dietrich K., Beaudry C., Berens M., Alberts D., Sondak V., Hayward N., Trent J. Molecular classification of cutaneous malignant melanoma by gene expression profiling. Nature 2000; 406:536-40.

Golub T.R., Slonim D.K., Tamayo P., Huard C., Gaasenbeek M., Mesirov J.P., Coller H., Loh M.L., Downing J.R., Caligiuri M.A., Bloomfield C.D., Lander E.S. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. Science 1999; 286:531-7.

Gruvberger S., Ringnér M., Chen Y., Panavally S., Saal L.H., Borg Å., Fernö M., Peterson C., Meltzer P.S. Estrogen receptor status in breast cancer is associated with remarkably distinct gene expression patterns. Cancer Res 2001; 61:5979-84.

Hampshire J.B., Pearlmutter B. Equivalence proofs for multi-layer perceptron classifiers and the Bayesian discriminant function. Proceedings of the 1990 connectionist models summer school. San Mateo, CA: Morgan Kaufman, 1990.

Khan J., Simon R., Bittner M., Chen Y., Leighton S.B., Pohida T., Smith P.D., Jiang Y., Gooden G.C., Trent J.M., Meltzer P.S. Gene expression profiling of alveolar rhabdomyosarcoma with cDNA microarrays. Cancer Res 1998; 58:5009-13.

Khan J., Wei J.S., Ringnér M., Saal L.H., Ladanyi M., Westermann F., Berthold F., Schwab M., Atonescu C.R., Peterson C., Meltzer P.S. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nat Med 2001; 7:673-79.

Krogh A., Vedelsby J. Neural network ensembles, cross validation and active learning. Advances in Neural Information Processing Systems, Volume 7. Cambridge, MA: MIT Press (1995).

Nguyen D.V., Rocke D.M. Tumor classification by partial least squares using microarray gene expression data. Bioinformatics 2002; 18:39-50.

Pesarin F. *Multivariate Permutation Tests : With Applications in Biostatistics.* Hoboken, NJ: John Wiley & Sons, 2001.

Sarle W.S. How to measure the importance of inputs? Technical Report, SAS Institute Inc, Cary, NC, USA, ftp://ftp.sas.com/pub/neural/FAQ.html, 1998.

Selaru F.M., Xu Y., Yin J., Zou T., Liu T.C., Mori Y., Abraham J.M., Sato F., Wang S., Twigg C., Olaru A., Shustova V., Leytin A., Hytiroglou P., Shibata D., Harpaz N., Meltzer S.J. Artificial neural networks distinguish among subtypes of neoplastic colorectal lesions. Gastroenterology 2002; 122:606-13.