Class Prediction with Microarray Datasets

Simon Rogers†, Richard D. Williams‡ and Colin Campbell†
†Advanced Computing Research Centre,
University of Bristol, BS8 1TR, United Kingdom
‡Dept. of Paediatric Oncology, Institute of Cancer Research,
Sutton, SM2 5NG, United Kingdom

Abstract

Microarray technology is having a significant impact in the biological and medical sciences and class prediction will play an increasingly important role in the use and interpretation of microarray data. For example, classifiers could be constructed indicating the detailed subtype of a disease, its expected progression and the best treatment strategy. In this chapter we outline the main stages involved in the development of a successful class predictor for microarray datasets, including data normalisation, the different classifiers which can be used, different feature selection strategies and a method for determining how much data is required for a classification task given an initial sample set. We illustrate this process with both public domain datasets and a new dataset for predicting relapse versus non-relapse for a paediatric tumour.

1 Introduction

Microarray technology enables the simultaneous determination of the expression levels of thousands of genes. This has opened up a wealth of opportunities which could revolutionise our understanding in many areas of biological and medical research. cDNA and oligonucleotide microarrays have been used for a number of purposes. For example, in cancer research, comparisons of expression profiles have been used to find genes consistently over- or under-expressed in a tumour relative to a normal sample. In a comparison of

samples drawn from multiple sclerosis lesions and normal tissue, microarrays highlighted a number of over-expressing genes associated with the immune response [9]. As an auto-immune disease this is to be expected. However, a further set of genes with unanticipated relevance was also found to be associated with the disease. In a study of schizophrenia, microarrays were used to compare expression profiles from different areas of the brain. For cells in the prefrontal cortex this highlighted the under-expression of genes associated with functioning of the pre-synaptic juntion [16].

Apart from discovering the significance of individual genes, collective expression profiles can give new insights. For example, cluster analysis has been used to detect previously unrecognised tumour subcategories. Alizadeh et al [6] analysed lymphoma tissue samples and found evidence for two previously unrecognised subtypes of this disease. These subtypes had distinct expression profiles at the molecular level and patients belonging to the two subclasses had different clinical prognoses. Similarly Bhattacharjee et al [7] and Sorlie et al [14] have identified subtypes of lung and breast cancer.

A further important application of microarray technology is class prediction. For cancer applications, classifiers could be built which may reliably indicate subtype, invasiveness potential, expected progression and the best treatment strategy. This could have important clinical consequences. For example, for prostate cancer, tumours can range from indolent to aggressive and microarray technology could be used to predict where on this range a new sample lies. If an indolent tumour type is reliably indicated, invasive treatment could be avoided and replaced by regular surveillence.

In this chapter we will discuss the main issues arising in the use of microarray data for class prediction. To build a successful classifier we first need to prepare and normalise the data. Performance of the classifier can be affected by the methods used for handling missing values or poor readings and by our use of normalisation within and between slides. Thus, in section 2 we will start by briefly discussing these issues with illustrations from real-life datasets. In section 3 we will discuss different types of classifiers which can be used with microarray datasets, the use of feature selection and methods which implement a confidence measure for the class assignment. For some classification tasks the available data may be insufficient in size to construct a classifier with good generalisation. Thus in section 4 we will consider the dependence of generalisation ability on sample size and how we may use the currently available data to determine how much data is required to achieve a given prediction performance. We will focus on cDNA and oligonucleotide

arrays though related technologies such as small molecule or protein microarrays can pose similar class prediction problems.

2 Data preparation and normalisation

Successful data preparation obviously plays a crucial part in the construction of a reliable classifier since the experimental process can introduce serious artefacts in the data. The pre-processing steps required will depend on the experimental procedure used. For example, different procedures will be required for spotted cDNA and synthesised oligonucleotide microarrays (for example, for Affymetrix arrays most of the normalisations described below are commonly implemented by supplied data analysis software). In this section we will only sketch some of the main problems which can appear. We will assume that the ith slide in a microarray can be represented as a vector $\mathbf{x}_i$ with an associated class label y_i. The vector $\mathbf{x}_i$ has component *attributes* or *features* derived from the measured expression level of each spot on the microarray. Thus a feature may have a direct association with a particular gene or an expressed sequence tag (EST), or it may correspond to a replicate measurement or a probe with no known associations. Depending on the experimental procedure, it is common practice to first take the log of the data. Thus, for a cDNA array, suppose we derive expression ratios for a two channel experiment with one channel the control and the second derived from the sample of interest. For a doubling of expression over control the ratio is 2.0, whereas for a halving of expression the ratio is 0.5. The base 2 log of these ratios are 1.0 and -1.0 respectively, introducing a symmetry between over- and under- expression.

Next we need to normalise the dataset to adjust for any effects due to the experimental process rather than the biology. For cDNA microarrays, for example, there may be uneven hybridisation of dyes across the slide. This can give large scale systematic deviations in the data which negatively impact on eventual performance of the classifier. One approach is to determine a 2-dimensional Lowess (Loess) surface [26, 22] through the data to find any abnormal trends and correct the data values using this surface. The Lowess surface is a locally weighted polynomial regression surface with a low-degree polynomial fitted to a subset of the data around each point and the coefficients for this polynomial are found using a weighted least squares

procedure which gives most weight to points nearest the point of interest. A global Lowess correction has the possible drawback that a localised cluster of differentially expressed genes could be biologically significant, but they could be modified by this spatial trend adjustment. Alternative strategies include step-wise print-tip and scaled print-tip normalisations [17].

Apart from uneven hybridisation of the dyes, the dyes themselves may have different dynamic ranges. In Figure 1 we show a plot (from breast cancer) for a cDNA microarray in which two fluorescent dyes, Cy3 and Cy5, were used for the control and experimental signal of interest. Unfortunately these dyes can be detected by the scanner with different efficiencies. In Figure 1 there is a systematic deviation away from the $y{=}x$ line as the intensity is increased. A variety of strategies have been proposed to correct for this problem. One of the simplest is to rotate the data clockwise by 45°, determine a one dimensional Lowess curve through the data, correct for any systematic deviation and counter-rotate the data. Alternatives are available such as the use of dye-swapping replicate measurements.

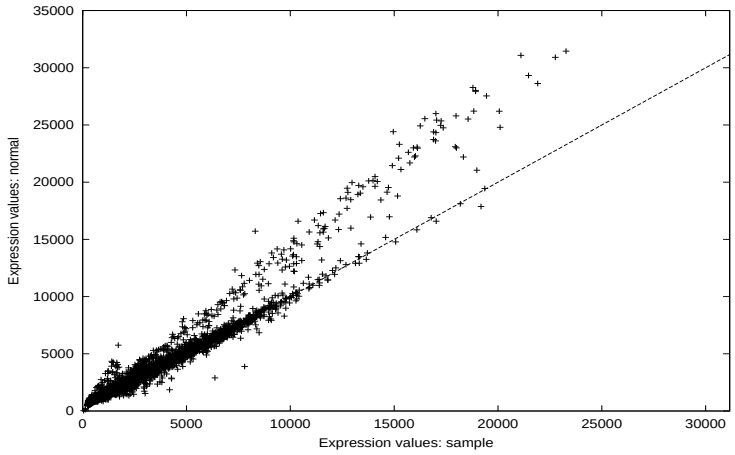

Figure 1: The different dynamic range of the dyes in this cDNA experiment leads to a systematic deviation with increasing intensity (illustrated by a deviation from the $y = x$ line). This is a relatively mild instance: in some cases the deviation can be substantial.

Having normalised data within-slide, the next step is to normalise the data between slides. The implicit scale and variability within each slide

can also make a difference. Again there are a variety of ways to normalise data between slides. For example, housekeeping genes (with expected little variability between slides) could be used to normalise the data. As a simple statistical approach, the median absolute deviation is a robust measure of variability in the data. For a given slide i with median d the median absolute deviation is the median of the differences $|x_{1i} - d|, |x_{2i} - d|, etc$ where x_{1i} is the first measured attribute, x_{2i} is the second, etc. We therefore normalise variability by applying a global rescaling of all the attribute values per slide to standardise the median absolute deviation across all slides.

After image processing there are usually poor readings from some spots on the slide. Suppose the combined dataset is presented with columns corresponding to different samples and rows corresponding to different features. Rows with discardable or missing values could simply be deleted from the subsequent analysis - however, this means we may not use the corresponding gene, and it could be important. If the number of good readings in a row is sufficiently high we could choose to retain the row and impute the small number of poor or missing entries. There is a large body of literature on statistical methods for handling missing values. However, one straightforward method (KNN-impute) is to select a set of K features each of which has an expression profile (row) similar to the row containing the missing or discarded entry. Weighting by similarity of expression profile, we calculate and use the weighted average of the K most similar rows as the new entry. Troyanskaya et al [15] report that using the 15 most similar rows gave the lowest error for the range of datasets they considered. Generally it is good practice to flag missing and poor readings and avoid their use in the intra-slide and inter-slide normalisations mentioned above.

3 Classification Techniques

Typically microarray datasets have a large number of features and a small number of examples. Thus, for example, the prostate cancer dataset of Singh et al [13] consists of 50 normal and 52 tumour samples with 12600 features each. Given that the data can therefore be viewed as a sparse set of points in a high-dimensional space (corresponding to a large number of features), it is not surprising that binary class datasets of this type are usually linearly separable: a hyperplane can readily be found which correctly separates both classes. This suggests a preference for simpler algorithms. For example, the

perceptron algorithm and its variants can efficiently handle linearly separable problems and can readily be used with these datasets. However, other factors are important. For example, in addition to assigning a class label to a new test point, it would be worth stating a confidence measure too. For some classifiers a confidence or probability measure is given and this will be an advantage in practical applications. A large number of classification algorithms could potentially be used, ranging from discriminant methods, to Gaussian Processes and classification trees. In this section we will only describe three popular choices for illustration: k-nearest neighbours, perceptrons and Support Vector Machines (SVMs).

3.1 K-nearest neighbour classifiers

One of the simplest classifiers to use is k-Nearest Neighbours (kNN). This classifier requires no training and the class of a new point is simply predicted to be the most common class among the k nearest neighbours. In its simplest form and for binary classification with $y_i = \pm 1$, the decision function is:

$$y = sign\left(\sum_{i|x_i \in \mathbf{K}} y_i\right) \tag{1}$$

where $\mathbf{K}$ is the set of neighbours closest to the new point, $\mathbf{x}$. This method can easily be extended to multi-class classification with the class of a new point determined by the consensus of its neighbours.

The set of nearest neighbours is determined by a distance metric that is usually the Euclidean distance in input space. Generally it is best to scale the influence of each neighbour depending on distance from the new point. This is easily accomplished by multiplying the class label of each neighbour by a weighting term, e.g. the reciprocal of the distance between the points. k-nearest neighbours was introduced in 1951 and since then there have been many extensions and variations proposed. One variation is the probabilistic nearest neighbour model (see e.g. [4]): not only does this assign a probability to the predicted label but it also automatically calculates the value of k resulting in a fully autonamous classifier.

3.2 Perceptron classifiers

The perceptron is a simple and effective classifier which can readily handle linearly separable datasets (Fig. 2). Indeed, using the idea of kernel substitution mentioned below, it can also handle non-linearly separable datasets. The *perceptron convergence theorem* states that the perceptron algorithm will converge on a solution after a finite number of passes through the dataset, provided a solution exists. If $\mathbf{x}_i$ are the feature vectors and z are binary-valued outputs then the decision function is:

$$z = sign(\mathbf{w} \cdot \mathbf{x}_i + b). \tag{2}$$

and the learning task is to find a weight vector $\mathbf{w}$ and bias b such that training feature vectors $\mathbf{x}_i$ map correctly to the corresponding labels y_i. This is achieved using an iterative training process with multiple passes through the data using the weight changes:

$$\Delta\mathbf{w} = y_i \mathbf{x}_i H\left[y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right)\right] \tag{3}$$

where $H(\theta)$ is the Heaviside Step function, we use $y_i = \pm 1$ and the bias b can be found by adding an extra component (say 0) to the feature vectors fixed at $x_0 = +1$ and with the bias given as $b = w_0$. The weight vector is therefore only corrected by an additive factor $y_i \mathbf{x}_i$ if the decision function gives the wrong label z on presentation of the ith feature vector.

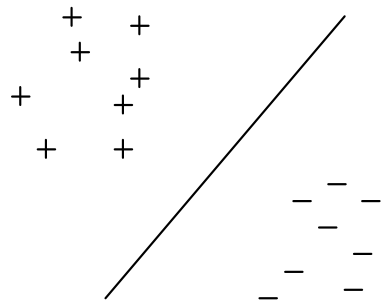

Figure 2: In input space a linear classifier finds a hyperplane separating the two classes

From (2) we see that the separatrix in the decision function is the linear hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ depicted in Figure 2. In an alternative view (the

geometric dual) hyperplanes become points and points become hyperplanes (Figure 3). This alternative representation gives insights about the solution found for both the perceptron and SVMs. In this dual representation we have a *version space* inside which each point represents a possible hyperplane correctly separating the datapoints in the original input space (Figure 2). The boundaries of version space derive from the datapoints. The iterations of the perceptron algorithm can be viewed as a trajectory which terminates inside version space starting from outside if the initial weights do not classify the training data correctly. Unfortunately, this indicates that the solution can be biased since it can depend on the starting point and the order of presentation of the patterns in (3). In addition, there is no simple way of implementing a confidence measure for class prediction with the perceptron. Next we will describe SVM classifiers which have a unique solution independent of the order of presentation - in addition a confidence can be assigned to the class label.

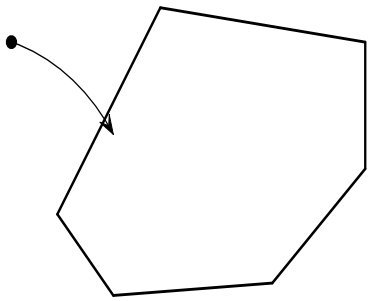

Figure 3: For the dual representation of Figure 2 datapoints become hyperplanes and hyperplanes correctly separating the data become points inside the *version space* (pictured, version space can be open in general). Thus the iterations of the perceptron algorithm corresponds to a trajectory into version space (shown) where the algorithm terminates with zero training error.

3.3 Support Vector Machines

Support Vector Machines (SVM's) [25, 2, 3] have also been used extensively for classification of microarray data. For binary classification, the motivation for this approach comes from theorems in learning theory which show that good generalisation does not depend on the dimensionality of the space (the

number of features used) but on maximising the *margin* or closest distance between the separating hyperplane and closest points on both sides (these are the *support vectors*, Figure 4). For m vectors $\mathbf{x}_i$ each with n features and corresponding labels y_i the task of maximising the margin is provably equivalent to maximising the following function with respect to the parameters α_i

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{4}$$

subject to the constraints

$$\alpha_i \geq 0 \qquad \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{5}$$

This is a standard constrained quadratic programming problem and can therefore be solved using any one of a number of readily available packages. Since quadratic programming is involved there are no local minima and the learning process always converges to the global minimum (the unique solution described above). For class prediction with microarrays one advantage is that learning is dependent on the number of samples m and not on the number of features (generally the larger number), hence learning is rapid.

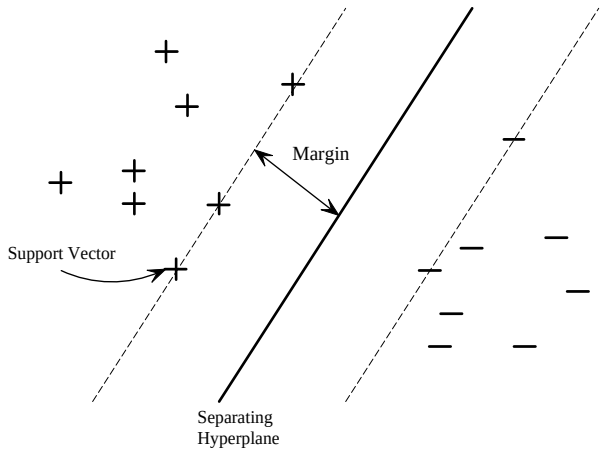

Figure 4: The SVM solution amounts to finding a separating hyperplane with maximal distance (*margin*) between itself and the closest points of each class on both sides (these are the *support vectors*).

The weight matrix in (2) is given by $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$, and if the values of α_i are determined at the optimum of (4,5) the decision function is:

$$f(\mathbf{z}) = sign\left(\sum_{i=1}^{m} y_i \alpha_i (\mathbf{x}_i \cdot \mathbf{z}) + b\right) \tag{6}$$

on presentation of a new input $\mathbf{z}$, and where the bias b is determined from:

$$b = -\frac{1}{2}\left[\max_{\{i|y_i=-1\}}\left(\sum_{j=1}^{m} y_j \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j)\right) + \min_{\{i|y_i=+1\}}\left(\sum_{j=1}^{m} y_j \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j)\right)\right]. \tag{7}$$

In the solution only some of the α_i values will be non-zero: these correspond to support vectors. Points further away have no influence on the orientation of the separating hyperplane, hence $\alpha_i = 0$ and they make no contribution in the decision function. The spectrum of α-values also carries information about the importance of particular datapoints in the training set, hence enabling *data cleaning*. In particular a large α_i relative to the others can indicate an outlier with undue influence on the orientation of the

separating hyperplane. For an early leukaemia microarray dataset [8] a mislabeled datapoint was detected this way, with subsequent correction of the diagnostic category.

Non-separable datasets. SVMs (and perceptrons) can readily handle non-separable datasets by *kernel substitution*. For SVMs good generalisation does not depend on the dimensionality of the space so mapping non-linearly separable data to a higher dimensional space (called *feature space*) can be implemented without loss of predictive ability. In a higher (or infinite) dimensional space linear separability by a hyperplane can be achieved. In the objective function (4) the datapoints, $\mathbf{x}_i$ only appear inside an inner product. Thus the mapping into feature space involves a mapping of the inner product:

$$\mathbf{x}_i \rightarrow \phi(\mathbf{x}_i) \qquad \text{therefore} \qquad \mathbf{x}_i \cdot \mathbf{x}_j \rightarrow \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \tag{8}$$

and feature space must necessarily be an *inner product* or *Hilbert* space. It is not necessary to define the functional form of the mapping $\phi(\mathbf{x}_i)$ as it is implicitly defined by the choice of mapped inner product or *kernel function*: $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(x_j)$. Only certain choices for the kernel function are allowed (*Mercer's conditions* must be satisfied [25, 2]). One example of a possible kernel function is the RBF kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left\{\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right\} \tag{9}$$

Given the choice of kernel function, learning a dataset for binary classification amounts to finding α_i which maximise the objective function:

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i.\mathbf{x}_j) \tag{10}$$

subject to the constraints (5). Once the optimal solution has been found, the decision function for a new point $\mathbf{z}$ is given by the sign of

$$f(\mathbf{z}) = \sum_{i=1}^{m} y_i \alpha_i K(\mathbf{z}, \mathbf{x}_i) + b \tag{11}$$

If there is no mapping to feature space then (4,5) apply and we have a *linear kernel*.

Multi-class classification. So far we have discussed binary classification, however, some tasks will involve multi-class prediction. A number of methods

have been proposed for performing multi-class prediction with SVMs but few are demonstrably better than using a set of 'one-against-all' classifiers (this approach can be used for perceptrons too). 'One-against-all' classifiers have the drawback that several members of the set may respond to a given input. As an alternative it is possible to use a series of binary classifiers in a tree (Fig. 5) to determine the outcome, and this method works satisfactorily if the number of classes is small.

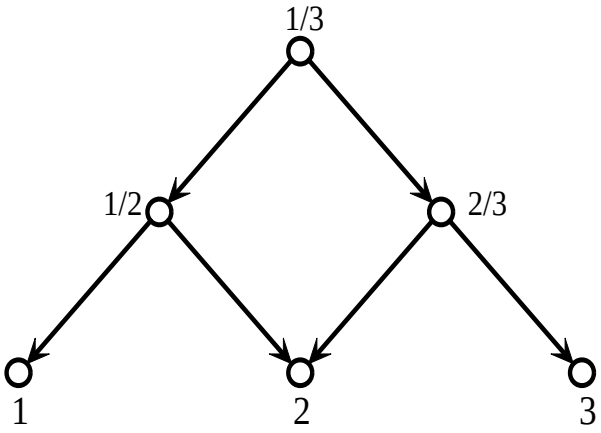

Figure 5: A multi-class classification problem can be reduced to a series of binary classification tasks (represented by the top two levels in the tree).

Soft Margins. Microarray datasets are typically noisy. An SVM could potentially fit the data well, including the noise, leading to a decrease in predictive accuracy. Noise can be handled using a *soft margin* and two methods are possible. For an L_1-*soft margin*, the optimisation task (4,5) is the same as before except the constraint $\alpha_i \geq 0$ is replaced by $C \geq \alpha_i \geq 0$ where C is the soft margin parameter, while for the L_2-*soft margin* the optimisation task is (4,5) except a small positive quantity λ is added to the kernel diagonal i.e. $K(\mathbf{x}_i, \mathbf{x}_i) \rightarrow K(\mathbf{x}_i, \mathbf{x}_i) + \lambda$. For most microarray datasets the use of a soft margin can lower the test error. However, *a priori* we do not know how much noise is present in the data nor the best value for C or λ, though a numerical cross-validation study (see below) might help to determine the appropriate setting. Without a soft margin and enforcing maximal separation between hyperplane and closest points (Fig. 4) the solution is known as a *hard margin*

classifier.

Confidence measures. If used for predicting diagnostic categories, for example, it is useful to have a confidence measure for the class assignment in addition to determining the class label. An SVM with linear kernel does have an inbuilt measure of confidence that could be exploited to provide a confidence measure for the assigned class, i.e. the distance of a new point from the separating hyperplane (Figure 4). A test point a large distance from the separating hyperplane should be assigned a higher degree of confidence than a point which lies close to the hyperplane.

The task of mapping this distance to probabilities has been solved in various ways. Here, we concentrate on one particular method, proposed by Platt [21]. Recall that the output of an SVM, before thresholding to ± 1, is given by

$$f(\mathbf{z}) = h(\mathbf{z}) + b \tag{12}$$

where:

$$h(\mathbf{z}) = \sum_i y_i \alpha_i K(\mathbf{x}_i, \mathbf{z}) \tag{13}$$

We use a parametric model to fit the posterior probability $P(y = 1|f)$ directly. The choice of parametric function for the posterior is the sigmoid (for more details see [21]):

$$P(y = 1|f) = \frac{1}{1 + \exp(Af + B)} \tag{14}$$

with the parameters A and B found from a training set (f_i, y_i). Define t_i as the target probabilities:

$$t_i = \frac{y_i + 1}{2} \tag{15}$$

i.e. using $y_i \in \{-1, 1\}$ we have $t_i \in \{0, 1\}$. We now minimize the log likelihood of the training data:

$$min \left[-\sum_i t_i \log(p_i) + (1 - t_i) \log(1 - p_i) \right] \tag{16}$$

where p_i is simply (14) evaluated at f_i. This is a straightforward 2-dimensional minimisation that can be solved using any one of a number of optimisation routines. Once the sigmoid has been found using this training set, we can use (14) to calculate the probability that a given test point belongs to each class. One question remains: how do we construct a training set to fit the sigmoid. The obvious choice would be the examples from the training set that were used to train the classifier. However, the training process biases the outputs for the support vectors to be ± 1 which is a very unlikely value for a new test point. However, for a linear SVM, the number of support vectors is usually reasonably low and so the bias should not be too severe. Therefore, it is generally acceptable to use the values from the training set to fit the posterior sigmoid. Figure 6 shows the training values and fitted sigmoid from a microarray dataset for an ovarian cancer dataset [12]. Note that no points are present in a large band between $+1$ and -1, due to the use of a hard margin and the data being linearly separable.

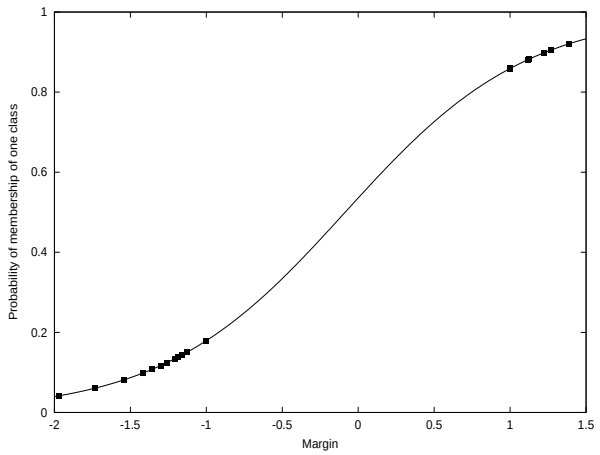

Figure 6: Probability of membership of one class (y-axis) versus margin. The plot shows the training points and fitted sigmoid for an ovarian cancer data set [12]. A hard margin was used which explains the absense of points in the central band.

3.4 Evaluating the test performance.

With a sparse population of points in a high-dimensional space, for most microarray datasets it is not difficult finding a separating hyperplane and therefore achieving zero training error. Zero training error and separability do not guarantee a low test error (it is possible to construct classifiers which can learn a training set with zero error but exhibit no generalisation ability).

For biological datasets (e.g. for yeast) there may be sufficient data for large training and test sets. However, for medical applications (e.g. cancer) this may not be the case and the paucity of samples can pose a problem. The standard approach to evaluating the test error is to use n-fold cross validation in which the dataset is split into $(n-m)$ training points and m test points with possible multiple resampling to establish a mean and standard deviation for the test error. The limiting case is leave-one-out (LOO) cross-validation with $(n-1)$ training points and 1 test point, with the single test point rotated successively through the data. LOO cross-validation gives the most *unbiased* estimate of the test error, it is least influenced by individual samples and most representative of performance for the distribution as a whole. However, LOO cross-validation can, depending on classifier, give a low bias but high *variance* solution which will be sub-optimal in terms of generalisation performance (see [22] for a detailed discussion). This *bias-variance tradeoff* can be handled using n-fold cross validation and $n = 5$ and $n = 10$ have been suggested as possible balanced estimates for many classifiers [22].

4 Feature selection

One characteristic of microarray data is that the number of features is usually very large and typically of the order of tens of thousands, often approximating the set of known genes in size. If a broad search is being pursued, the vast majority of these features are likely to be irrelevant for a given classification task and ideally we would like to remove them. Feature selection has two benefits. Firstly, large numbers of irrelevant features effectively inject noise into the classification task and can destroy generalisation, as we will illustrate later with an example. Secondly, from the viewpoint of interpretation, feature selection also highlights the most relevant features or genes in the data. Two general approaches may be used: *filter methods* in which features are scored individually (e.g. using statistical methods) prior to use of the classifier,

and *wrapper methods* in which the algorithm uses an internal procedure to eliminate redundant features.

4.1 Filter Methods

The choice of filter method can amount to a prior assumption about the way in which the significance of individual features are ranked. Roughly speaking, filter methods can be viewed as falling into two groupings - those measures more influenced by the consistency of the difference between classes and those more influenced by magnitude differences. For example, the set of ratios (1.1,1.1,1.1,1.1) is consistently different from the set (0.9,0.9,0.9,0.9) and features in both can be separated by a simple threshold (1.0). On the other hand there is a significant difference in the means of the set of ratios (1.0,1.0,5.0,5.0) and (1.0,1.0,0.1,0.1) even though the first two members of each set are the same. As for classification algorithms there are a large number of methods which can be employed. In this section we will therefore describe three commonly used approaches: the Mann-Whitney test and the TNoM score, the Fisher and Golub scores and the t-test (which also comes in many variants). To give an impression of their performance we will also evaluate these scores on a new dataset from cancer research.

Scoring by ranking. Statistical scoring based on ranking will be most influenced by consistency of a difference since if all values belonging to one class are ranked higher than all members of the other class this is determined as an improbable event even if the means of both classes do not differ a great deal. For example, for binary classification, the Mann-Whitney U test provides a measure of the difference between the medians of two populations [5] based on ranking. For a particular feature, two populations are determined from the expression ratios of the two distinctly labelled sample sets. The aim of the test is to estimate the probability that the two populations were drawn from an identical distribution. If this probability is very low, the feature is consistent with the class labels and will be significant. We start by combining the samples from the two classes and ranking them in numerical order. Each sample is then ranked with a value equal to its position in a line (i.e. a rank between 1 and (n_1+n_2) where n_1 and n_2 are the number of samples in classes 1 and 2.). If expression ratios are tied, they are given the average rank. The ranks for each of the two sample sets are summed. If the sums are R_1 and R_2, we now calculate the U statistic from $U_1 = n_1 n_2 + 0.5 n_1(n_1 + 1) - R_1$

and $U_2 = n_1 n_2 + 0.5 n_2(n_2+1) - R_2$. Choosing the smaller of U_1 and U_2 as the test statistic U we calculate $z = (U - \mu_U)/\sigma_U$ using:

$$\mu_U = \frac{n_1 n_2}{2} \qquad \sigma_U = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}} \tag{17}$$

z is a normally distributed random variable ($\mathcal{N}(0,1)$) and hence it is straightforward to determine the probability: the lower the probability the more significant the feature for discriminating the two classes.

Designed specifically for microarray datasets the Threshold Number of Misclassifications (TNoM) score [1] is closely related to ranking scores and will give a similar feature ranking to the Mann Whitney test, for example. We calculate the TNoM score by finding the best classification performance possible for the given feature. For the m values of the feature x_i (components of the feature vector) and label y_i we evaluate:

$$TNoM = \min_{a,b} \sum_{i=1}^{m} H\left[y_i \left(a x_i + b\right)\right] \tag{18}$$

thus we count an error every time $y_i \neq sign(a x_i + b)$. Since the argument inside the Heaviside step function is invariant under an arbitrary positive rescaling we can set $a = 1$ and evaluate the score using a 1-dimensional minimisation on b. If we are able to find a threshold such that all values above it belong to one class and all below belong to the other, then the TNoM score is 0. If the best threshold involves one misclassification the TNoM score is 1, etc. Given a particular class distribution we can also calculate the probability of getting a feature with any particular TNoM score. We can then compare the cumulative distribution function of the TNoM scores for a microarray dataset with the theoretical scores for a null model (i.e. purely random data) and thus determine if there is a significantly high number of low (i.e. good) scoring features compared to random occurrence and therefore if the data carries a significant information load. For example, in Figure 7 we give a plot of the cumulative distributions for an ovarian cancer data set [12]. Here, the curve determined from the ovarian cancer dataset lies considerably to the left of the theoretical curve expected for purely random data, suggesting a large number of discriminating features and significant information content.

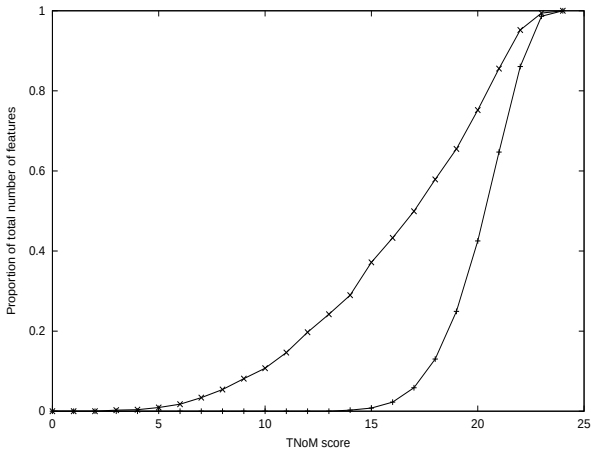

Figure 7: TNoM cumulative score curves for an ovarian cancer dataset [12]. The x-axis is TNoM score and y axis is the proportion of the total number of features. The left-hand curve corresponds to actual scores and the right-hand curve to the theoretical score for a null (random) model. As the curve from the actual scores lies significantly to the left this means the data contains a significant information content.

The Fisher and Golub scores. The second family of feature scoring techniques has more of an emphasis on the differences in magnitudes of the expression values between classes. Thus if we derive the means and standard deviations for the samples in each class, a good discriminating feature would have a large separation between the means and small standard deviations. If we define the means of the samples in the two classes as μ_1 and μ_2 and their standard deviations as σ_1 and σ_2, the Fisher (F) and Golub (G) scores are defined as follows

$$F = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \qquad\qquad G = \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2} \tag{19}$$

Scoring using the t-test. The final score we will consider is the well-known t-test for the difference between means of two populations. We calculate the means and variances (μ_1, μ_2, σ_1^2 and σ_2^2) for the two classes and then a weighted average of the two variances, with

$$s^2 = \frac{(n_1 - 1)\sigma_1^2 + (n_2 - 1)\sigma_2^2}{n_1 + n_2 - 2}. \tag{20}$$

and a test statistic t using

$$t = \frac{\mu_1 - \mu_2}{s\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{21}$$

from which a probability measure can readily be obtained from Student's distribution.

4.2 Recursive Elimination of Features

Rather than a prior scoring of features we could use a procedure within the algorithm to eliminate redundant features. As an illustration we will consider one method for SVMs which removes irrelevant features during the training process. When using a linear kernel, we noted that the weight matrix for an SVM can be expressed as $\mathbf{w} = \sum_{i=1}^{m} y_i \alpha_i \mathbf{x}_i$. The smallest component values of the weight matrix will have least influence in the decision function and will therefore be the best candidates for removal. For the *Recursive Feature Elimination* (RFE) method proposed by Guyon et al [20], the SVM is trained with the current set of features and the best candidate feature for removal is identified via the weight vector. This feature is removed and the process repeated until termination. One disadvantage of this method is that the process will be slow if there are a large number of features (typically the case for modern high density microarrays), though features could be removed in batches, of course. The algorithm can be terminated if the test error is logged throughout and passes through a minimum though, for SVMs, theoretical criteria such as LOO bounds (which estimate generalisation performance) can also be used [24]. This approach is general and can be applied to the weights generated using the perceptron algorithm, for example.

Aside from RFE there are a number of other approaches where feature selection is implemented directly within the algorithm. For example, with Bayesian approaches a *Bayesian prior* can be incorporated into the design of a classifier favouring sparse solutions, i.e. there is an explicit preference for a solution with a very limited number of features (this is known as Automatic Relevance Determination (ARD), see [23]). Bayesian ARD algorithms for

this purpose have been developed and work well with microarray datasets [27].

4.3 Feature Selection: A Case Study.

To illustrate the above we will apply these techniques to a new dataset for predicting relapse versus non-relapse for a paediatric malignancy (R.D. Williams *et al.* manuscript in preparation). In this study cDNA microarrays were used with an approximately balanced dataset of 27 samples. The normalisations mentioned in section 2 were very important and without proper intra- and inter-slide normalisation we found little predictive ability. Normalisations were implemented using a pre-existing software package [18] based on the Statistics for Microarray Analysis (SMA) R package of Speed et al [19].

A Support Vector Machine with linear kernel was used with filter methods for the feature selection. Recursive elimination of features was not used because of the large number of features (17790). Given the small size of the dataset leave-one-out (LOO) appears the best first strategy for evaluating the test error. During evaluation of the test error the 26 examples in the training set change with every leave-one-out rotation. Consequently, to derive a fair test statistic, the statistical scores were determined for each of the 27 evaluations on the test point without incorporating it into the computation of the score.

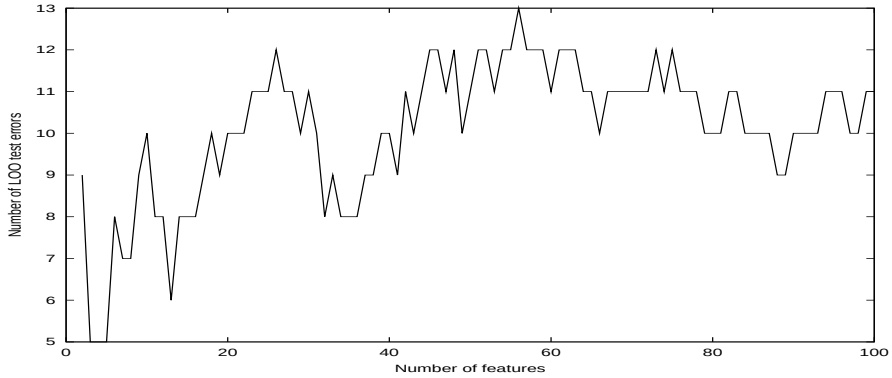

Figure 8: The number of LOO test errors (y-axis) versus number of top-ranked features (x-axis) remaining with feature-ranking by the Fisher score for predicting relapse or non-relapse. For 4 or fewer features a non-zero training error appears and the test error rises from the minimum of 5/27.

In Figures 8, 9 and 10 we show the LOO test error (y-axis) versus number of top-ranked features (x-axis) remaining for Fisher, Mann-Whitney and t-test scoring of features. All three scores indicate prediction is poor if all the features are used. However, good prediction is achieveable with a small number of features. Thus for the Fisher score the minimal LOO test error is 5/27, for Mann-Whitney 4/27 and for the t-test 1/27. Generally the t-test performs better than non-parametric methods such as the Mann-Whitney test so this comparative performance is to be expected. For 9-fold cross-validation with 1000 random reshufflings of the order we get a $6.4 \pm 1.2\%$ precentage test error with 3 features remaining. These results indicate prediction of relapse can be achieved, though any final confirmation would await evaluation on *de novo* data and biological confirmation of the genes used. If, as here, prediction can be achieved with the expression profile of a small set of genes then the result is interesting but should be viewed with caution. If a set of genes have highly correlated expression then only one member may be needed for building a successful predictor (the rest are effectively redundant), though this gene may not be as significant as some of the others. If, for example, the expression pattern of a particular gene is associated with positional effects (e.g. when a biologically significant overexpressed gene is located in an open chromatin domain, or in a region of the genome that has become duplicated in tumour cells), other genes that are not relevant to tumour outcome may be

co-regulated by the same mechanism. Their apparent relevance only derives from their position in this genomic region. Consequently the significance of genes should be evaluated by other methods independently of the feature selection used by the classifier.

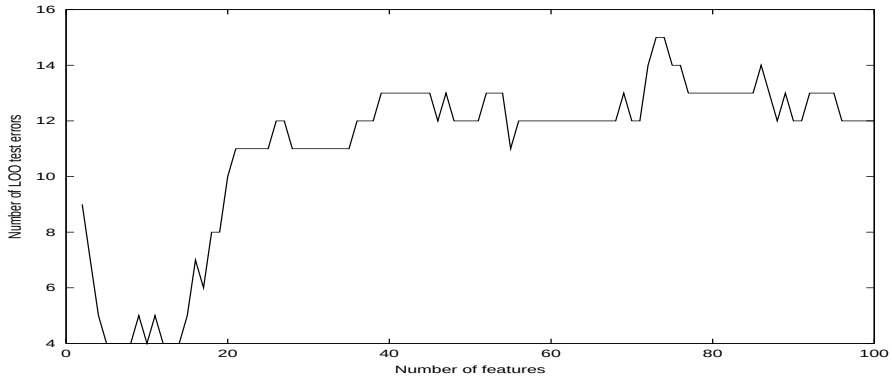

Figure 9: The number of LOO test errors (y-axis) versus number of top-ranked features remaining (x-axis) with ranking of features by the Mann-Whitney score. The minimum LOO test error improves on the Fisher score in Fig. 8 with a minimum of 4/27.

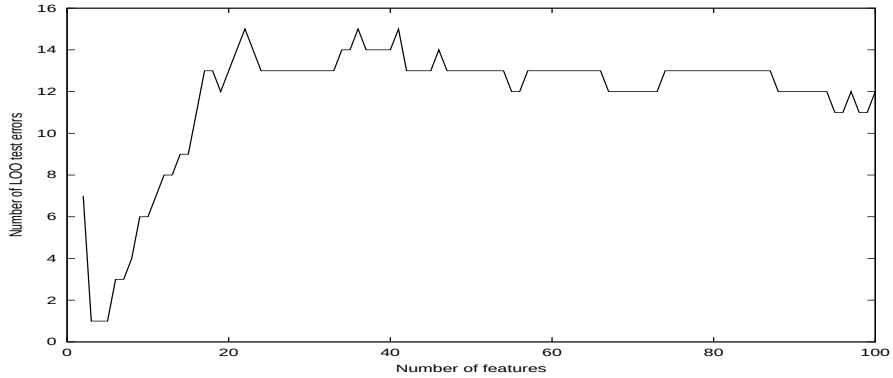

Figure 10: The number of LOO test errors (y-axis) versus number of top-ranked features remaining (x-axis) with ranking of features by the t-test. The minimum LOO test error was 1 from 27 with 3-5 features remaining. With 2 features remaining a non-zero training error was recorded. The minimal error is lower than for Fisher (Fig. 8) and Mann Whitney (Fig. 9) and the curve is noticeably smoother.

5 Estimating sample size requirements

Suppose for a dataset of size m we evaluate a test error $e(m)$. This test performance may not be adequate for the given task. For example, if the objective is to use a classifier for predicting invasive or non-invasive tumour types the predictor may need to have greater than 95% test accuracy to be acceptable in clinical practice. Hence given the current dataset size and test error we may want to know the dataset size m' which would give a required test error $e(m')$. The answer to this problem lies within learning theory where the theoretical dependence of generalisation error on sample size has been well studied from a number of viewpoints for the perceptron, SVM and other classifiers. *Learning curves* depict this dependence as the sample size varies. The shape of the learning curve depends on the data and the efficiency of the algorithm and in general, for typical datasets, these learning curves have an inverse power-law dependence:

$$e(m) = am^{-\alpha} + b \tag{22}$$

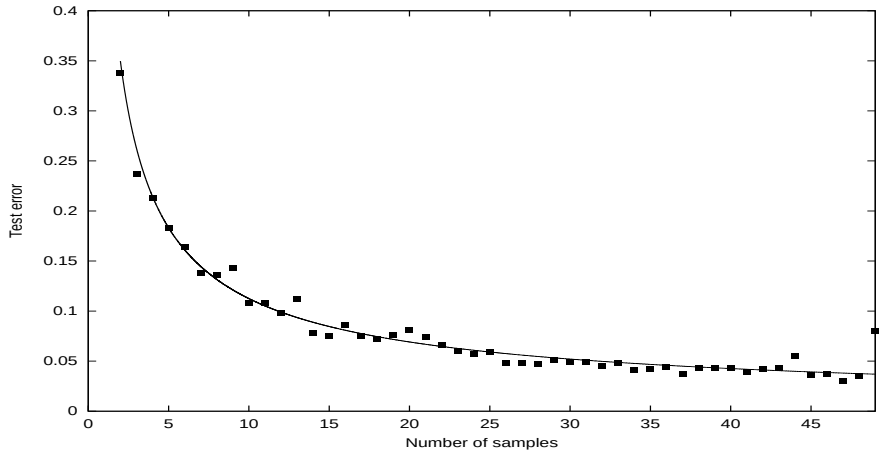

Figure 11: The learning curve for a lymphoma classification problem [6] distinguishing diffuse large B-cell lymphoma from other types. The fractional error rate is given on the y-axis and the sample size on the x-axis. Both leave-one-out test errors and power law fit are given.

for expected error $e(m)$ given m training samples, *learning rate* a, *decay rate* α and *Bayes error* b, which indicates the minimum test error achievable. The theoretical maximal decay rate is $\alpha = 1$, falling from this optimum for less efficient algorithms or complex learning problems. This functional form is expected to apply to the 25% and 75% quartiles in addition to the mean trend curve. Mukherjee et al [10] have investigated learning curves for a variety of microarray datasets, predominantly for cancer. In Figure 11 and 12 we show numerical test errors (errors on holdout data with resampling) and the learning curves for binary classification using a lymphoma [6] and a colon cancer [11] microarray dataset.

For very small training sets, the above inverse power-law model usually breaks down and there is not enough data to make an accurate extrapolation of the test error. Thus, for a workable method, we must determine the minimum training set size that will give sound results. This is done by finding at what training set size the test error becomes significant when compared with the null hypothesis of a random classifier:

$$H_0 : p(y = 1|x, \{x_1, y_1, \ldots, x_m, y_m\}) = p(y = -1|x, \{x_1, y_1, \ldots, x_m, y_m\}) \tag{23}$$

i.e. the conditional probability of a label being 1 or -1 is equal. The random classifier is constructed by training on the same input data but with the class labels randomly permuted. Suppose we have a total of m data points that we sub-sample into T_1 different training sets of size p and test sets of size $(m - p)$. Now, for each of our T_1 train/test realisations, we construct T_2 random data sets by permuting the training labels. We now use these $T_1 \times T_2$ random datasets to train classifiers and record the error of each classifier when evaluated with the respective non-random test set. Using these errors we construct an empirical distribution function for the random classifier,

$$P_p^{ran}(x) = \frac{1}{(T_1 \times T_2)} \sum_{i=1}^{T_1} \sum_{j=1}^{T_2} \theta(x - e_{n,i,j}) \tag{24}$$

where $e_{n,i,j}$ is the error of the j^{th} random dataset created from the i^{th} subsampling, with training size p. The significance of the classifier is $P_p^{ran}(\bar{e}_p)$ which is the percentage of random classifiers with error rate smaller than $\bar{e}_p$, the mean error of classifiers trained on the T_1 subsamplings with the *true* labels. So, for example, if it is decided that in order for sample sizes to be used, they should be significant with a probability of 95%, one would simply find the lowest value of p for which $P_p^{ran} > 0.05$. All values of p above this value are deemed to be significant and can be used in fitting the inverse power-law model.

Once a set of p pairs of training set sizes and empirical errors has been gathered, the learning curve can be fitted by minimising:

$$\min_{\alpha,a,b} \sum_{l=1}^{p} (a m_l^{-\alpha} + b - \bar{e}_{m_l})^2 \qquad \alpha, a, b \geq 0. \tag{25}$$

This is a convex optimisation problem when b is fixed. If we fix b, one can estimate α and a by taking logarithms and solving the equivalent linearised minimisation problem:

$$\min_{\alpha,a,b} \sum_{l=1}^{p} (\ln(a) - \alpha m_l + \ln(b - \bar{e}_{m_l}))^2 \qquad \alpha, a, b \geq 0. \tag{26}$$

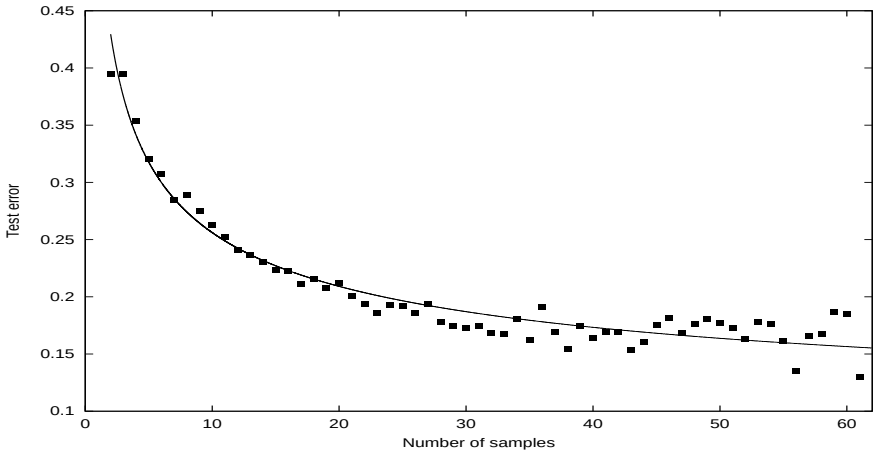

Figure 12: The learning curve and LOO test errors for a colon cancer classification problem [11].

A straightforward line search procedure over b enables us to find an estimate for α, a and b.

6 Conclusion

The interpretation of microarray datasets poses new problems for data analysis. It will be important to compare the performance of different classifiers on these datasets, to find the best feature selection strategies, the best method for implementing confidence measures and suitable procedures for handling imbalanced datasets. In addition, the performance of a classifier could be improved by using prior knowledge or the integration of knowledge from other domains. Many other problems arise in this context. For example, a novelty detector might be used to identify new instances which do not apparently belong to any class currently in the data. For some cancer microarray datasets there are samples with little apparent connection to the others, which could have arisen from other locations, belong to functionally different cell types (e.g. stem cells) or be sufficiently abnormal as to be unrepresentative of any assigned classes. In short, a better classifier could be built by not using all the data but by only using prototypical instances with removal of some ab-

normal datapoints (therefore given as 'unclassifiable'). The best strategies for quality pre-filtering prior to construction of a classifier is an interesting topic for future research.

Aside from classification, microarray datasets prompt many other problems for data analysis. One obvious issue is that the label may be continuous-valued. For example, rather than predicting indolent or aggressive for a tumour sample, it would be better to output a continuous value indicating where on a spectrum the new sample lies. This is the problem of regression and most of the normalisations and feature selection strategies described above will be relevant to this problem also.

The arrival of microarray technology is already giving important insights and this trend will only accelerate in the future. We can expect accurate classifiers which will indicate the detailed sub-type of a disease, the expected future progression and the optimal treatment strategy. The ability to predict the future course of a disease also means an implicit understanding of the role and significance of particular genes. This knowledge will be crucial for drug development. Aside from medical uses, microarray technology has the potential to revolutionise our understanding of biological processes.

References

[1] Ben Dor A. Tissue classification of gene expression profiles. *Journal of Computational Biology*, 7:559–583, 2000.

[2] Scholkopf B. and Smola A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2001.

[3] Campbell C. Kernel methods: a survey of current techniques. *Neurocomputing*, 48:63–84, 2002.

[4] Holmes C.C. A probabilistic nearest neighbour method for statistical pattern recognition. *Jounal Roy. Statist. Soc. B*, 64(2):295–306, 2002.

[5] Rees D.G. *Essential Statistics*. Chapman and Hall, 2001.

[6] Alizadeh A.A. et al. Different types of diffuse large b-cell lymphoma identified by gene expressing profiling. *Nature*, 403:503–511, 200.

[7] Bhattacharjee A. et al. Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma sub-classes. *Proc. Natl. Acad. Sci.*, 98:13790–13795, 2001.

[8] Golub T.R. et al. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[9] Lock C. et al. Gene-microarray analysis of multiple sclerosis lesions yields new targets validated in autoimmune encephalitis. *Nature Medicine*, 8:500–507, 2002.

[10] Mukherkee S. et al. Estimating dataset size requirements for classifying dna microarray data. *Journal of Computational Biology*, 10:119–142, 2003.

[11] Notterman D. et al. Transcriptional gene expression profiles of colorectal adenoma, adenocarcinoma and normal tissue examined by oligonucleotide arrays. *Cancer Research*, 61:3124–3130, 2001.

[12] Schummer M. et al. Comparative hybridization of an array of 21500 ovarian cdnas for the discovery of genes overexpressed in ovarian carcinomas. *International Journal on Genes and Genomes*, 238:375–385, 1999.

[13] Singh D. et al. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1:203–209, 2002.

[14] Sorlie T. et al. Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *Proc. Natl. Acad. Sci.*, 98:10869–10874, 2001.

[15] Troyanska O. et al. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17:520–525, 2001.

[16] Vawter M.P. et al. Microarray analysis of gene expression in the prefrontal cortex in schizophrenia: a preliminary study. *Schizophrenia Research*, 58:11–20, 2002.

[17] Yang Y.H. et al. Normalization for cdna microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Res.*, 30(4):e15, 2002.

[18] http://www.maths.lth.se/help/R/com.braju.sma/.

[19] http://www.stat.berkeley.edu/users/terry/zarray/Software/smacode.html.

[20] Guyon I., Weston J., Barnhill S., and Vapnik V. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[21] Platt J.C. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Alexander J. Smola, Peter Bartlett, Schölkopf Bernhard, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*, 1999.

[22] Hastie T. Tibshirani R. and Friedman J. *The Elements of Statistical Learning*. Springer, 2001.

[23] Neal R.M. *Bayesian Learning for Neural Networks (Lecture Notes in Statistics 118)*. Springer, 1996.

[24] Joachims T. Estimating the generalization performance of a svm efficiently. In *Proceedings of the Seventeenth International Conference on Machine Learning. Stanford, CA.*, 2000.

[25] Vapnik V.N. *Statistical Learning Theory*. John Wiley and Sons, inc., 1998.

[26] Cleveland W. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74:829–836, 1979.

[27] Li Y., Campbell C., and Tipping M. Bayesian automatic relevance determination algorithms for classifying gene expression data. *Bioinformatics*, 18:1332–1339, 2002.