Chapter 6

FEATURE SELECTION IN MICROARRAY ANALYSIS

Eric P. Xing
Computer Science Division
University of California, Berkeley
epxing@cs.berkeley.edu

1. Introduction

Microarray technology makes it possible to put the probes for the genes of an entire genome onto a chip, such that each data point provided by an experimenter lies in the high-dimensional space defined by the size of the genome under investigation. However, the sample size in these experiments is often severely limited. For example, in the popular leukemia dataset (Golub et al., 1999), which is used as a running example in this chapter, there are only 72 observations of the expression levels of each of 7,130 genes. This problem exemplifies a situation that will be increasingly common in the analysis of microarray data using machine learning techniques such as classification or clustering.

In high-dimensional problems such as these, feature selection methods are essential if the investigator is to make sense of his/her data, particularly if the goal of the study is to identify genes whose expression patterns have meaningful biological relationships to the classification or clustering problem. For example, for a microarray classification problem, it is of great clinical and mechanistic interest to identify those genes that directly contribute to the phenotype or symptom that we are trying to predict. Computational constraints can also impose important limitations. Many induction methods[1] suffer from the *curse of dimensionality*, that is, the time required for an algorithm grows dramatically,

[1] **Induction** (or inductive inference, inductive learning) refers to the following learning task: given a collection of examples $(x, f(x))$, find a function h that approximates f. The function h is called a **hypothesis**.

sometimes exponentially with the number of features involved, rendering the algorithm intractable in extremely high-dimensional problems we are facing with microarray data. Furthermore, a large number of features inevitably lead to a complex hypothesis and a large number of parameters for model induction or density estimation, which can result in serious overfitting over small datasets and thus a poor bound on generalization error.(Indeed we may never be able to obtain a 'sufficiently large' dataset. For example, theoretical and experimental results suggest that the number of training examples needed for a classifier to reach a given accuracy, or *sample complexity*, grow exponentially with the number of irrelevant features.)

The goal of feature selection is to select relevant features and eliminate irrelevant ones. This can be achieved by either explicitly looking for a good subset of features, or by assigning all features appropriate weights. Explicit feature selection is generally most natural when the result is intended to be understood by humans or fed into different induction algorithms. Feature weighting, on the other hand, is more directly motivated by pure modeling or performance concerns. The weighting process is usually an integral part of the induction algorithm and the weights often come out as a byproduct of the learned hypothesis.

In this chapter, we survey several important feature selection techniques developed in the classic *supervised learning* paradigm. We will first introduce the classic *filter* and *wrapper* approaches and some recent variants for explicit feature selection. Then we discuss several feature weighting techniques including WINNOW and Bayesian feature selection. We also include a brief section describing recent works on feature selection in the *unsupervised learning* paradigm, which will be useful for clustering analysis in the high-dimensional gene space.

Before proceeding, we should clarify the scope of this survey. There has been substantial work on feature selection in machine learning, pattern recognition and statistics. Due to space limit and the practical nature of this volume, we will refrain from detailed formal discussions and focus more on algorithmic solutions for practical problems from a machine learning perspective. Readers can follow the references of this chapter for more details. We use the leukemia microarray profile from the Whitehead Institute as our running example in the presentation.

2. Explicit Feature Selection

In explicit feature selection, we look for the subset of features that leads to optimal performance in our learning task, such as classifying biological samples according to their mRNA expression profiles.

Explicit feature selection can be formulated as a heuristic search problem, with each state in the search space specifying a specific subset of features (Blum and Langley, 1997). Any feature selection algorithm needs to deal with the following four issues which determine the nature of the heuristic search process: 1) How to start the search. One can either begin with an empty set and successively add features (*forward selection*) or start with all features and successively discard them (*backward elimination*) or other variations in between. 2) How to explore the search space. Popular strategies include a *hill-climbing* type of greedy scheme or a more exhaustive *best-first search*. 3) How to evaluate a feature subset. A common metric involves the degree of consistency of a feature with the target concept (e.g. sample labels) in the training data; more sophisticated criteria concern how selected features interact with specific induction algorithms. 4) When to stop the search. Depending on which search and evaluation scheme is used, one can use thresholding or a significance test, or simply stop when performance stops improving. It should be clear that all the above design decisions must be made for a feature selection procedure, which leaves practitioners substantial freedom in designing their algorithms.

2.1 The Filter Methods

The filter model relies on general characteristics of the training data to select a feature subset, doing so without reference to the learning algorithm. Filter strategies range from sequentially evaluating each feature based on simple statistics from the empirical distribution of the training data to using an embedded learning algorithm (independent of the induction algorithm that uses its output) to produce a feature subset.

Discretization and discriminability assessment of features. The measurements we obtained from microarrays are continuous values. In many situations in functional annotation (e.g., constructing regulatory networks) or data analysis (e.g. the information-theoretic-based filter technique we will discuss later), however, it is convenient to assume discrete values. One way to achieve this is to deduce the functional states of the genes based on their observed measurements.

A widely adopted empirical assumption about the activity of genes, and hence their expression, is that they generally assume distinct functional states such as 'on' or 'off'. (We assume binary states for simplicity but generalization to more states is straightforward.) The combination of such binary patterns from multiple genes determines the sample phenotype. For concreteness, consider a particular gene i (feature F_i). Suppose that the expression levels of F_i in those samples where F_i is in the

'on' state can be modeled by a probability distribution, such as a Gaussian distribution $\mathcal{N}(x|\mu_1, \sigma_1)$ where μ_1 and σ_1 are the mean and standard deviation. Similarly, another Gaussian distribution $\mathcal{N}(x|\mu_2, \sigma_2)$ can be assumed to model the expression levels of F_i in those samples where F_i is in the 'off' state. Given the above assumptions, the marginal probability of any given expression level x_i of gene i can be modeled by a weighted sum of the two Gaussian probability functions corresponding to the two functional states of this gene (where the weights $\pi_{1/2}$ correspond to the prior probabilities of gene i being in the on/off states):

$$P(x_i) = \pi_1 \mathcal{N}(x_i|\mu_1, \sigma_1) + \pi_2 \mathcal{N}(x_i|\mu_2, \sigma_2). \tag{6.1}$$

Such a model is called a *univariate mixture model* with two components (which includes the degenerate case of a single component when either of the weights is zero). The histogram in Figure 6.1a gives the empirical marginal of gene 109, which clearly demonstrates the case of a two-component mixture distribution of the expression levels of this gene in the 72 leukemia samples (which indicates this gene can be either 'on' or 'off' in these samples), whereas Figure 6.1b is an example of a nearly uni-component distribution (which indicates that gene 1902 remains in the same functional state in all the 72 samples).

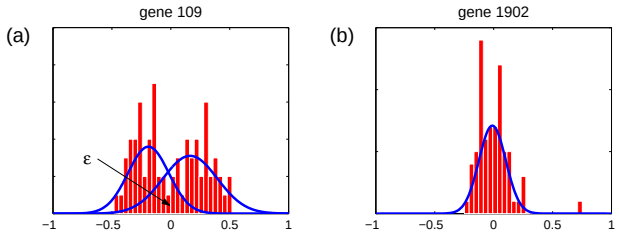


Figure 6.1. The histograms and estimated density functions of the expression profiles of two representative genes. The x-axes represent the normalized expression level.

For feature selection, if the underlying binary state of the gene does not vary between the two classes, then the gene is not discriminative for the classification problem and should be discarded. This suggests a heuristic procedure in which we measure the separability of the mixture components as an assay of the discriminability of the feature.

Given N microarray experiments for which gene i is measured in each experiment, the complete likelihood of all observations $X_i = \{x_{1i}, \ldots, x_{Ni}\}$ and their corresponding state indicator $Z_i = \{z_{1i}, \ldots, z_{Ni}\}$ is:

$$P_c(X_i, Z_i|\theta_i) = \prod_{n=1}^{N} \prod_{k=0}^{1} \left(\pi_{i,k} \left[\frac{1}{\sqrt{2\pi}\sigma_{i,k}} \exp\left\{ -\frac{(x_{ni} - \mu_{i,k})^2}{2(\sigma_{i,k})^2} \right\} \right] \right)^{z_{ni}^k}. \tag{6.2}$$

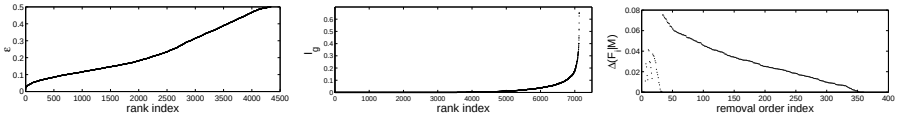
Random variable $z_{ni} \in \{0, 1\}$ indicates the underlying state of gene i in sample n (we omit sample index n in the subscript in the later presentation for simplicity) and is usually latent. We can fit the model parameters using the EM algorithm (Dempster et al., 1977). The solid curves in Figure 6.1a depict the density functions of the two Gaussian components fitted on the observed expression levels of gene 109. The curve in Figure 6.1b is the density of the single-component Gaussian distribution fitted on gene 1902. Note that each feature F_i is fitted independently based on its measurements in all N microarray experiments.

Suppose we define a decision $d(F_i)$ on feature F_i to be 0 if the posterior probability of $\{z_i = 0\}$ is greater than 0.5 under the mixture model, and let $d(F_i)$ equal 1 otherwise. We can define a *mixture-overlap probability*:

$$\epsilon = P(z_i = 0)P(d(F_i) = 1|z_i = 0) + P(z_i = 1)P(d(F_i) = 0|z_i = 1). \quad (6.3)$$

If the mixture model were a true representation of the probability of gene expression, then ϵ would represent the Bayes error of classification under this model (which equals to the area indicated by the arrow in Figure 6.1a). We can use this probability as a heuristic surrogate for the discriminating potential of the gene. Figure 6.2(a) shows the mixture overlap probability ϵ for the genes in the leukemia dataset in ascending order. It can be seen that only a small percentage of the genes have an overlap probability significantly smaller than $\epsilon \ll 0.5$, where 0.5 would constitute a random guessing under a Gaussian model if the underlying mixture components were construed as class labels.

The mixture model can be used as a quantizer, allowing us to discretize the measurements for a given feature. We can simply replace the continuous measurement f_i with the associated binary value $d(f_i)$.



(a) (b) (c)

Figure 6.2. Feature selection using filter methods. (a) Genes ranked by mixture-overlap probability ϵ. Only 2-state genes (i.e. those whose distributions of expressions in all samples have two mixture components corresponding to the 'on' and 'off' states) are displayed. (b) Genes ranked by their information gains I_g with respect to the reference partition induced by the sample labels. (c) The $\Delta(F_i|\mathbf{M})$ of the last 360 genes removed during MB filter. (The x axis indexes the inverse removal order of the genes. $\{x = 1\}$ refers to the gene that is removed last.)

Correlation-based feature ranking. We now turn to methods that make use of the class labels. Perhaps the simplest filter scheme of this category is to rank each feature individually based on its correlation to the target function. The goal of these methods is to find a good

approximation of the conditional distribution, $P(C \mid \mathbf{F})$, where $\mathbf{F}$ is the overall feature vector and C is the class label.

The *information gain* is commonly used as a surrogate for approximating a conditional distribution in the classification setting (Cover and Thomas, 1991). Let the class labels induce a *reference partition* $S_1, \ldots, S_C$ (e.g. different types of cancers). Let the probability of this partition be the empirical proportions: $P(T) = |T|/|S|$ for any subset T. Suppose a test on feature F_i induces a partition of the training set into $E_1, \ldots, E_K$. Let $P(S_c|E_k) = P(S_c \cap E_k)/P(E_k)$. We define the information gain due to F_i with respect to the reference partition as:

$$I_g = H(P(S_1), \ldots, P(S_C)) - \sum_{k=1}^{K} P(E_k) H(P(S_1|E_k), \ldots, P(S_C|E_k)), \tag{6.4}$$

where H is the entropy function[2]. To calculate the information gain, we need to quantize the values of the features. This is achieved to the mixture model quantization discussed earlier. Back to the leukemia example: quantization of all the 72 measurement of gene 109 results in 36 samples in the 'on' state (of which 20 are of type I leukemia and 16 type II) and 36 samples in the 'off' state (27 type I and 9 type II). According to Eq. 6.4, the information gain induced by gene 109 with respect to the original sample partition (47 type I and 25 type II) is:

$$I_g(F_{109}) = H(\tfrac{47}{72}, \tfrac{25}{72}) - \left(\tfrac{36}{72} H(\tfrac{20}{36}, \tfrac{16}{36}) + \tfrac{36}{72} H(\tfrac{27}{36}, \tfrac{9}{36})\right) = 0.0304.$$

The information gain reveals the degree of relevance of a feature to the reference partition. The greater the information gain, the more relevant the feature is to the reference partition. Figure 6.2(b) shows the information gain due to each individual gene with respect to the leukemia cancer labels. Indeed, only a very small fraction of the genes induce a significant information gain. One can rank all genes in the order of increasing information gain and select genes conservatively via a statistical significance test (Ben-Dor et al., 2000).

Markov blanket filtering. If we have a large number of similar or redundant genes in a dataset, all of them will score similarly in information gain[3]. This will cause undesirable dominance of the resulting classifier by a few gene families whose members have coherent expression

[2] For discrete cases, the entropy of distribution $\{P_1, \ldots, P_c\}$ is given by $H = \sum_{i=1}^{c} -P_i \log P_i$.
[3] Such situations could either arise from true functional redundancy, or result from artifacts of the microarray (e.g. the probe of a particular gene is accidentally spotted k times and appears as k 'similar genes' to a user who is unaware of the erroneous manufacturing process).

patterns, or even by a group of replicates of genes. This will seriously compromise the predictive power of the classifier. To alleviate this problem, we turn to *Markov blanket filtering*, a technique due to Koller and Sahami (1996), which can screen out redundant features.

Let $\mathbf{G}$ be a subset of the overall feature set $\mathbf{F}$. Let $\mathbf{f}_G$ denote the projection of $\mathbf{f}$ onto the variables in $\mathbf{G}$. Markov blanket filtering aims to minimize the discrepancy between the conditional distributions $P(C|\mathbf{F} = \mathbf{f})$ and $P(C|\mathbf{G} = \mathbf{f}_G)$, as measured by a conditional entropy:

$$\Delta_{\mathbf{G}} = \sum_{\mathbf{f}} P(\mathbf{f}) D(P(C|\mathbf{F} = \mathbf{f}) \parallel P(C|\mathbf{G} = f_{\mathbf{G}})), \tag{6.5}$$

where $D(P\|Q) = \sum_x P(x) \log(P(x)/Q(x))$ is the *Kullback-Leibler divergence*. The goal is to find a small set $\mathbf{G}$ for which $\Delta_{\mathbf{G}}$ is small.

Intuitively, if a feature F_i is conditionally independent of the class label given some small subset of other features, then we should be able to omit F_i without compromising the accuracy of class prediction. Koller and Sahami formalize this idea using the notion of a Markov blanket.

Definition 1 (Markov blanket) *For a feature set* $\mathbf{G}$ *and class label* C*, the set* $\mathbf{M}_i \subseteq \mathbf{G}$ *(*$F_i \notin \mathbf{M}_i$*) is a* Markov Blanket *of* F_i *(*$F_i \in \mathbf{G}$*) if: given* $\mathbf{M}_i$*,* F_i *is conditionally independent of* $\mathbf{G} - \mathbf{M}_i - \{F_i\}$ *and* C*.*

Biologically speaking, one can view the Markov blanket $\mathbf{M}_i$ of gene i as a subset of genes that exhibit similar expression patterns as gene i in all the samples under investigation. Such a subset could correspond to genes of isozymes, coregulated genes, or even (erroneous) experimental/manufactural replicates of probes of the same gene in an array.

Theoretically, it can be shown that once we find a Markov blanket of feature F_i in a feature set $\mathbf{G}$, we can safely remove F_i from $\mathbf{G}$ without increasing the divergence from the desired distribution (Xing et al., 2001). Furthermore, in a sequential filtering process in which unnecessary features are removed one by one, a feature tagged as unnecessary based on the existence of a Markov blanket $\mathbf{M}_i$ remains unnecessary in later stages when more features have been removed.

In most cases, however, few if any features will have a Markov blanket of limited size. Hence we must instead look for features that have an "approximate Markov blanket." For this purpose we define

$$\begin{aligned}\Delta(F_i|\mathbf{M}) = \sum_{f_{\mathbf{M}}, f_i} & P(\mathbf{M} = f_{\mathbf{M}}, F_i = f_i) \\ & D(P(C|\mathbf{M} = f_M, F_i = f_i) \parallel P(C|\mathbf{M} = f_M)).\end{aligned} \tag{6.6}$$

If $\mathbf{M}$ is a Markov blanket for F_i then $\Delta(F_i|\mathbf{M}) = 0$ (following the definition of Markov blanket), which means all information carried by

F_i about the sample is also carried by feature subset $\mathbf{M}_i$. Since an exact zero is unlikely to occur, we relax the condition and seek a set $\mathbf{M}$ such that $\Delta(F_i|\mathbf{M})$ is small. It can be proved that those features that form an approximate Markov blanket of feature F_i are most likely to be more strongly correlated to F_i. We can construct a candidate Markov blanket of F_i by collecting the k features that have the highest correlations (defined by the Pearson correlations between the original feature vectors that are not discretized) with F_i, where k is a small integer. This suggests an easy heuristic way to search for features with approximate Markov blankets (Koller and Sahami, 1996):

Initialize
- $\mathbf{G} = \mathbf{F}$

Iterate
- For each feature $F_i \in \mathbf{G}$, let $\mathbf{M}_i$ be the set of k features $F_j \in \mathbf{G} - \{F_i\}$ for which the correlations between F_i and F_j are the highest.
- Compute $\Delta(F_i|\mathbf{M}_i)$ for each i
- Choose the i that minimizes $\Delta(F_i|\mathbf{M}_i)$, and define $\mathbf{G} = \mathbf{G} - \{F_i\}$

This heuristic method requires computation of quantities of the form $P(C|\mathbf{M} = \mathbf{f}_M, F_i = \mathbf{f}_i)$ and $P(C|\mathbf{M} = \mathbf{f}_M)$, which can be easily computed using the discretization technique described in Sec. 2.1. When working on a small dataset, one should keep the Markov blankets small to avoid fragmenting the data[4]. The fact that in a real biological regulatory network the fan-in and fan-out will generally be small provides some justification for enforcing small Markov blankets.

Figure 6.2(c) displays the values of $\Delta(F_i|\mathbf{M}_i)$ (Eq. 6.6) for each F_i, an assessment of the extent to which the approximate Markov blanket $\mathbf{M}_i$ subsumes information carried by F_i and thus renders F_i redundant. Genes are ordered in their removal sequence from right to left. Note the increasing trend of $\Delta(F_i|\mathbf{M}_i)$ with more genes being removed, which reveals the expected decrease of redundancy of the remaining genes.

Decision Tree Filtering. A decision tree is itself an induction algorithm and learns a decision rule (a Boolean function) mapping relevant attributes to the target concept. Since a decision tree typically contains only a subset of the features, those included in the final tree can be viewed as a relevant feature subset and fed into another classification algorithm of choice. Thus, we can use the decision-tree algorithm as

[4] This refers to the situation in which, given small number of samples, one has to estimate, for example, $P(C|\mathbf{M} = \mathbf{f}_M)$ for many different possible configurations of $\mathbf{f}_M$. When $\mathbf{M}$ is large, each $\mathbf{f}_M$ configuration is seen only in a very small number of samples, making estimation of the conditional probabilities based on empirical frequency very inaccurate.

an embedded selection scheme under the filter model[5]. This approach has worked well for some datasets, but does not have a guarantee of performance gain on an arbitrary classifier since features that are good for a decision tree are not necessarily useful in other models. Essentially, a decision tree is itself a classifier (or an hypothesis), the features admitted to the learned tree inevitably bears *inductive bias*[6]. For high-dimensional microarray data, current methods of building decision trees may also suffer from data fragmentation and lack of sufficient samples. These shortcomings will result in a feature subset of possibly insufficient size. Nevertheless, if users have a strong prior belief that only a small number of genes are involved in a biological process of his/her interest, decision tree filtering could be a highly efficient way to pick them out.

2.2 The Wrapper Methods

The wrapper model makes use of the algorithm that will be used to build the final classifier to select a feature subset. Thus, given a classifier $\mathcal{C}$, and given a set of features F, a wrapper method searches in the space of subsets of F, using cross-validation to compare the performance of the trained classifier $\mathcal{C}$ on each tested subset. While the wrapper model tends to be more computationally expensive, it also tends to find feature sets better suited to the inductive biases of the learning algorithm and tends to give superior performance.

A key issue of the wrapper methods is how to search the space of subsets of features. Note that when performing the search, enumeration over all 2^N possible feature sets is usually intractable for the high-dimensional problems in microarray analysis. There is no known algorithm for otherwise performing this optimization tractably. Indeed, the feature selection problem in general is NP-hard[7], but much work over recent years has developed a large number of heuristics for performing

[5] If at each tree-growing step, we choose to incorporate the feature whose information gain with respect to the target concept is the highest among all features not yet in the tree, then decision tree filtering is in a sense similar to information gain ranking mentioned previously. However, general decision tree learning algorithm can also use other criteria to choose qualified features (e.g. classification performance of the intermediate tree resulted from addition of one more feature), and usually a learned tree needs to be pruned and cross-validated. These differences distinguish decision tree filtering from information gain ranking.

[6] Any preference for one hypothesis over another, beyond mere consistency with the examples, is called a **inductive bias**. For example, over many possible decision trees that are consistent with all training examples, the learning algorithm may prefer the smallest one, but the features included in such a tree may be insufficient for obtaining a good classifier of another type, e.g. support vector machines.

[7] NP stands for **nondeterministic polynomial**. In short, the NP-hard problems are a class of problems for which no polynomial-time solution is known.

this search efficiently. A thorough review on search heuristics can be found in (Russell and Norvig, 1995).

It is convenient to view the search process as building up a search tree that is superimposed over the state space (which, in our case, means each node in the tree corresponds to a particular feature subset, and adjacent nodes correspond to two feature subsets that differ by one feature). The root of this tree is the initial feature set which could be full, empty, or randomly chosen. At each search step, the search algorithm chooses one leaf node in the tree to expand by applying an **operator** (i.e. adding, removing, or replacing one of the features) to the feature subset corresponding to the node to produce a child. The first two search strategies described in the following can be best understood in this way.

Hill-climbing search. Hill-climbing search is one of the simplest search techniques also known as greedy search or steepest ascent. In fact, to perform this search one does not even need to maintain a search tree because all the algorithm does is to make the locally best changes to the feature subset. Essentially, it expands the current node and moves to the child with the highest accuracy based on cross-validation, terminating when no child improves over the current node. An important drawback of hill-climbing search is that it tends to suffer from the presence of local maxima, plateaux and ridges of the value surface of the evaluation function. *Simulated annealing* (occasionally picking a random expansion) provides a way to escape possible sub-optimality.

Best-First search. Best-first search is a more robust search strategy than the hill-climbing search. Basically, it chooses to expand the best-valued leaf that has been generated so far in the search tree (for this purpose we need to maintain a record of the search tree to provide us the tree frontier). To explore the state space more throughly, we do not stop immediately when node values stop increasing, but keep on expanding the tree until no improvement (within ϵ error) is found over the last k expansions.

Probabilistic search. For large search problems, it is desirable to concentrate the search in the regions of the search space that has appeared promising in the past yet still allow sufficient chance of exploration (in contrast to the greedy methods). A possible way to do so is to sample from a distribution of only the front-runners of the previously seen feature combinations. Define a random variable $z \in \{0,1\}^n$: a string of n bits that indicates whether each of the n features is relevant. We can hypothesize a parametric probabilistic model, for example, a de-

pendence tree or even a more elaborated Bayesian network, for random variable z and learn its distribution via an incremental procedure.

A dependence tree model is of the following form:

$$p(z) = p(z_r) \prod_{i \neq r} p(z_i | z_{\pi_i}), \tag{6.7}$$

where z_r is the root node and π_i indexes the parent of node i. This tree should be distinguished from the search tree we mentioned earlier where a node represents a feature subset and the size of the tree grows during search up to 2^n. In a dependence tree each node corresponds to an indicator random variable concerning the inclusion or exclusion of a particular feature, and the size of the tree is fixed. Any particular composition of feature subset we may select is a *sample* from the distribution determined by this tree. Given a collection of previously tested feature subsets, we can use the Chow-Liu algorithm (Chow and Liu, 1968) to find the optimal tree model that fits the data (in the sense of maximizing the likelihood of the tested instances)[8]. Then given the tree model, we can apply a depth first tree-traversal[9] that allows candidate feature subsets to be sampled from a concentrated subspace that is more likely to contain good solutions than mere random search. Figure 6.3 gives the pseudo-code of dependence-tree search. A detailed example of this algorithm can be found in (Baluja and Davies, 1997).

Initialization
- Generate N random bit-strings as candidate feature subsets

Iterate
- Evaluate each of the N candidate feature subsets by training the classifier on each feature subset and cross-validating
- Collect the αN top performing feature subsets (bit-strings), use them to update (with decay factor β) all pairwise mutual information between each pair of bits in the bit-strings
- Generate a maximum spanning tree for the bit-strings using Kruskal's algorithm
- Generate N bit-strings based on joint probability encoded by the dependence tree (using depth first traversal)

if *performance* converges, **end** iteration

Figure 6.3. The dependence-tree search algorithm

[8]We skip the details of the Chow-Liu algorithm due to the space limit. Essentially, it constructs a maximum spanning tree from a complete graph of the feature nodes whose edges are weighted by the mutual information of the random variables connected by the edge.

[9]A strategy of touching every node in a tree by always visit the child-node of the current node before going back to its parent-node and visit a sibling-node.

2.3 The ORDERED-FS Algorithm

For microarray data which have thousands of features, filter methods have the key advantage of significantly smaller computational complexity than wrapper methods. Therefore, these methods have been widely applied in the analysis of microarray data (Golub et al., 1999; Chow et al., 2002; Dudoit et al., 2000). But since a wrapper method searches for feature combinations that minimize classification error of a specific classifier, it can perform better than filter algorithms although at the cost of orders of magnitude of more computation time.

An additional problem with wrapper methods is that the repeated use of cross-validation on a single dataset can potentially cause severe overfitting for problems with a few samples but very large hypothesis spaces, which is not uncommon for microarray data. While theoretical results show that exponentially many data points are needed to provide guarantees of choosing good feature subsets under the classic wrapper setting (Ng, 1998), Ng has recently described a generic feature selection methodology, referred to as ORDERED-FS, which leads to more optimistic conclusions (Ng, 1998). In this approach, cross-validation is used only to compare between feature subsets of different cardinality. Ng proves that this approach yields a generalization error that is upper-bounded by the logarithm of the number of irrelevant features.

Filter$(D = \{X_{N\times M}, C\})$
- Quantize each feature via mixture modeling (MM)
- Rank all features via information gain (IG) filter
- Pick l features with highest IG, determine a removal order via Markov Blanket (MB) filter

Return an order π of the l features

Wrapper(D, H, π)
For $k = 1 : l$
- Train hypothesis $h_k \in H$ using the best k features
- Leave-One-Out CV on h_k, compute ϵ_k

End
$k^* = \arg\min_k \epsilon_k$
Return h_{k*} (optimal hypothesis), k^* (optimal cardinality)

Figure 6.4. The ORDERED-FS algorithm

Figure 6.4 presents an algorithmic instantiation of the ORDERED-FS approach in which filtering methods are used to choose best subsets for a given cardinality. We can use simple filter methods described earlier to carry out the major pruning of the hypothesis space, and use cross-validation for final comparisons to determine the optimal cardinality. This is in essence a hybrid of a filter and a wrapper method.

In Figure 6.5, we show training set and test set errors observed for the leukemia data when applying the ORDERED-FS algorithm[10]. Three different classifiers: a Gaussian quadratic classifier, a logistic linear classifier and a nearest neighbor classifier, are used (Xing et al., 2001). For all classifiers, after an initial coevolving trend of the training and testing curves for low-dimensional feature spaces, the classifiers quickly overfit the training data. For the logistic classifier and kNN, the test error tops out at approximately 20 percent when the entire feature set of 7130 genes is used. The Gaussian classifier overfits less severely in the full feature space. For all three classifiers, the best performance is achieved only in a significantly lower dimensional feature space.

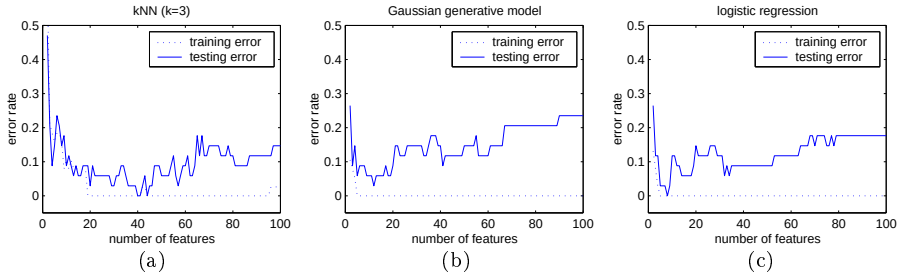


Figure 6.5. Classification in a sequence of different feature spaces with increasing dimensionality due to inclusion of gradually less qualified features. (a) Classification using kNN classifier; (b) A quadratic Bayesian classifier given by a Gaussian generative model; (c) A linear classifier obtained from logistic regression. All three classifiers use the same 2-100 genes selected by the three stages of feature filtering.

Figure 6.5 shows that by an optimal choice of the number of features it is possible to achieve error rates of 2.9%, 0% and 0% for the Gaussian classifier, the logistic regression classifier and kNN, respectively. (Note that due to inductive bias, different types of classifiers admit different optimal feature subsets.) Of course, in actual diagnostic practice we do not have the test set available, so these numbers are optimistic. To choose the number of features in an automatic way, we make use of leave-one-out cross-validation on the training data.

The results of leave-one-out cross-validation are shown in Figure 6.6. Note that we have several minima for each of the cross-validation curves. Breaking ties by choosing the minima having the smallest cardinality, and running the resulting classifier on the test set, we obtain error rates of 8.8%, 0% and 5.9% for the Gaussian classifier, the logistic regression classifier and kNN, respectively. The size of the optimal feature subsets determined hereby for the three classifiers are 6, 8 and 32, respectively.

[10]The 72 leukemia samples are split into two sets, with 38 (typeI/typeII=27/11) serving as a training set and the remaining 34 (20/14) as a test set.

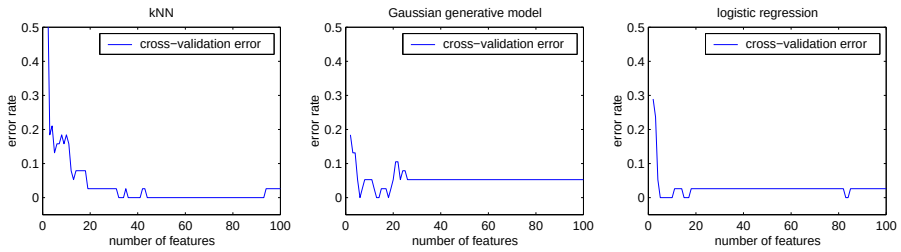


Figure 6.6. Plots of leave-one-out cross-validation error for the three classifiers.

3. Feature Weighting

Essentially, feature selection methods search in the combinatorial space of feature subsets, and pick an optimal subset of 'relevant' features as input to a learning algorithm. In contrast, feature weighting applies a weighting function to features, in effect assigning them a degree of perceived relevance and thereby performing feature selection implicitly during learning. In the following, we describe both a classic feature weighting scheme called WINNOW and a more general-purpose Bayesian learning technique that integrates feature weighting into the learning. For concreteness we consider the generalized linear model (GLIM) for classification, where the input $x \in X$ (i.e. the measure on the microarray) enters into the model via a linear combination $\xi = \theta^T x$ and the predictor, for example, the conditional distribution $p(y|x)$ of the corresponding label $y \in \{0, 1\}$ is characterized by an exponential family distribution with conditional mean $f(\xi)$, where f is known as a *response function*. Many popular classifiers belong to this family, for example, the logistic regression classifier:

$$P(y = 1|x, \theta) = \frac{1}{1 + e^{-\theta^T x}}. \tag{6.8}$$

3.1 The WINNOW Algorithm

The WINNOW algorithm is originally designed for learning Boolean monomials, or more generally, also k-DNF[11] formulas and r-of-k threshold functions[12], from noiseless data (Littlestone, 1988). Under these settings it enjoys worst-case loss logarithmic in the number of irrelevant

[11] A boolean formula is in **k-disjunctive normal form** (k-DNF) if it is expressed as a OR of clauses, each of which is the AND of k literals

[12] For a chosen set of $k(k <= n)$ variables and a given number $r(1 <= r <= k)$, an r-of-k threshold function is true if and only if at least r of the k relevant variables are true. The learning problem arises when both r and k are unknown.

features (i.e. the error rate is a function of the logarithm of the number of irrelevant features) . For more realistic learning tasks encountered in microarray analysis, such as building a classifier from training set $\{(x^1, y^1), \ldots, (x^k, y^k)\}$, we can use the following multiplicative update rule for the weight of feature j: if the classifier misclassifies an input training vector x^i with true label y^i, then we update each component j of the weight vector w as:

$$w_j \leftarrow w_j \exp(\eta x_j^i y^i), \tag{6.9}$$

where η is a learning rate parameter, and the initial weight vector is set to $w_j = w_{j,0} > 0$. Where does w appear in the classifier? Back to the GLIM model, this simply means a slight change of the linear term ξ in the *response function*: $\xi = \theta^T(w \star x)$, where $w \star x$ means element-wise product of vectors w and x.

There are a number of variants of the WINNOW algorithm, such as normalized WINNOW, balanced WINNOW and large margin WINNOW. See (Zhang, 2000) and reference therein for more details.

3.2 Bayesian Feature Selection

Bayesian methods for feature selection have a natural appeal, because they model uncertainties present in the feature selection problems, and allow prior knowledge to be incorporated. In Bayesian feature selection, each feature is associated with a selection probability, and the feature selection process translates into estimating the posterior distribution over the feature-indicator variables. Irrelevant features quickly receive low albeit non-zero probability of being selected (Jebara and Jaakkola, 2000). This type of feature selection (which is carried out jointly with inductive learning) is most beneficial when the number of training examples is relatively small compared to their dimensionality.

Again consider the classification of cancerous and non-cancerous samples measured on microarrays spanning n genes. Following the representation introduced in Section 2.2, we can index each of the possible 2^n subsets of features by a random variable z, then the linear combination term ξ in the response function $f(\xi)$ essentially becomes $\xi = \sum_{i=1}^n \theta_i z_i x_i$ (which obviates the effect of z_i as relevance indicator). Since the appropriate value of z is unknown, we can model the uncertainty underlying feature selection by a mixing prior:

$$P(\theta, z) = P_\theta(\theta) \prod_{i=1}^n P_z(z_i), \tag{6.10}$$

where P_θ is a (conjugate) prior for the model parameters θ, and

$$P_z(z_i) = p_i^{z_i}(1-p_i)^{1-z_i}, \tag{6.11}$$

where p_i controls the overall prior probability of including feature i.

For a training set $\mathcal{D} = \{X, Y\}$, the marginal posterior distribution $P(z|\mathcal{D})$ contains the information for feature selection, and the Bayesian optimal classifier is obtained by calculating:

$$P(y=1|x,\mathcal{D}) = \sum_z \int_\theta p(y=1|x,\theta)P(\theta,z|\mathcal{D})d\theta. \tag{6.12}$$

For high dimensional problems and complex models we may encounter in microarray analysis, exact probabilistic computation of the posterior distribution $P(z,\theta|X,Y)$ as well as evaluation of the decision rule is intractable. Therefore we need to use approximation techniques. George and McCulloch presented a detailed study of Markov Chain Monte Carlo methods such as Gibbs sampler or Metropolis-Hasting algorithm to explore the posterior distribution(George and McCulloch, 1997). Jebara and Jaakkola, on the other hand, took a Maximum Entropy Discrimination approach, and derived a closed-form solution of the posterior distribution $P(z,\theta|X,Y)$ for some model families such as logistic regression and support vector machines (Jebara and Jaakkola, 2000).

Recently, Ng and Jordan presented a Voting Gibbs classifier that solves the Bayesian feature selection problem in a surprisingly simple way (Ng and Jordan, 2001). Rather than taking Eq.6.11, they use a prior $P(\theta)$ assuming that the subset of relevant features is picked randomly according to the following procedure: **first**, sample the number r of relevant features uniformly from $\{0, 1, \ldots, n\}$; **then** a bit-string z in which r features are relevant is chosen randomly from one of the $\binom{n}{r}$ possible configurations. The prior $P(\theta)$ is constrained such that only the feature corresponding to an 'on' bit in z has a non-zero prior. Thus we have a parameter prior conditioned on z, $P_\Theta(\theta|z)$. Then we proceed to the usual Gibbs Voting classifier procedure where we sample N replicates of parameters θ from the posterior distribution $p(\theta|\mathcal{D})$, followed by N samples of y each from a particular $p(y=1|x,\theta)$. Finally, we vote for the result. A notable merit of this algorithm is its high tolerance to the presence of large number of irrelevant features. Ng and Jordan proved that their algorithm has sample complexity that is logarithmic in the number of irrelevant features.

4. Feature Selection for Clustering

Clustering is another important type of analysis for microarray data. In contrast to classification, in this paradigm (known as unsupervised

learning) a labeled training set is unavailable, and users are supposed to discover "meaningful" patterns (i.e. the existence of homogeneous groups that may correspond to particular macroscopic phenotypes such as clinical syndromes or cancer types) based on intrinsic properties of the data. Since microarrays usually measure thousands of genes for each sample, clustering a few hundred samples in such a high dimensional space may fail to yield a statistically significant pattern.

Eigenvector-based dimensionality reduction techniques such as Multidimensional Scaling (MDS) (Cox and Cox, 1994) and Principal Component Analysis (PCA) (Jolliffe, 1989) handle this problem by trying to map the data onto a lower-dimensional space spanned by a small number of "virtual" features (e.g. the principal eigenvectors of the sample covariance matrix in case of PCA). However, microarray measurement is usually a highly noisy data source. Results from matrix stability theory suggest that even small perturbation may cause the eigenvector methods to pick a different set of eigenvectors (Ng et al., 2001). Moreover, in methods like PCA, the principal eigenvectors represent those directions in the original feature space along which data has the greatest variance, the presence of a few highly variable but not informative "noisy" genes tends to mislead the algorithm to a wrong set of discriminative eigenfeatures. Finally, identifiability remains an outstanding issue. In many situations we would like to explicitly recover genes that significantly contribute to the sample partition of interest. Eigenvector methods do not offer a convenient way to do so. (Each eigenvector from PCA is a linear combination of all the original features, eigenvectors from the Gram matrix in MDS even lack an explicit connection to the original features.)

Feature selection under the clustering paradigm is substantially more difficult than that for classification. The main difficulty lies in the absence of reference information for evaluating the relevance of features. Before concluding this chapter, we briefly introduce some of the recent attempts on this problem.

Category utility. In the absence of class labels, one possible measure combining feature quality with the clustering performance is the average accuracy of predicting the value of each of the features in the data. The *category utility* metric is such a measure (Fisher, 1987). For a partition produced during clustering, the category utility is calculated as:

$$U = \frac{1}{K}\Big[\sum_{k=1}^{K} P(C_k) \sum_{i=1}^{I}\sum_{j=1}^{J(i)} P(F_i = x_{ij}|C_k)^2 - \sum_{i=1}^{I}\sum_{j=1}^{J(i)} P(F_i = x_{ij})^2\Big], \tag{6.13}$$

where $P(F_i = x_{ij}|C_k)$ is the probability of feature F_i taking value x_{ij} conditional on class membership C_k, and $P(F_i = x_{ij})$ is the marginal probability of feature F_i taking value x_{ij} in the dataset. Replacing the innermost summations with integration, category utility can be readily computed in the continuous domain for some distribution models (i.e. the mixture of two Gaussians we assumed in Section2.1).

Devaney and Ram proposed a wrapper-like feature selection strategy using category utility as an evaluation function (Devaney and Ram, 1997). Essentially, any clustering algorithm can be used to evaluate the candidate feature subsets produced by a search heuristic based on this metric. The search terminates when category utility stops improving.

Entropy-based Feature Ranking. Dash and Liu made an interesting empirical assumption on the relationship between the entropy and data distribution: two points belonging to the same cluster or in two different clusters will contribute less to the total entropy than if they were uniformly separated. They further reasoned that the former situation is more likely to happen if the *similarity* between the two points is either very high or low (rather than intermediate) (Dash and Liu, 2000). Then given distance measure (e.g. Euclidean distance or Pearson correlation) $D_{i,j}$ between point i and j, we can compute the entropy of a dataset as:

$$E = -\sum_{i \neq j} \sum_{j} \Big(S_{i,j} \log S_{i,j} + (1 - S_{i,j}) \log(1 - S_{i,j}) \Big), \tag{6.14}$$

where $S_{i,j} = \exp(-\alpha D_{i,j})$.

Based on this measure, one can rank features sequentially by discarding, one at a time, the feature whose removal results in minimum E. The optimal cardinality of the feature subset can be determined by an independent clustering algorithm (similar to the ORDERED-FS approach). However, the entropy assumption underlying this measure is only plausible when clusters are well separated and symmetric in shape. Under less ideal conditions, the performance is likely to break down.

The CLICK Algorithm. Xing and Karp proposed a strategy for feature selection in clustering that goes beyond the purely unsupervised feature evaluation techniques such as the entropy-based ranking or mixture-overlapping probability ranking (Xing and Karp, 2001). In their CLICK algorithm, they bootstrap an iterative feature selection and clustering process by using the most discriminative subset of features identified by the unsupervised mixture modeling to generate an *initial partition* of the samples. This partition is then used as an approximate reference for supervised feature selection based on informa-

tion gain ranking and Markov blanket filtering, and then the algorithm alternates between computing a new reference partition given the currently selected features, and selecting a new set of features based on the current reference partition. It is hoped that at each iteration one can expect to obtain an approximate partition that is close to the target one, and thus allows the selection of an approximately good feature subset, which will hopefully draw the partition even closer to the target partition in the next iteration.

5. Conclusion

At a conceptual level, one can divide the task of concept learning into the subtask of selecting a proper subset of features to use in describing the concept, and learning a hypothesis based on these features. This directly leads to a modular design of the learning algorithm which allows flexible combinations of explicit feature selection methods with model induction algorithms and sometimes leads to powerful variants. Many recent works, however, tend to take a more general view of feature selection as part of model selection and therefore integrate feature selection more closely into the learning algorithms (i.e. the Bayesian feature selection methods). Feature selection for clustering is a largely untouched problem, and there has been little theoretical characterization of the heuristic approaches we described in the chapter. In summary, although no universal strategy can be prescribed, for high-dimensional problems frequently encountered in microarray analysis, feature selection offers a promising suite of techniques to improve interpretability, performance and computation efficiency in learning.

Acknowledgments

I thank Professor Richard Karp and Dr. Wei Wu for helpful comments on the manuscript.

References

Baluja, S. and Davies, S. (1997). Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In *Proceedings of the Fourteenth International Conference on Machine Learning*.

Ben-Dor, A., Friedman, N., and Yakhini, Z. (2000). Scoring genes for relevance. In *Agilent Technologies Technical Report AGL-2000-19*.

Blum, A. and Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271.

Chow, C. and Liu, C. (1968). Approximating discrete probability distribution with dependency tree. *IEEE Transactions on Information Theory*, 14:462–367.

Chow, M. L., Moler, E. J., and Mian, I. S. (2002). Identification marker genes in transcription profiling data using a mixture of feature relevance experts. In Physiological Genomics (in press).

Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. Wiley, New York.

Cox, T. and Cox, M. (1994). *Multidimensional Scaling*. Chapman & Hall, London.

Dash, M. and Liu, H. (2000). Feature selection for clustering. In *PAKDD*, pages 110–121.

Dempster, A., Laird, N., and Revow, M. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, B 39(1):1–38.

Devaney, M. and Ram, A. (1997). Efficient feature selection in conceptual clustering. In *Proceedings of the Fourteenth International Conference on Machine Learning*.

Dudoit, S., Fridlyand, J., and Speed, T. (2000). Comparison of discrimination methods for the classification of tumors using gene expression data. In *Technical report 576, Department of Statistics, UC Berkeley*.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172.

George, E. I. and McCulloch, R. E. (1997). Approaches for Bayesian variable selection. *Statistica Sinica*, 7:339–373.

Golub, T., D.K., S., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M. L., Downing, J., Caligiuri, M., Bloomfield, C., and Lander, E. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537.

Jebara, T. and Jaakkola, T. (2000). Feature selection and dualities in maximum entropy discrimination. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann.

Jolliffe, I. (1989). *Principal Component Analysis*. Springer-Verlag, New York.

Koller, D. and Sahami, M. (1996). Toward optimal feature selection. In *Proceedings of the Thirteenth International Conference on Machine Learning*.

Littlestone, N. (1988). Learning quickly when irrelevant attribute abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318.

Ng, A. (1998). On feature selection: Learning with exponentially many irrelevant features as training examples. In *Proceedings of the Fifteenth International Conference on Machine Learning*.

Ng, A. Y. and Jordan, M. (2001). Convergence rates of the voting Gibbs classifier, with application to Bayesian feature selection. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

Ng, A. Y., Zheng, A. X., and Jordan, M. (2001). Link analysis, eigenvectors, and stability. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence, A Modern Approach*. Prentice Hall, New Jersey.

Xing, E., Jordan, M., and Karp, R. (2001). Feature selection for high-dimensional genomic microarray data (long version). In *Proceedings of the Eighteenth International Conference on Machine Learning*.

Xing, E. and Karp, R. (2001). Cliff: Clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. In *Proceedings of the Ninth International Conference on Intelligence Systems for Molecular Biology*.

Zhang, T. (2000). Large margin winnow methods for text categorization. In *KDD-2000 Workshop on Text Mining*, pages 81–87.