Analysis of Gene Expression Data **Spring Semester, 2002**

Lecture 7: May 23, 2002

Lecturer: Ron Shamir *Scribe: Simon Kamenkovich and Erez Greenstein*

7.1 Molecular Classification of Cancer

This part of the lecture is based on the papers of T. Golub, P. Tamayo, D. Slonim et al. [5, 11].

7.1.1 Overview

Although cancer classification has improved over the past 30 years, there has been no general approach for identifying new cancer classes(class discovery) or for assigning tumors to known classes(class prediction). In [5] and [11] a generic approach to cancer classification based on gene expression monitoring by DNA microarrays is described and applied to human acute leukemias as a test case. A class discovery procedure automatically discovered the distinction between acute myeloid leukemia (AML) [1] and acute lymphoblastic leukemia (ALL) [2] without previous knowledge of these classes. An automatically derived class predictor was able to determine the class of new leukemia cases. The results demonstrate the feasibility of cancer classification based solely on gene expression monitoring and suggest a general strategy for discovering and predicting cancer classes for other types of cancers, independent of previous biological knowledge.

7.1.2 Introduction

Determination of cancer type helps assigning an appropriate treatment to a patient. Cancer classification is based primarily on location or on morphological appearance of the tumor, that requests experienced biologist to distinguish known forms. The morphology classification has the limitation, that tumors with similar histopathological [3] appearance can have significantly different clinical courses and response to therapy. In general, cancer classification has been difficult because it has historically relied on specific biological insights, rather

[1] AML affects various white blood cells including granulocytes, monocytes and platelets. Leukemic cells accumulate in the bone marrow, replace normal blood cells and spread to the liver, spleen, lymph nodes, central nervous system, kidneys and gonads.

[2] ALL is a cancer of immature lymphocytes, called lymphoblasts (sometimes referred to as blast cells). Normally, white blood cells repair and reproduce themselves in an orderly and controlled manner but in leukaemia the process gets out of control and the cells continue to divide, but do not mature.

[3] Histopathology is the science that studies pathologic tissues.

then systematic and unbiased approaches for recognizing tumor subtypes. In [5] and [11] an approach based on global gene expression analysis is described.

The authors study the following three challenges:

1. Feature selection:
   Identifying the most informative genes for prediction. Clearly most genes are not relevant to cancer so we want to choose only the best features(criteria) for class prediction.

2. Class prediction or Classification:
   Assignment of particular tumor samples to already defined classes, which could reflect current states or future outcomes.

3. Class discovery or Clustering:
   Finding previously unrecognized tumor subtypes.

Thus, difference between classification and clustering is that clustering is unsupervised - we do not know anything about division, whereas classification is a supervised learning process, where division to subtypes is already known.

The authors studied the problem of classifying acute leukemia. Classification of acute leukemia began with the observation of variability in clinical outcome and subtle differences in nuclear morphology. Enzyme-based histochemical analysis, introduced in the 1960s, provided the first basis for classification of acute leukemias into acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML). Later ALL was divided into 2 subcategories: T-lineage ALL and B-lineage ALL. Some particular subtypes of acute leukemia have been found to be associated with specific chromosomal translocations.

Although the distinction between AML and ALL has been well established, no single test is currently sufficient to establish the diagnosis. Rather, current clinical practice involves an experienced hematopothologist's interpretation of the tumor morphology, histochemistry and immunophenotyping analysis performed in highly specialized laboratory. Although usually accurate, leukemia classification remains imperfect and errors do occur, for example when one type of cancer pretends to be another or when a mix of cancers accidently is identified as cancer of only one type.
The goal is to develop a systematic approach to cancer classification based on gene expression data.

Two data sets were taken:

- Learning set, containing 38 bone marrow samples (27 ALL and 11 AML), that were obtained at the same stage of the disease, but from different patients. On this set features will be learned and predictors will be developed to validate the test set.

- Test set, containing 34 leukemia samples (20 ALL and 14 AML), that consisted of 24 bone marrow and 10 peripheral blood samples.

RNA prepared from bone marrow mononuclear cells was hybridized to old generation Affymetrix oligonucleotide microarrays with 6817 human genes.

7.1.3 Feature Selection

The first goal was to find set of predicting genes,whose expression pattern are strongly correlated with the class distinction to be predicted.

Let c = (1,1,1,1,0,0,0,0) be a binary class vector, containing the class assigned to each sample (0 or 1).

Let $gene_i = (e_1, e_2, e_3, ..., e_{12})$ is expression vector for $gene_i$, consisting of its expression levels in each of the tumor samples.

We score a gene as a distinctor by $P(gene_i, c) = \frac{\mu_1 - \mu_0}{\sigma_1 + \sigma_0}$,where μ_i is mean expression level of samples in class i and σ_i is standard deviation of the expression levels in these samples, $i = 0, 1$. The fact that this metric accounts for both the class separation and the spread around the class means is the reason why the best performance(most accurate prediction) was obtained, comparing to other distance-based metrics such that Manhattan and Battacharayya distances, which have traditionally been employed as measures of class separation [7],[10]. The larger $P(gene, c)$, the better the gene distinction. Hence genes with highest $|P(g, c)|$ are chosen as predictor set.

Neighborhood analysis

The 6817 genes were sorted by their degree of correlation with distinction. To establish whether the observed correlation would be stronger then expected by chance, the researchers developed a method called "Neighborhood analysis". Assume that range of an expression levels is $[-1, 1]$, so expression vector of an ideal class distinctor would be represented by an "idealized expression pattern" c, in which the expression level is uniformly high in class 1 and uniformly low in class 2: $c = (1, 1, 1, 1, -1, -1, -1)$.

The idea of neighborhood analysis is simply look at a neighborhood of a fixed size around c, count the number of gene expression vectors within it, compare it to the number of expression vectors within the neighborhood of the same size around a random permutation of c - $\pi(c)$. Let $N(g)$ be a number of genes such that $P(g, c) > \alpha$, for some constant α. Let R(g) be a number of genes such that $P(g, \pi(c)) > \alpha$. By trying many random permutations we can determine if the neighborhood around c holds more gene expression vectors that we expect to see by chance. If we find that $N(g) \gg E(R(G))$, we can conclude that the class distinction represented by c is likely to be predictable from the expression data. This analysis is illustrated in Figure7.1. Note that permuting preserves class sizes.

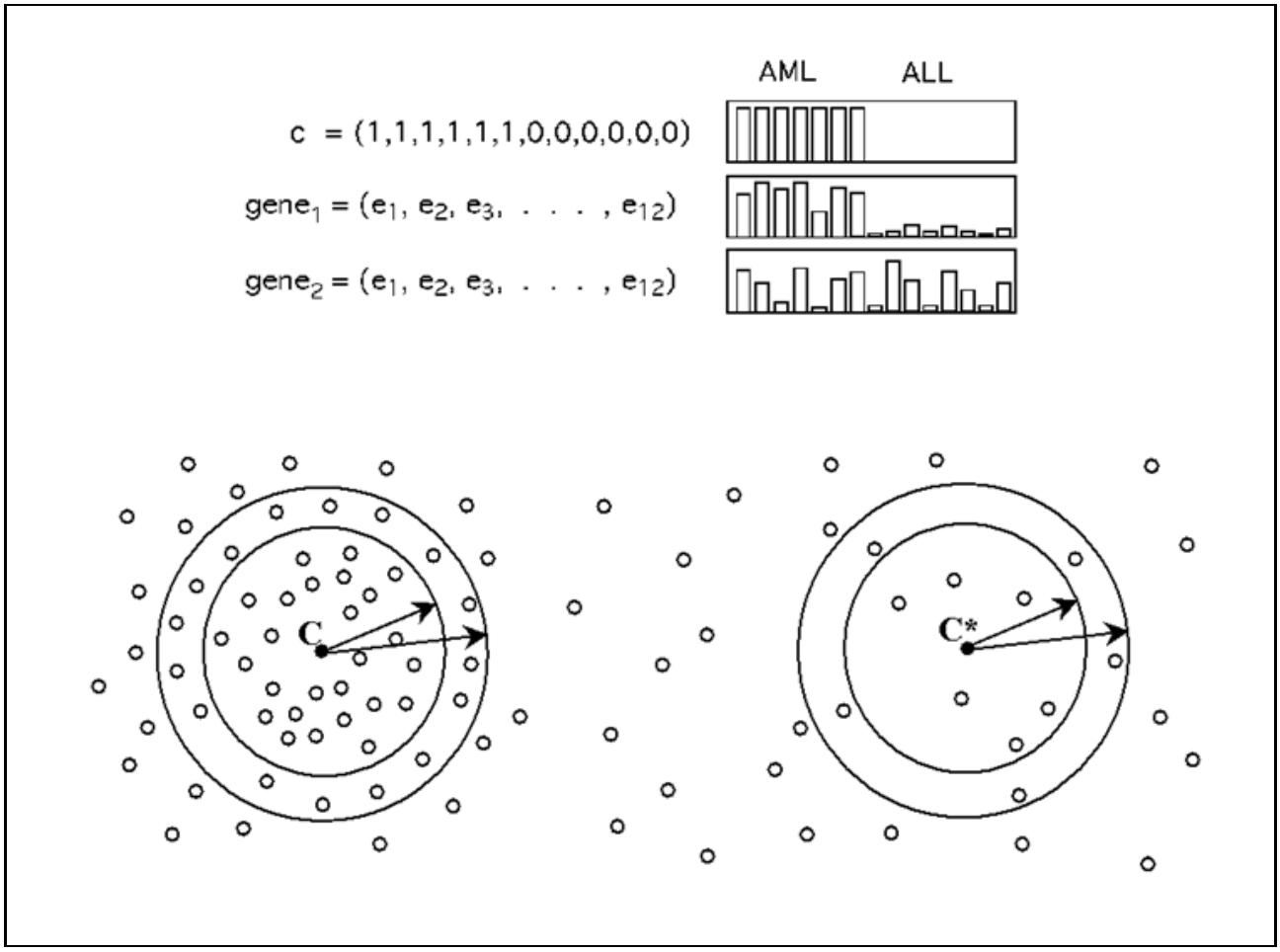

Figure 7.1: Schematic illustration of Neighborhood analysis.
The class distinction is represented by c. Each gene is represented by expression level in each of the tumor samples. In the figure, the data set is composed of 6 AMLs and six ALLs. Gene g_1 is well correlated with the class distinction, whereas g_2 is poorly correlated with c. The results are compared to the corresponding distribution obtained for random idealized vector c^*, obtained by randomly permuting the coordinates of c.

Another approach is to present neighborhoods, whose radii are a function of the expression value i.e., for each class c we count genes with $P(g,c) > x$ as a function of x. In this approach we do not aspire to get more genes in the same circle, as in previous one, but to obtain small circles, containing many genes.

We can calculate this distribution for known subsets and for random permutations. In Figure 7.2 we can see how much genes(axis y) has P at least the value on axis x. If our data was random we were expecting that observed data curve should be much closer to the median, but it's not so and for $P(g,c) > 0.3$ it's located far even from 1% significance level, that means there are a lot of informative genes in our data set. In summary we have about 1000 more genes that are highly correlated with the AML-ALL class distinction, then we could expect in random, indicating sufficient information for class prediction (classification).

Once the neighborhood analysis shows that there are genes significantly correlated with class distinction c, we want to use known samples to create a "class predictor" capable of assigning a new sample to one of two classes. Our goal is to choose the k genes most closely correlated with AML-ALL distinction in the known samples.

Choosing a prediction set

We could simple choose the top k genes by the absolute value of $P(g,c)$, but this allows the possibility that, for example, all genes might be expressed at a high level in class 1 and a low level(or not at all)in class 2. However, predictors do better when they include some gene expressed at high levels in each class, because to assign new sample to a class, it must correlate with highly expressed genes in this class, so we choose the top k_1 genes (highly expressed in class 1) and the bottom k_2 genes (highly expressed in class 2) so that

How we choose k_1 and k_2:

- They must be roughly equal to prevent a sample, that is located somewhere between 2 classes, to be assigned to class 1 just because $k_1 > k_2$ or the opposite.
- The fewer genes we choose the more statistically significant they will be.
- The more genes we choose the robust the results will be obtained. We know that gene expression is different in different people ,different tissues etc., so it is not reasonable that one gene is enough to predict. Also we know that cancer is related to many biological processes, so we expect to find several genes to represent each of these processes.
- Too many genes are not helpful, because if all of them are significantly correlated with a class distinction, it is unlikely that they're all represent different biological mechanisms. Their expression patterns are probably dependent, so that the are unlikely to add information not already provided by the others.

Thus, generally we pick few tens of genes. Let S be the set of the chosen informative genes.

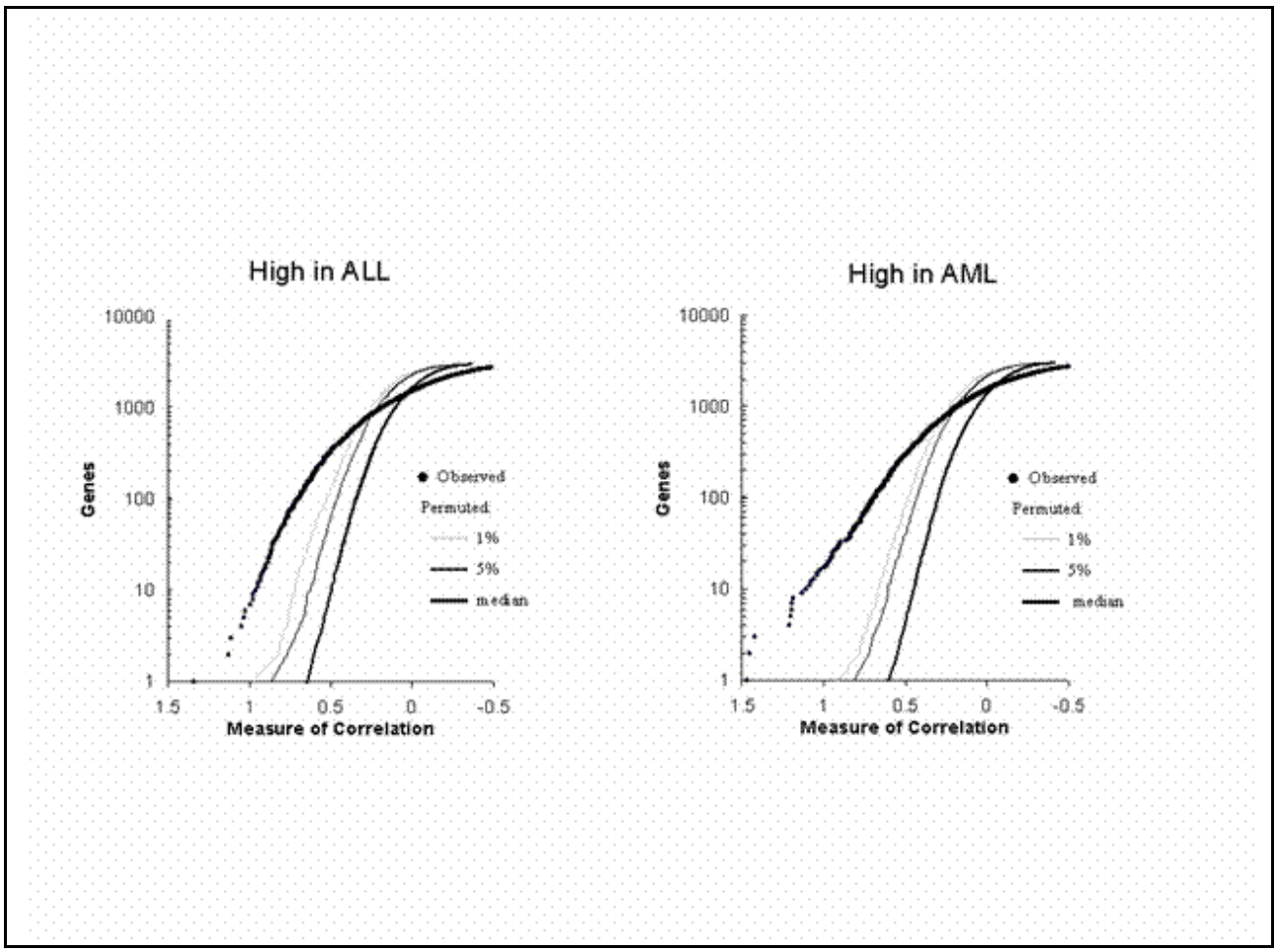

Figure 7.2: Neighborhood analysis: ALL versus AML.
For the 38 leukemia samples in the initial data set, the plot shows the number of genes within various "neighborhoods" of the ALL-AML class distinction together with curves showing the 5% and 1% significance levels for the number of genes within corresponding neighborhoods of the randomly permuted class distinction. Genes highly expressed in ALL compared to AML are shown in the left panel; those more highly expressed in ALL compared to AML are shown in the left panel. The large number of genes highly correlated with the class distinction is apparent. For example, in the left panel the number of genes with correlation $P(g,c) > 0.3$ was 709 for the AML-ALL distinction, but had a median of 173 genes for random class distinctions. $P(g,c) = 0.3$ is the point where the observed data intersect the 1% significance level, meaning that 1% of random neighborhoods contain as many points as the observed neighborhood around the AML-ALL distinction.

7.1.4 Class prediction

Prediction by weighted voting

We now describe the voting scheme presented in [5]. Each gene in S gets to cast its vote for exactly one class. The gene's vote on a new sample x is weighted by how closely its expression in the learning set correlates with c: $w(g) = P(g, c)$. The vote is the product of this weight and a measure of how informative the gene appears to be for predicting the new sample.

Intuitively, we expect the gene's level in x to look like that of either a typical class 1 sample or a typical class 2 sample in the learning set, so we compare expression in the new sample to the class means in the learning set. We define a "decision boundary" as halfway between the two class means: $b_g = \frac{\mu_1 - \mu_2}{2}$. The vote corresponds to the distance between the decision boundary and the gene's expression in the new sample. So each gene casts a weighted vote $V = w(g)(x_g - b_g)$.

The weights are defined so that positive votes count as votes for membership in class 1, negative ones for membership in class 2. The votes for all genes in S are combined; V_+ is the sum of positive votes and V_- is the sum of negative votes. The winner is simply the class receiving the larger total absolute vote.

Intuitively, if one class receives most of the votes, it seems reasonable to predict this majority. However, if the margin of victory is slight, a prediction for the majority class seems somewhat arbitrary and can only be done with low confidence. The authors therefore define the "prediction strength" to measure the margin of victory as: $PS = \frac{V_{winner} - V_{loser}}{V_{winner} + V_{loser}}$. Since V_{winner} is always greater then V_{loser}, PS varies between 0 and 1. Empirically, the researchers decided on a threshold of 0.3 i.e., if $PS > 0.3$, then x will be assigned to the winning class, otherwise x is left undetermined. Figure 7.3 shows a graphic presentation of the solution: Each gene g_i votes for either AML or ALL, depending on its expression level x_i in the sample. Summing separately votes for AML and ALL, shows that ALL wins.

Testing class predictors

There are two possibilities to test the validity of class predictors:

1. Leave-one-out Cross Validation (LOOCV) on initial data set:
   Withhold a sample, choose informative features, build a predictor based on the remaining samples, and predict the class of the withheld sample. The process is repeated for each sample and the cumulative error rate is calculated.

2. Validation on an independent set:
   Assess the accuracy on an independent set of samples.

Generally, the two above procedures are carried out together. Testing on an independent is better, but we are forced to do LOOCV, when samples are very scarce.

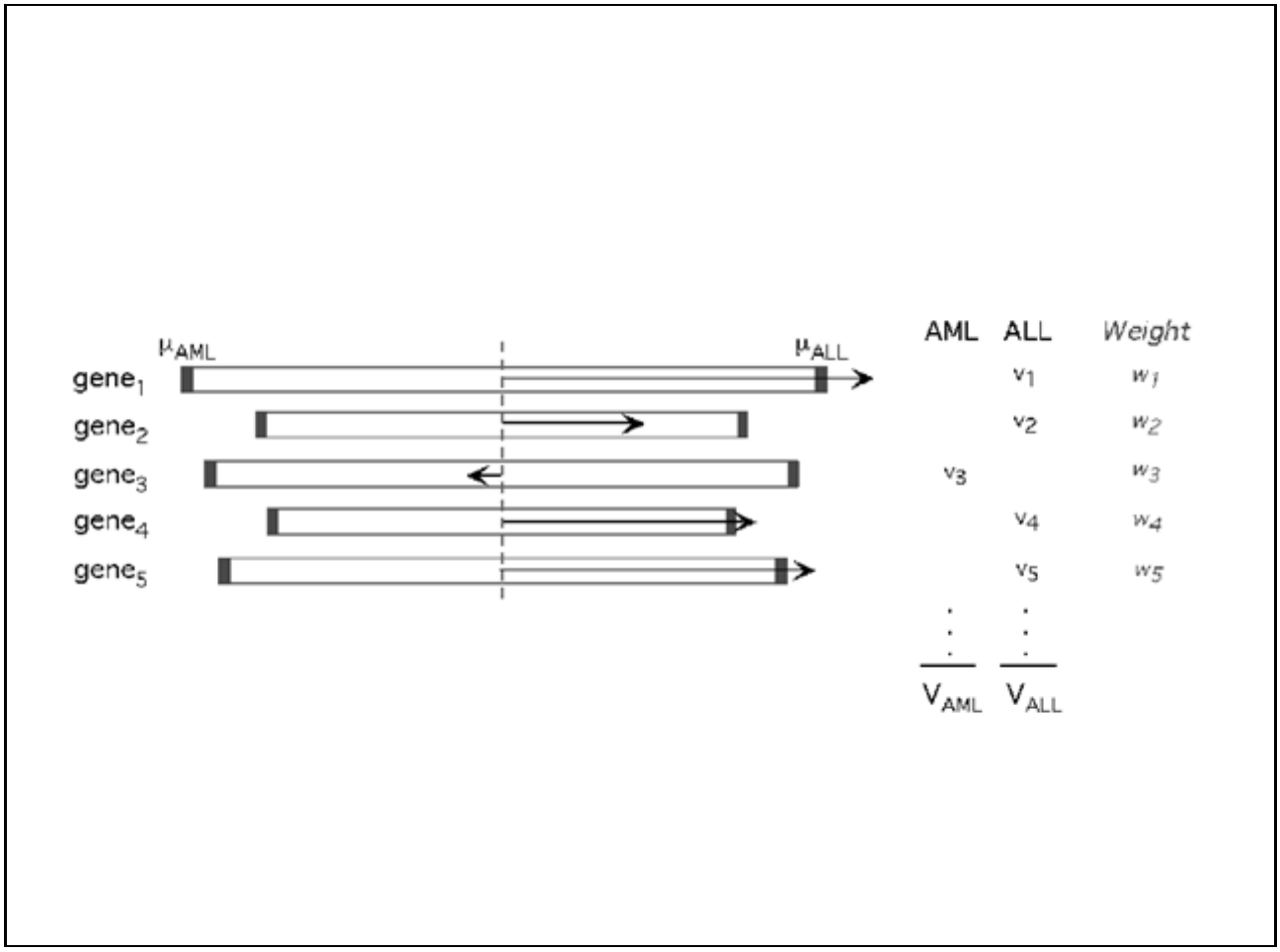

Figure 7.3: Class predictor.
The prediction of a new sample is based on "weighted votes of a set of informative genes. Each such gene g_i votes for either AML or ALL, depending on whether its expression level x_i is closer to μ_{ALL} or μ_{AML} The votes of each class are summed to obtain V_{AML} and V_{ALL}. The sample is assigned to the class with the higher total vote, provided that the prediction strength exceeds a predetermined threshold.

The authors applied this approach to the acute leukemia samples. The set of informative gene to be used as predictors was to chosen to be the 50 genes most closely correlated with AML-ALL distinction in the known samples. The following results were obtained:

1. On learning set:
   These predictors assigned 36 of 38 samples as either AML or ALL and the remaining two as uncertain($PS < 0.3$); all predictions agreed with the patients' clinical diagnosis.

2. On independent test set:
   These predictors assigned 29 of 34 samples with 100% accuracy.

The success is notable, as the test set included samples from peripheral blood,from childhood AML patients and from different reference laboratories that used different sample preparation protocols.
Overall, as shown in Figure 7.4, the prediction strength was quite high. The median in cross validation was $PS = 0.73$. On the test the median was PS = 0.77. The average prediction strength was lower for samples from one laboratory, that used very different protocol for sample preparation. This suggests that clinical implementation of such approach should include standardization of sample preparation.

The choice to use 50 informative genes in the predictor was somewhat arbitrary.In fact, the results were insensitive to the particular choice: predictors based on between 10 to 200 genes were all found to be 100% accurate, reflecting the strong correlation with the AML-ALL distinction.

Informative genes

The list of informative genes used in the AML versus ALL predictor is highly instructive. Figure 7.5 show the list of used informative genes.

Some genes, including CD11c, CD33 and MB-1 encode cell surface protein useful in distinguishing lymphoid from myeloid lineage cells. Others provide new markers of acute leukemia subtype, for example, the leptin receptor [4] , originally identified through its role in weight regulation, showed high relative expression in AML. These data suggest that genes useful for cancer class prediction may also provide insights into cancer pathogenesis and pharmacology.
The researchers explored the ability to predict response to chemotherapy among the 15 adult AML patients who had been treated and for whom long-term clinical follow-up was available. Eight patients failed to achieve remission after induction chemotherapy, while the remaining seven remained in remission for 46 to 84 months. Neighborhood analysis found no striking excess of genes correlated with the response to chemotherapy. Class predictors that used 10 to 50 genes were not highly accurate in cross-validation. Thus, no evidence

[4]leptin receptor is a molecule that identifies leptines

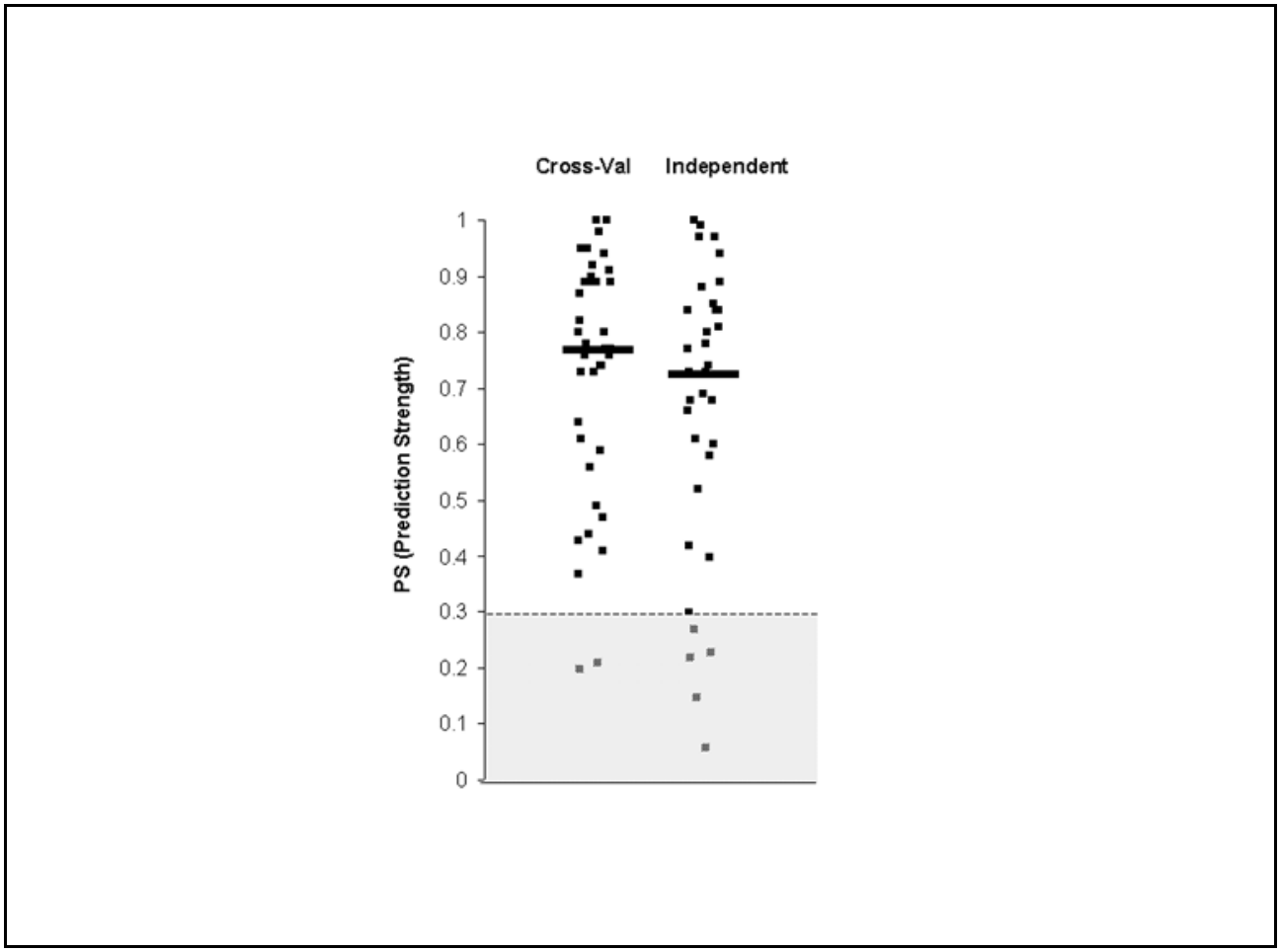

Figure 7.4: Prediction strengths.

The scatter-plots show the prediction strengths(PS) for the samples in cross-validation(left) and on the independent set(right). Median PS is denoted by a horizontal line. Predictors with PS less 0.3 are considered uncertain.

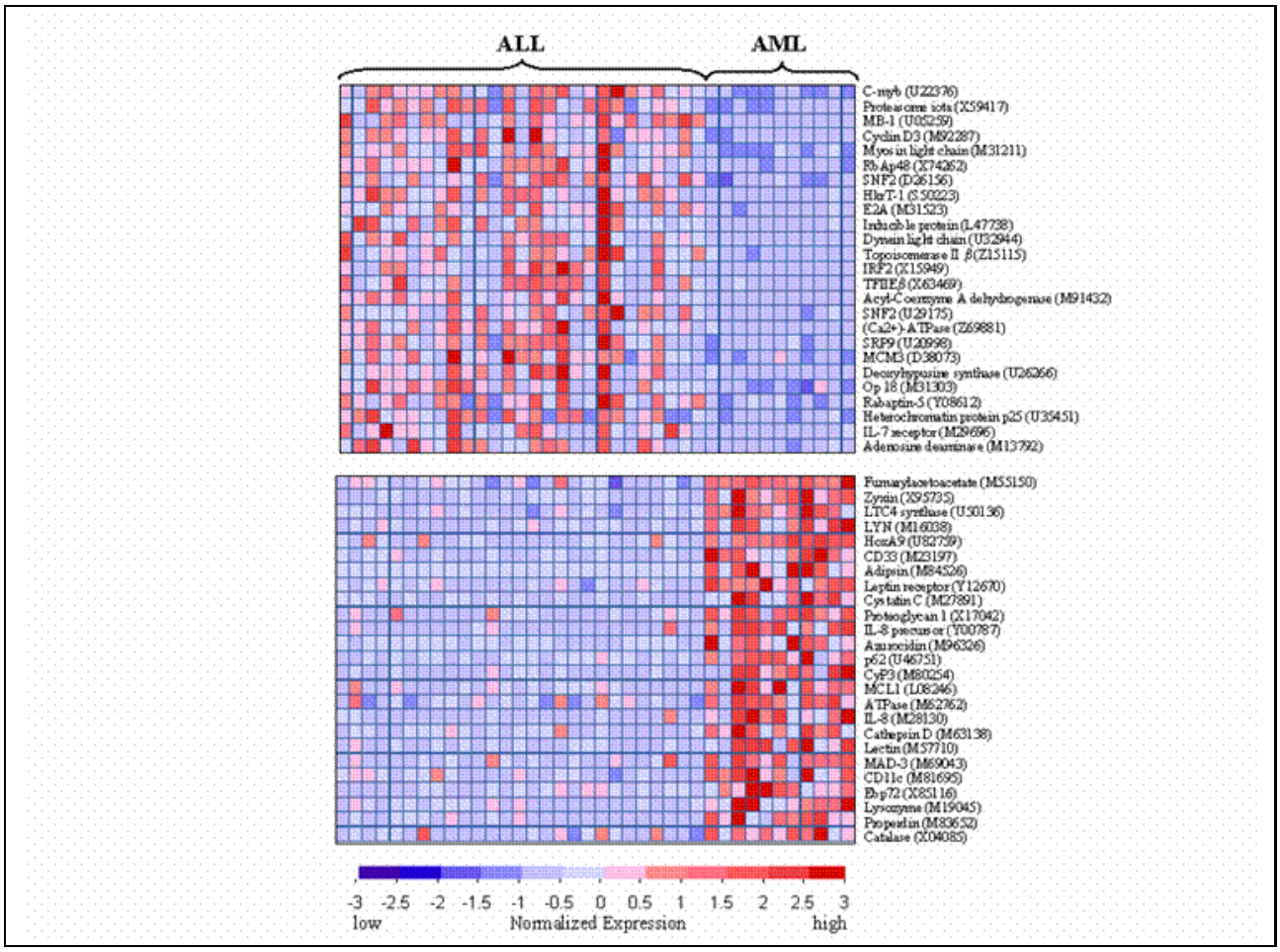

Figure 7.5: Genes distinguishing ALL from AML.
The 50 genes most highly correlated with the ALL-AML class are shown. Each row corresponds to a gene, with the columns corresponding to expression level in different samples. Expression levels for each gene are normalized across the samples such that the mean is 0 and the standard deviation(SD) is 1. Expressions level greater then the mean are shaded in red, and those below the mean are shaded in blue. The scale indicates SDs above or below the mean.

of strong multigene expression signature was correlated with clinical outcome, although this could reflect the relatively small sample size.

7.1.5 Class Discovery

Next the researchers turn to the question of class discovery. They explored whether cancer classes could be discovered automatically. For example, if the AML-ALL distinction was not already known, could we discover it simply on the basis of gene expression. So we use some clustering algorithm to cluster samples. One way to check the validity of the resulting class is through prediction experiment. The idea is to predict classes for new samples based on the classes' centroids and compare the results on random classes.

To cluster tumors Golub et al. used the SOM(self-organizing maps) algorithm. At first a 2-cluster SOM was applied to cluster the 38 initial leukemia samples on the basis of the expression patterns of all 6817 genes. The clusters were evaluated by comparing them to the known AML-ALL classes (Figure 7.6A). The results were as follows: cluster A1 contained mostly ALL samples (24 of 25) and cluster A2 contained mostly AML samples (10 of 13).

On basis of the clusters the authors constructed predictors to assign new samples as "type A1" or "type A2" and tested their validity. Predictors that used a wide range of different numbers of informative genes preformed well in cross-validation. The cross-validation not only showed high accuracy , but actually refined the SOM-defined classes: the subset of samples accurately classified in cross-validation were those perfectly subdivided by SOM into ALL and AML. Then the class predictor of A1-A2 distinction was tested on the independent test set. The prediction strengths were quite high: the median PS was 0.61 and 74% of samples were above threshold (Figure 7.6B), indicating that the structure seen in the initial set is also seen in the test set. In contrast, random clusters consistently yielded predictors with poor accuracy in cross-validation and low PS on the independent data set.

On basis of such analysis, the A1-A2 distinction can be seen to be meaningful, rather then simply a statistical artifact of the initial data set. The results thus show that the AML-ALL distinction could have been automatically discovered and confirmed without previous biological knowledge.

Finally, the researchers tried to extend the class discovery by searching for finer subclasses of the leukemias. 4-cluster SOM divided the samples into four classes, that largely corresponded to AML, T-lineage ALL, B-lineage ALL and B-lineage ALL, respectively (Figure 7.6C). When these classes were evaluated by constructing class predictors, all pairs could be distinguished one from another, with exception of B3 versus B4 (Figure 7.6D). The prediction tests thus confirmed the distinction corresponding to AML, B-ALL and T-ALL and suggested that it may be appropriate to merge classes B3 and B4, composed primarily of B-lineage ALL.

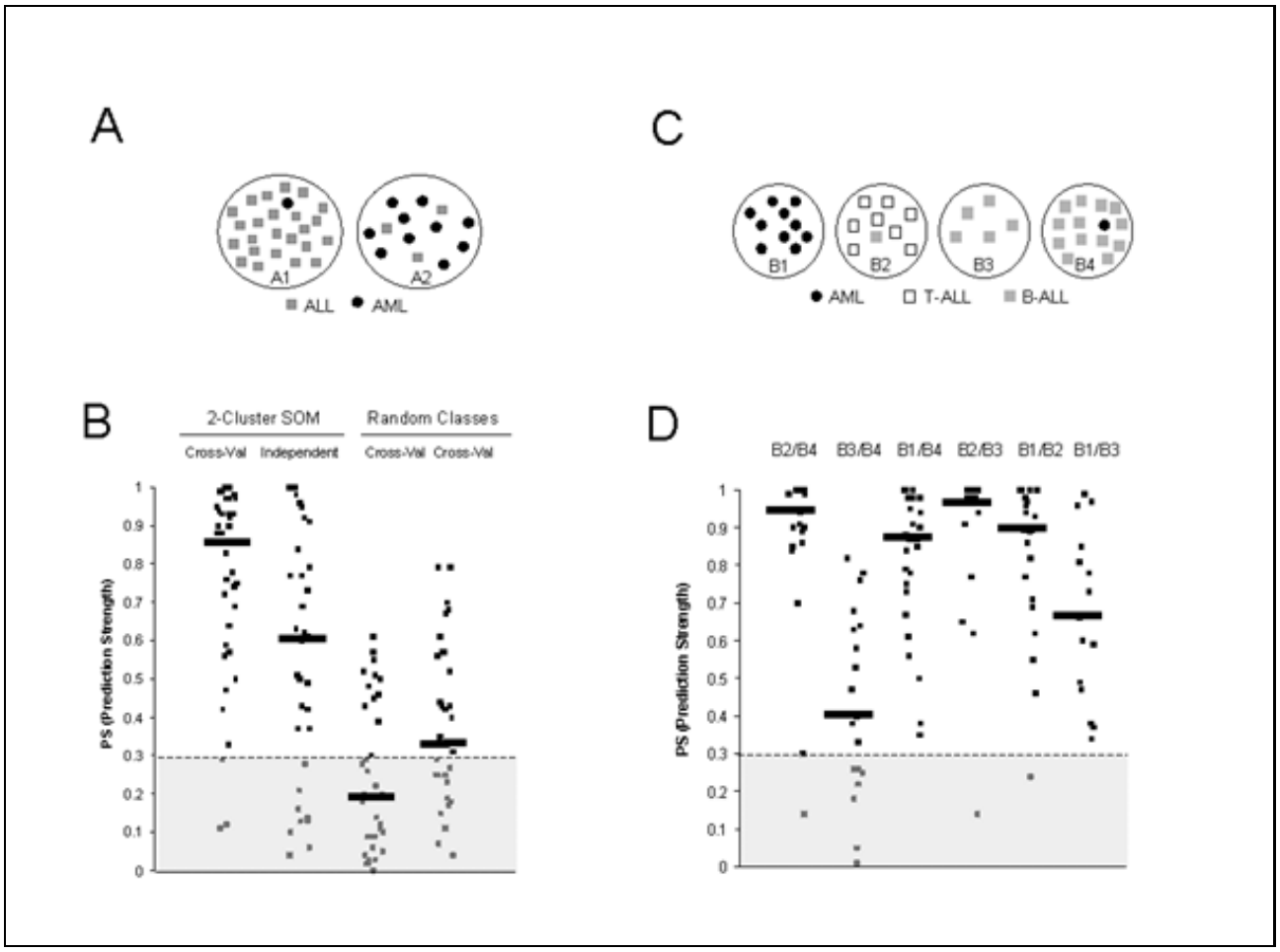

Figure 7.6: ALL-AML class discovery.
(A) Schematic presentation of 2-cluster SOM. A 2-cluster (2 by 1) SOM was generated from the 38 initial leukemia samples with a modification of the GENECLUSTER computer package. Cluster A1 contains the majority of ALL samples(grey squares) and cluster A2 contains the majority of AML samples(black circles).
(B) Prediction strength distribution. The scatter plots show the distribution of PS scores for class predictors.
(C) Schematic presentation of the 4-cluster SOM. ALL samples are shown as black circles, T - lineage ALL as open squares, and B-lineage ALL as grey squares.
(D) Prediction strength distribution for pairwise comparison among classes. Cross-validation studies show that the four classes could be distinguished with high prediction scores, with the exception of classes B3 and B4.

7.2 Support Vector Machines

7.2.1 Introduction

The theory of support vector machines (**SVM**), has its origins in the late seventies, in the work of Vapnik [12] on the theory of statistical learning. Lately it has been receiving increasing attention, and many applications as well as important theoretical results are based on this theory. The main area in which the methods described below had been used is in the field of pattern recognition . Just to state a few recent works:

- isolating handwritten digit recognition.
- object recognition.
- speaker identification.
- face detection in images.

In most of the cases SVM *generalization performance* either matches or is significantly better then that of competing methods. In the next section we will describe in detail the usage of SVM in the analysis of microarray gene expression data. This exposition is based mainly on [2, 3, 8].

7.2.2 General Motivation

One of the basic notions in the theory of SVM is the *capacity* of the machine, i.e. the ability of the machine to learn any training set of data without error. A machine with too much capacity is like a botanist with a photographic memory - who, when presented with a new tree, concludes that it is not a tree since it has a different number of leaves from anything she has seen before. On the other hand, a machine with too little capacity is like the botanist lazy brother - who declares that if it is green it is a tree.Neither can generalize well. Roughly speaking, for a given learning task with a given amount of training data, the best generalization performance will be achieved if the right balance is found, between the accuracy attained on the particular training set and the capacity of the machine.

7.2.3 General Mathematical Setting

Suppose we are given l observations .Each observation consists of a pair - a vector $x_i \in \mathbf{R}^n$, $1 \leq i \leq l$, and an associated label y_i, given to us by a trusted source. In the tree recognition problem, the vector x_i might be some representation of an image, e.g., by its pixel values, and y_i would be $+1$ if the image contains a tree, and 0 otherwise. Furthermore, it is assumed

that there exists some unknown probability distribution $P(x, y)$, from which the above observations are assumed to be independently drawn and identically distributed.

Now suppose we have a deterministic machine whose task is to learn the mapping $x_i \rightarrow y_i$. The machine is actually defined by a set of possible mappings $x_i \rightarrow f(x, \alpha)$, where the function f depends on a parameter α, i.e, a choice of the parameter α specifies one particular machine from the set $f(x, \alpha)$. For a particular choice of the parameter, the machine will be called a *trained machine*.

For a given trained machine one can measure its performance by its *expected risk*

$$R(\alpha) = \int |y - f(x, \alpha)| dP(x, y)$$

where the integration is over the distribution space. The ideal goal is to find a trained machine, or in other words a parameter α for the function $f(x, \alpha)$ that minimizes the expected risk.

The problem in the usage of this estimation is that usually the data distribution $P(x, y)$ is unknown. Therefore we define the *empirical risk* $R_{emp}(\alpha)$ as the measured mean error rate on a finite training set:

$$R_{emp}(\alpha) = \frac{1}{l} \sum_{1}^{l} |y_i - f(x_i, \alpha)|$$

The key idea is to minimize the empirical risk, R_{emp}, instead of the expected risk. The quantity $|y_i - f(x_i, \alpha)|$ is called the loss. Now choose some η, $0 \leq \eta \leq 1$. Vapnik has shown that the following upper bound holds with probability $1 - \eta$:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\ln(2l/h) + 1) - \ln(\eta/4)}{l}} \tag{7.1}$$

h is a non negative integer called the Vapnik Chervonenkis (VC) dimension. It is a measure of the notion of the capacity of the machine discussed above. The right hand side of the above equation is called the risk bound. This bound is the basis of a technique called Structural Risk Minimization(**SRM**), that will be discussed below.

7.2.4 The VC dimension

The VC dimension is a property of a set of functions $\{f(\alpha)\}$. It can be defined in a more general manner, but we will assume families of functions that obtain binary values.

If a given set of l points can be labelled in all possible 2^l ways, and for each labelling, a member of the set $\{f(\alpha)\}$ can be found which correctly assigns those labels, we say that the set of points is **shattered** by the set of functions. The VC dimension for the set of functions $\{f(\alpha)\}$ is defined as the maximum size of a set of points that can be shattered by $\{f(\alpha)\}$.

In other words, if the VC dimension is h there exists at least one set of h points that can be shattered.

Example: Shattering points with oriented lines in $\mathbf{R}^2$.
Suppose that the data are points in $\mathbf{R^2}$, and the set $\{f(\alpha)\}$ consists of oriented straight lines, such that for a given line all points on one side are assigned the value 1, and all points on the other are assigned the value 0.
It is possible to find a set of three points that can be shattered by the oriented lines (see 7.7), but it is not possible to shatter a set of 4 points with the set of oriented lines. Thus, the VC dimension of the set of oriented lines in $\mathbf{R}^2$ is 3. In the more general case the VC dimension of a set of oriented hyperplanes in $\mathbf{R}^n$ is $n + 1$.

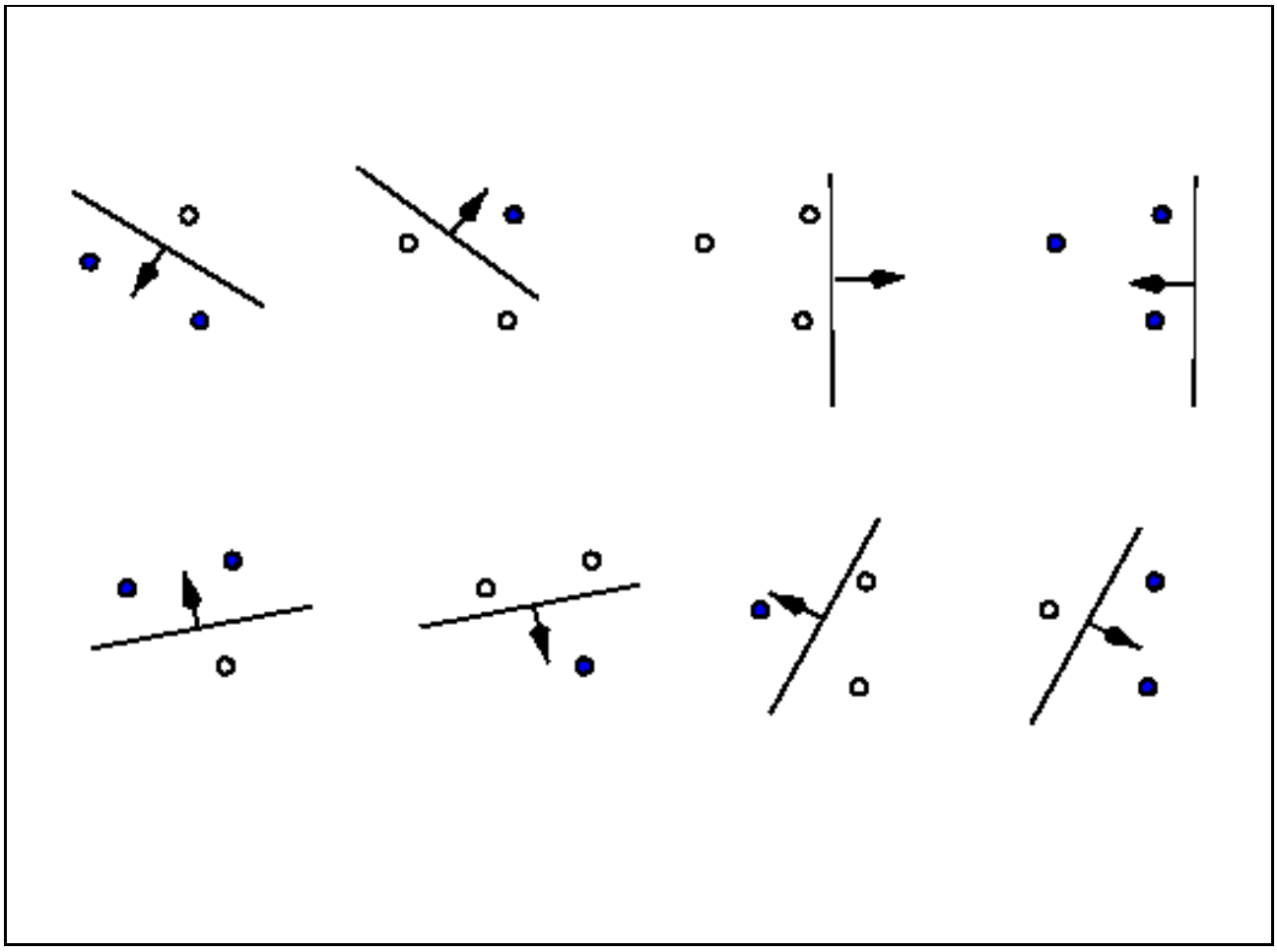

Figure 7.7: Three points in the plane shattered by oriented lines.

7.2.5 Structural Risk Minimization(SRM)

Returning back to equation 7.1

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\ln(2l/h) + 1) - \ln(\eta/4)}{l}}$$

We note three points about this bound:

1. It is independent of the distribution $P(x, y)$.
2. It is usually impossible to compute the left hand side of the equation.
3. If we know h it is easy to compute the right hand side.

Thus, given several different learning machines(i.e., different families of functions $\{f(\alpha)\}$) and choosing a fixed small η, by taking a machine which minimizes the right hand side, we are choosing the machine which gives the lowest upper bound on the expected risk $R(\alpha)$. The right hand side of the above bound depends on the chosen class of functions - via the VC dimension, and on the other side on a particular function chosen by the training procedure. We would like to find that subset of the chosen set of functions, such that the risk bound for that subset is minimized. It is not possible to arrange things so that the VC dimension varies smoothly - since it is an integer.Instead we introduce a "structure" by dividing the entire class of functions into nested subsets , for each subset we must be able to compute, or to bound h itself. The way we introduce this structure defines the specific heuristic for SRM.

SRM then consists of finding that subset of functions which minimizes the bound on the actual risk. This can be done by simply training a series of machines, one for each subset, where for a given subset the goal of training is simply to minimize the empirical risk, $R_{emp}(\alpha)$. One then takes that trained machine in the series, whose sum of empirical risk and VC confidence (the term depending on the VC dimension in 7.1) is minimal.

7.2.6 Support Vector Machines

We will start with the simplest case, linear machines trained on separable data. Given a training set $x_i, y_i, x_i \in \mathbf{R}^n, y_i \in \{-1, 1\}$, we assume that the data is linearly separable, i.e., there exists a separating hyperplane which separates the positive examples ($y_i = 1$) from the negative ones ($y_i = -1$). The points x which lie on the hyperplane satisfy $w \cdot x + b = 0$, where w is a normal to the hyperplane and $\frac{|b|}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin. Let $d_+(d_-)$ be the shortest distance from the separating hyperplane to the closest positive (negative) example. Define the *margin* of a separating hyperplane to be $d_+ + d_-$. For the linearly separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin.

Thus the goal is to find the optimal linear classifier (a hyperplane), such that it classifies every training example correctly, and maximizes the classification margin. The above description can be formulated in the following way: Suppose that all the training data satisfy the following constraints:

$$x_i \cdot w + b \geq +1, y_i = +1 \tag{7.2}$$

$$x_i \cdot w + b \leq -1, y_i = -1 \tag{7.3}$$

Now consider the points for which the equality in (7.2) holds. These points lie on the hyperplane $H_1 : x_i \cdot w + b = 1$. Similarly, the points for which the equality in (7.3) holds lie on the hyperplane $H_2 : x_i \cdot w + b = -1$. In this case $d_+ = d_- = \frac{1}{\|w\|}$, therefore the margin is $\frac{2}{\|w\|}$. Note that H_1 and H_2 are parallel (they have the same normal) and that no training points fall between them. Thus we can find the pair of hyperplanes which gives the maximum margin by minimizing $\|w\|^2$, subject to the above constraints.
To summarize our discussion so far, our goal is to find an optimal linear classifier (a hyperplane) such that:

1. It classifies every training example correctly.

2. It maximizes the classification margin.

$$minimize \qquad \|w\|^2 \tag{7.4}$$

$$s.t. \quad y_i(w \cdot x_i + b) \geq 1, i = 1 \ldots l \tag{7.5}$$

Lagrange formulation of the problem

We will now switch to a Lagrangian formulation of the problem. There are two reasons for doing this. The first is that the constraints (7.2),(7.3) will be replaced by constraints on the

Lagrange multipliers themselves, which will be much easier to handle. The second is that in this reformulation of the problem, the training data will only appear (in the actual training and test algorithms) in the form of dot products between vectors. This is a crucial property which will allow us to generalize the procedure to the nonlinear case.

Thus,utilizing the theory of Lagrange multipliers, we introduce positive Lagrange multipliers $\alpha_i, i = 1, \ldots, l$ one for each of the inequality constraints . We form the Lagrangian:

$$L_P = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^{l} \alpha_i$$

We have to minimize L_P with respect to w, b and simultaneously require that the derivatives of L_P with respect to all the α_i vanish. This is a convex quadratic problem, which has a dual formulation- maximize L_P, subject to the constraint that the gradients of L_P with respect to w and b vanish, and subject to the constraints that $\alpha_i \geq 0$. This gives:

$$maximize \qquad L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{7.6}$$

$$s.t. \qquad \sum_{i=1}^{l} \alpha_i y_i = 0, \alpha_i \geq 0 \tag{7.7}$$

Support vector training (for the separable, linear case) therefore amounts to maximizing L_D with respect to the α_i, subject to above constraints and non-negativity of the α_i, with solution given by $w = \sum_{i=1}^{l} \alpha_i y_i x_i$. Notice that there is a Lagrange multiplier α_i for every training point. In the solution, those points for which $\alpha_i > 0$ are called *support vectors*, and lie on one of the hyperplanes H_1 or H_2. All other training points have $\alpha_i = 0$ and lie to the side of H_1 or H_2 such strict inequality holds. For these machines, the support vectors are the critical elements of the training set. They lie closest to the decision boundary. If all other training points were removed (or moved around, but so as not to cross H_1 or H_2), and training was repeated, the same separating hyperplane would be found.

7.2.7 Linear Non Separable case

When applied to non-separable data, the above algorithm will find no feasible solution. This will be evidenced by the objective function (i.e. the dual Lagrangian) growing arbitrarily large. So how can we extend these ideas to handle non-separable data? We would like to relax the constraints:

$$x_i \cdot w + b \geq +1, y_i = +1 \tag{7.8}$$

$$x_i \cdot w + b \leq -1, y_i = -1 \tag{7.9}$$

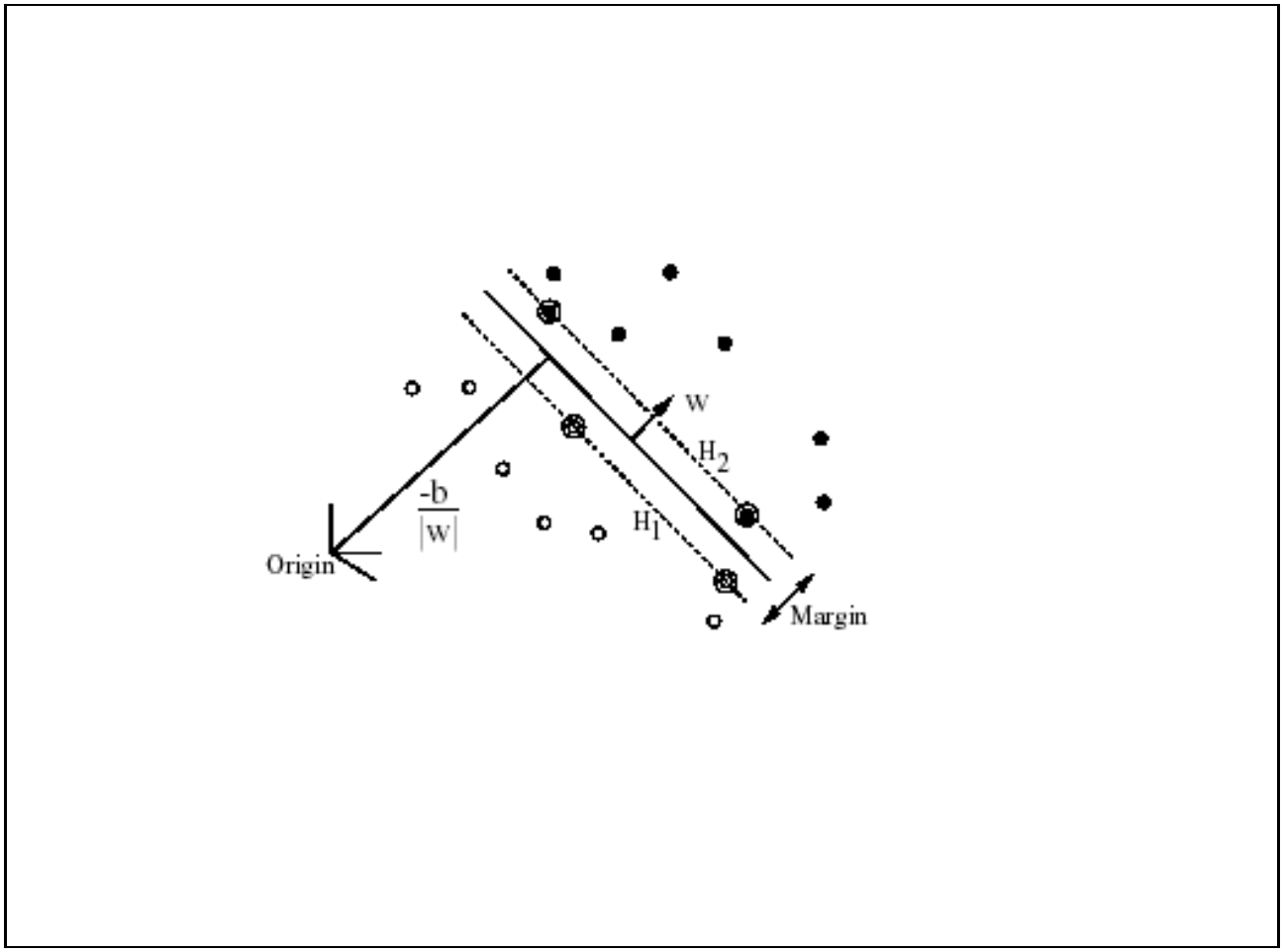

Figure 7.8: **Linear Separating hyperplanes for the separable case. The support vectors are circled**

but only when necessary, that is, we would like to introduce a further cost (i.e. an increase in the primal objective function) for doing so. This can be done by introducing positive slack variables $\xi_i, i = 1, \ldots, l$ in the constraints (Cortes and Vapnik, 1995), which then become:

$$x_i \cdot w + b \geq +1 - \xi_i, y_i = +1 \tag{7.10}$$

$$x_i \cdot w + b \leq -1 + \xi_i, y_i = -1 \tag{7.11}$$

Thus, for an error to occur, the corresponding ξ_i must exceed unity, so $\sum_i \xi_i$ is an upper bound on the number of training errors. Hence a natural way to assign an extra cost for errors is to change the objective function to be minimized from $\frac{1}{2}\|w\|^2$ to $\frac{1}{2}\|w\|^2 + C\sum_i^l \xi_i$, where C is a parameter to be chosen by the user. A larger C corresponding to assigning a higher penalty to errors. This is again a convex quadratic programming problem. Thus the dual formulation becomes:

$$maximize \qquad L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{7.12}$$

$$s.t \qquad 0 \leq \alpha_i \leq C, \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{7.13}$$

The only difference from the optimal hyperplane case is that the α_i have an upper bound of C. The parameter C controls the range of the α_i and avoids over emphasizing some examples. C is called the complementary slackness when C tends to infinity we return to the separable case.

7.2.8 Non Linear case

When coming to generalize the above ideas to the non-linear case - the idea is to use the same techniques as above for linear machines . Since finding a linear machine is not possible in the original space of the training set, we first map the training set to another Euclidean space with higher dimension(even of infinite dimension) ,this higher-dimensional space is called the *feature space*, as opposed to the *input space* occupied by the training set. With an appropriately chosen feature space of sufficient dimensionality, any consistent training set can be made separable. However, translating the training set into a higher-dimensional space incurs both computational and learning-theoretic costs.

The key idea is to notice that in the dual representation of the above problems - the training data appeared only in the form of dot products. Now suppose we first map the data to some other space H, using a mapping $\Phi : R^d \rightarrow H$. Then the training algorithm would only depend on the data through dot products in H, i.e. on functions of the form $\Phi(x_i) \cdot \Phi(x_j)$. All we need in order to perform the training algorithm in H, is a function that satisfy $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$. These type of functions are called *kernel functions*.

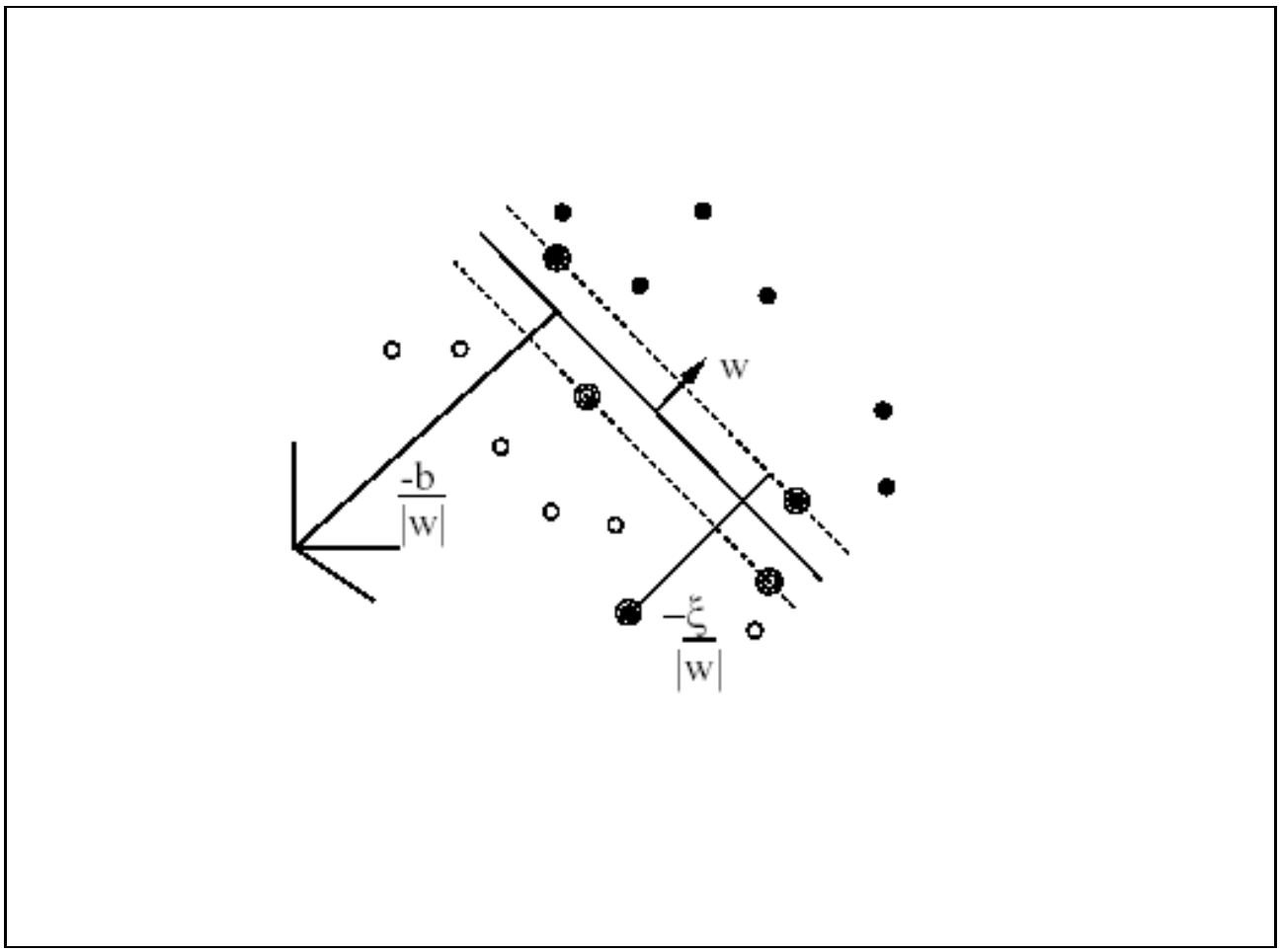

Figure 7.9: **Linear Separating hyperplanes for the non separable case.**

The kernel function is used in the higher dimension space as a dot product. Using the kernel function the training set is mapped to a higher dimension space, with more degrees of freedom which results with a new set of separable data. Examples of kernel functions are:

- $K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$ - radial basis kernel.
- $K(x_i, x_j) = (x_i \cdot x_j + 1)^k$ - polynomial kernel.

7.2.9 Training SVM , Problems and Heuristics

There are several heuristics for solving the problem of training an SVM. As we have seen above the training of an SVM is a quadratic optimization problem. Zoutendijk's Method [6] aims to solve a linear programming problem: Find the direction that minimizes the objective function, and make the largest move along this direction ,while still satisfying all the constraints. Another Problem in the training of SVM is that in the general case the computation of the kernel function $K(x_i, x_j)$ for each pair of elements might be computationally expensive. The solution is to use only part of the training data, using the assumption that only part of the data contributes to the decision boundary. Then we define a strategy to increase the objective function, while updating the set of data points contributing to the formation of the decision boundary.

7.2.10 Heuristic for training SVM with large data set

1. Divide the training examples into two sets A,B.
2. Use the set A of training examples to find out the optimal decision boundary.
3. Find an example x_i in A with no contribution to the decision boundary, $\alpha_i = 0$.
4. Find another example, x_m in B that can not be classified correctly by the current decision boundary.
5. Remove x_i from A and add x_m to A.
6. Repeat steps 2-5 till some stopping criterion is satisfied.

7.3 Knowledge-based analysis of microarray gene expression data by using SVM

7.3.1 Introduction

After laying the theoretic foundations for SVM, we explore its applications in the analysis of microarray gene expression data. The work described here is due to [1]. In this work the authors applied SVM for functional annotation of genes. The idea is to begin with a set of genes that have a common function: for example, genes coding for ribosomal proteins or genes coding for components of the proteasome. In addition, a separate set of genes that are known not to be members of the functional class is specified. These two sets of genes are combined to form a set of training examples in which the genes are labelled positively if they are in the functional class and are labelled negatively if they are known not to be in the functional class. A set of training examples can easily be assembled from literature and database sources. Using this training set, an SVM would learn to discriminate between the members and non-members of a given functional class based on expression data. Having learned the expression features of the class, the SVM could recognize new genes as members or as non-members of the class based on their expression data.

We describe here the use of SVM to classify genes based on gene expression. Analyzing expression data from 2,467 genes from the budding yeast Saccharomyces cerevisiae measured in 79 different DNA microarray hybridization experiments [4]. From these data, the authors learn to recognize five functional classes from the Munich Information Center for Protein Sequences Yeast Genome Database (MYGD) (http://www.mips.biochem.mpg.de/proj/yeast). In addition to SVM classification, the authors subject the data to analyses by four competing machine learning techniques, including Fisher's linear discriminant, Parzen windows, and two decision tree learners. The SVM method out-performed all other methods investigated here.

The work described here experimented with several kernel functions:

$$K(X,Y) = (X \cdot Y + 1)^d, d = 1, 2, 3$$

and:

$$K(X,Y) = e^{-\|X-Y\|^2/2\sigma^2}$$

7.3.2 Balancing positive and negative examples

The gene functional classes examined here contain very few members relative to the total number of genes in the data set. This leads to an imbalance in the number of positive and negative training examples that, in combination with noise in the data, is likely to

cause the SVM to make incorrect classifications. When the magnitude of the noise in the negative examples outweighs the total number of positive examples, the optimal hyperplane located by the SVM will be uninformative, classifying all members of the training set as negative examples. The authors overcame this problem by modifying the matrix of kernel values computed during SVM optimization. Let $X_1, \ldots, X_n$ be the genes in the training set, and let $\mathbf{K}$ be the matrix defined by the kernel function K on this training set i.e., $K_{ij} = K(X_i, X_j)$.We define X_i to be the logarithm of the ratio of expression level E_i for gene $\mathbf{X}$ in experiment i to the expression level R_i of gene $\mathbf{X}$ in the reference state, normalized so that the expression vector $X = (X_1, \ldots, X_{79})$ has Euclidean length 1:

$$X_i = \frac{\log(E_i/R_i)}{\sqrt{\sum_{j=1}^{79} \log^2 E_j/R_j}}$$

By adding to the diagonal of the kernel matrix a constant whose magnitude depends on the class of the training example, one can control the fraction of misclassified points in the two classes. This technique ensures that the positive points are not regarded as noisy labels. For positive examples, the diagonal element is modified by $K_{ij} := K_{ij} + \lambda n^+/N$, where n^+ is the number of positive training examples, N is the total number of training examples, and λ is scale factor. A similar formula is used for the negative examples, with n^+ replaced by n^-. In the experiments reported here, the scale factor is set to 0.1.

7.3.3 Experimental design

Using the class definitions made by the MYGD, the authors trained SVM to recognize six functional classes: tricarboxylic acid (TCA) cycle, respiration, cytoplasmic ribosomes, proteasome, histones, and helix-turn-helix proteins. The MYGD class definitions come from biochemical and genetic studies of gene function whereas the microarray expression data measures mRNA levels of genes. Many classes in MYGD, especially structural classes such as protein kinases, will be unlearnable from expression data by any classifier. The first five classes were selected because they represent categories of genes that are expected, on biological grounds, to exhibit similar expression profiles. The sixth class, the helix-turn-helix proteins, is included as a control group.

The performance of the SVM classifiers was compared with that of four standard machine learning algorithms: Parzen windows, Fisher's linear discriminant, and two decision tree learners (C4.5 and MOC1). Descriptions of these algorithms can be found at http://www.cse.ucsc.edu/research/compbio/genex. Performance was tested by using a three-way cross-validation. The gene expression vectors were randomly divided into three groups. Classifiers were trained by using two-thirds of the data and were tested on the remaining third. This procedure was then repeated two more times, each time using a different third of the genes as test genes.

To judge overall performance, the authors defined the cost of using the method M as $C(M) = fp(M) + 2fn(M)$, where $fp(M)$ is the number of false positives for method M, and $fn(M)$ is the number of false negatives for method M. The false negatives are weighted more heavily than the false positives because, for these data, the number of positive examples is small compared with the number of negative ones. The cost for each method is compared with the cost $C(N)$ for using the null learning procedure, which classifies all test examples as negative. We define the cost savings of using the learning procedure M as $S(M) = C(N) - C(M)$. Experiments predicting functions of unknown genes were performed by first training SVM classifiers on the 2,467 annotated genes for the five learnable classes. For each class, the remaining 3,754 genes were then classified by the SVM. The results are summarized in Table 7.1. The methods tested are the SVM using the scaled dot product kernel raised to the first, second, and third power, the radial basis function SVM, Parzen windows, Fisher's Linear Discriminant, and the two decision tree learners, C4.5 and MOC1. For every class (except the helix-turn-helix class), the best-performing method is a SVM using the radial basis or a higher-dimensional dot product kernel. Other cost functions, with different relative weights of the false positive and false negative rates, yield similar rankings of performance. The results shown in Table 7.1 for higher-order SVM are considerably better than the corresponding error rates for clusters derived in an unsupervised fashion. For example, using hierarchical clustering, the histone cluster only identified 8 of the 11 histones, and the ribosome cluster only found 112 of the 121 genes and included 14 others that were not ribosomal genes. Across all five functional classes (excluding helix-turn-helix) and all the 20 experiments, 25 genes are misclassified in at least 19 of the 20 experiments. Disagreements between the SVM and MYGD(the original annotations) in the form of false negatives may occur for a number of reasons. First, genes that are classified in MYGD primarily by structure (e.g., protein kinases) may have very different expression patterns. Second, genes that are regulated at the translational level or protein level, rather than at the transcriptional level as measured by the microarray experiments, may not be correctly classified by expression data alone. False positives and false negatives represent cases in which further biological experimentation may be fruitful.

Class	Method	FP	FN	TP	TN	SM
TCA	D-p 1 SVM	18	5	12	2,432	6
	D-p 2 SVM	7	9	8	2,443	9
	D-p 3 SVM	4	9	8	2,446	12
	Radial SVM	5	9	8	2,445	11
	Parzen	4	12	5	2,446	6
	FLD	9	10	7	2,441	5
	C4.5	7	17	0	2,443	27
	MOC1	3	16	1	2,446	21
Resp	D-p 1 SVM	15	7	23	2,422	31
	D-p 2 SVM	7	7	23	2,430	39
	D-p 3 SVM	6	8	22	2,431	38
	Radial SVM	5	11	19	2,432	33
	Parzen	22	10	20	2,415	18
	FLD	10	10	20	2,427	30
	C4.5	18	17	13	2,419	8
	MOC1	12	26	4	2,425	24
Ribo	D-p 1 SVM	14	2	119	2,332	224
	D-p 2 SVM	9	2	119	2,337	229
	D-p 3 SVM	7	3	118	2,339	229
	Radial SVM	6	5	116	2,340	226
	Parzen	6	8	113	2,340	220
	FLD	15	5	116	2,331	217
	C4.5	31	21	100	2,315	169
	MOC1	26	26	95	2,320	164
Prot	D-p 1 SVM	21	7	28	2,411	35
	D-p 2 SVM	6	8	27	2,426	48
	D-p 3 SVM	3	8	27	2,429	51
	Radial SVM	2	8	27	2,430	52
	Parzen	21	5	30	2,411	39
	FLD	7	12	23	2,425	39
	C4.5	17	10	25	2,415	33
	MOC1	10	17	18	2,422	26
Hist	D-p 1 SVM	0	2	9	2,456	18
	D-p 2 SVM	0	2	9	2,456	18
	D-p 3 SVM	0	2	9	2,456	18
	Radial SVM	0	2	9	2,456	18
	Parzen	2	3	8	2,454	14
	FLD	0	3	8	2,456	16
	C4.5	2	2	9	2,454	16
	MOC1	2	5	6	2,454	10
HTH	D-p 1 SVM	60	14	2	2,391	256
	D-p 2 SVM	3	16	0	2,448	23
	D-p 3 SVM	1	16	0	2,450	21
	Radial SVM	0	16	0	2,451	0
	Parzen	14	16	0	2,437	214
	FLD	14	16	0	2,437	214
	C4.5	2	16	0	2,449	22

Table 7.1: Performance of different classifiers. The last five columns are the false positive, false negative, true positive, and true negative rates summed over three cross-validation splits, followed by the total cost savings [S(M)].

7.4 Classification into multiple classes

7.4.1 Introduction

We now describe the work of Ramaswamy et al. [9] that deals with multiclass cancer diagnosis.

To establish analytic methods capable of solving complex, multiclass gene expression-based classification problems the researchers created a gene expression database, containing the expression profiles of 218 tumor samples, representing 14 common human cancer classes, and 90 normal tissue samples. Hybridization targets were prepared with RNA from whole tumors. The targets were hybridized sequentially to oligonucleotide arrays, containing a total of 16063 probe sets. Expression values for each gene were calculated by using Affymetrix GENECHIP analysis software. Two fundamentally different approaches to data analysis were explored: clustering (unsupervised learning) and classification (supervised learning).

7.4.2 Clustering

As we already know, this approach allows the dominant structure in a dataset to dictate the separation of samples into clusters based on overall similarity in expression, without prior knowledge of sample identity.

Of 16063 expression values considered, 11322 passed some variation filter(see [9] for details) and were used for clustering. The dataset was normalized by standardizing each row (gene) to mean = 0 and variance = 1. Average-linkage hierarchical clustering was performed by using CLUSTER and TREEVIEW software. Self-organizing map analysis was performed by using GENECLUSTER analysis package. Figure 7.10 shows the results of both hierarchical and self-organizing map clustering of this data set.

7.4.3 Classification

To make multiclass distinctions the researchers devised an analytic scheme, depicted in Figure 7.11.

One vs. All (OVA) SVM scheme

For each known type a binary classifier is built. The classifier uses the SVM algorithm to define a hyperplane that best separates training samples into two classes: samples from this class vs. all other samples. An unknown test sample's position relative to the hyperplane determines its class, and the confidence of each SVM prediction is based on the distance of the test sample from the hyperplane; that distance is calculated in 16063-dimensional gene space, corresponding to the total number of expression values considered.

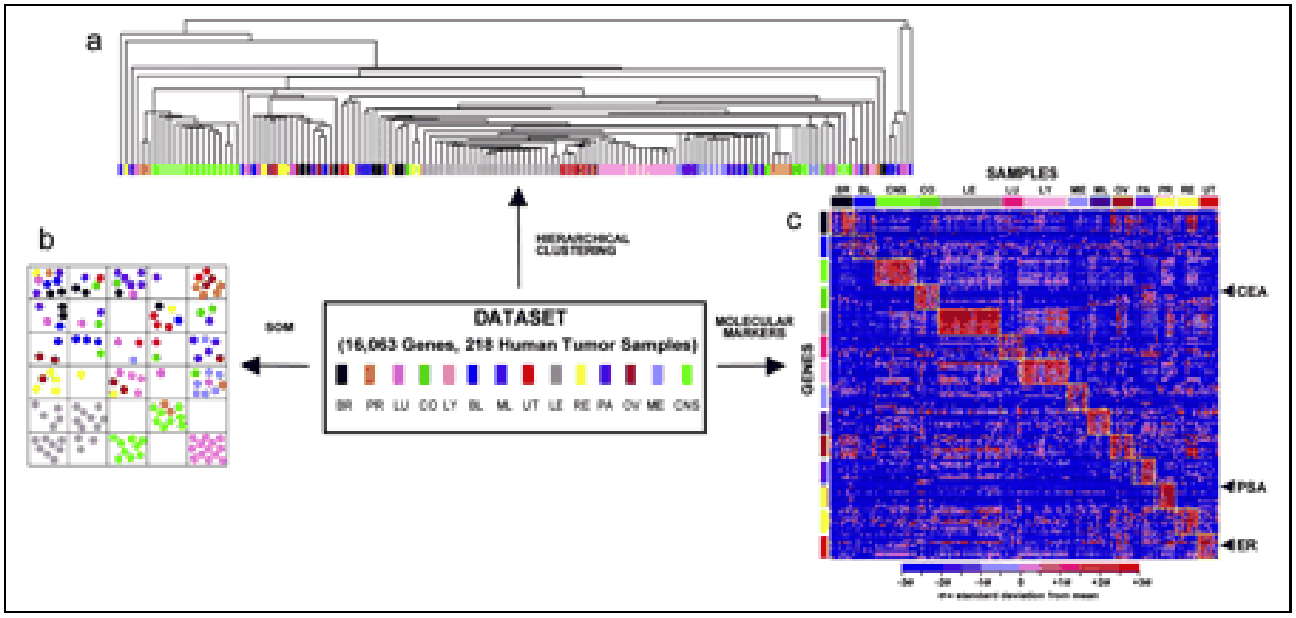

Figure 7.10: Clustering of tumor gene expression data and identification of tumor-specific molecular markers.
Hierarchical clustering (a) and a 5 x 5 self-organizing map (SOM) (b) were used to cluster 144 tumors spanning 14 tumor classes according to their gene expression patterns. (c) Gene expression values for class-specific OVA markers are shown. Columns represent 190 primary human tumor samples ordered by class. Rows represent 10 genes most highly correlated with each OVA distinction. Red indicates high relative level of expression, and blue represents low relative level of expression. The known cancer markers prostate-specific antigen (PSA), carcinoembryonic antigen (CEA), and estrogen receptor (ER) are identified. BR, breast adenocarcinoma; PR, prostate adenocarcinoma; LU, lung adenocarcinoma; CR, colorectal adenocarcinoma; LY, lymphoma; BL, bladder transitional cell carcinoma; ML, melanoma; UT, uterine adenocarcinoma; LE, leukemia; RE, renal cell carcinoma; PA, pancreatic adenocarcinoma; OV, ovarian adenocarcinoma; ME, pleural mesothelioma; CNS, central nervous system.

Recursive feature elimination

Given microarray data with n genes per sample, each OVA SVM classifier outputs a hyperplane w, that can be thought of as a vector with n elements each corresponding to the expression of a particular gene. Assuming the expression values of each gene have similar ranges, the absolute magnitude of each element in w determines its importance in classifying the sample, since the class label is $sign[f(x)]$. Each OVA SVM classifier is first trained with all genes, 10 % of the genes with least $|w_i|$ are removed , and each classifier is retrained with the smaller gene set. This procedure is repeated iteratively to study prediction accuracy as a function of gene number.

Prediction

Each test sample is presented sequentially to the 14 OVA classifiers, each of which either claims or rejects that sample as belonging to a single class with an associated confidence. Finally, each test sample is assigned to the class with the highest OVA classifier confidence. If the confidences were low no prediction is made.

Testing and Results

As we mentioned above, the number of genes contributing to the high accuracy of the SVM OVA classifier was also investigated. The SVM algorithm considers all 16063 input genes and naturally utilizes all genes that contain information for each OVA distinction. Genes are assigned weights based on their relative contribution to the determination of each hyperplane, and genes that do not contribute to a distinction are weighted zero. Virtually all genes on the array were assigned weakly positive and negative weights in each OVA classifier, indicating that thousands of genes potentially carry information relevant for the 14 OVA class distinctions. To determine whether the inclusion of this large number of genes was actually required for the observed high-accuracy predictions, the authors examined the relationship between classification accuracy and gene number by using recursive feature elimination. As shown in Figure 7.13, maximal classification accuracy is achieved when the predictor utilizes all genes for each OVA distinction. Nevertheless, significant prediction can still be achieved by using smaller gene numbers.

The accuracy of the multiclass SVM-based classifier in cancer diagnosis was first evaluated by leave-one-out cross-validation in a set of 144 training samples. As shown in Figure 7.12, the majority (80%) of the 144 calls was high confidence (defined as confidence > 0) and these had an accuracy of 90%, using the patient's clinical diagnosis as the "gold standard". The remaining 20% of the tumors had low confidence calls (confidence 0), and these predictions had an accuracy of 28%. Overall, the multiclass prediction corresponded to the correct assignment for 78% of the tumors, far exceeding the accuracy of random classification(9%).

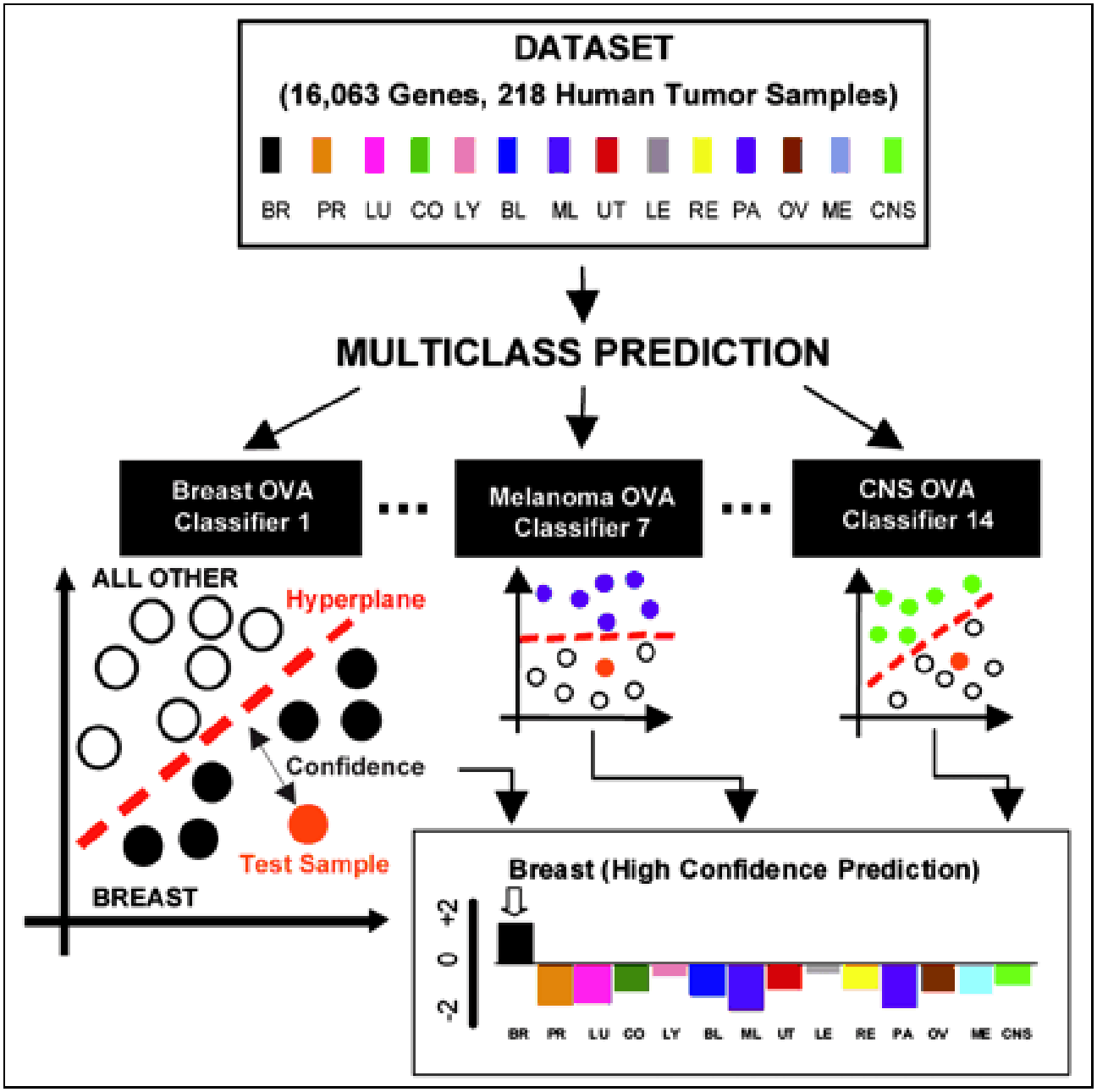

Figure 7.11: Multiclass classification scheme.
The multiclass cancer classification problem is divided into a series of 14 OVA problems, and each OVA problem is addressed by a different class-specific classifier (e.g., "breast cancer" vs. "not breast cancer"). Each classifier uses the SVM algorithm to define a hyperplane that best separates training samples into two classes. In the example shown, a test sample is sequentially presented to each of 14 OVA classifiers and is predicted to be breast cancer, based on the breast OVA classifier having the highest confidence.

For half of the errors, the correct classification corresponded to the second- or third-most confident OVA prediction.

These results were confirmed by training the multiclass SVM classifier on the entire set of 144 samples and applying this classifier to an independent set of 54 tumor samples. Overall prediction accuracy on this test set was 78%.

Poorly differentiated samples yielded low-confidence prediction in cross-validation and could not be accurately classified according to the tissue origin, indicating that they are molecularly distinct entities with different gene expression patterns compared with their well differentiated counterparts.

Overall, these results demonstrate the feasibility of accurate multiclass molecular cancer classification and suggests a strategy for future clinical implementation of molecular cancer diagnosis.

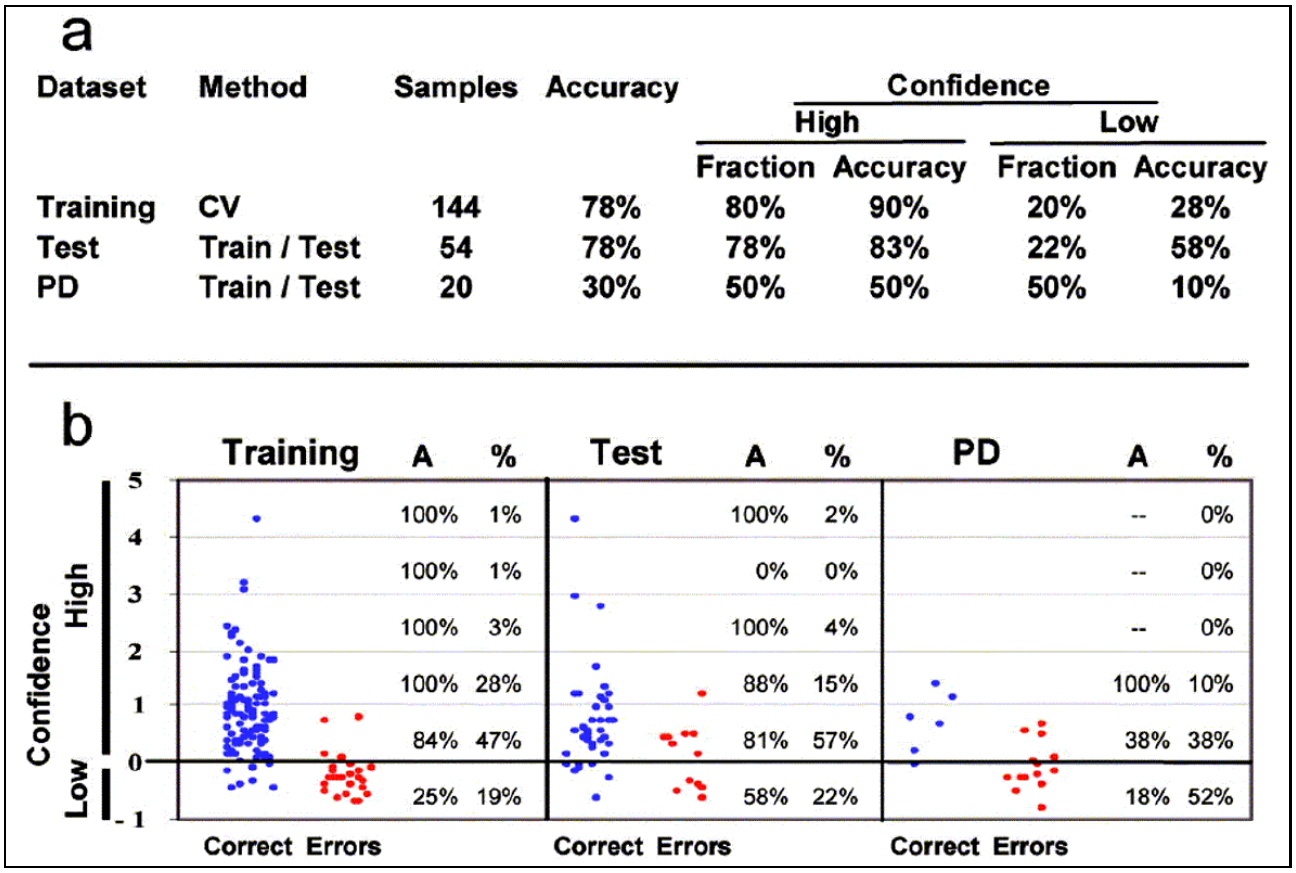

Figure 7.12: Multiclass classification results.
(a) Results of multiclass classification by using cross-validation on a training set (144 primary tumors) and independent testing with 2 test sets: Test (54 tumors; 46 primary and 8 metastatic) and PD (20 poorly differentiated tumors; 14 primary and 6 metastatic). (b) Scatter plot showing SVM OVA classifier confidence as a function of correct calls (blue) or errors (red) for Training, Test, and PD samples. A - accuracy of prediction; % - percentage of total sample number

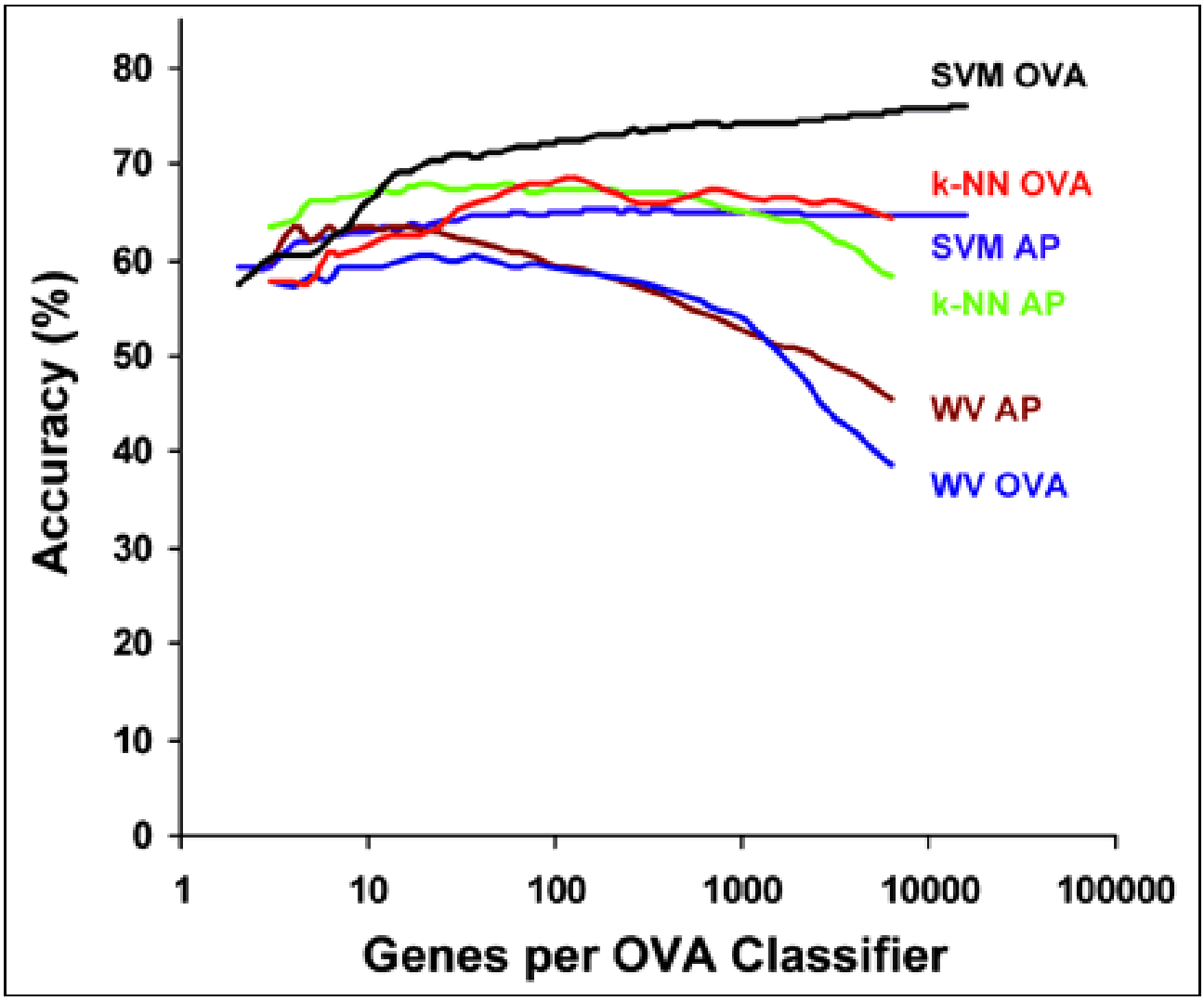

Figure 7.13: Multiclass classification as a function of gene number.
Training and test datasets were combined (190 tumors; 14 classes), then were randomly split into 100 training and test sets of 144 and 46 samples (all primary tumors) in a class-proportional manner. SVM OVA prediction was performed, and mean classification accuracy for the 100 splits was plotted as a function of number of genes used by each of the 14 OVA classifiers, showing decreasing prediction accuracy with decreasing gene number. Results using other algorithms (k-NN, k-nearest neighbors; WV, weighted voting) and classification schemes (AP, all-pairs) are also shown.

Bibliography

[1] M. P. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci. U. S. A.*, 97(1):262–267, 2000.

[2] C.J.C Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:1–47, 1998.

[3] R. Freund E. Osuna and F. Girosi. Support vector machines: Training and applications. Technical Report AIM-1602, MIT, 1996.

[4] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–14868, 1998.

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

[6] G.Zoutendijk. *Methods of Feasible Directions: A study in linear and non-linear programming*. Elsevier, 1970.

[7] K.Fukunga. *Pattern Recognition and Neural Network*. Academic Press, second edition, 1990.

[8] J. Shawe-Taylor N. Cristianini. *An Introduction to support vector machines and other Kernel based learning methods*. Cambridge University Press, 2000.

[9] S. Ramaswamy et al. Multiclass cancer diagnosis using tumor gene expression signature. *Proc. Natl. Acad. Sci. USA*, 98(26):15149–15154, 2001.

[10] B.D. Ripley. *Pattern recognition and Neural netwoks*. Cambridge University Press, 1960.

[11] D. K. Slonim, P. Tamayo, J. P. Mesirov, T. R. Golub, and E. S. Lander. Class prediction and discovery using gene expression data. In *Proceedings of the Fourth Annual International Conference on Computational Moleculaer Biology (RECOMB 2000)*, 2000.

[12] V.N.Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1999.