1

FEATURE SELECTION FOR GENOMIC AND PROTEOMIC DATA MINING

Sun-Yuan Kung and Man-Wai Mak

1.1 INTRODUCTION

The extreme dimensionality (also known as the curse of dimensionality) in genomic data has been traditionally a serious concern in many applications. This has motivated a lot of research in feature representation and selection, both aiming at reducing dimensionality of features to facilitate training and prediction of genomic data.

In this chapter, *N* denotes the number of training data samples, *M* the original feature dimension, and the full feature is expressed as an *M*-dimensional vector process

$$\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_M(t)]^T, \quad t = 1, \ldots, N.$$

The subset of features is denoted as an *m*-dimensional vector process

$$\mathbf{y}(t) = [y_1(t), y_2(t), \cdots, y_m(t)]^T \tag{1.1}$$

$$= [x_{s_1}(t), x_{s_2}(t), \cdots, x_{s_m}(t)]^T, \tag{1.2}$$

where $m \leq M$ and s_i stands for index of a selected feature.

From the machine learning's perspective, one metric of special interest is the sample–feature ratio *N*/*M*. For many multimedia applications, the sample–feature

Machine Learning in Bioinformatics. Edited by Yan-Qing Zhang and Jagath C. Rajapakse ratios lie in a desirable range. For example, for speech data, the ratio can be as high as 100 : 1 or 1000 : 1 in favor of training data size. For machine learning, such a favorable ratio plays a vital role in ensuring the statistical significance of training and validation.

Unfortunately, for genomic data, this is often not the case. It is common that the number of samples is barely compatible with, and sometimes severely outnumbered by, the dimension of features. In such situation, it becomes imperative to remove the less relevant features, that is, features with low signal-to-noise ratio (SNR) [1].

It is commonly acknowledged that more features means more information available for disposal, that is,

$$I[A] \leq I[A \cup B)] \leq \cdots, \tag{1.3}$$

where A and B represent two features, say x_i and x_j, respectively, and $I(X)$ denotes information of X. However, the redundant and noisy nature of genomic data makes it not always advantageous but sometimes imperative to work with properly selected features.

1.1.1 Reduction of Dimensionality (Biological Perspectives)

In genomic applications, each gene (or protein sequence) corresponds to a feature in gene profiling (or protein sequencing) applications. Feature selection/representation has its own special appeal from the genomic data mining perspective. For example, it is a vital preprocessing stage critical for processing microarray data. For gene expression profiles, the following factors necessitate an efficient gene selection strategy.

1. *Unproportionate Feature Dimension w.r.t. Number of Training Samples.* For most genomic applications, the feature dimension is excessively higher than the size of the training data set. Some examples of the sample–feature ratios N/M are

protein sequences	$\rightarrow$	1 : 1
microarray data	$\rightarrow$	1 : 10 or 1 : 100

   Such an extremely high dimensionality has a serious and adverse effect on the performance. First, high dimensionality in feature spaces increases the computational cost in both (1) the learning phase and (2) the prediction phase. In the prediction phase, the more the features used, the more the computation required and the lower the retrieval speed. Fortunately, the prediction time is often linearly proportional to the number of features selected. Unfortunately, in the learning phase, the computational demand may grow exponentially with the number of features. To effectively hold down the cost of computing, the features are usually quantified on either *individual* or *pairwise* basis. Nevertheless, the quantification cost is in the order of $O(M)$ and $O(M^2)$ for individual and pairwise quantification, respectively (see Section 1.2).

3. *Plenty of Irrelevant Genes.* From the biological viewpoint, only a small portion of genes are strongly indicative of a targeted disease. The remaining "housekeeping" genes would not contribute relevant information. Moreover, their participation in the training and prediction phases could adversely affect the classification performance.
4. *Presence of Coexpressed Genes.* The presence of coexpressed genes implies that there exists abundant redundancy among the genes. Such redundancy plays a vital role and has a great influence on how to select features as well as how many to select.
5. *Insight into Biological Networks.* A good feature selection is also essential for us to study the underlying biological process that lead to the type of genomic phenomenon observed. Feature selection can be instrumental for interpretation/tracking as well as visualization of a selective few of most critical genes for *in vitro* and *in vivo* gene profiling experiments. The selective genes closely relevant to a targeted disease are called biomarkers. Concentrating on such a compact subset of biomarkers would facilitate a better interpretation and understanding of the role of the relevant genes. For example, for *in vivo* microarray data, the size of the subset must be carefully controlled in order to facilitate an effective tracking/interpretation of the underlying regulation behavior and intergene networking.

1.1.2 Reduction of Dimensionality (Computational Perspectives)

High dimensionality in feature spaces also increases uncertainty in classification. An excessive dimensionality could severely jeopardize the generalization capability due to overfitting and unpredictability of the numerical behavior. Thus, feature selection must consider a joint optimization and sometimes a delicate trade-off of the computational cost and prediction performance. Its success lies in a systematic approach to an effective dimension reduction while conceding minimum sacrifice of accuracy.

Recall from Equation 1.3 that the more the features the higher the achievable performance. This results in a monotonically increasing property: the more the features selected, the more the information is made available, as shown in the lower curve in Fig. 1.1a.

However, there are a lot of not-so-informative genomic features that are noisy and unreliable. Their inclusion is actually much more detrimental (than beneficial), especially in terms of numeric computation. Two major and serious adverse effects are elaborated below:

- *Data Overfitting.* Note that overoptimizing the training accuracy as the exclusive performance measure often results in overfitting the data set, which in turn degrades generalization and prediction ability.

  It is well known that data overfitting may happen in two situations: one is when the feature dimension is reasonable but too few training data are available; the other is when the feature dimension is too high even though there is a

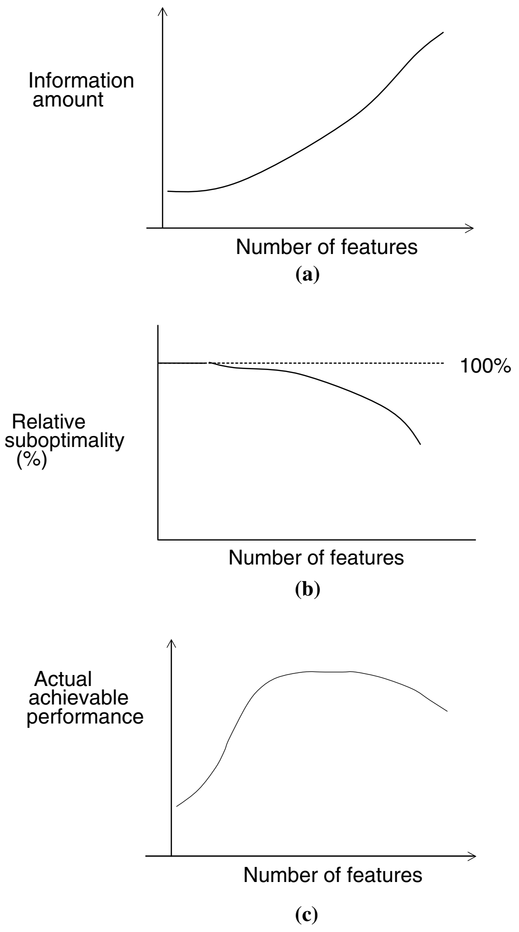


Figure 1.1 (a) Monotonic increasing property of the total information available. (b) Relative performance versus the feature size taking into consideration data overfitting and limited computational resources. (c) Nonmonotonic increasing property of the actual classification performance achievable. The best performance is often achieved by selecting an optimal size instead of the full set of available features.

reasonable amount of training data. What matters most is the ratio between the feature dimension and the size of the training data set. In short, classification/generalization depends on the sample–feature ratio.

Unfortunately, for many genomic applications, the feature dimension can be as high or much higher than the size of the training data set. For these applications, overtraining could significantly harm generalization and feature reduction is an effective way to alleviate the overtraining problem.

- *Suboptimal Search.* Practically, the computational resources available for most researchers are deemed to be inadequate, given the astronomical amounts of genomic data to be processed. High dimensionality in feature spaces increases uncertainty in the numerical behaviors. As a result, a computational process often converges to a solution far inferior to the true optimum, which may compromise the prediction accuracy.

In conclusion, when the feature size is too large, the degree of suboptimality must reflect the performance degradation caused by data overfitting and limiting computational resource (see Fig. 1.1b). This implies a nonmonotonic property on achievable performance w.r.t. feature size, as shown in Fig. 1.1c. Accordingly, but not surprisingly, the best performance is often achieved by selecting an optimal subset of features. The use of any oversized feature subsets will be harmful to the performance. Such a nonmonotonic performance curve, together with the concern on the processing speed and cost, prompts the search for an optimal feature selection and dimension reduction.

Before we proceed, let us use a subcellular localization example to highlight the importance of feature selection.

Example 1 (Subcellular localization). *Profile alignment support vector machines (SVMs) [2] are applied to predict the subcellular location of proteins in an eukaryotic protein data set provided by Reinhardt and Hubbard [3]. The data set comprises 2427 annotated sequences extracted from SWISSPROT 33.0, which amounts to 684 cytoplasm, 325 extracellular, 321 mitochondrial, and 1097 nuclear proteins. Fivefold cross-validation was used to obtain the prediction accuracy. The accuracy and testing time for different number of features selected by a Fisher-based method [4] are shown in Fig. 1.2. This example offers an evidence of the nonmonotonic performance property based on real genomic data.* □

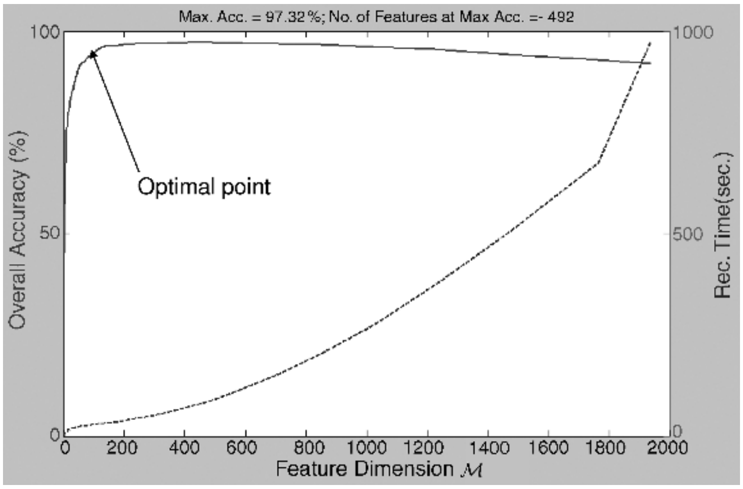


Figure 1.2 Real data supporting the monotonic increasing property. *Upper curve*: performance reaches a peak by selecting an optimal size instead of the full set of the features available. *Lower curve*: the computational time goes up (more than linear rate) as the number of features increases.

1.1.3 How Many Features to Select or Eliminate?

The question now is how many features should be retained, or equivalently how many should be eliminated? There are two ways to determine this number.

1. *Predetermined Feature Size.* A common practice is to have a user-defined threshold, but it is hard to determine the most appropriate threshold. For some applications, we occasionally may have a good empirical knowledge of the desirable size of the subset. For example, how many genes should be selected from, say, the 7129 genes in the leukemia data set [5]? Some plausible feature dimensions are as follows:
   (a) From classification/generalization performance perspective, a sufficient sample–feature ratio would be very desirable. For this case, empirically, an order of 100 genes seems to be a good compromise.
   (b) If the study concerns a regulation network, then a few extremely selective genes would allow the tracking and interpretation of cause–effect between them. For such an application, 10 genes would be the right order of magnitude.
   (c) For visualization, two to three genes are often selected for simultaneous display.
2. *Prespecified Performance Threshold.* For most applications, one usually does not know *a priori* the right size of the subset. Thus, it is useful to have a preliminary indication (formulated in a simple and closed-form mathematical criterion) on the final performance corresponding to a given size. Thereafter, it takes a straightforward practice to select/eliminate the features whose corresponding criterion functions are above/below a predefined threshold.

1.1.4 Unsupervised and Supervised Selection Criteria

The features selected serve very different objectives for unsupervised versus supervised learning scenarios (see Fig 1.3). Therefore, each scenario induces its own type of criterion functions.

1.1.4.1 Feature Selection Criteria for Unsupervised Cases In terms of unsupervised cases, there are two very different ways of designing the selection criteria. They depend closely on the performance metric, which can be either fidelity-driven or classification-driven.

1. *Fidelity-Driven Criterion.* The fidelity-driven criterion is motivated by how much of the original information is retained (or lost) when the feature dimension is reduced. The extent of the pattern spread associated with that feature is evidently reflected in the second-order moment for each feature x_i, $i = 1, \cdots, M$. The larger the second-order moment, the wider the spread, thus the more likely the feature x_i contains useful information.

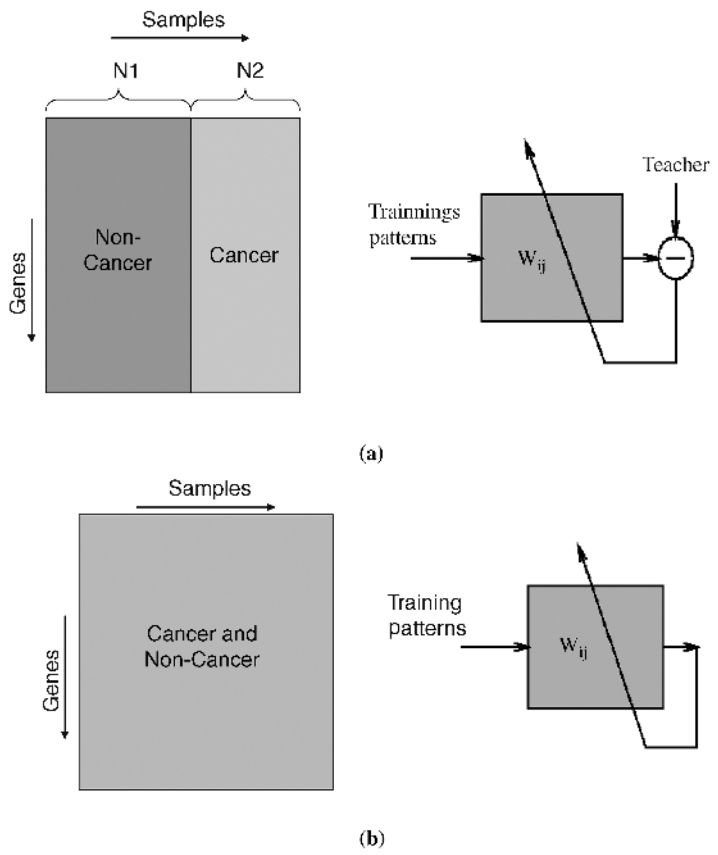


Figure 1.3 Difference between (a) supervised and (b) unsupervised feature selection.

There are two major types of fidelity-driven metrics:

- A performance metric could be based on the so-called mutual information: $I(\mathbf{x}|\mathbf{y})$.
- An alternative measure could be one which minimizes the reconstruction error:

$$\epsilon(\mathbf{x}|\mathbf{y}) \equiv \min_{\mathbf{y}\in\Re^m} ||\mathbf{x} - \hat{\mathbf{x}}_\mathbf{y}||,$$

where $\hat{\mathbf{x}}_\mathbf{y}$ denotes the estimate of $\mathbf{x}$ based on $\mathbf{y}$.

2. *Classification-Driven Criterion.* From the classification perspective, separability of data subclusters plays an important role. Thus, the corresponding criterion depends on how well can the selected features reveal the subcluster structure. The higher-order statistics, known as independent component analysis (ICA), has been adopted as a popular metric. For more discussion on this subject, see Ref. [6].

1.1.4.2 Feature Selection Criteria for Supervised Cases The ultimate objective for supervised cases lies in a high classification/predition accuracy. Ideally speaking, if the classification information is known, denoted by C, the simplest criterion will be $I(C|\mathbf{y})$. However, the comparison between $I(C|\mathbf{x})$ and $I(C|\mathbf{y})$ often provides a more useful metric. For example, it is desirable to have

$$I(C|\mathbf{y}) \rightarrow I(C|\mathbf{x}),$$

while keeping the feature dimension m as small as possible. However, the above formulation is numerically difficult to achieve. The only practical solution known to exist is the one making the full use of the feedback from the actual classification result, which is computationally very demanding. (The feedback-based method is related to the wrapper approach to be discussed in Section 1.4.5.)

To overcome this problem, an SNR-type criterion based on the Fisher discriminant analysis is very appealing. (Note that the Fisher discriminant offers a convenient metric to measure the interclass separability embedded in each feature.) Such a feature selection approach entails computing Fisher's discriminant denoted as FD_i, $i = 1, \ldots, M$, which represents the ratio of intercluster distance to intracluster variance for each individual feature. (This related to the filter approach to be discussed in Section 1.4.1.)

1.1.5 Chapter Organization

The organization of the chapter is as follows. Section 1.2 provides a systematic way to quantify the information/redundancy of/among features, which is followed by discussions on the approaches to ranking the relevant features and eliminating the irrelevant ones in Section 1.3. Then, in Section 1.4, two supervised feature selection methods, namely filter and warper, are introduced. For the former, the features are selected without explicit information on classifiers nor classification results, whereas for the latter, the select requires such information explicitly. Section 1.5 introduces a new scenario called self-supervised learning in which prior known group labels are assigned to the features, instead of the vectors. A novel SVM-based feature selection method called Vector-Index-Adaptive SVM, or simply VIA-SVM, is proposed for this new scenario. The chapter finishes with experimental procedures showing how self-supervised learning and VIA-SVM can be applied to (protein-sequence-based) subcellular localization analysis.

1.2 QUANTIFYING INFORMATION/REDUNDANCY OF/AMONG FEATURES

Quantification of information and redundancy depends on how the information is represented. A representative feature is the one that can represent a group of similar features. Denote S as a feature subset, that is, $S \equiv \{y_i\}$, $i = 1, \ldots, m$. In addition to the general case, what of most interest is either a single individual feature $m = 1$ or a pair of features $m = 2$. A generic term $I(S)$ will be used temporarily to denote the information pertaining to S, as the exact form of it has to depend on the application scenarios.

Recall that there are often a large number of features in genomic data sets. To effectively hold down the cost of computing, we have to limit the number of features simultaneously considered in dealing with the interfeature relationship. More exactly, such computational consideration restricts us to three types of quantitative measurements of the feature information:

1. *Individual Information*: The quantification cost is in the order of $O(M)$.
2. *Pairwise Information*: The quantification cost becomes now $O(M^2)$.
3. *Groupwise Information:* (with three or more features).

The details can be found in the following text.

1.2.1 Individual Feature Information

Given a single feature x_i, its information is denoted as $I(x_i)$. Such a measure is often the most effective when the features are statistically independent. This leads to the individual ranking scheme in which only the information and/or discriminative ability of individual features are considered. This scheme is the most straightforward, since each individual feature is independently (and simultaneously) evaluated. Let us use a hypothetical example to illustrate the individual ranking scheme.

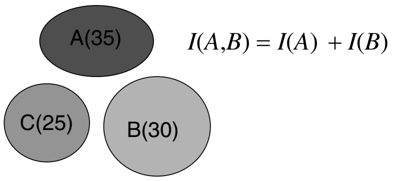
Example 2 (Three-party problem—without interfeature redundancy). *The individual ranking method works the best when the redundancy plays no or minimal role in affecting the final ranking. In this example, each area in Fig. 1.4 represents one feature. The size of the area indicates the information or discriminativeness pertaining to a feature. In the figure, no "overlapping" between elements symbolizes the fact that there exists no mutual redundancy between the features. In this case, the combined information of any two features is simply the sum of two individual amounts. For example,* $I(A \cup B) = I(A) + I(B) = 35 + 30 = 65$.

When all the features are statistically independent, it corresponds to the fact that there is no overlap pictorially. All methods lead to the same and correct result. It is, however, a totally different story with the statistically dependent cases. □

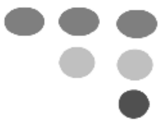
Unfortunately, the downside of considering the feature individually is that it does not fully account for the redundancy among the features. For example, it is very possible that two highest-rank individual features share a great degree of similarity. As a result, the inclusion of both features would amount to a waste of resource. In fact, one needs to take the interfeature relationship (such as mutual similarity/redundancy) into account. This problem can be alleviated by adopting either pairwise or groupwise information to be discussed next.

1.2.2 Pairwise Feature Information

Given a pair of features x_i and x_j, its information is denoted as $I(x_i \cup x_j)$. The main advantage of studying the pairwise relationship is to provide a means to identify the *similariy/redundancy* of the pair. A fundamental and popular criterion is based on



(a) Three-party problem



(b) Forward selection

(c) Backward elimination

Selection Method	Select 1 Feature	Select 2 out of 3 Features
Independent ranking	A	AB
Forward selection	A	AB
Backward elimination	A	AB
Correct solution	A	AB

(d)

Figure 1.4 (a) Three-party problem without redundancy. No "overlapping" between elements symbolizes the fact that no mutual redundancy exists between the features. (b) Consecutive result of the step-by-step forward selection. (c) Consecutive result of the step-by-step backward elimination. (d) Table illustrating search results of different strategies.

correlation. For example, the Pearson correlation coefficient is defined as

$$r_{x_i x_j} = \frac{E[x_i x_j]}{\mathrm{var}(x_i)\mathrm{var}(x_j)} = E\left[x_i x_j\right].$$

Without loss of generality, here we shall simply assume that both the features x_i and x_j are zero mean with unit variance $\mathrm{var}(x_i)\mathrm{var}(x_j) = 1$.

From the practical decision perspective, there are again two pairwise criteria: (1) mutual predictability and (2) mutual information.

1. *Mutual Predictability.* The mutual predictability represents the ability of estimating one feature from another feature. Such a metric is also closely tied with the (Pearson) correlation coefficients, that is,

$$\hat{x}_j = E[x_j|x_i] = r_{x_i x_j} x_i. \qquad (1.4)$$

   When there is no correlation, that is, $r_{x_i x_j} = 0$, then $\hat{x}_j = 0$ regardless of whatever the value of x_i is. In other words, the information of x_i offers no information about x_j. In general, the predictability is a function of $r_{x_i x_j}$; the higher the correlation, the more predictable is x_j given x_i.
2. *Mutual Information.* Suppose that there exists pairwise redundancy, then $I(x_i \cup x_j) \leq I(x_i) + I(x_j)$. The mutual information $I(x_i, x_j)$ is also a function of $r_{x_i x_j}$, the higher the correlation, the greater the mutual information.

Individual information and the pairwise information complement each other very well. In fact,

- Individual information reveals the separability of subclusters (termed as SNR in supervised case, see Section 1.4.1), while the pairwise information does not.
- Conversely, pairwise information reveals the redundancy between features, while the individual information does not.

1.2.3 Groupwise Information (with Three or More Features)

In order to optimally evaluate total information contained in a subset of multiple features, the most prudent approach is to have the entire group's information content, $I(\mathbf{y})$, evaluated collectively as an undivided entity.

Sometimes, it is more meaningful or direct to evaluate the information loss in terms of the difference between $I(\mathbf{x})$ and $I(\mathbf{y})$ due to the features missing from the subset. A common objective is to drop as many irrelevant features as possible but suffering from a minimum loss.

The importance of *groupwise information* is evident:

- When compared with individual ranking, the group evaluation has a clear advantage in harnessing information about interfeature redundancy. When compared with pairwise information, it can reveal the SNR as well as a fuller picture of interfeature redundancy.
- When compared with the consecutive ranking (to be discussed next), it delivers a relatively fair solution, because no feature has a better or poorer chance of being selected/eliminated just because it is being evaluated earlier than others.

The downside of group ranking is that it is computationally demanding to find the best combination of features, since it involves exhaustive search to find such an optimal subset of features. If every possible combination has to be considered, it would involve $2^M - 1$ possible combinations.[1] This represents a very formidable computation cost, rendering the group evaluation approach impractical.

1.3 CONSECUTIVE RANKING OF FEATURES

Based on the different types of information discussed previously, this section will look into computationally effective selection strategies.

In seeking a reasonable compromise between accuracy and cost, consecutive search approach arises as a very viable option. The consecutive ranking evaluates features on a one-by-one basis. Because of the interfeature redundancy, the order of

[1]Even if only those subsets of size m or lower are searched, it still amounts to $C_m^M \times 2^m - 1$ possible combinations. The search space can be substantially reduced to only C_m^M groups, if the size of subsets is predetermined to be exactly m.

feature selection can significantly affect the outcome of the selection. There are two ways to conduct a consecutive search:

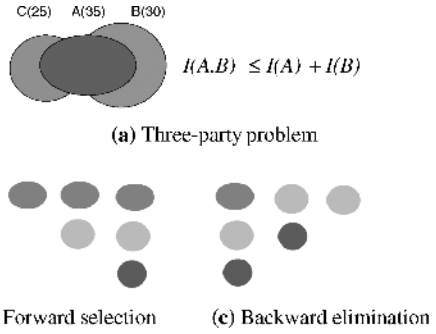
1. *Forward Selection (Augmenting).* Search usually begins at an empty feature set.
2. *Backward Elimination.* Search can start at either the full set (i.e., backward elimination) or, more generally, any subset. This can be considered as a special case of the overselect-then-prune strategy.

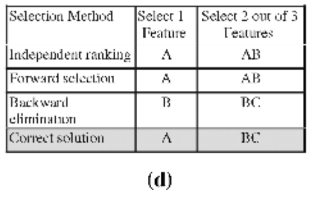
1.3.1 Forward Search: Most Innovative First Admitted

Forward search is a recursive search scheme, which begins at an empty feature set. It adopts a most innovative first admitted (MIFA) strategy. In this scheme, one feature is added to the existing chosen subset at each step. (Once a feature is admitted, it will not be dropped anymore, except in an iterative procedure.) The *importance* of the candidate features is not judged by its individual merit but rather by how much extra value it can add to the existing subset. In other words, the usefulness of a feature depends on how well does it complement the current subset. The search continues until either a preset number of features is reached or a prespecified performance is achieved.

Example 3 (Three-Party problem—with interfeature redundancy. *The scenario is illustrated in Fig. 1.5. If only one feature is to be selected, the optimal and correct solution is* A. *This is because*

$$I(A) > I(B) > I(C).$$





Selection Method	Select 1 Feature	Select 2 out of 3 Features
Independent ranking	A	AB
Forward selection	A	AB
Backward elimination	B	BC
Correct solution	A	BC

(d)

Figure 1.5 (a) Three-party problem with redundancy, where $A(35)$, $B(30)$, and $C(25)$ exhibit a peculiar overlapping pattern. "Overlapping" between elements symbolizes the fact that mutual redundancy exists between the features. (b) Consecutive result of the step-by-step forward selection. (c) Consecutive result of the step-by-step backward elimination. (d) Table illustrating search results of different strategies.

If more than one feature are to be selected, then the redundancy among the features selected plays a role. For example, it is often the case that a gene A is very discriminative on its own, and so is gene B. But the information revealed by gene A overlaps significantly with that of gene B. The interfeature redundancy implies that

$$I(A \cup B) \leq I(A) + I(B). \tag{1.5}$$

In Fig. 1.5, such interfeature redundancy is pictorially displayed as "overlapping" of two corresponding elliptic areas. The size of overlap indicates the amount of "information discount" in Equation 1.5.

In this example, there is a large overlap between A and *B*, which is, visually speaking, about half of the size of $B(\approx 30/2 = 15)$. Therefore,

$$I(A \cup B) \approx 50 \leq 55 = I(A) + I(B).$$

In contrast, there is virtually no overlap between B and C.

$$I(B \cup C) \approx 55 = I(A) + I(B).$$

Therefore, $I(A \cup B) \leq I(B \cup C)$. If a two-feature subset is to be selected, the optimal and correct solution is B and C. Note that the individual ranking would select A and B, because they have the two highest scores, which is an incorrect selection.

In the forward ranking, A will be chosen in the first round. In the second round of selection, B will be a better choice to be teamed up with A. So the best two features would be A and B, again an incorrect selection. See Fig. 1.5b. □

1.3.2 Backward Elimination: Least Useful First Eliminated

This is a recursive search scheme, usually starting at the full set. In this scheme, at each step, one feature is removed from the current subset based on strategy in which the feature with least useful (or indispensable) is first eliminated.[2] The *impact* of the candidate features is not judged by its individual merit but rather by how much loss of information is caused by its removal from the current subset. The one that incurs the least damage is deemed to have the least importance (or impact), which is also a natural candidate for elimination at that step. In this sense, the elimination rule is that the feature causing least loss gets first eliminated.

To minimize the incurred loss of information, we compute

$$\text{Loss} = I(S) - I(S^-),$$

where S denotes the current subset and S^- denoted the subset after the elimination.

Example 4 (Backward elimination: three-party problem). *Let us continue the example illustrated in Fig. 1.5. In backward elimination, A will be eliminated in the first round. This is a surprising result, because A would be considered to be the "most important"*

[2]Once a feature is eliminated, it will not be reselected anymore.

according to independent ranking. However, according to the backward elimination rule, A is considered to be the "least important" because its elimination will incur least loss. Therefore, A is eliminated and the best two-feature subset is B and C (see Fig. 1.5c). This matches with the correct answer derived via the group evaluation approach. In this case, only the backward elimination reaches the correct solution; cf. Fig. 1.5d.

Unfortunately, this also means that the backward approach would result in an incorrect selection if only one feature is to be selected, because the correct solution, that is, feature A, would be eliminated in the very first round, since it is regarded as having the least impact (and thus most dispensable) in the first iteration of the backward elimination. Among the remaining two features, feature B would beat feature C in the second round and get selected if only one feature is wanted, as shown in Fig. 1.5d. □

1.4 SUPERVISED FEATURE SELECTION AND EXTRACTION

For supervised selection, there are two types of selection criteria: closed-loop type and open-loop type.

- *Filter (Open-Loop) Approach.* In order to facilitate a quick search of candidate features (they may or may not be the final choice), some simple open-loop criterion functions may be desirable for guiding or conducting an efficient preliminary search. In this case, the selection method/criterion is self-determined, that is, it is independent of the classifier.
- *Wrapper (Closed-Loop) Approach.* The classification method is predetermined and it has substantial influence on the choice of the feature subset. For supervised learning cases, the ultimate goal is to yield the highest possible classification accuracy while using only a subset of features within the given size constraint. In order to estimate the exact accuracy, the only known approach is via a closed-loop solution, that is, it involves the complete training and testing phase of the adaptive classification. This unfortunately demands an enormous computational burden, which may prove too costly.

While the filter method is simple to compute, it fails to consider the critical inter-feature redundancy. The wrapper approach, however, can fully rectify this problem, although its closed-loop process is very computational demanding. It is natural to combine or consider both methods in order to reach an optimal strategy.

1.4.1 Filter Method—An Open-Loop Approach

In the filter method, feature selection and classifier design are separated in that a subset of features is first selected and then classifiers are trained based on the selected features. From the structural dependence perspective, while the classifier has to depend on the selection strategy, the features are selected with no regard of what classifier will be adopted.

All the filter approaches are open-loop approaches, that is, no information on the postprocessing classifier is needed. In addition, they are all based an individual ranking scheme, mostly using an SNR-type criterion. An SNR-type criterion, which is intuitively a normalized distance, is defined as follows:

$$\mathrm{SNR} = \frac{\text{signal}}{\text{noise}} = \frac{\text{distance}}{\text{variance}},$$

where the signal is the numerator representing the distance (separation) between the two centroids (one for positive class and the other negative class) and the noise is the denominator representing the variance of each class of data.[3]

Example 5. *Golub's SNR*:

$$\mathrm{SNR}(j) = \text{signed-FDR}(j) = \frac{\mu_j^+ - \mu_j^-}{\sigma_j^+ + \sigma_j^-}, \tag{1.6}$$

where μ_j^+, μ_j^-, σ_j^+, and σ_j^- represent the class-conditional means and standard derivations of the jth feature, respectively. Note that this criterion follows the idea of Fisher discriminant, and therefore, will be referred to as signed Fisher discriminant ratio (signed-FDR) in the sequel. □

If our goal is to identify a single feature, then the best choice can be obtained as

$$\arg\max_j \mathrm{SNR}(j). \tag{1.7}$$

Such formulation can be easily extended to multiple-feature selection, where we retain the *m* features with the highest scores.

In filter approaches, because feature selection acts as a preprocessor and the classifier does not play any role in the selection process, the feature selector can be considered as a "master" governing the final set of feature to be used and the classifier can be considered as a "slave" that uses whatever features provided by the feature selector.

Several SNR-type ranking criteria have been proposed in the literature. They are summarized in Table 1.1.

1.4.1.1 First Order SNR-Type Criteria

- *Signed-FDR.* The discriminative power of features is evaluated independently according to [5]

$$\text{signed} - \mathrm{FDR}(j) = \frac{\mu_j^+ - \mu_j^-}{\sigma_j^+ + \sigma_j^-}, \tag{1.8}$$

[3]Here, we assume that the positive and negative classes have the same variance. When the variances are different, then the average of the two variances will be adopted instead.

Table1.1 SNR-based filter methods

Name	Order	Criterion	Reference
Signed-FDR	First	$\frac{\mu_j^+ - \mu_j^-}{\sigma_j^+ + \sigma_j^-}$	[5]
t-Statistics	First	$\frac{\mu_j^+ - \mu_j^-}{\sqrt{\frac{(s_j^+)^2}{N^+} + \frac{(s_j^-)^2}{N^-}}}$	[7]
FDR	Second	$\frac{(\mu_j^+ - \mu_j^-)^2}{(\sigma_j^+)^2 + (\sigma_j^-)^2}$	[8]
SD	Second	$\frac{1}{2}\left(\frac{(\sigma_j^+)^2}{(\sigma_j^-)^2} + \frac{(\sigma_j^-)^2}{(\sigma_j^+)^2}\right) - 1 + \frac{1}{2}\left(\frac{(\mu_j^+ - \mu_j^-)^2}{(\sigma_j^+)^2 + (\sigma_j^-)^2}\right)$	[4]

FDR: Fisher discriminant ratio; SD: symmetric divergence. "+" and "−" represent positive and negative classes, respectively. Refer to Equation 1.10 for s_j^+ and s_j^-.

where μ_j^+, μ_j^-, σ_j^+, and σ_j^- represent the class-conditional means and standard derivations of the jth feature, respectively. Furey et al. [9] proposed to use the absolute value of signed-FDR(j) as ranking criterion. Features with high signed-FDR(j) or |signed-FDR(j)| are selected for classification. The method is intuitively appealing because high signed-FDR(j) or |signed-FDR(j)| means that the corresponding features produce maximum separation between the positive and negative classes.

- *T-Statistics.* A similar ranking criterion is based on the t-statistics [7]:

$$z_j = \frac{\mu_j^+ - \mu_j^-}{\sqrt{\frac{(s_j^+)^2}{N^+} + \frac{(s_j^-)^2}{N^-}}}, \tag{1.9}$$

where

$$(s_j^+)^2 = \frac{\sum_{k\in C^+}(x_{kj} - \mu_j^+)^2}{N^+ - 1} \quad \text{and} \quad (s_j^-)^2 = \frac{\sum_{k\in C^-}(x_{kj} - m_j^-)^2}{N^- - 1}, \tag{1.10}$$

where N^+ and N^- are the numbers of training samples in the positive class C^+ and negative class C^-, respectively.

1.4.1.2 Second Order SNR-Type Criteria

- *Fisher Discriminant Ratio (FDR).* A slight variant of signed-SNR is the FDR [8]:

$$FDR(j) = \frac{(\mu_j^+ - \mu_j^-)^2}{(\sigma_j^+)^2 + (\sigma_j^-)^2}. \tag{1.11}$$

As an illustrative example, Fig. 1.6 shows the FDR of 7129 genes in an acute leukemia data set [5] that contains two types of acute leukemia: ALL and AML.

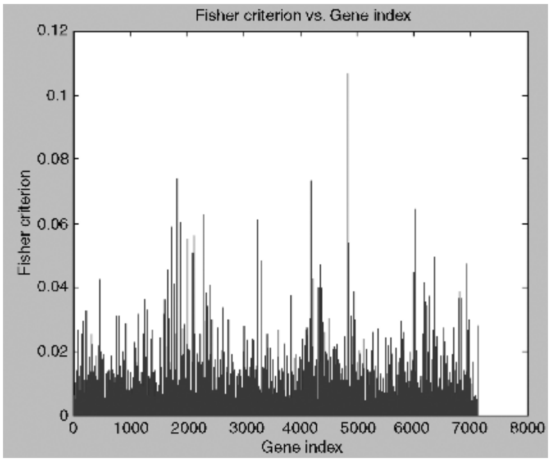


Figure 1.6 The Fisher discriminant ratio of 7192 genes in the acute leukemia data set 5. The FDR cutoff point (threshold) and the corresponding number of remaining genes are shown in Table 1.2.

Evidently, only a small number of genes have large FDR, meaning that only a few genes are useful for differentiating the two types of acute leukemia. Table 1.2 shows the numbers of selected genes at different cutoff points.

- *Symmetric Divergence (SD).* By assuming the distributions of the positive and negative classes as Gaussians, Mak and Kung [4] proposed ranking the features according to the symmetric divergence between the positive and negative class distributions:

$$D(p(x_j^+)||p(x_j^-)) = E\left\{\log\frac{p(x_j^+)}{p(x_j^-)}\middle|C^+\right\} + E\left\{\log\frac{p(x_j^-)}{p(x_j^+)}\middle|C^-\right\}$$
$$= \frac{1}{2}\left(\frac{(\sigma_j^+)^2}{(\sigma_j^-)^2} + \frac{(\sigma_j^-)^2}{(\sigma_j^+)^2}\right) - 1 + \frac{1}{2}\left(\frac{(\mu_j^+ - \mu_j^-)^2}{(\sigma_j^+)^2 + (\sigma_j^-)^2}\right), \tag{1.12}$$

where $p(x_j^+)$ and $p(x_j^-)$ represent the density functions of the positive and negative classes, respectively. Figure 1.7 illustrates the procedure of computing the symmetric divergence of feature j and Fig. 1.8 shows the histograms of the symmetric divergences. Apparently, only a small fraction of the features have

Table1.2 The FDR cutoff point (threshold) and the corresponding number of remaining genes

Threshold	0.01	0.02	0.04	0.06
No. of remaining genes	596	142	19	8

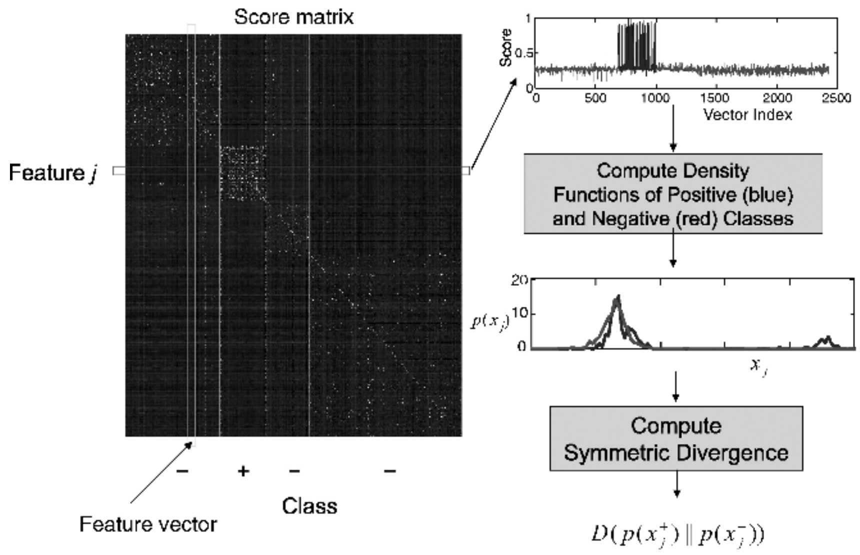

Score matrix
Feature j
Feature vector
− + − −
Class
Score
Vector Index
Compute Density Functions of Positive (blue) and Negative (red) Classes
$p(x_j)$
x_j
Compute Symmetric Divergence
$D(p(x_j^+) \| p(x_j^-))$

Figure 1.7 Computation of symmetric divergence (Eq. 1.12) in a subcellular localization task.

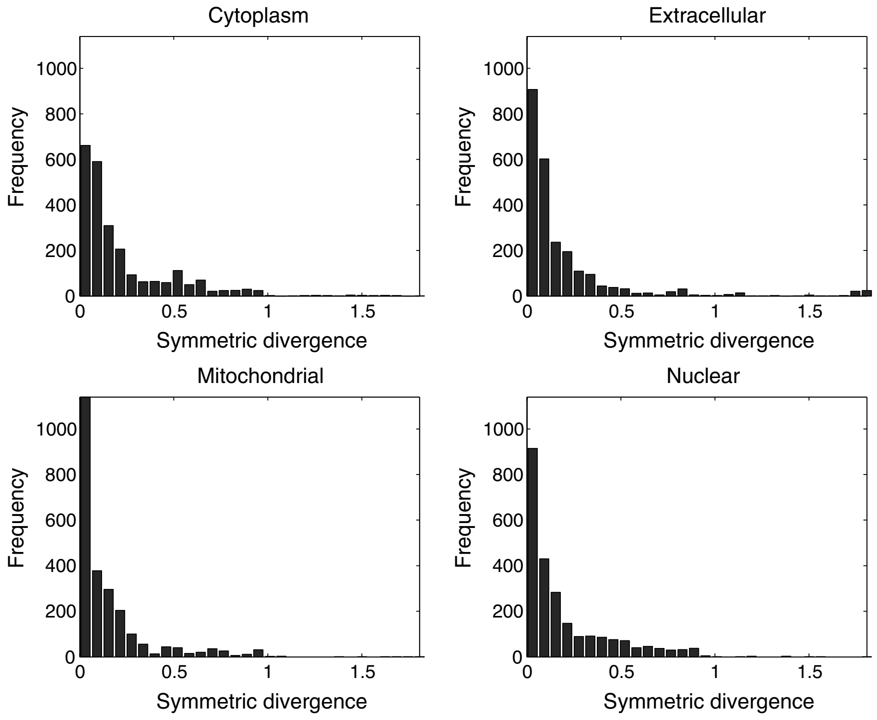


Figure 1.8 Histograms of symmetric divergences in a subcellular localization task.

large symmetric divergences, which means that only a few feature dimensions are relevant for classification. This hypothesis is confirmed by our experimental results below.

1.4.1.3 Comparing Different Filters The ranking methods based on signed-FDR, *t*-statistics, FDR, and symmetric divergence share a common property: Features with small variances but large difference in class means will be ranked high. However, there are also important differences. For example, Equations 1.11 and 1.12 differ in the additional term that depends only on the ratio of variances. Therefore, features with high variance ratio between positive and negative classes will be ranked high under the symmetric divergence criterion. *t*-Statistic is used to assess the statistical significance of the difference between the sample means of two normal distributed population when the sample size is small. However, Kullback–Leibler divergence (from which symmetric divergence is derived) is a distance measure between two probability distributions. In most cases, one distribution represents the observations and the other represents a model that attempts to approximate the former one.

To compare the capability of different filter methods, we applied signed-FDR (Eq. 1.6), absolute value of signed-FDR, FDR (Eq. 1.11), and symmetric divergence (Eq. 1.12) to select features for classifying subcellular location of proteins. The data set [3] that we used contains 2427 annotated amino acid sequences (684 cytoplasm, 325 extracellular, 321 mitochondrial, and 1097 nuclear proteins) extracted from SWISSPROT 33.0. We applied pairwise profile alignment to create a score matrix for classification by RBF-SVMs [2].

Figure 1.9 shows the accuracy of subcellular localization when the number of selected features is progressively increased from 1 to the full size. Evidently, the accuracy increases rapidly at small feature size and becomes saturated at around 200

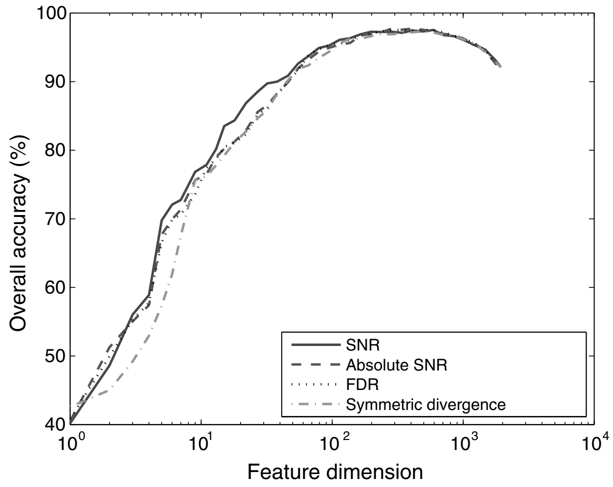


Figure 1.9 Accuracy based on features selected by signed-FDR, absolute value of signed-FDR (|Signed-FDR|), FDR, and symmetric divergence.

(the accuracy even drops slightly because of the curse of dimensionality), suggesting only one-tenth of the features are useful for classification. The results show that the performance of signed-FDR is better than that of |signed-FDR| and FDR, suggesting that the sign is important for feature selection.

1.4.1.4 Limitations of Filter Methods

Selecting features based on the filter methods suffers from several serious drawbacks. First, these methods require users to set a cutoff point in the ranking scores under which features are deemed to be irrelevant for classification. However, the optimal number of features (i.e., the best cutoff point) is usually unknown. Second, the features selected may be highly correlated because the feature set may contain many highly discriminative features but with almost identical characteristics. This means that some of these *redundant* features can be removed without affecting the classification accuracy. Third, the ranking criteria do not take the combined effect of features into account. For example, a low-ranked feature may become very useful for classification when combined with another low-ranked feature. To overcome these limitations, researchers have proposed using the performance of classifiers to guide the feature selection progress, which is to be discussed in Sections 1.4.3 and 1.4.5.

1.4.1.5 Cope with Redundancy

To develop a more appropriate open-loop criterion, one wants not only to maximize the discriminativeness of individual features, but also to minimize the redundancy between the features selected. Because features may be mutually redundant, that is, two highly ranked features may carry similar discriminative information to the target class, redundancy removal can be done by (1) forward selection based on the "most innovative first admitted" approach; and (2) backward elimination based on the "least useful first eliminated" approach. To illustrate the idea, let us consider the first strategy in the following example.

Example 6 (Forward selection of two out of three—with versus without dependence). *Without loss of generality, assume that we have a total of three features: x_1, x_2, and x_3, and that x_1 is preselected a priori. The question is which of x_2 and x_3 could be the preferred choice to team up with x_1. The answer hinges upon which one can best complement x_1 rather than which is more discriminative by itself. Mathematically, let us denote*

$$\hat{x}_2 = x_2/x_1,$$
$$\hat{x}_3 = x_3/x_1,$$

where x_i/x_1 denotes the innovation component of x_i w.r.t. x_1. *There is no unique way to define the innovation component for the supervised cases. One approach is to (1) identify the mutually dependent subspace and (2) perform coordinate transformation so that the innovative component is represented by its own axis, which is perpendicular to the dependent subspace. For example, the features x_1 and x_2 are*

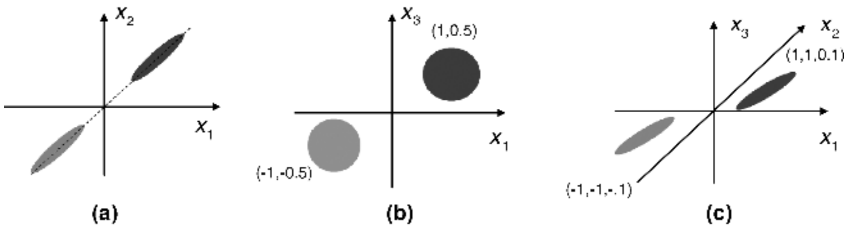


Figure 1.10 Selecting two out of three features in supervised feature selection. x_1 and x_2 are mutually dependent. x_1 and x_3 are independent. x_3 is independent of both x_1 and x_2.

mutually dependent as shown in Fig. 1.10a . Following the above-mentioned procedure, let us use a coordinate transformation:

$$\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}.$$

After the transformation, the vertical axis would represent the innovative component $\hat{x}_2$. Thus, the (innovative) discriminative power is represented by the normalized distance (FD) of $\hat{x}_2$. In contrast, the features x_1 and x_3 are mutually independent as shown in Fig. 1.10b. Therefore, one can say, given that we already have the knowledge on x_1, the innovative component of x_3 is greater than that of x_2.

Apply the same procedures to all the features. In this example, features x_2 and x_3 can be transformed to $\hat{x}_2$ and $\hat{x}_3$ *via a three-dimensional coordinate transformation:*

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

After the transformation, the best choice (to team up with x_1) can then be identified as

$$\arg\max_{i\neq 1} \ \mathrm{FD}(\hat{x}_i).$$

The interfeature dependence is mostly due to the mutual information or redundancy between two features. If two features have too much redundancy (i.e., they share very similar common information), then selection of one feature would immediately obviate the selection of the other, and vice versa.

In this example, features x_1 and x_2 have by themselves high discriminative power (in FD sense), but the two features have a great statistical similarity. Therefore, selection of x_1 would naturally obviate any chance for x_2. In this case, the second best candidate is x_3. In conclusion, the best feature pair is x_1 and x_3. □

This formulation can be easily extended to the backward elimination strategy.

1.4.2 Weighted Voting and Linear Combination of Features

After the features are selected, say, by a filter method, the next task is classification. For this, a popular approach was proposed by Golub et al. [5]. In this method, an SNR filter-based selection is combined with a weighted voting scheme. Each selected feature independently contributes a vote for a class. More exactly,

- The vote from the jth feature is represented by $v_j(x_j) = x_j - b_j$. Here the decision threshold b_j is set halfway between the class means, that is,

$$b_j = \frac{\mu_j^+ + \mu_j^-}{2}.$$

  Each feature j casts a vote to one of the two classes. A positive value is for the positive class and negative value for the negative class.
- Furthermore, every vote will be given a different weighting w_j. The weighting factor w_j is introduced to reflect how trustworthy is the feature—it depends on how well the feature is correlated with the class distinction. In Ref. [5], the weighting factor w_j is based on the SNR of the individual feature:

$$w_i = \frac{\mu_i^+ - \mu_i^-}{\sigma_i^+ + \sigma_i^-},$$

  that is, it is a function of the mean and standard deviation of the feature values.
- This leads to

$$\text{weighted total score} = \sum_j w_j v_j(x_j).$$

If the weighted total score is positive, then the sample is identified, otherwise it is rejected.

To illustrate the role of weighting, let us take a look at the data distribution shown in Fig. 1.12b. It is obvious that a higher weighting must be placed on x_2. Such a conclusion may also be reached either via the weighted voting approach (due to the higher SNR associated with x_2) or by almost any optimal linear classifier, such as LSE, Bayesian classifier, or linear SVM.

1.4.3 Linear Feature Representation

Note that

$$\text{weighted total score} = \sum_j w_j v_j(x_j) = \sum_j w_j(x_j - b_j) = \mathbf{w}^T\mathbf{x} - b,$$

where $b = \sum_j w_j b_j$. This shows that a weighted voting is basically equivalent to the use of a linear decision boundary. Then, it is natural and appealing to pursue a more direct solution for optimal linear decision boundary. This is in fact equivalent to the pursuit of an optimal linear representation of the features:

$$y = \sum_j w_j x_j$$

and an associated threshold b for optimal linear classification.

The linear feature representation is formulated as follows. Given a set of (say N) M-dimensional vectors, $X = \{\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(N)\}$, each of which is known to belong to either the positive class or the negative class, find an optimal linear combination of features that best separates two classes of data samples. More exactly, we are to find $\{w_1, w_2, \ldots, w_M\}$ to best classify the test vectors into two classes, depending on the sign of the discriminant function:

$$f(x) = w^T x - b.$$

For M-dimensional training vectors, the decision boundary represents an $(M-1)$-dimensional hyperplane.

In fact, linear classifiers via supervised training have a long history.

1. Rosenblatt's perceptron [10] was the first neural classifier proposed in 1958.
2. Least squares or MMSE formulation is to best approximate the teacher values.
3. Fisher's discriminant [11] laid the groundwork for statistical pattern recognition in 1936.
4. SVM developed by Vapnik [12] in 1995 and by Boser et al. [13] in 1992.

One would be naturally concerned that the restriction to linear classification could render the proposed methods in this chapter to have very limited application domains. Indeed, in general, the best prediction result can only be achieved via using a more sophisticated nonlinear decision boundary.

Fortunately, linear classifiers often work well for most genomic data because they usually posses a very low sample-to-feature ratio. For example, the sample-to-feature ratio for gene expression data is around 1:100 and that for vectorized data produced by pairwise sequence (e.g., protein or DNA sequences) comparison is 1:1. It is well known that linear classifiers work well under such situations. Therefore, it should be of no surprise that the two inherently linear schemes, that is, the weighted voting and wrapper approach, work well for genomic data. For example, Golub et al. [5] show that the weighted votes of a set of informative genes yield a good prediction of a new sample. More exactly, a 50-gene predictor derived in cross-validation tests correctly assigned 36 of the 38 samples as either AML or ALL, while the remaining two were labeled as uncertain (PS = 0.3).

1.4.4 Comparison of Weighted Voting and Linear Classification

The advantage of weighted voting is that it is numerically stable; thus, it is preferable especially when the number of samples is too small for conducting a robust statistical estimation. The disadvantage of weighted voting, however, is that it fails to consider redundancy among the features. Consequently, it could lead to a highly undesirable scenario in which two highly redundant features get to vote twice (one vote from each feature). This is very different from the optimal weighting strategy according to the Bayesian classification. This can be best explained in Example 7.

Example 7. *Without loss of generality, assume that we have two classes of data represented by three features: x_1, x_2, and x_3, and that x_1 is preselected a priori. The centroids for the positive and negative classes are [+1, +1, +1] and [−1, −1, −1], respectively. Assume that the two classes have the same intraclass covariance matrix:*

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \epsilon \\ 0 & \epsilon & 1 \end{bmatrix}.$$

When $\epsilon \rightarrow 0$, then all the features are statistically independent (in intraclass sense); therefore, the optimal weighting will be one with the uniform weighting, $w_1 = 1$, $w_2 = 1$, and $w_3 = 1$, a result derivable via either the weighted voting or the optimal Bayesian classification. However, when $\epsilon \rightarrow 1$, then the features x_2 and x_3 are highly correlated or they have a high mutual redundancy. Notwithstanding, the weighted voting approach would still reach a uniform weighting result, $w_1 = 1$, $w_2 = 1$, and $w_3 = 1$, because it fails to take such redundancy into account. This can be effectively handled by wrapper approaches based on linear classifier.

For example, features x_2 and x_3 are mutually dependent as shown in Fig. 1.11a . It can be derived (omitted here) that the optimal Bayesian classification is [1 1 0] or [1 0 1]. □

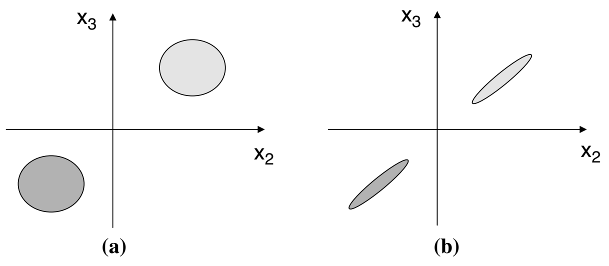
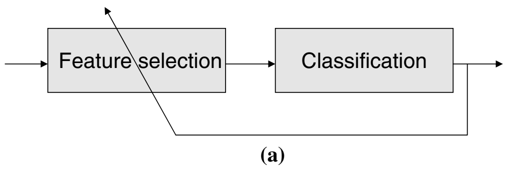


Figure 1.11 (a) Equal weights are placed on x_2 and x_3. (b) When $\epsilon \rightarrow 1$, x_2 and x_3 are highly correlated or they have a high mutual redundancy. The weighted voting approach would fail to take such redundancy into account. This can be effectively handled by wrapper approach based on linear classifier, as discussed in Example 7.



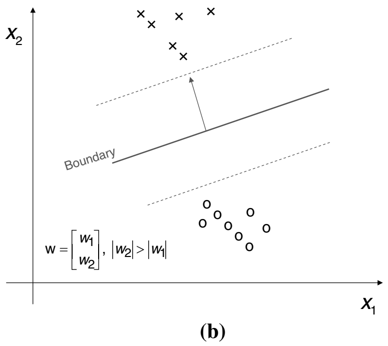

x_2
Boundary
$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, |w_2| > |w_1|$
x_1
(b)

Figure 1.12 (a) The architecture of wrappers. The classifier acts as a master guiding the feature selection process. (b) Linear separable case in which x_2 will be chosen over x_1.

This example highlights the fact that the weakness of the weighted voting scheme lies in its failure to consider the important role of redundancy. There are two possible solutions:

1. *Direct Solution.* As shown in the above example, the weighted voting is very effective when the features are relatively independent. So, it would be wise that we weed out those highly redundant features during feature selection, using for example the notion of the innovation components. See Example 6.
2. *Use a Wrapper Approach.* In this case, an optimal linear classifier would have taken into account the underlying redundancy. See Fig. 1.12.

1.4.5 SVM-Based Wrapper Methods—A Closed-Loop Approach

All of the linear classifiers mentioned earlier can be considered as candidates in the wrapper approach. Nevertheless, the literature in genomic applications, especially microarray data, seem to strongly favor the SVM approach.

It is nearly impossible to have an open-loop solution (in a closed form) to faithfully reflect the exact optimal classification. Let us highlight this point via an example. Suppose we have M individual FDs, each representing the discriminative power of a feature. A closed-form criterion would mean that there exists a function of the FDs: $f(\mathrm{FD}_1, \mathrm{FD}_2, \ldots, \mathrm{FD}_M)$, for example, $\sum_i \mathrm{FD}_i$, which can be used as a bona fide measure of the classification performance. Unfortunately, it is a naive and impractical goal. In practice, there exists no such function $f(\mathrm{FD}_1, \mathrm{FD}_2, \ldots, \mathrm{FD}_M)$ meeting the full spectrum of applications. Even with the benefit of the statistically independent assumption, the pessimistic assessment will remain true.

The fact that there will be no easy-to-apply open-loop criterion necessitates the use of closed-loop solution. This would incur an enormous amount of computation costs, because classification performance is used as critical feedbacks to assess whether the final feature subset can truly deliver an optimal performance (see Fig. 1.12). This feedback strategy is adopted by most wrapper approaches to be discussed in this subsection. Briefly, assuming that the full set (FS) of features delivers the best accuracy, our goal is to find a subset (SS) of features such that

$$|A(FS) - A(SS)|$$

is minimal, where $A(\cdot)$ stands for accuracy.

Because the ultimate goal of feature selection is to increase classification accuracy, it is intuitive to choose a particular classification method and use its parameters or its performance on training data to guide the feature selection process (see Fig. 1.12a). Typically, this is done by selecting a subset of features and evaluating its performance on the chosen classifier, and the process is repeated until the best performing subset is obtained. Methods based on this approach are known as wrappers in the literature [14].

From the structural dependence perspective, this approach is very much unlike the filter approach, the feature selection strategy depends on the classifier adopted. In fact, the classifiers in wrappers act as the master guiding the feature selection process.

A number of wrapper approaches have been proposed for bioinformatics data mining:

- *SVM Approach.* This kind of approach includes the recursive feature elimination (RFE) [15] and recursive SVM [16, 17].
- *Nearest Neighbor Approach.* A typical example of this approach is the ReliefF [18].

This subsection will focus on the SVM-RFE approach.

SVM-RFE. Guyon et al. [15] proposed a backward elimination algorithm, namely, SVM-RFE, that ranks features based on the weights of a linear SVM. The algorithm begins with using the full-feature training vectors $\mathbf{x}_i \in \Re^{D_0}$ to train a linear SVM. Features are then ranked by sorting the square of the SVM's weights $\{w_j^2\}_{j=1}^{D_0}$ in descending order, where the weight vector is given by

$$\mathbf{w} = \sum\nolimits_{i \in \mathrm{SV}} \alpha_i y_i x_i, \quad \mathbf{w} \in \Re^{D_0}. \tag{1.13}$$

A subset of features corresponding to the end of the sorted list (i.e., those with small w_j^2) is then removed.[4] The remaining features are used to construct a new set of training vectors $x_i \in \Re^{D_1}$, where $D_1 < D_0$. These vectors are then used to train another linear SVM and the process is repeated until all features have been eliminated. At the end of

[4]Intuitively, a small weight w_j means that the jth axis in the feature space is irrelevant to classification and can be removed without affecting performances.

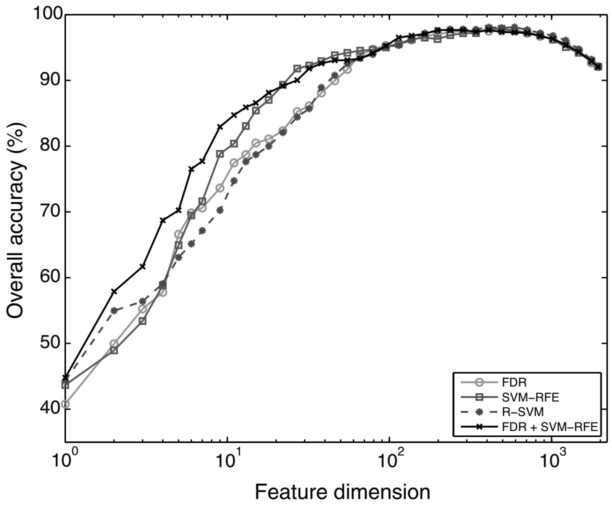


Figure 1.13 Performance of FDR, SVM-RFE, R-SVM, and the fusion of FDR and SVM-RFE in the subcellular localization task.

the iterative process, a ranked list of features is produced. Figure 1.13 compares the performance of SVM-RFE, FDR, and the fusion of SVM-RFE and FDR.

1.5 FEATURE SELECTION VIA VECTOR-INDEX-ADAPTIVE SVM FOR FAST SUBCELLULAR LOCALIZATION

The comparison of two temporal sequences is often hampered by the fact that two sequences often have different lengths whether or not they belong to the same family. To overcome this problem, pairwise comparison between a sequence and a set of known sequences has been a popular scheme for creating fixed-size feature vectors from variable-length sequences [2, 19, 20]. This process is referred to as *vectorization*. Suppose that we have M sequences, then each sequence is converted into a (column) vector of dimension M with entries representing the pairwise similarity between that sequence and all of the M sequences. In total, there will be M such M-dimensional (column) vectors. For example, in Fig. 1.14, three sequences $S^{(1)}$, $S^{(2)}$, and $S^{(3)}$ are converted to three three-dimensional column vectors. Together these vectors form an $M \times M$ matrix, named the *kernel matrix*.

In the previous sections, we have constantly used an axis to represent a feature, and therefore, feature selection/elimination means axis selection/elimination. Note that under the pairwise kernel representation, we have as many features as vectors. Then, does a feature correspond to an axis or a data point? The somewhat surprising answer is: *both*, because the symmetrical kernel matrix exhibits a useful reflexive property, which is best illustrated by an example shown in Fig. 1.15. It can be advantageous to harness such a symmetry property, which motivates the discussion of this section.

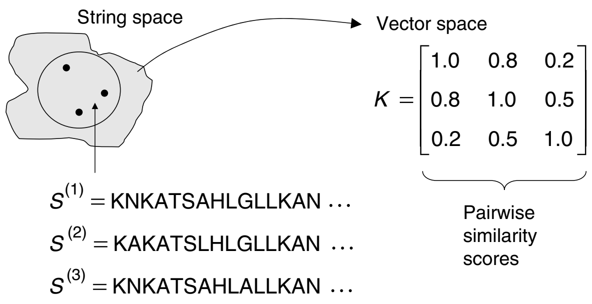
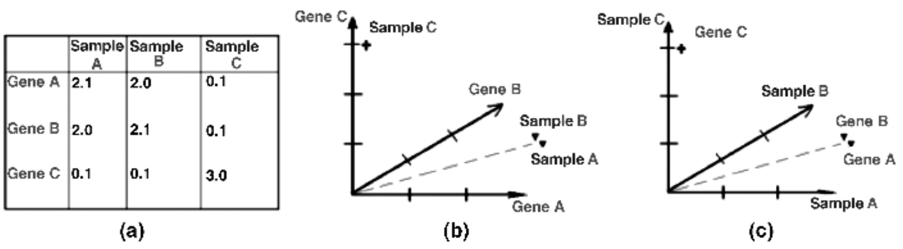

String space
Vector space
$K = \begin{bmatrix} 1.0 & 0.8 & 0.2 \\ 0.8 & 1.0 & 0.5 \\ 0.2 & 0.5 & 1.0 \end{bmatrix}$
Pairwise similarity scores
$S^{(1)}$ = KNKATSAHLGLLKAN ...
$S^{(2)}$ = KAKATSLHLGLLKAN ...
$S^{(3)}$ = KNKATSAHLALLKAN ...

Figure 1.14 Vectorizing sequences. Because strings are composed of alphabets and have different lengths, they need to be converted to vectors of the same dimension for classification via kernel methods.

The downside of using such symmetry property is that the feature dimension is the same as the number of training patterns. In fact, for the applications addressed in this chapter, they are in the range of several thousands. Therefore, such curse of dimensionality could hurt both training and recognition speed. Because a large number of sequences are being added to sequence databases on a daily basis, it is imperative to design feature selection methods that can weed out irrelevant features to reduce training and recognition time.

In this section, we propose a method that makes use of the symmetric property of pairwise scoring matrices to select relevant features. The method considers the columns of a pairwise scoring matrix as high-dimensional vectors and uses the column vectors to train a linear SVM. Because of the symmetric property of the score matrix, the row vectors with row indexes equal to the support vector indexes are identical to the support vectors. Also, because the support vectors define the decision boundary and margins of the SVM, they are critical for classification performance. Therefore, the support vector indexes are good candidates for selecting features for the column vectors, that is, only the rows corresponding to the support vectors are retained. The column vectors with reduced dimensions are then used to train another SVM for classification. Because the indexes of support vectors are used for selecting features, we referred this method to as vector-index-adaptive SVM, or simply VIA-SVM [26].



	Sample A	Sample B	Sample C
Gene A	2.1	2.0	0.1
Gene B	2.0	2.1	0.1
Gene C	0.1	0.1	3.0

Figure 1.15 Symmetry (reflexive) property of pairwise data. (a) Table showing the feature/sample data. (b) Features are displayed as axes and samples as data vectors. (c) Features are displayed as data vectors and samples as axes.

Later in the section, we shall compare the VIA-SVM with Guyon et al.'s SVM-RFE [15] in a subcellular localization benchmark and show that the VIA-SVM not only avoids setting a cutoff point but also is insensitive to the penalty factor in SVM training.

1.5.1 Pairwise Scoring Kernels

Denote $\mathcal{D} = \{S^{(1)}, \ldots, S^{(T)}\}$ as a training set containing T protein sequences. Let us further denote the operation of PSI-BLAST[5] search given the query sequence $S^{(i)}$ as

$$\phi^{(i)} \equiv \phi(S^{(i)}) : S^{(i)} \rightarrow \{\mathbf{P}^{(i)}, \mathbf{Q}^{(i)}\},$$

where $\mathbf{P}^{(i)}$ and $\mathbf{Q}^{(i)}$ are the PSSM and PSFM of $S^{(i)}$, respectively.[6] Because these matrices are based on the information of a large number of sequences that are similar to the query sequence, they contain rich information about the remote homolog of the query sequence, which may help improve the prediction of subcellular locations and protein functions. Given the profiles of two sequences $S^{(i)}$ and $S^{(j)}$, we can apply the Smith–Waterman algorithm [23] and its affine gap extension [24] to align $\mathbf{P}^{(i)}$, $\mathbf{Q}^{(i)}$, $\mathbf{P}^{(j)}$, and $\mathbf{Q}^{(j)}$ to obtain the normalized profile alignment score $\zeta(\phi^{(i)}, \phi^{(j)})$.[7]

The scores $\{\zeta(\phi^{(i)}, \phi^{(j)})\}_{i,j=1}^{T}$ constitute a symmetric matrix $\mathbf{Z}$ whose columns can be considered as T-dimensional vectors:

$$\boldsymbol{\zeta}^{(j)} = [\boldsymbol{\zeta}(\phi^{(1)}, \phi^{(j)}) \cdots \zeta(\phi^{(T)}, \phi^{(j)})]^{\mathrm{T}}, \quad j = 1, \ldots, T. \tag{1.14}$$

An M-class protein prediction problem can now be solved by M one-versus-rest SVMs:

$$f_m(S) = \sum\nolimits_{j \in S_m} y_{m,j}\alpha_{m,j} K(\phi(S)),\ \phi(S^{(j)})) + b_m, \tag{1.15}$$

where S is an unknown sequence, $m = 1, \ldots, M$, $y_{m,j} \in \{+1, -1\}$, S_m contains the indexes of support vectors, $\alpha_{m,\,j}$ are Lagrange multipliers, and

$$K(\phi(S)), \phi(S^{(j)}) = g(\boldsymbol{\zeta}, \boldsymbol{\zeta}^{(j)})$$

is a kernel function.

Now we may consider the columns of a pairwise scoring matrix as high-dimensional vectors. This means that there are T feature vectors with dimension equal to the training

[5]To efficiently produce the profile of a protein sequence (called query sequence), the sequence is used as a seed to search and align homologous sequences from protein databases such as SWISSPROT [21] using the PSI-BLAST program [22].

[6]The homolog information pertaining to the aligned sequences is represented by two matrices (profiles): position-specific scoring matrix (PSSM) and position-specific frequency matrix (PSFM). Both PSSM and PSFM have 20 rows and L columns, where L is the number of amino acids in the query sequence.

[7]See http://www.eie.polyu.edu.hk/~mwmak/BSIG/PairProSVM.htm.

set size. The T T-dimensional column vectors can be used to train M SVMs. Because of the high dimensionality, linear SVM is a preferred choice, that is, $g(\boldsymbol{\zeta}, \boldsymbol{\zeta}^{(j)}) = \langle \boldsymbol{\zeta}, \boldsymbol{\zeta}^{(j)} \rangle$. The class of S can then be obtained by $y(S) = \arg\max_{m=1}^{M} f_m(S)$, where M is the number of classes.

1.5.2 VIA-SVM Approach to Pairwise Scoring Kernels

The pairwise approach always results in feature vectors with extremely high dimensions. This creates a problem known as the curse of dimensionality. An obvious solution is to reduce the feature size and yet retaining the most important information critical for classification. The challenge thus lies in how to effectively determine those relevant features. The approaches mentioned in Section 1.4 do not make use of the symmetric property of the pairwise scoring matrices in the selection process (the symmetric property is illustrated in Fig. 1.16), because they are designed for general cases. In fact, they are primarily designed for gene selections in microarray data where expression matrices are neither square nor symmetric. However, it can be advantageous to adopt a feature selection method that is tailor designed for the pairwise scoring vectors.

To design a feature selection algorithm for pairwise scoring vectors, we need to exploit the reflexive property of pairwise scoring matrices. The idea is based on the notion that support vectors are important for classification and pairwise scoring matrices are symmetric. (Namely, the elements of the ith column of $\mathbf{Z}$ are identical to those in the ith row.) This suggests a possible hypothesis:

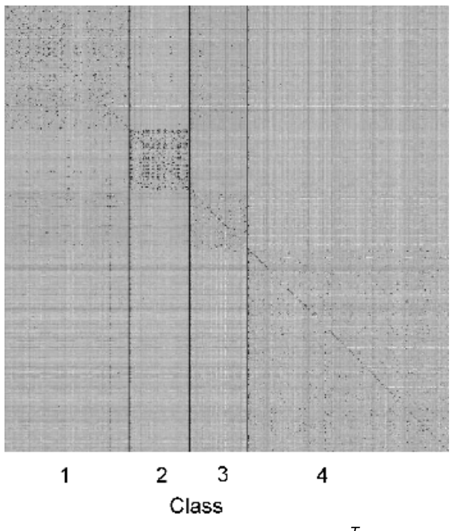


Figure 1.16 Profile alignment score matrix $\mathbf{Z} = \{\zeta(\phi^{(i)}, \phi^{(j)})\}_{i,j=1}^{T}$. The training vectors have been prearranged such that the vectors belonging to the same class are all grouped together, that is, they are consecutively indexed. The three vertical lines were artificially added to divide the column vectors into four classes.

Hypothesis. *The support vector indexes are good candidates for selecting features for the column vectors, that is, only the rows corresponding to the support vectors are retained.*

Heuristically, due to the symmetry property, if the jth vector is a critical (supporting) vector for the decision boundary, then the jth feature would also be a critical feature and therefore should be selected. We refer to this selection scheme as vector-index-adaptive SVM, or simply VIA-SAM. In fact, our simulation results also strongly support this hypothesis.

1.5.2.1 Why Consider Only Support Vectors?

In VIA-SVM, the support vector indexes are reused as feature selection indexes. The use of support vectors to select relevant features is intuitively appealing because they are "critical" for establishing the decision boundary of SVM classifiers. Because of the symmetrical property of kernel matrices, the elements of the ith column of $\mathbf{Z}$ are identical to those in the ith row. If the ith column of $\mathbf{Z}$ happens to be a support vector, the corresponding feature dimension (the ith row of $\mathbf{Z}$) will also be critical for classification. However, nonsupport vectors are irrelevant for classification, so are their corresponding feature dimensions.

The diagonal dominance implies that a large value of α_i in Equation 1.34 is likely to lead to a large value of w_i. By the same token, the nonsupport vectors are those that correspond to $\alpha_i = 0$, and therefore their corresponding weight values w_i are more likely to be smaller. Therefore, only those features corresponding to $\alpha_i > 0$ are considered for selection. Those corresponding to $\alpha_i = 0$ will be eliminated automatically. This concept is further elaborated in Fig. 1.17.

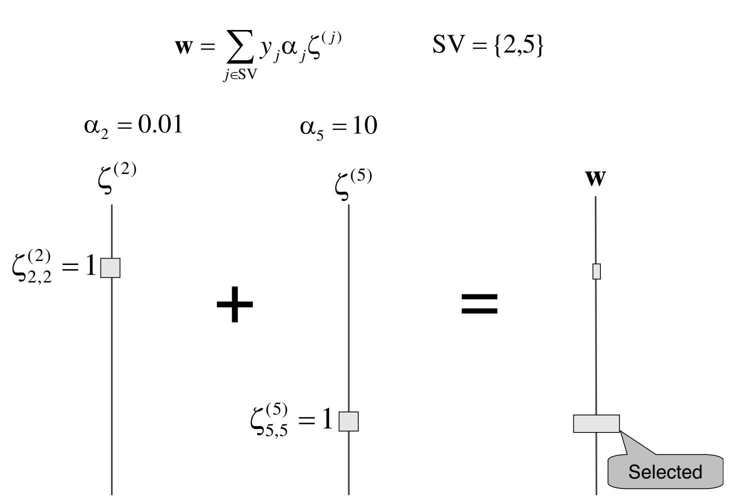


Figure 1.17 Diagram illustrating how VIA-SVM uses the values of Lagrange multipliers α_j to select features. The vertical bars represent support vectors $\zeta^{(j)}$ and the weight vector **w**, and the magnitude of the vector components are proportional to the thickness along the bars. It is assumed that the matrix **Z** has been normalized such that all diagonal elements are equal to 1.

The above interpretation of VIA-SVM is consistent with how SVM-RFE selects features in that, in both methods, indexes with large weight will be chosen first. Moreover, they both prune the vectors/features corresponding to zero α_i. However, there is also an important difference, which lies in the treatment of the vectors/features corresponding to nonzero α_i. More exactly, in VIA-SVM, different types of support vectors receive different levels of preferences.

1.5.2.2 Differential Treatments of Support Vectors Because the SVM-RFE takes the overall weight vector **w** into account, it only considers the Lagrange multipliers α_i but not the slack variables ξ_i. In contrast, the VIA-SVM considers both α_i (related to SV) and ξ_i (indicates safety margin or, sometimes, outlier). In this sense, the VIA-SVM offers a more comprehensive coverage of all the critical factors made available by the SVM classifier.

It is important to recognize the fact that *not all SVs are created equal*. Therefore, in the VIA-SVM, support vectors are differentially treated. In fact, they are divided into four levels of preferences as specified by the four regions in Fig. 1.18:

Level 1 *Most-Preferred.* The SV is on the margin, that is, $0 < \alpha_i < C$ and $\xi_i = 0$, where C is the penalty factor in SVM training.

Level 2 *Preferred.* The SV is in the fuzzy region and on the correct side of the decision boundary, that is, $\alpha_i = C$ and $0 < \xi_i < 1$.

Level 3 *Marginally Preferred.* The SV is in the fuzzy region but on the wrong side of the decision boundary, that is, $\alpha_i = C$ and $1 \leq \xi_i < 2$.

Level 4 *Nonpreferred.* The SV is regarded as an outlier, that is, $\alpha_i = C$ and $\xi_i \geq 2$.

The reason of ruling out the outlier SVs is self-explanatory. The decision to have the marginal support vectors assigned the highest preference level can be justified on the basis that they offer relatively higher confidence than the fuzzy SVs.

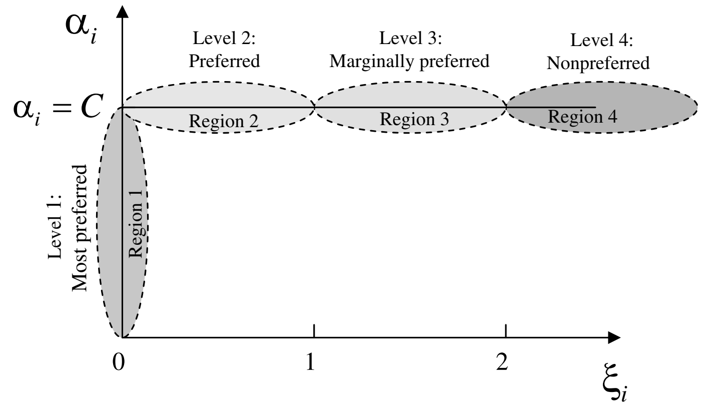


Figure 1.18 The four levels of preferences for the support vectors in VIA-SVM. Regions 1–4 correspond to preference levels 1–4, respectively.

Although it appears that two parameters (α_i and ξ_i) are required to define the preferences, ξ_i alone can already provide sufficient information to determine the preference level of an SV.[8] In fact, the preference decreases with increasing ξ_i. More exactly, $\xi_i = 0$, $0 < \xi_i < 1$, $1 \leq \xi_i < 2$, and $2 \leq \xi_i < \infty$ define Level 1 to Level 4, respectively (cf. Fig. 1.18).

1.5.3 The VIA-SVM Algorithm

Feature selection in VIA-SVM is divided into two steps:

Step 1 The score matrix $\mathbf{Z} = \{\boldsymbol{\zeta}(\phi^{(i)}, \phi^{(j)})\}$ is used to train M SVMs (Eq. 1.15) from which M sets of support vector indexes S_m are determined. This results in a set of support vectors $\zeta^{(j)} = [\zeta(\phi^{(1)}, \phi^{(j)}) \cdots \zeta(\phi^{(T)}, \phi^{(j)})]^{\mathrm{T}}$ for each class, where $j \in S_m$.

Step 2 For the mth class, the indexes in S_m are used to select the feature dimensions (rows of $\mathbf{Z}$) of the column vectors to obtain vectors $\zeta'^{(j)}$ of reduced dimensions, where $j = 1, \ldots, T$. These vectors are then used to train another SVM for classification. This process is repeated for all classes.

These two steps are iterated N times (five times in this work). Specifically, the features selected at the nth iteration are used to train a new SVM in the $(n+1)$th iteration, whose support vectors are subsequently used for determining the feature set in the $(n+2)$th iteration, and so on. The classification accuracy of the training data at each iteration is recorded. At the end of the Nth iteration, the support vectors of the SVM with the highest training accuracy are used for selecting the final set of features. The column vectors with reduced dimensions are then used to train another SVM for classification. Figure 1.19 shows the pseudocode of VIA-SVM.

1.5.3.1 Level-Dependent VIA-SVM Selection Strategies For the actual implementation, it is worth noting that *not* all SVs are created equal. Here, we propose four level-dependent VIA-SVM selection strategies:

Strategy 1 (*Level 1 only*) Select the most-preferred SVs, that is, select only the "pure" marginal SVs ($\alpha_i < C$) while excluding those fuzzy and outlier SVs ($\alpha_i = C$).

Strategy 2 (*Levels 1 and 2*) Select the correctly classified SVs only, that is, select the "pure" marginal SVs ($\alpha_i < C$) and the correctly classified SVs that are falling on the fuzzy region ($\alpha_i = C$ and $0 < \xi_i < 1$).

Strategy 3 (*Levels 1–3*) Remove the nonpreferred outlier SVs, that is, only keep those SVs with $\xi_i < 2$.

[8]Under some rare situations in which extremely small numbers of features are desirable, we may use α_i to further divide Level 1 into sublevels. However, this subdivision is unlikely to be useful in bioinformatic applications.

```
Algorithm VIA-SVM
Input:   $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \cdots \ \mathbf{x}_T], \mathbf{y} = [y_1 \ y_2 \ \cdots \ y_T]$,
         and $C \in \Re$, where $\mathbf{x}_t \in \Re^T$, $y \in \{+1,-1\}$
Initialization: $\mathbf{X}' = \mathbf{X}$;
for k = 1 to N
do
        // Train an SVM to obtain the indexes to the support vectors in $\mathbf{i}$
        $[\boldsymbol{\alpha}, b, \mathbf{i}]$ = SVM_train($\mathbf{X}'$, $\mathbf{y}$, $C$);

        // Find the support vector indexes $\mathbf{j}$ such that $0 < \alpha_j \le C$ and $0 \le \xi_j < 2$
        // where $\xi_j = 1 - \alpha_j(\mathbf{x}_j \cdot \mathbf{w} + b)$ and $\mathbf{w} = \sum_{i \in \mathbf{i}} \alpha_i y_i \mathbf{x}_i$
        $\mathbf{j}${k} = find_sv_index($\mathbf{X}'$, $\mathbf{y}$, $C$, $\boldsymbol{\alpha}$, $b$, $\mathbf{i}$);

        // Select a feature subset based on the selected support vector indexes
        $\mathbf{X}'$ = X($\mathbf{j}${k}, :);

        // Train another SVM based on the selected features
        $[\boldsymbol{\alpha}, b, \mathbf{i}]$ = SVM_train($\mathbf{X}'$, $\mathbf{y}$, $C$);

        // Computing classification accuracy on training data
        a(k) = SVM_test($\mathbf{X}'$, $\mathbf{y}$, $\boldsymbol{\alpha}$, $b$, $\mathbf{i}$);
end
Output:  $\mathbf{j}\{k'\}$ where $k' = \arg\max_k a(k)$
```

Figure 1.19 Pseudocode of VIA-SVM.

Strategy 4 (*Levels 1–4*) Select ALL SVs, that is, select all marginal and fuzzy SVs with $0 < \alpha_i \le C$.

Experiments on subcellular localization support that Strategies 2 and 3 appear to produce the best performance.

1.5.3.2 Comparing VIA-SVM and SVM-RFE Although both VIA-SVM and SVM-RFE are based on SVMs, they do have an important difference in terms of information used. Figure 1.20 illustrates the information used by VIA-SVM and SVM-RFE in selecting and ranking features. Clearly, SVM-RFE uses the weight

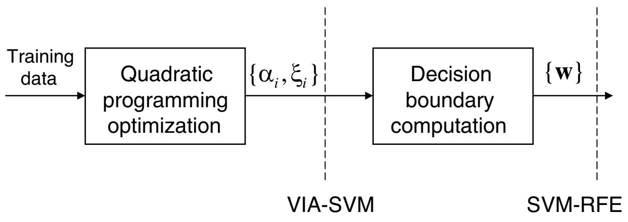

Training data
Quadratic programming optimization
{α_i, ξ_i}
Decision boundary computation
{w}
VIA-SVM
SVM-RFE

Figure 1.20 Algorithmic difference between VIA-SVM and SVM-RFE. The former uses the Lagrange multipliers α_i and slack variables ξ_i, whereas the latter uses the weight vector **w**.

vector **w**, whereas VIA-SVM uses the Lagrange multipliers α_i and slack variables ξ_i. In terms of the usefulness of the respective parameters, there are key similarity and difference:

- *Similarity.* Because of the symmetric property and diagonal dominance of the scoring matrix **Z**, a large α_i will lead to a large w_i, as exemplified in Fig. 1.17. Therefore, when all training samples are linearly separable, that is, $\xi_i = 0\ \forall i$, VIA-SVM and SVM-RFE share a similar feature selection strategy. Such a similarity will play a role when α_i have different values (cf. Region 1 in Fig. 1.18). However, such situation can only occur in the ideal, linearly separable case.
- *Difference.* In reality, most training samples are not linearly separable even in the kernel space. In such cases, many support vectors satisfy $\alpha_i = C$ and $\xi_i > 0$, that is, they belong to Regions 2–4 in Fig. 1.18. The problem is that SVM-RFE offers no means to directly differentiate the three regions; particularly, it cannot discriminate whether the support vectors are outliers or not, because they all share the same constant $\alpha_i = C$. In contrast, VIA-SVM is sensitive to the value of $\xi_i > 0$. In particular, the VIA scheme uses ξ_i to divide support vectors into three levels of preferences (Levels 2–4 in Fig. 1.18) when $\alpha_i = C$. The uses of ξ_i to select support vectors (features) is intuitive appealing because when $\xi_i > 0$, α_i—which is a constant—can no longer provide information regarding the degree of relevance of a feature. This additional information provided by ξ_i plays an important role in selecting more vital and "relevant" features and disregarding the "misleading" (i.e., outlier) ones.

1.5.4 Combined with Other Selection Schemes

1.5.4.1 Redundance Removal for VIA-SVM A common weakness of both VIA and RFE approaches is that they do not explicitly consider the redundancy factor. This will result in wasteful selection because some of the selected features can be either repetition of each other or highly redundant. To minimize feature redundancy, we propose two pruning methods for VIA-SVM.

1. *Euclidean Distance.* For those support vectors in Region 2 or 3, their pairwise Euclidean distances (eDist) in the reduced kernel space (defined by the selected feature dimensions) are examined. If two features (with similar ξ_i) are close to each other, one of them may be removed without affecting the decision boundary. The degree of closeness can be defined by multiplying the average distance of all support vectors by a user-defined constant η. More specifically, we remove feature j if $d(\zeta_{\mathrm{r}}^{(i)}, \zeta_{\mathrm{r}}^{(j)}) < \eta\bar{d}$, where $\bar{d}$ represents the average distance and the subscript r signifies that the distance is evaluated in the reduced kernel space.
2. *K-Means.* Support vectors that have similar ξ_i in Region 2 or 3 are clustered by K-means in the reduced kernel space. The SVs closest to the centers are retained and all remaining SVs in Regions 2 and 3 are removed.

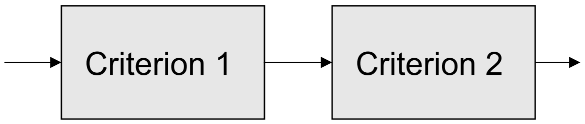


Figure 1.21 In the cascade fusion architecture, Criterion 1 is meant for coarse and fast overselection and Criterion 2 represents pruning. The threshold for Criterion 1 can be more relaxed (i.e., its value does not have to be very close to the optimal number of features) to include more candidate features for the second stage.

If the allowable number of features is very small, the redundance-reduced feature set can be further pruned by a filter method such as symmetric divergence. Results of cascading VIA-SVM, redundance removal, and feature pruning will be shown in Section 1.5.5.

Note that while SVM-RFE can also apply a postprocessing redundancy removal selection process (such as applying K-means or double checking the similarity score in the kernel matrix), it does not enjoy the numeric information provided by ξ_i.

1.5.4.2 Fusion of Selection Criteria

It is natural to combine/consider both the filter and wrapper methods in order to reach an optimal strategy. Specifically, the features and/or criterion functions obtained from the filter and wrapper methods can be combined via a cascaded fusion scheme shown in Fig. 1.21.

It is desirable to seek an optimal compromise between accuracy and cost. A possibility is via a comprehensive selection procedure comprising two consecutive phases. Here, we refer to these procedures as overselect-and-prune strategy. The strategy has two steps.

Step 1 *Overselection Phase.* This is a stage in which only those obviously irrelevant features are quickly weeded out. Preferably, it involves a quick and coarse (suboptimal) evaluation, for example, individual ranking. The goal is to have most unwanted features filtered out while guarantee a retention rate of the eligible features. This phase can be implemented via Criterion 1 of a cascade architecture shown in Fig. 1.21.

Step 2 *Pruning Phase.* The second stage may serve as a fine-tuning process. The goal is to remove redundant features with minimum information loss. If major redundancy exists between A and B, then one of the two may be pruned without incurring much loss of information.[9]

1.5.5 Experiments on Subcellular Localization

Two data sets were used for evaluating the performance of VIA-SVM and for comparing it against other feature selection algorithms. The first data set is provided by Reinhardt and Hubbard [3]. It comprises 2427 amino acid sequences extracted from

[9]The second stage may also serve as a second opinion. This can be implemented by a cascaded architecture (Fig. 1.21) with two complementary criteria applied in series. More precisely, a feature can be retained only if it has good scores in both criteria.

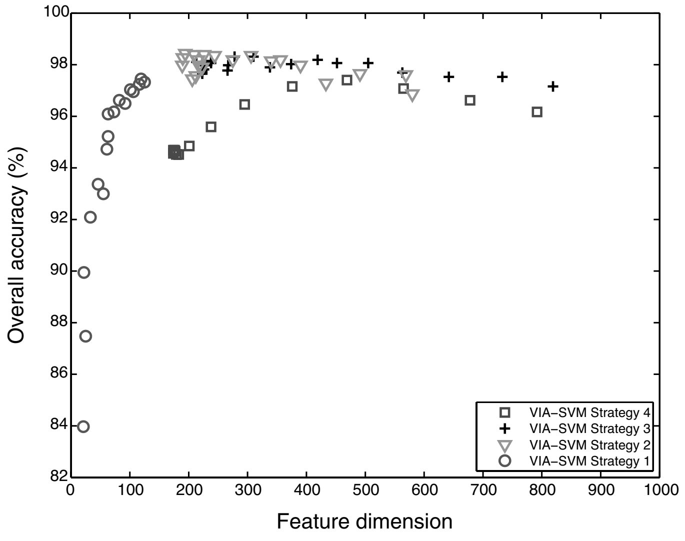
SWISSPROT 3.3, with each protein annotated with one of the four subcellular locations: cytoplasm, extracellular, mitochondrial, and nuclear. The second data set was provided by Huang and Li [25]. It was created by selecting all eukaryotic proteins with annotated subcellular locations from SWISSPROT 41.0 and by setting the identity cutoff to 50%. The data set comprises 3572 proteins (622 cytoplasm, 1188 nuclear, 424 mitochondrial, 915 extracellular, 26 golgi apparatus, 225 chloroplast, 45 endoplasmic reticulum, 7 cytoskeleton, 29 vacuole, 47 peroxisome, and 44 lysosome). We used fivefold cross-validation for performance evaluation so that every sequence in the data sets will be tested.

1.5.5.1 Performance of VIA-SVM Now let us discuss the case study results based on the four selection strategies mentioned in Section 1.5.3.1. Because Strategy 1 includes only very few SVs, the features are extremely underselected. In contrast, because Strategy 4 includes all SVs regardless of their types, it is likely to cause overselection, particularly when the penalty factor C is small. In Strategy 2, all the SVs that are incorrectly misclassified will be excluded. However, this may lead to underselection as there are some useful SVs falling on the fuzzy regions. Strategy 3 is a compromise between the overselection in Strategy 2 and the underselection in Strategy 4. More exactly, Strategy 3 excludes all the outlier SVs. (The SVs lying beyond the margin of the opposite class are deemed to be outliers.) In this strategy misclassified SVs that lie within the margin of separation will still be selected, leading to overselection, especially when the penalty factor C is very small.

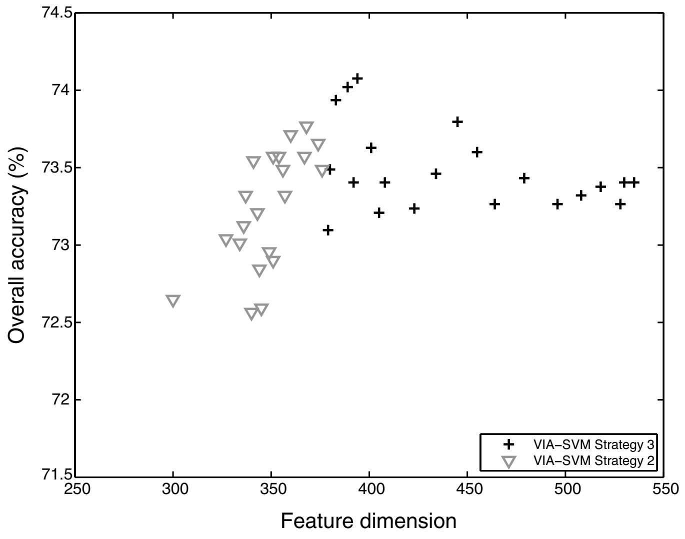
Figure 1.22 shows the performance of Strategies 1–4 when the penalty factor C varies from 0.004 to 4096. Note that the number of selected features (feature dimension) is automatically determined by the SVMs. For each strategy, a smaller penalty factor C will generally lead to a larger number of features, and vice versa for a larger C. Therefore, markers on the right region of the figure correspond mainly to small C's. The results show that the optimal number of features found by Strategy 4 is considerably higher than those found by the other strategies. Notwithstanding the larger number of features, the maximum accuracy attained by Strategy 4 is still lower than those achieved by the other strategies. This confirms our earlier hypothesis that including all SVs will lead to overselection. Results also show that Strategy 1 will lead to underselection when the penalty factor C becomes large. These case studies suggest that Strategies 2 and 3, which exclude either the nonpreferred SVs or both the marginally preferred and nonpreferred SVs (cf. Fig. 1.18), seem to be the least sensitive to the penalty factor, because they can keep the number of features within a small range and maintain the accuracy at a constant level for a wide range of C.

1.5.5.2 Comparison Between VIA-SVM and SVM-RFE We compared the proposed VIA-SVM (Strategy 2) with SVM-RFE [15] in the subcellular localization benchmarks mentioned earlier.[10] Note that SVM-RFE does not make use of the

[10]Because the performances of SVM-RFE, R-SVM, and symmetric divergence are comparable in the two benchmarks, we only report the results of SVM-RFE for clarity of presentation.



(a)



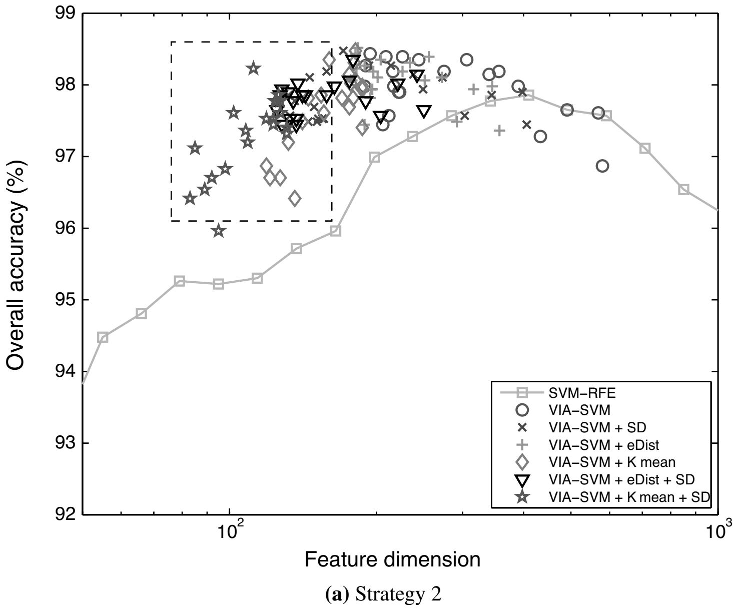
(b)

Figure 1.22 Prediction performance of different strategies of VIA-SVM on (a) Reinhardt and Hubbard's data set and (b) Huang and Li's data set when the penalty factor *C* varies from 0.004 to 4096. See Section 1.5.5 for details of the strategies.

symmetric property of the pairwise scoring matrices in the selection process, because it is primarily designed for gene selections in microarray data where expression matrices are neither square nor symmetric.

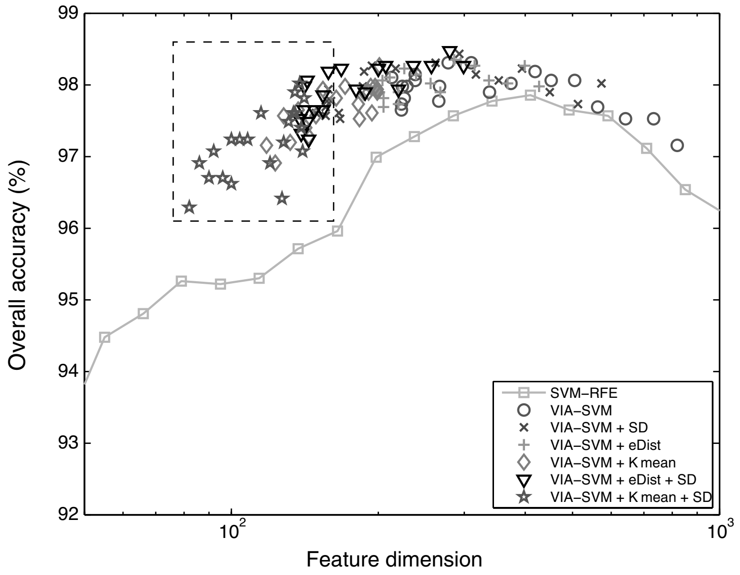
Figures 1.23 and 1.24 show the performance of SVM-RFE (–□– and VIA-SVM (○). Evidently, VIA-SVM is superior to SVM-RFE in two aspects: (1) It outperforms SVM-RFE at almost all feature dimensions, particularly at low feature dimensions, and (2) it automatically bounds the number of selected features within a small range. A drawback of SVM-RFE is that it requires a cutoff point for stopping the selection. However, VIA-SVM is insensitive to the penalty factor in SVM training and can avoid the need to set a cutoff point for stopping the feature selection process.

1.5.5.3 Fusion of VIA-SVM and SD Given a particular axis, which corresponds to one feature, two factors can be considered: VIA-SVM parameters (C and ξ_i) and symmetric divergence of feature i (SD_i). We adopt the overselect-and-prune cascaded fusion architecture, as proposed in Section 1.5.4.2, to combine VIA-SVM and SD. Based on this cascade fusion strategy, the selection process is divided into two stages.



(a) Strategy 2

Figure 1.23 Prediction performance of SVM-RFE, VIA-SVM, and VIA-SVM cascaded with various pruning methods on Reinhardt and Hubbard's data set. (a) The VIA-SVM uses Strategy 2 for feature selection. (b) Strategy 3 was used. (c) The means and standard derivations (in parentheses) of the classification accuracies and feature dimensions for 21 penalty factors ranging from 0.004 to 4096; for SVM-RFE in (c), the means and standard derivations are based on 11 points in (a) and (b) whose feature dimensions range from 95 to 591. Pruning was applied according to the order indicated in the legend; for example, "VIA-SVM + eDist + SD" means that the features selected by VIA-SVM were pruned by eDis-based method followed by the symmetric divergence-based method.

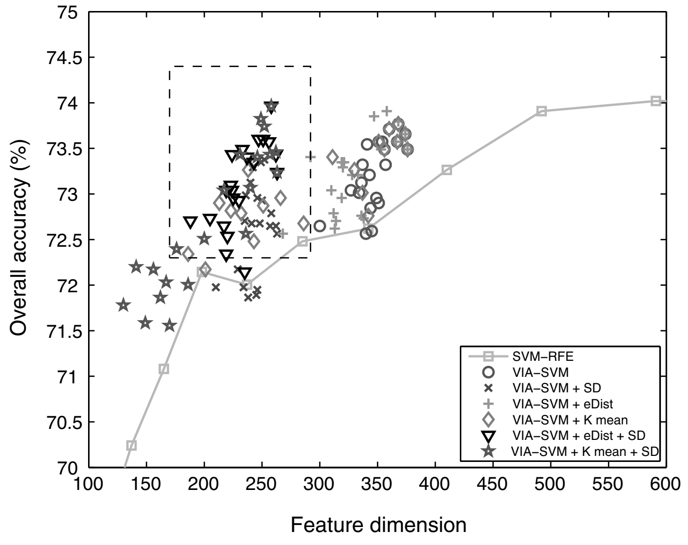


(b) Strategy 3

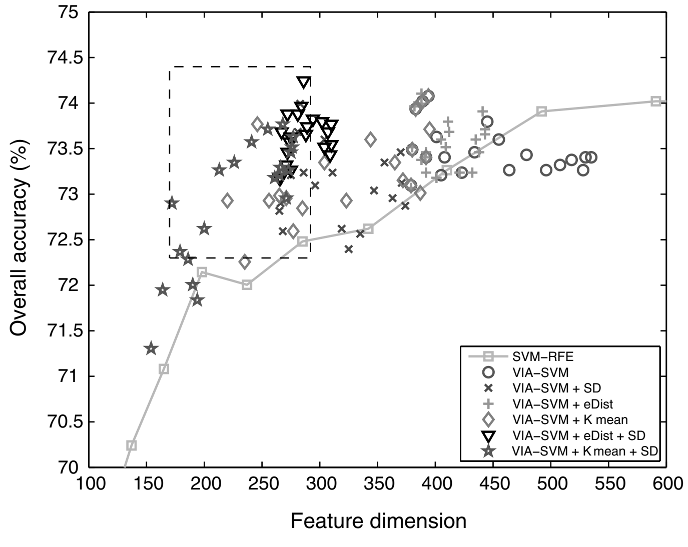
	Accuracy (%)		Feature Dimension	
Method	Strategy 2	Strategy 3	Strategy 2	Strategy 3
VIA-SVM	97.96 (0.42)	97.90 (0.28)	299 (125.20)	370 (183.18)
VIA-SVM + SD	97.84 (0.32)	97.96 (0.28)	210 (87.60)	259 (128.18)
VIA-SVM + eDist	98.05 (0.32)	98.03 (0.19)	230 (56.02)	261 (71.00)
VIA-SVM + Kmean	97.65 (0.58)	97.68 (0.34)	161 (23.47)	166 (28.81)
VIA-SVM + eDist+SD	97.82 (0.24)	97.92 (0.35)	161 (39.16)	183 (49.69)
VIA-SVM + Kmean + SD	97.30 (0.55)	97.19 (0.48)	112 (16.49)	116 (20.24)
SVM-RFE	96.81 (1.04)		279 (163.23)	

(c) Accuracy and feature dimension

Figure 1.23 (*Continued*).



(a) Strategy2



(b) Strategy3

Figure 1.24 Same as Fig. 1.23 but based on Huang and Li's data set. For SVM-RFE in (c), the means and standard derivations are based on nine points in (a) and (b) whose feature dimensions range from 114 to 492.

	Accuracy (%)		Feature Dimension	
Method	Strategy 2	Strategy 3	Strategy 2	Strategy 3
VIA-SVM	73.23 (0.38)	73.48 (0.27)	348 (17.22)	445 (55.77)
VIA-SVM + SD	72.66 (0.51)	73.12 (0.41)	244 (12.07)	312 (39.01)
VIA-SVM + eDist	73.23 (0.42)	73.56 (0.29)	332 (27.77)	408 (22.69)
VIA-SVM + Kmean	73.09 (0.47)	73.26 (0.48)	296 (65.29)	327 (61.84)
VIA-SVM + eDis + SD	73.11 (0.47)	73.63 (0.27)	232 (19.35)	286 (15.83)
VIA-SVM + Kmean + SD	72.73 (0.78)	72.93 (0.71)	207 (45.76)	229 (43.39)
SVM-RFE	71.90 (1.45)		264 (129.10)	

(c) Accuracy and feature dimension

Figure 1.24 (*Continued*).

•

Stage 1 Use VIA-SVM (Strategy 2 or 3) to select all-but-outlier SVs, that is, only keep those with $\xi_i < 2$.

Stage 2 Use SD to sort the features found in Stage 1 and keep the most relevant $x\%$.

In this work, we set x to 70. Figures 1.23 and 1.24 show the fusion results ($\times$), which suggest that fusion can produce more compact feature subsets without significant reduction in prediction accuracy. We also note that although VIA-SVM is inferior to SVM-RFE for large feature-set size, the combination of SD and VIA-SVM performs better at small feature-set size.

1.5.5.4 Redundance Removal The eDist- and K-means-based methods mentioned in Section 1.5.4.1 were applied to reduce the redundance among the features selected by VIA-SVM. For the former, the constant η was set to 0.3; for the latter, the number of centers in *K*-means was set to 100 for Reinhardt and Hubbard's data set and 300 for Huang and Li's data set. The setting of these values was based on the observation from Fig. 1.22 that the accuracy drops rapidly when the number of features is smaller than these values. The results ($+$ and $\diamond$) shown in Figs 1.23 and 1.24 suggest that both methods can reduce the feature size without scarifying accuracy. This once again demonstrates the merit of using the slack variables ξ_i in selecting features.

To further reduce the feature size, we applied SD to prune the features after redundance removal. Figures 1.23 and 1.24 ($\bigtriangledown$ and $\star$) show that the resulting feature sets achieve a significantly higher accuracy when compared with SVM-RFE.

The tables in Figs 1.23c and 1.24c compare Strategy 2 and Strategy 3 in terms of the mean accuracy and mean feature dimension obtained by VIA-SVM and VIA-SVM cascaded with various pruning methods. Three observations can be obtained from these tables:

1. For each pruning method, there is no significant difference between the accuracy obtained by Strategies 2 and 3.
2. Strategy 2 generally leads to a smaller number of features than Strategy 3 with almost the same performance statistically.
3. SVM-RFE not only gives lower average accuracies but also leads to a larger variation in both accuracy and feature dimension.

These observations suggest that Strategy 2 is a winner because it can keep the number of features to a minimum without scarifying accuracy. Strategy 2 is also a better choice for applications where feature dimension (and hence recognition time) should be kept to a minimum. However, for applications where accuracy is of primary importance, Strategy 3 is a better option.

1.5.5.5 Range of Desirable Feature Dimension In most pattern recognition problems, a smaller feature size can result in faster recognition speed, but at the expense of lower classification accuracy. The opposite situation occurs for large (but not excessively large) feature size. We advocate that there is a range of desirable feature dimension for a particular problem and that whether the lower or upper limit of the range should be used depends on the applications. For example, in biometric applications where real-time recognition is essential, we may opt for minimum feature size that produces the best compromise between recognition speed and accuracy. However, in bioinformatic applications where accuracy is far more important than speed, we may prefer using the upper limit that gives high accuracy but not to the point that causes the curse of dimensionality. This notion of desirable range of feature dimension is highlighted by the dashed rectangles in Figs 1.23 and 1.24. Evidently, the redundance removal and the cascade fusion of VIA-SVM and SD enable us to find the feature sizes falling on the desirable range. In particular, when recognition speed is a concern, we can overselect features by using VIA-SVM (Strategy 2 or 3) and then remove feature redundance by using K-means, and finally, we can further prune the features by using SD ("VIA-SVM + Kmean + SD"). On the other hand, if accuracy is more important, we may skip the final pruning stage, that is, using "VIA-SVM + Kmean", or do not use pruning at all.

1.6 CONCLUSION

This chapter discusses the feature selection methods from many different aspects, including selection criteria, the order of feature selection, and open-loop (filter-type) and closed-loop (wrapper-type) approaches.

A special method (VIA-SVM) designed exclusively for pairwise scoring kernels is introduced. This is the first method that fully utilizes the reflexive property uniquely available in this scenario. Based on several subcellular localization experiments, the VIA-SVM when combined with some filter-type metrics appears to reach a significantly dimension reduction (one-order of magnitude) while conceding minimum sacrifice of accuracy.

It is well known that fusion of complementary criteria or methods can substantially enhance classification performance. While this chapter has offered several exemplar cases of successful fusion, there are still more variants of fusion techniques that need to be pursued. This appears to be a promising direction worth further investigation.

ACKNOWLEDGMENTS

This work was in part supported by The Research Grant Council of the Hong Kong SAR (Project Nos. PolyU5241/07E and PolyU5251/08E.)

REFERENCES

1. Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3: 1157–1182, 2003.
2. Mak, M. W. Guo, J. and Kung, S. Y. PairProSVM: Protein subcellular localization based on local pairwise profile alignment and SVM. *IEEE/ACM Transaction on Computational Biology and Bioinformatics*, 5 (3): 416–422, 2008.
3. Reinhardt, A. and Hubbard, T. Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Research*, 26: 2230–2236, 1998.
4. Mak, M. W. and Kung, S. Y. A solution to the curse of dimensionality problem in pairwise scoring techniques. *International Conference on Neural Information Processing*, pp. 314–323, 2006.
5. Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286: 531–537, 1999.
6. Hyvarinen, A. and Oja, E. Independent component analysis: algorithms and applications. *Neural Networks*, 13: 411–430, 2000.
7. Pan, W. A comparative review of statistical methods for discovering differentially expressed genes in replicated microarray experiments. *Bioinformatics*, 18: 546–554, 2002.
8. Pavlidis, P., Weston, J., Cai, J., and Grundy, W. N. Gene functional classification from heterogeneous data. *International Conference on Computational Biology, Pittsburgh, PA*, pp. 249–255, 2001.
9. Furey, T. S., Cristianini, N., Duffy, N., Bednarski, D. W., Schummer, M., and Haussler, D. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16: 906–914, 2000.

10. Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization of the brain. *Psychology Review*, 65: 42–99, 1958.
11. Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7: 179–188, 1936.
12. Vapnik, V. N. *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.
13. Boser, B. E., Guyon, I. M., and Vapnik, V. N. A training algorithm for optimal margin classifiers. In: D. Haussler, Ed., *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152, 1992.
14. Kohavi, R. and John, G. H. Wrappers for feature selection. *Artificial Intelligence*, 97(1–2): 273–324, 1997.
15. Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46: 389422, 2002.
16. Duan, K.-B., Rajapakse, J. C., Wang, H., and Azuaje, F. Multiple SVM-RFE for gene selection in cancer classification witn expression data. *IEEE Transactions on Nanobioscience*, 4(3): 228–234, 2005.
17. Zhang, X. G., Lu, X., Shi, Q., Xu, X. Q., Leung, H. C. E., Harris, L. N., Iglehart, J. D., Miron, A., Liu, J. S., and Wong, W. H. Recursive SVM feature selection and sample classification for mass-spectrometry and microarray data. *BMC Bioinformatics*, 7(197): 2006.
18. Kira, K. and Rendell, L. A practical approach to feature selection. *International Conference on Machine Learning, Aberdeen*, pp. 368–377, July 1992.
19. Liao, L. and Noble, W. S. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *Journal of Computational Biology*, 10(6): 857–868, 2003.
20. Kim, J. K., Raghava, G. P. S., Bang, S. Y., and Choi, S. Prediction of subcellular localization of proteins using pairwise sequence alignment and support vector machine. *Pattern Recognition Letters*, 27(9): 996–1001, 2006.
21. http://www.expasy.org/sprot.
22. Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25: 3389–3402, 1997.
23. Smith, T. F. and Waterman, M. S. Comparison of biosequences. *Advances in Applied Mathematics*, 2: 482–489, 1981.
24. Gotoh, O. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162: 705–708, 1982.
25. Huang, Y. and Li, Y. D. Prediction of protein subcellular locations using fuzzy K-NN method. *Bioinformatics*, 20(1): 21–28, 2004.
26. Kung, S. Y., and Mak, M. W. Feature selection for self-supervised classification with applications to microarray and sequence Data. *IEEE Journal of Selected Topics in Signal Processing, Special Issue on Genomic and Proteomic Signal Processing*, 2 (3): 297–309, 2008.