9. Feature Selection for Dimensionality Reduction and Classification

9.1 Introduction

In previous parts, we used all features available in the design of the classifier. A problem associated with pattern recognition is the so-called curse of dimensionality, which refers to the problems that arise when working with high-dimensional data. The dimension of a dataset corresponds to the number of features that exist in a dataset. The difficulties related to training machine learning models due to high dimensional data is referred to as 'curse of dimensionality'. The main aspects of the curse of dimensionality are data sparsity and distance concentration.

There are more than one reason to reduce the number of features to a sufficient minimum. Computational complexity is the obvious one. Another major reason is the generalization capacity of the classifier. Usually, the higher the ratio of the number of training samples to the number of free classifier parameters, the better the generalization properties of the resulting classifier.

A large number of features are directly translated into a large number of classifier parameters (e.g., synaptic weights in a neural network, weights in a linear classifier). Thus, for a finite and usually limited number of training samples, keeping the number of features as small as possible is in line with our desire to design classifiers with good generalization capabilities.

The major task of this part can now be summarized as follows:

Given a number of features, how can we select the most important ones so as to reduce their number and at the same time retain as much as possible of their class discriminatory information?

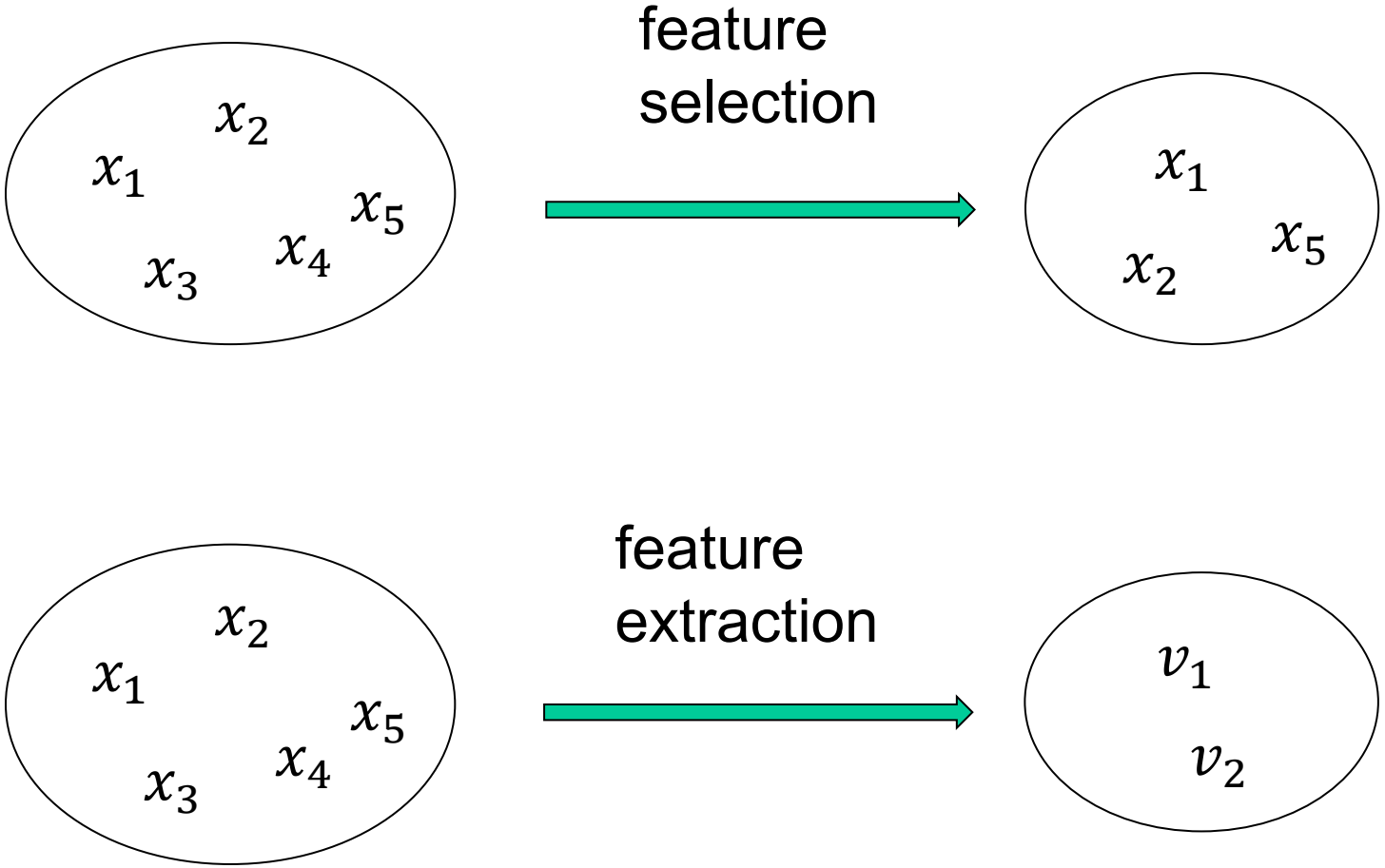
This problem is known as **feature selection** or reduction.

This step is very crucial. If we selected features with little discrimination power, the subsequent design of a classifier would lead to poor performance. On the other hand, if information-rich features are selected, the classifier would have good performance.

it must be pointed out that there is some confusion in the literature concerning the terminology of feature selection and feature extraction.

Simply speaking, feature selection is about selecting a subset of features out of the original features in order to reduce model complexity, enhance the computational efficiency of the models and improve generalization.

Feature extraction is about extracting or deriving information from the original features set to create a new feature subspace. The primary idea behind feature extraction is to compress the data with the goal of maintaining most of the relevant information. The commonly used principal component analysis (PCA) is a feature extraction method.



$$v_1 = f_1(x_1, x_2, x_3, x_4, x_5)$$

$$v_2 = f_2(x_1, x_2, x_3, x_4, x_5)$$

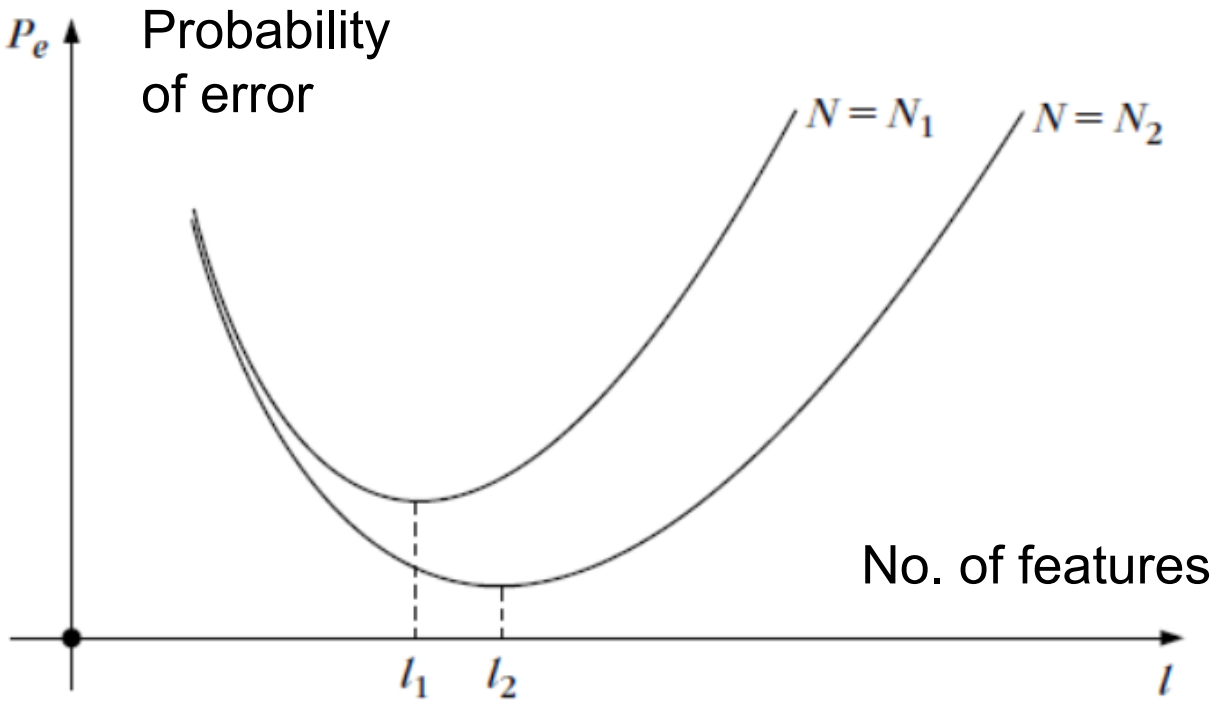
9.2 The Peaking Phenomenon

As stated above, in order to design a classifier with good generalization performance, the number of training samples, N, must be large enough with respect to the number of features, l. i.e. the dimensionality of the feature space. Take as an example the case of designing a linear classifier:

$$y = \mathbf{w}^T\mathbf{x} + w_0$$

The number of the unknown parameters is $l + 1$. In order to get a good estimate of these parameters, the number of samples N must be larger than $l + 1$. The larger the N, the better the estimate, since we can filter out the effects of the noise and also minimize the effects of the outliers.

In practice, for a finite N, by increasing the number of features, one obtains an initial improvement in performance, but after a critical value, further increase of the number of features results in an increase of the probability of error. This phenomenon is known as the peaking phenomenon as illustrated below:



The above figure illustrates the general trend that one expects to experience in practice by playing with the number of features, l, and the size of the training data set, N.

(a) For $N_2 \gg N_1$, the error values corresponding to N_2 are lower than those resulting for N_1, and the peaking phenomenon occurs for a value $l_2 > l_1$.

(b) For each value of N, the probability of error starts decreasing with increasing l till a critical value where the error starts increasing.

Consequently, in practice, for a small number of training data, a small number of features must be used. If a large number of training data is available, a larger number of features can be used for better performance.

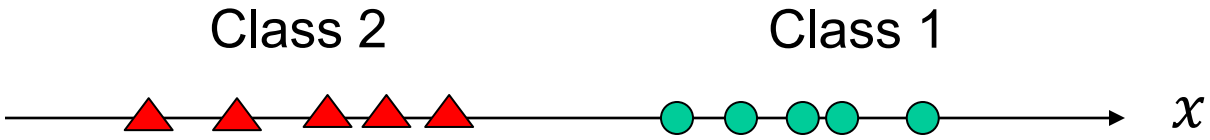
9.3 Individual Feature Evaluation

A first step in feature selection is to look at each of the features individually or separately and test their individual discriminatory capability for the problem at hand. Although looking at the features separately is far from optimal, this procedure helps us to identify and discard "bad" features. This will relieve the set-based feature evaluation and selection, which is to be introduced later, from unnecessary computational burden.

Here, we will introduce two methods for individual feature evaluation, based on two different interpretations of good features, including class separability and relevance of a feature.

9.3.1 Fisher's ratio

As discussed previously, a good feature should take differ values in different classes.



The class separability of a feature can be evaluated using Fisher's ratio. Assume the mean value and standard deviation of the samples in the two classes are m_1, σ_1, m_2 and σ_2, respectively, the Fisher's ratio is defined as:

$$R = \frac{(m_1 - m_2)^2}{\sigma_1^2 + \sigma_2^2}$$

The numerator of Fisher's ratio is actually the inter-class (or called between-class) difference, while the denominator is the intra-class (or within-class) scatter.

From Fisher's ratio, we can see that a good feature should have:

(i) large inter-class difference and

(ii) small intra-class scatter or variance.

The larger the Fisher's ratio, the more discriminative the feature. Fisher ratio is usually used for continuous variables.

9.3.2 Mutual information

A good feature should contain significant amount of information to decide the value of the class label. This relevance of a feature can be evaluated based on mutual information between the feature and the class label.

The mutual information (MI) of two variables is a measure of the mutual dependence between the two variables. More specifically, it quantifies the amount of information obtained about one variable by observing another variable.

The mutual information between feature x and class label y can be measured as follows:

$$MI(x, y) = E(x) + E(y) - E(x, y)$$

Where $E(x)$, $E(y)$ are the entropy for x and y, and $E(x,y)$ is the joint entropy for x and y:

$$E(x) = -\sum_{v_i \in x} p(x = u_i) \times \log p(x = u_i)$$

$$E(y) = -\sum_{l_i \in y} p(y = c_i) \times \log p(y = c_i)$$

$$E(x,y) = -\sum_{c_i \in y} \sum_{u_j \in x} p(x = u_j, y = c_i) \times \log p(\ x = u_j, y = c_i)$$

Where $p(x = u_i)$ denotes probability of future x taking value of u_i. Similarly for $p(y = c_i)$, and $p(x = u_j, y = c_i)$. Mutual information is usually used for discrete features.

9.4 Feature subset selection

The feature subset selection problem can be defined as follows:

Given a full feature set:

$$X = \{x_1, x_2, \ldots, x_n\}$$

Select a subset

$$Z = \{z_1, z_2, \ldots, z_m\}, \qquad z_i \in X$$

that produces the best classification performance.

Due to the peaking phenomenon, a subset of all the available features should be selected and used as inputs to the pattern classification model. In the above, the evaluation of individual feature is introduced. Whether the feature subset can be obtained by simply selecting the top-ranked features?

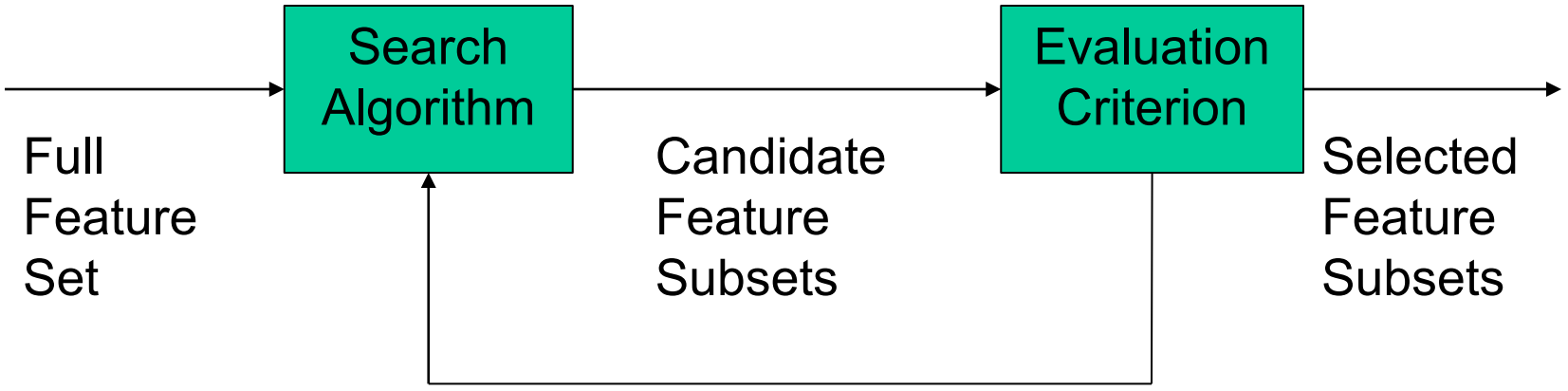
The answer is **no**.

This is because there may exit strong correlation between the top-ranked features, and the feature subsets thus selected may exhibit severe redundancy. A good feature subset should have maximum relevance and minimum redundancy.

A feature subset selection algorithm consists of two main components:

(i) Search algorithm
(ii) Evaluation criterion

The goal of the search algorithm is to generate candidate feature subsets, while the evaluation criterion aims to evaluate the goodness of the candidate feature subsets generated by the search algorithm as illustrate below



9.4.1 Search algorithm

(1) Exhaustive search method

Exhaustive search will, as the name suggests, exhaust all possible subsets. In other words, it generates all possibly combinations of the n features:

$$\{x_1\},\ldots,\ \{x_n\},\ \{x_1, x_2\},\ldots,\{x_1, x_n\},\ldots.,\{x_1, x_2, \ldots, x_n\}$$

Exhaustive search certainly will not miss the optimal feature subset, but the computational complexity is prohibitive if n is large. In practice, exhaustive search is used only when n is small.

(2) Sequential forward selection

Sequential forward search is a bottom-up method. It begins with an empty set of features, adds features to form a feature subset in a step-by-step manner. At every step, the best feature among unselected is chosen based on certain criterion. The feature subset grows until the stopping criterion is satisfied.

(3) Sequential backward elimination

Sequential backward elimination is a top-down method. It begins with a full set of features. Features are removed one at a time. At each time, the least important feature is removed based on certain criterion. The feature set shrinks until the stopping criterion is satisfied.

9.4.2 Evaluation criterion

In order to select a good feature subset, we require a means of measuring the ability of a feature subset to discriminate accurately two or more classes. This can be achieved by defining a class separability measure that is optimized with respect to the possible feature subsets.

We can evaluate the feature subsets by two ways:

(1) Classification performance of a classifier

In this approach, we train a classifier on the feature subset, and then use the classification performance as the evaluation criterion. The feature subset is chosen to match the classifier. Different classifiers may select different feature subsets.

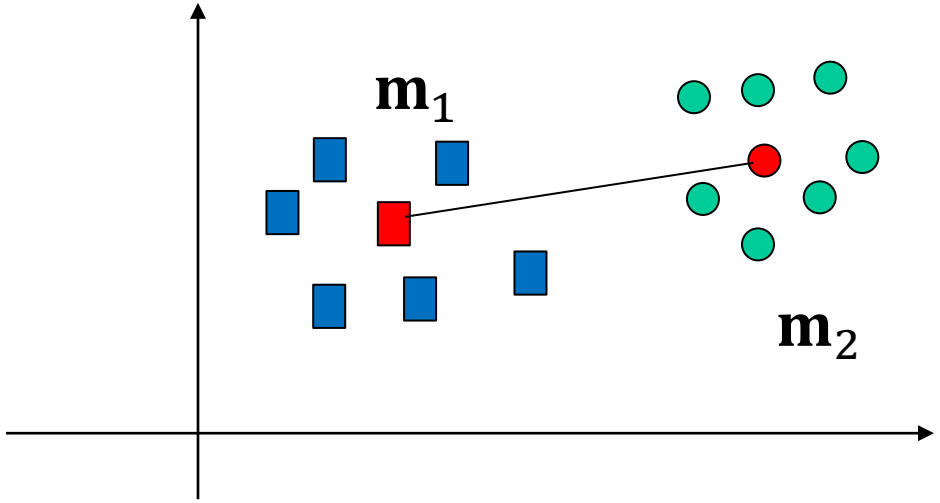
(2) Separability measures

This approach estimates the overlap between the distributions from which the data is drawn, and favours those with minimum overlap, i.e. the maximum separability. The feature subset thus selected is independent of the final classifier used. This criterion does not require training of a large number of pattern classifiers, and is therefore computationally more efficient.

The disadvantage is that the assumptions made in estimating data distribution overlap are often crude and may not match the true distribution well.

Next, a few separability measures are introduced.

(i) Mahalanobis distance-based separability measure



The separability can be measured by Mahalanobis distance between two mean vectors in the space of the feature subset:

$$J_{1,2} = (\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{C}^{-1} (\mathbf{m}_1 - \mathbf{m}_2)$$

Where $\mathbf{m}_1$ and $\mathbf{m}_1$ are mean vectors of the two classes, $\mathbf{C}$ is the covariance matrix.

For a multi-class problem, we may use the sum of pair-wise separability measures

$$J = \sum_{i} \sum_{j \neq i} J_{i,j}$$

(ii) Scatter-based separability measure

$$J = \mathrm{Tr}(\mathbf{S}_W^{-1}\mathbf{S}_B)$$

Or

$$J = \frac{\mathrm{Tr}(\mathbf{S}_B)}{\mathrm{Tr}(\mathbf{S}_W)}$$

Where $\mathbf{S}_B$ and $\mathbf{S}_W$ are the between-class and within-class scatter matrices. $\mathrm{Tr}(\mathbf{A})$ denotes the trace of matrix $\mathbf{A}$, defined as the sum of elements on the main diagonal of $\mathbf{A}$.

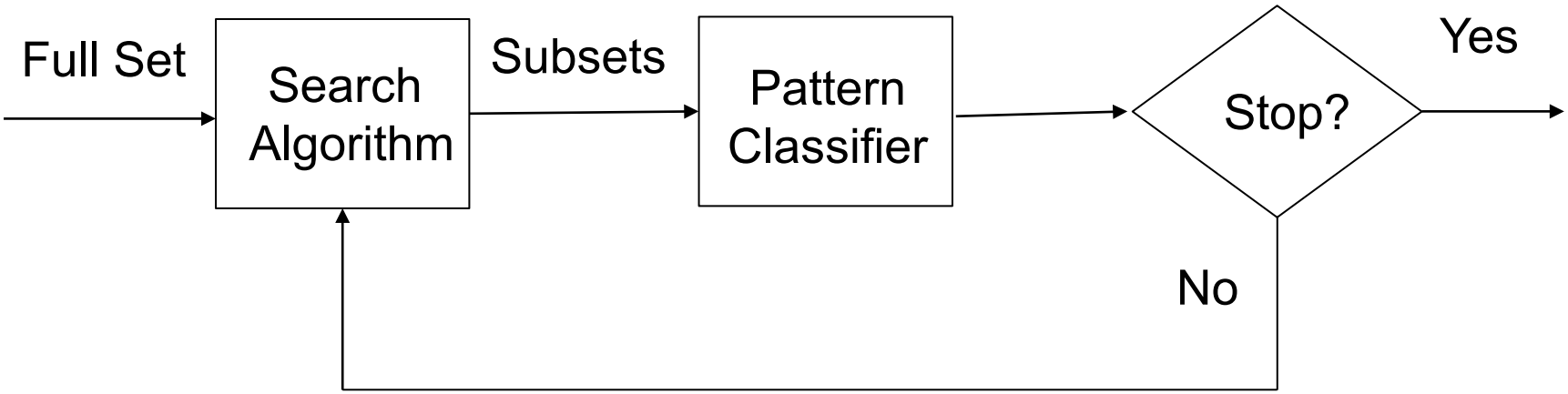
9.4.3 Feature subset selection algorithms

A feature subset selection algorithm consists of a search algorithm and a feature subset evaluation criterion. Based on the evaluation criterion used, the feature selection algorithms are often divided into three categories: the **filter** method, the **wrapper** methods and the **embedded** method.

The wrapper method includes a classifier in the loop and use classification performance such as accuracy or error rate as the feature subset evaluation criterion.

The filter method does not include a classifier in the loop, instead, it employs class separability measures as the feature subset evaluation criterion.

(1) The wrapper method



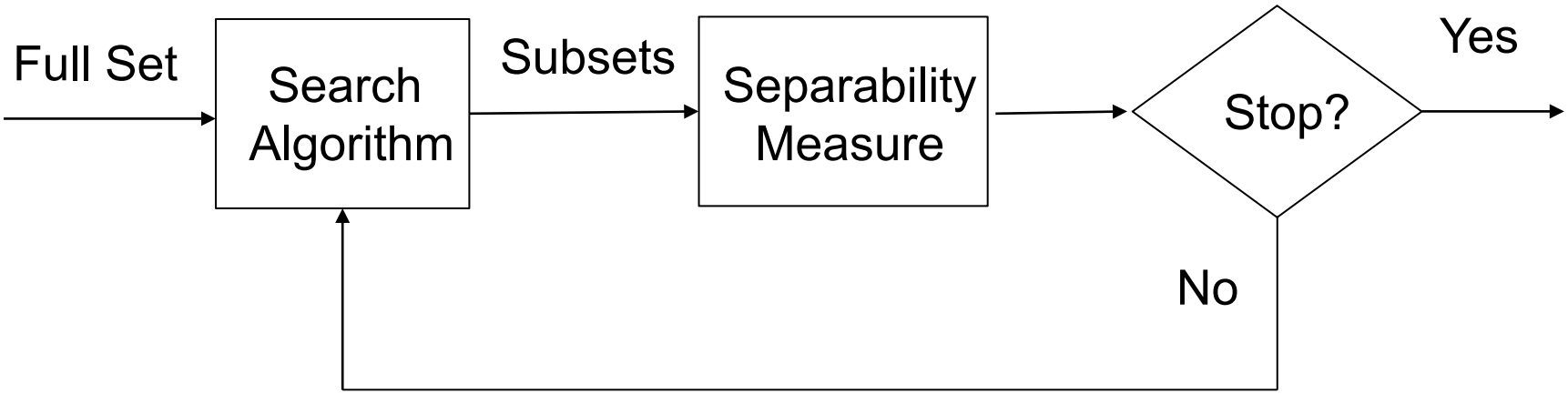
The stopping criterion could be

(i) Number of features to be selected is reached, or

(ii) Classification accuracy (on validation set) is optimal, or

(iii) Others

(2) The filter method



The stopping criterion could be

(i) Number of features to be selected is reached, or

(ii) Separability measure no longer improves, or

(iii) Others

Sequential forward feature selection algorithm

We next combine the sequential forward search and evaluation criteria to form feature selection algorithms.

(1) In Step 1, consider each feature as a candidate,

$$F^{(i)} = \{x_i\}, \qquad i = 1,2, \dots, n$$

evaluate the separability measure (filter)

$$J^{F(i)} = \frac{\mathrm{Tr}(S_B^{F(i)})}{\mathrm{Tr}(S_W^{F(i)})}$$

or train a classifier and find classification accuracy (wrapper)

$$C^{F(i)} = Classifier\ Trained\ on\ F(i)$$

$$J^{F(i)} = Accuracy(C^{F(i)})$$

The feature that has the largest separability (filter method) or the best classification performance (wrapper method) is selected as the first feature:

$$k = \text{argmax}(J^{F(i)})$$

$$z_1 = x_k$$

(2) In Step 2, consider each of the remaining $n - 1$ features as the candidate for the second feature,

$$F^{(i)} = \{z_1, x_i\}, i \neq k$$

evaluate the separability measure (filter method):

$$J^{F(i)} = \frac{Tr(S_B^{F(i)})}{Tr(S_W^{F(i)})}$$

or train a classifier and find the classification accuracy (wrapper method)

$$C^{F(i)} = Classifier\ Trained\ on\ F(i)$$

$$J^{F(i)} = Accuracy(C^{F(i)})$$

The feature that has led to the largest separability (filter method) or the best classification performance (wrapper method) is selected as the second feature.

(3) The selection process is continued until the stopping criterion is satisfied

Sequential backward feature elimination algorithm

(1) In Step 1, consider each feature as the candidate to be eliminated,

$$F^{(i)} = \{x_1, x_2, \dots, x_n\} - x_i, \qquad i = 1,2, \dots, n$$

evaluate corresponding separability measure (filter)

$$J^{F(i)} = \frac{Tr(S_B^{F(i)})}{Tr(S_W^{F(i)})}$$

or train a classifier and find corresponding classification accuracy (wrapper)

$$C^{F(i)} = Classifier\ Trained\ on\ F(i)$$

$$J^{F(i)} = Accuracy(C^{F(i)})$$

Identify the feature that least affects the separability measure (filter method) or the classification performance (wrapper method) and take it as the first feature to be removed.

$$k = \text{argmax}(J^{F(i)})$$

$$z_1 = x_k$$

(2) In Step 2, consider each of the remaining $n - 1$ features as the candidate for the second feature to be removed,

$$F^{(i)} = \{x_1, x_2, \dots, x_n\} - z_1 - x_i \,, i \neq k$$

evaluate the separability measure (filter method):

$$J^{F(i)} = \frac{Tr(S_B^{F(i)})}{Tr(S_W^{F(i)})}$$

or train a classifier and find the classification accuracy (wrapper method)

$$C^{F(i)} = Classifier\ Trained\ on\ F(i)$$

$$J^{F(i)} = Accuracy(C^{F(i)})$$

The feature that least affects the separability measure (filter method) or the classification performance (wrapper method) is selected as the second feature to be removed.

(3) The elimination process is continued until the stopping criterion is satisfied.

(3) Embedded method

Embedded methods combine the qualities of filter and wrapper methods. It is implemented by algorithms that have their own built-in feature selection methods. The most popular example of this method is Lasso (least absolute shrinkage and selection operator), which have inbuilt mechanisms to penalize the number of features in the model by shrinking the coefficients associated with the least important features to zero (refer to Chapter 8).

$$J_{lasso}(\boldsymbol{\theta}) = \frac{1}{2}\left\{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 + \lambda \sum_{j=0}^{m}\left|\theta_j\right|\right\}$$

$$= \frac{1}{2}\{(\mathbf{y} - \boldsymbol{\Phi\theta})^T(\mathbf{y} - \boldsymbol{\Phi\theta}) + \lambda\|\boldsymbol{\theta}\|\}$$