Feature Selection

- By: **Kaveh Kavousi**
- Department of Bioinformatics
- IBB (Institute of Biochemistry and Biophysics)
- University of Tehran

Feature Selection

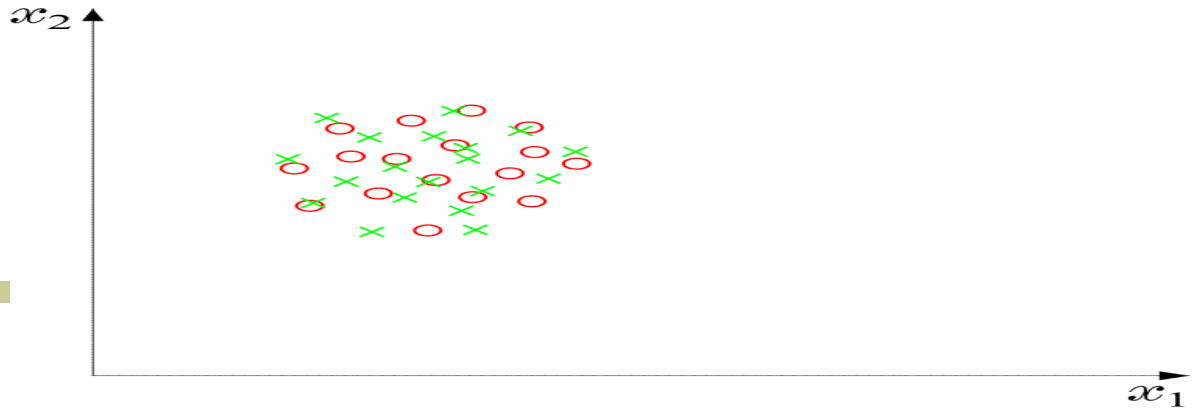
- The goals:
  - Select the “optimum” number l of features
  - Select the “best” l features

- Large l has a three-fold disadvantage:
  - High computational demands
  - Low generalization performance
  - Poor error estimates

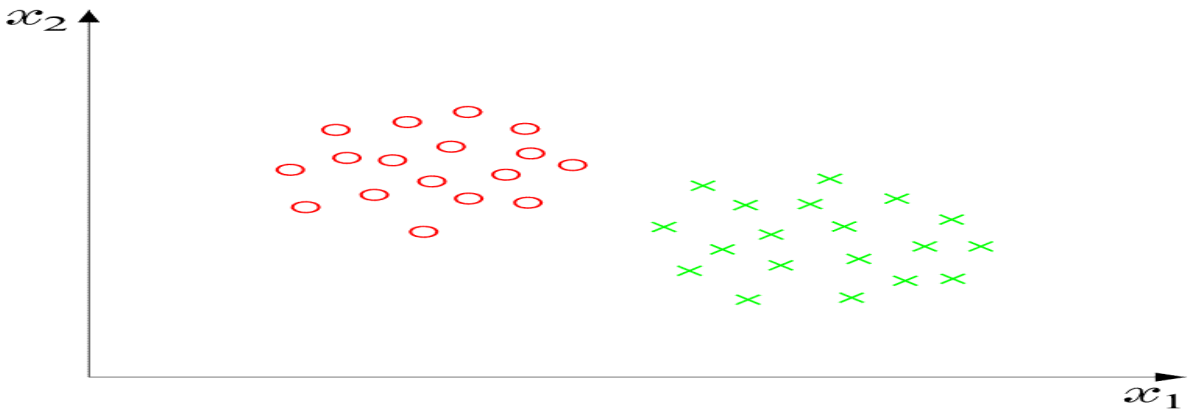
Feature Selection

- Given N
  - l must be large enough to learn
    - what makes classes different
    - what makes patterns in the same class similar
  - l must be small enough not to learn what makes patterns of the same class different
  - In practice, $l < N/3$ has been reported to be a sensible choice for a number of cases

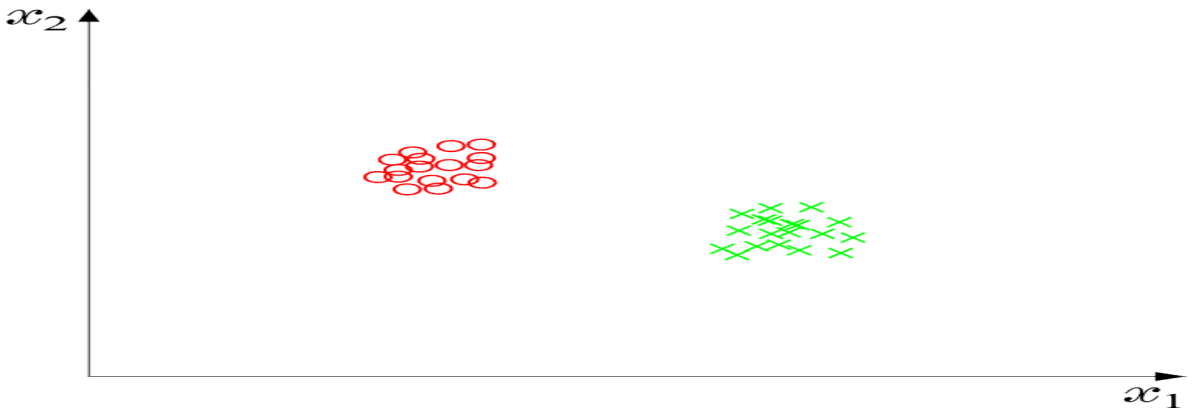
- Once l has been decided, choose the l most informative features
  - Best: **Large** between class distance,
    Small within class variance



Bad choice



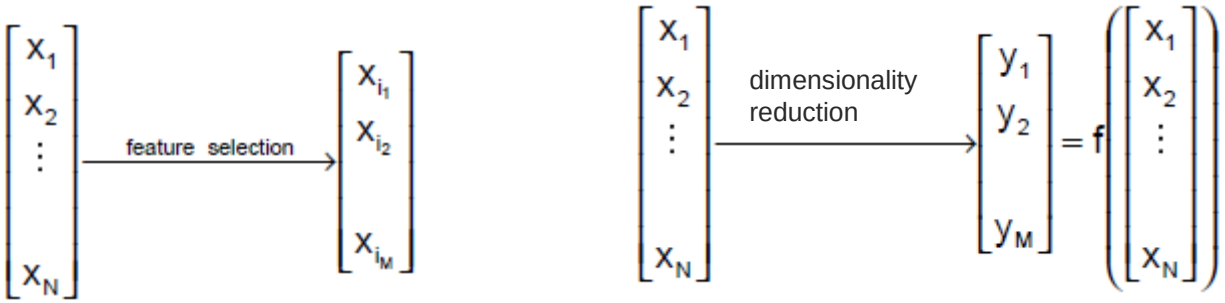
Not bad choice



Good choice

Feature Selection

- Given a set of **n** features, the role of **feature selection** is to select a subset of size **d** (**d** < **n**) that leads to the smallest classification error.



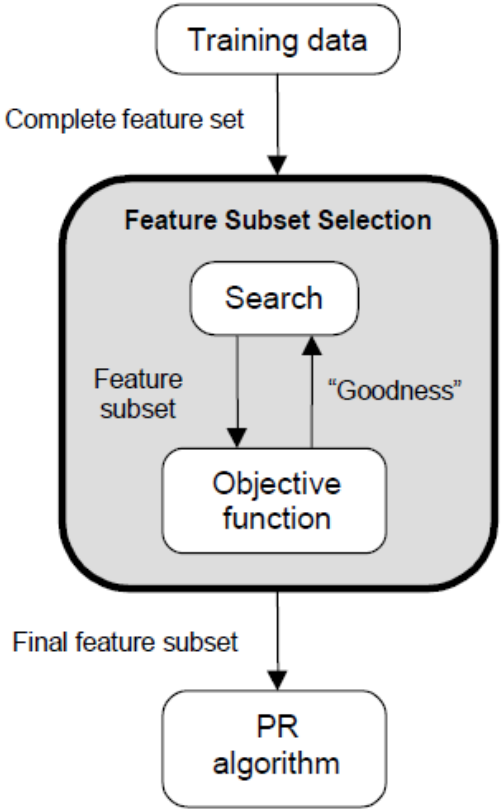
- Fundamentally different from dimensionality reduction (e.g., PCA or LDA)

Why is feature selection important?

- Features may be expensive to obtain
  - You evaluate a large number of features (sensors) in the test bed and select only a few for the final implementation
- You may want to extract meaningful rules from your classifier
  - When you transform or project, the measurement units (length, weight, etc.) of your features are lost
- Features may not be numeric
  - A typical situation in the machine learning domain

Feature Selection Methods

- Feature selection is an **optimization** problem.
  - Search the space of possible feature subsets.
  - Pick the subset that is optimal or near-optimal with respect to an objective function.



Feature Selection Methods

- Feature selection is an **optimization** problem.
  - Search the space of possible feature subsets.
  - Pick the subset that is optimal or near-optimal with respect to a certain criterion.

Search strategies

- Optimum
- Heuristic
- Randomized

Evaluation strategies

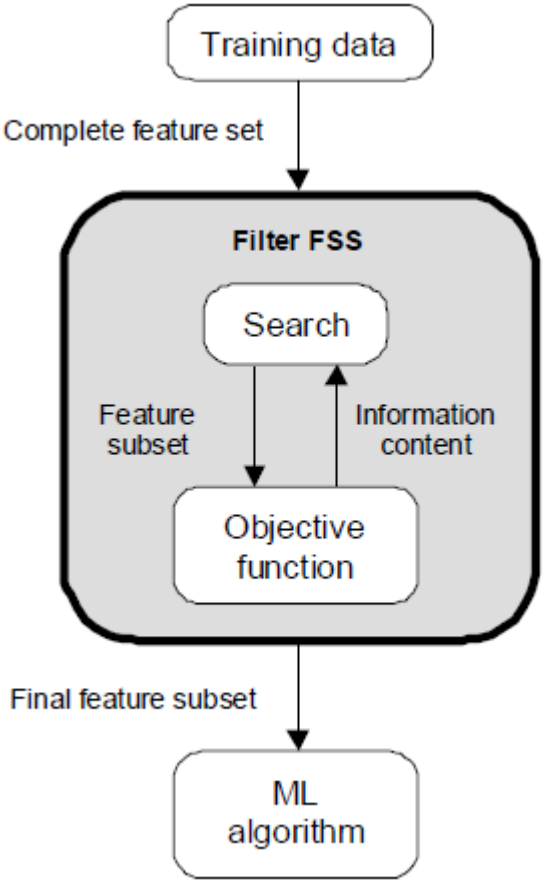
- Filter methods
- Wrapper methods
- Embedded Methods

Search Strategies

- Assuming *n* features, an exhaustive search would require:
  - Examining all $\binom{n}{d}$ possible subsets of size d.
  - Selecting the subset that performs the best according to the criterion function.
- The number of subsets grows combinatorially, making exhaustive search impractical.
- Iterative procedures are often used based on heuristics but they **cannot** guarantee the selection of the optimal subset.

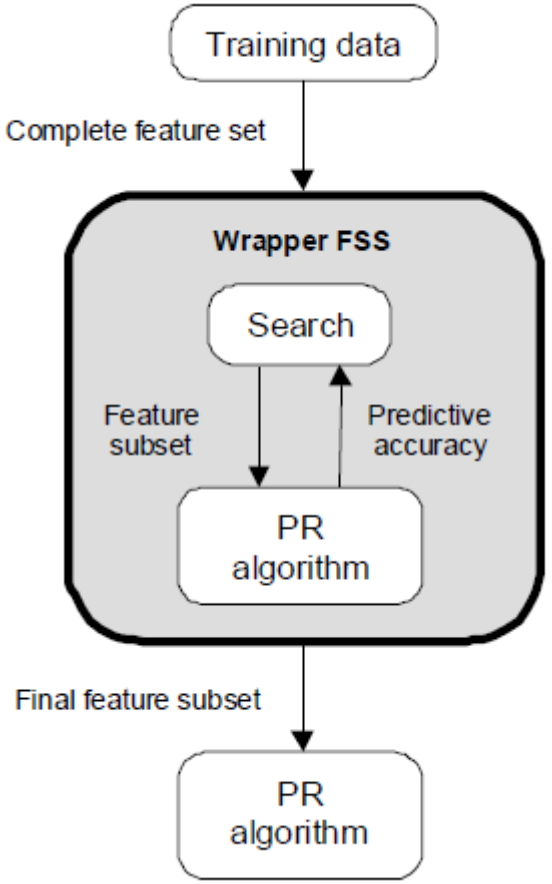
Evaluation Strategies

- Filter Methods
  - Evaluation is independent of the classification algorithm.
  - The objective function evaluates feature subsets by their information content, typically interclass distance, statistical dependence or information-theoretic measures.



Evaluation Strategies

- Wrapper Methods
  - Evaluation uses criteria related to the classification algorithm.
  - The objective function is a pattern classifier, which evaluates feature subsets by their predictive accuracy (recognition rate on test data) by statistical resampling or cross-validation.



Filter vs Wrapper Approaches

- **Wrappers**
  - Advantages
    - **Accuracy**: wrappers generally achieve better recognition rates than filters since they are tuned to the specific interactions between the classifier and the dataset
    - **Ability to generalize**: wrappers have a mechanism to avoid overfitting, since they typically use cross-validation measures of predictive accuracy
  - Disadvantages
    - **Slow execution**: since the wrapper must train a classifier for each feature subset (or several classifiers if cross-validation is used), the method can become unfeasible for computationally intensive methods
    - **Lack of generality**: the solution lacks generality since it is tied to the bias of the classifier used in the evaluation function. The "optimal" feature subset will be specific to the classifier under consideration

Filter vs Wrapper Approaches (cont'd)

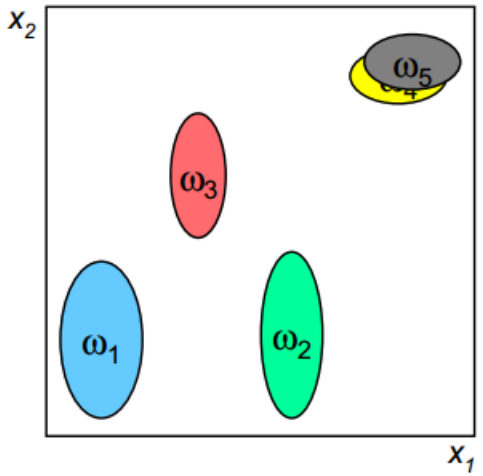
- **Filters**
  - Advantages
    - **Fast execution**: Filters generally involve a non-iterative computation on the dataset, which can execute much faster than a classifier training session
    - **Generality**: Since filters evaluate the intrinsic properties of the data, rather than their interactions with a particular classifier, their results exhibit more generality: the solution will be "good" for a larger family of classifiers
  - Disadvantages
    - **Tendency to select large subsets**: Since the filter objective functions are generally monotonic, the filter tends to select the full feature set as the optimal solution. This forces the user to select an arbitrary cutoff on the number of features to be selected

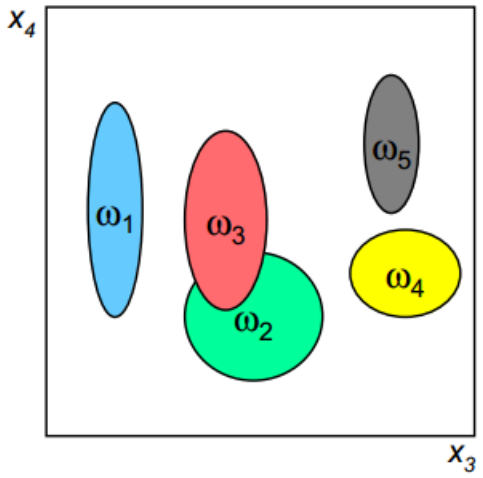
Naïve Search

- Sort the given n features in order of their probability of correct recognition.
- Select the top d features from this sorted list.
- Disadvantage
  - Feature correlation is not considered.
  - Best pair of features may not even contain the best individual feature.

Naïve Search

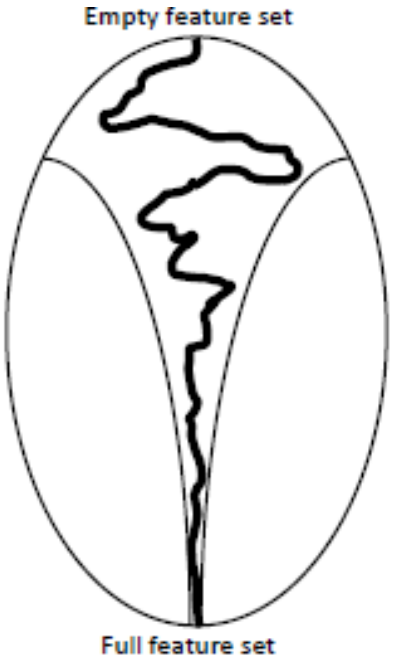
- **One may be tempted to evaluate each individual feature separately and select those M features with the highest scores**
  - Unfortunately, this strategy will VERY RARELY work since it does not account for feature dependence
- **An example will help illustrate the poor performance that can be expected from this naïve approach**
  - The figures show a 4-dimensional pattern recognition problem with 5 classes. Features are shown in pairs of 2D scatter plots
  - The objective is to select the best subset of 2 features using the naïve sequential feature selection procedure
  - Any reasonable objective function will rank features according to this sequence: $J(x_1)>J(x_2)\approx J(x_3)>J(x_4)$
    - x_1 is, without a doubt, the best feature. It clearly separates ω_1, ω_2, ω_3 and $\{\omega_4, \omega_5\}$
    - x_2 and x_3 have similar performance, separating classes in three groups
    - x_4 is the worst feature since it can only separate ω_4 from ω_5, the rest of the classes having a heavy overlap
  - The optimal feature subset turns out to be $\{x_1, x_4\}$, because x_4 provides the only information that x_1 needs: discrimination between classes ω_4 and ω_5
  - However, if we were to choose features according to the individual scores $J(x_k)$, we would choose x_4 and either x_2 or x_3, leaving classes ω_4 and ω_5 non separable
    - This naïve strategy fails because it does not take into account the interaction between features





Sequential forward selection (SFS)
(heuristic search)

- First, the best **single** feature is selected (i.e., using some criterion function).
- Then, **pairs** of features are formed using one of the remaining features and this best feature, and the best pair is selected.
- Next, **triplets** of features are formed using one of the remaining features and these two best features, and the best triplet is selected.
- This procedure continues until a predefined number of features are selected.

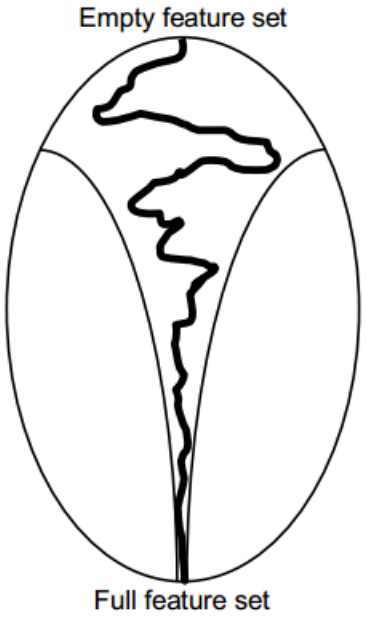


SFS performs best when the optimal subset is small.

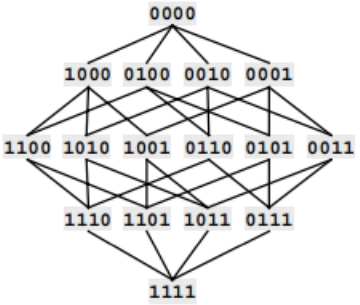
Sequential forward selection (SFS)
(heuristic search)

- **Sequential Forward Selection is the simplest greedy search algorithm**
  - Starting from the empty set, sequentially add the feature x^+ that results in the highest objective function $J(Y_k+x^+)$ when combined with the features Y_k that have already been selected
- **Algorithm**

1. Start with the empty set $Y=\{\varnothing\}$
2. Select the next best feature $x^+ = \underset{x \in X-Y_k}{\arg\max}[J(Y_k + x)]$
3. Update $Y_{k+1}=Y_k+x$; $k=k+1$
4. Go to 2

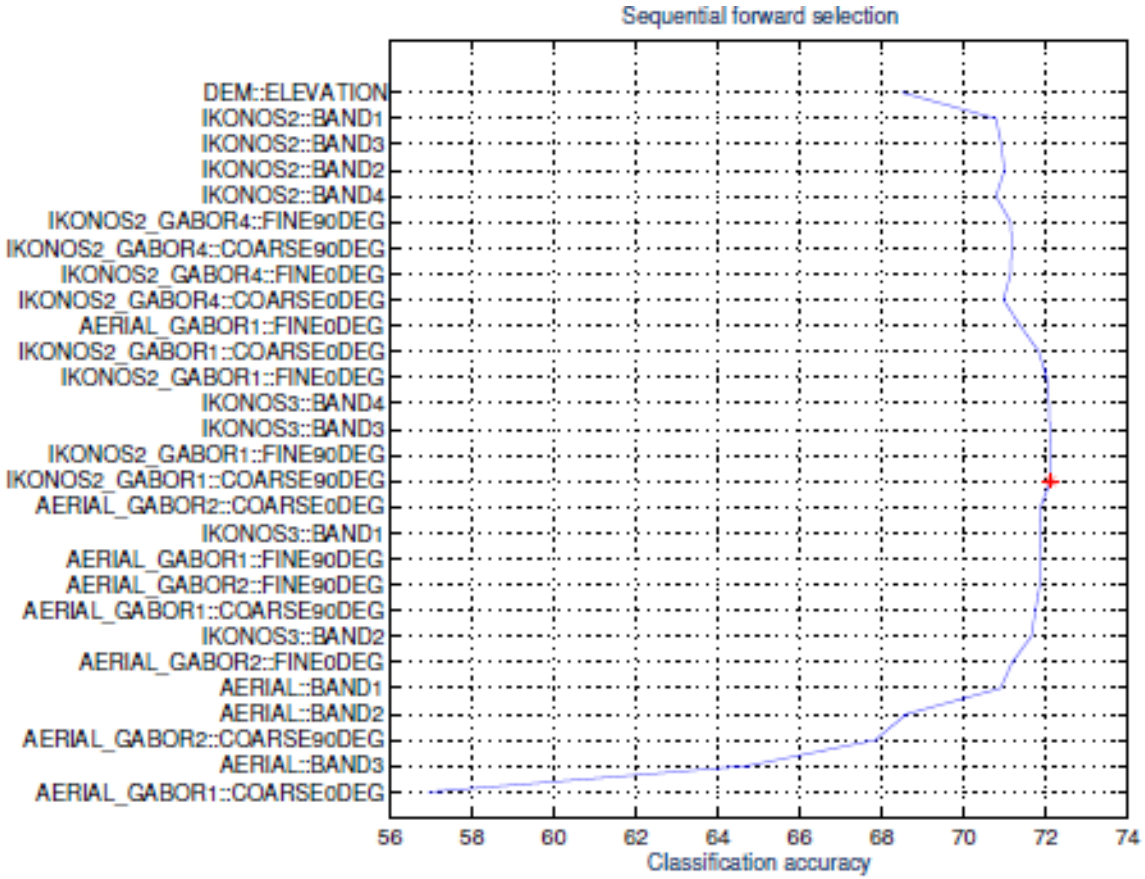


- **Notes**
  - SFS performs best when the optimal subset has a small number of features
    - When the search is near the empty set, a large number of states can be potentially evaluated
    - Towards the full set, the region examined by SFS is narrower since most of the features have already been selected
  - The search space is drawn like an ellipse to emphasize the fact that there are fewer states towards the full or empty sets
    - As an example, the state space for 4 features is shown. Notice that the number of states is larger in the middle of the search tree
    - The main disadvantage of SFS is that it is unable to remove features that become obsolete after the addition of other features



Example

features added at each iteration

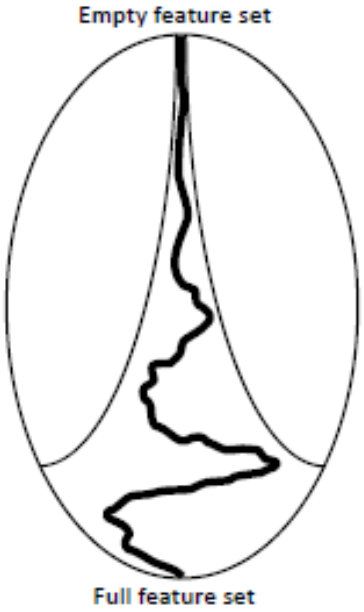


Results of sequential forward feature selection for classification of a satellite image using 28 features. x-axis shows the classification accuracy (%) and y-axis shows the features added at each iteration (the first iteration is at the bottom). The highest accuracy value is shown with a star.

Sequential backward selection (SBS)

(heuristic search)

- First, the criterion function is computed for all **n** features.
- Then, each feature is deleted one at a time, the criterion function is computed for all subsets with **n-1** features, and the worst feature is discarded.
- Next, each feature among the remaining **n-1** is deleted one at a time, and the worst feature is discarded to form a subset with **n-2** features.
- This procedure continues until a predefined number of features are left.



SBS performs best when the optimal subset is large.

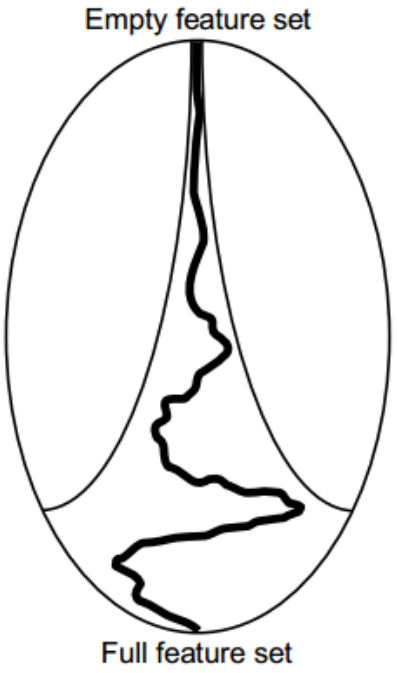
Sequential backward selection (SBS)

(heuristic search)

- **Sequential Backward Selection works in the opposite manner as SFS**
  - Starting from the full set, sequentially remove the feature x^+ that results in the smallest decrease in the value of the objective function $J(Y-x^+)$
    - Notice that removal of a feature may result in an increase in the objective function $J(Y_k-x^+)> J(Y_k)$. Such functions are said to be non-monotonic --more on this when we cover Branch and Bound.
- **Algorithm**

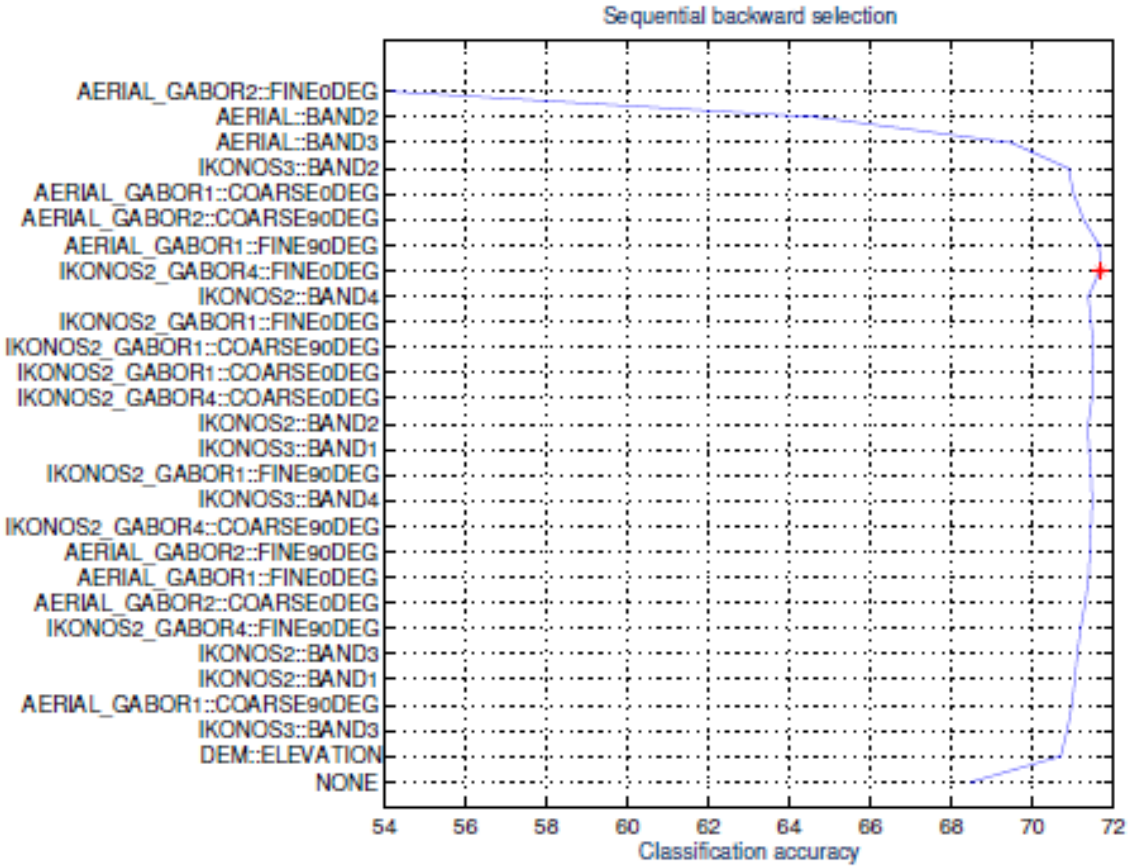
1. Start with the full set Y=X
2. Remove the worst feature $x^- = \arg\max_{x \in Y_k}[J(Y_k - x)]$
3. Update $Y_{k+1}=Y_k-x$; k=k+1
4. Go to 2

- **Notes**
  - SBS works best when the optimal feature subset has a large number of features, since SBS spends most of its time visiting large subsets
  - The main limitation of SBS is its inability to reevaluate the usefulness of a feature after it has been discarded



Example

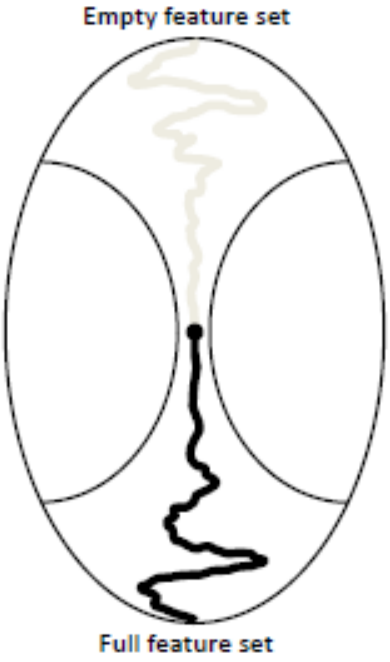
features removed at each iteration



Results of sequential backward feature selection for classification of a satellite image using 28 features. x-axis shows the classification accuracy (%) and y-axis shows the features removed at each iteration (the first iteration is at the top). The highest accuracy value is shown with a star.

Bidirectional Search (BDS)

- BDS applies SFS and SBS simultaneously:
  - SFS is performed from the empty set
  - SBS is performed from the full set
- To guarantee that SFS and SBS converge to the same solution
  - Features already selected by SFS are not removed by SBS
  - Features already removed by SBS are not selected by SFS



Bidirectional Search (BDS)

- **Bidirectional Search is a parallel implementation of SFS and SBS**
  - SFS is performed from the empty set
  - SBS is performed from the full set
  - To guarantee that SFS and SBS converge to the same solution, we must ensure that
    - Features already selected by SFS are not removed by SBS
    - Features already removed by SBS are not selected by SFS
    - For example, before SFS attempts to add a new feature, it checks if it has been removed by SBS and, if it has, attempts to add the second best feature, and so on. SBS operates in a similar fashion.
- **Algorithm**

1. Start SFS with the empty set $Y_F=\{\varnothing\}$
2. Start SBS with the full set $Y_B=X$
3. Select the best feature

$$x^+ = \underset{\substack{x\in X-Y_{F_k} \\ x\notin Y_{B_k}}}{\arg\max}\left[J\left(Y_{F_k}+x\right)\right]$$

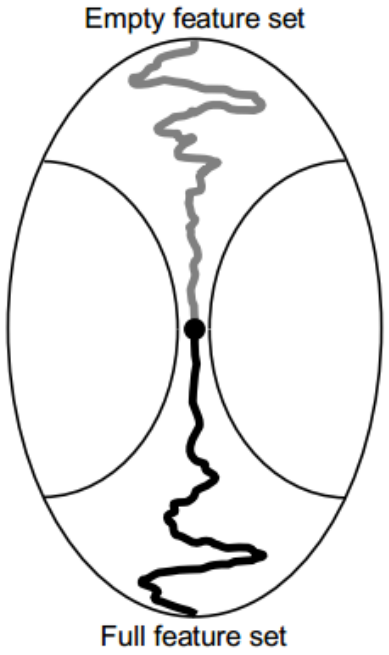
$$Y_{F_{k+1}} = Y_{F_k} + x^+$$

3. Remove the worst feature

$$x^- = \underset{\substack{x\in Y_{B_k} \\ x\notin Y_{F_{k+1}}}}{\arg\max}\left[J\left(Y_{B_k}-x\right)\right]$$

$$Y_{B_{k+1}} = Y_{B_k} - x^-; \quad k = k+1$$

4. Go to 2



“Plus-L, minus-R” selection (LRS)

- A generalization of SFS and SBS
  - If L>R, LRS starts from the empty set and:
    - Repeatedly add L features
    - Repeatedly remove R features
  - If L<R, LRS starts from the full set and:
    - Repeatedly removes R features
    - Repeatedly add L features

- LRS attempts to compensate for the weaknesses of SFS and SBS with some backtracking capabilities.

"Plus-L, minus-R" selection (LRS)

- **Plus-l Minus-r is a generalization of SFS and SBS**
  - If l>r, LRS starts from the empty set and repeatedly adds 'l' features and removes 'r' features
  - If l<r, LRS starts from the full set and repeatedly removes 'r' features'l' followed by 'r' feature additions
- **Algorithm**

1. If l>r then
   start with the empty set Y={∅}
   else
   start with the full set Y=X
   go to step 3
2. Repeat l times

$$x^+ = \arg\max_{x \in X - Y_k} [J(Y_k + x)]$$

$$Y_{k+1} = Y_k + x^+; \quad k = k + 1$$

3. Repeat r times

$$x^- = \arg\max_{x \in Y_k} [J(Y_k - x)]$$

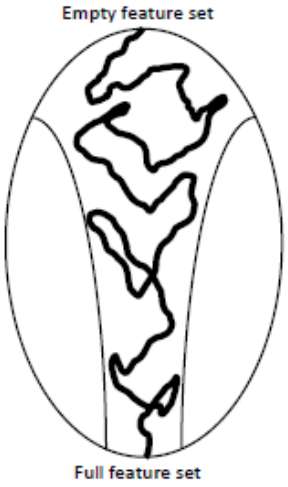
$$Y_{k+1} = Y_k - x^-; \quad k = k + 1$$

4. Go to 2

- **Notes**
  - LRS attempts to compensate for the weaknesses of SFS and SBS with some backtracking capabilities
  - Its main limitation is that there is no theoretical way of predicting the optimal values of l and r

Sequential floating selection (SFFS and SFBS)

- An extension to LRS with flexible backtracking capabilities
  - Rather than fixing the values of L and R, floating methods determine these values from the data.
  - The dimensionality of the subset during the search can be thought to be “floating” up and down



- There are two floating methods:
  - Sequential floating forward selection (SFFS)
  - Sequential floating backward selection (SFBS)

P. Pudil, J. Novovicova, J. Kittler, Floating search methods in feature selection, Pattern Recognition Lett. 15 (1994) 1119–1125.

Sequential floating selection (SFFS and SFBS)

- SFFS
  - Sequential floating forward selection (SFFS) starts from the empty set.
  - After each forward step, SFFS performs backward steps as long as the objective function increases.
- SFBS
  - Sequential floating backward selection (SFBS) starts from the full set.
  - After each backward step, SFBS performs forward steps as long as the objective function increases.

Sequential floating selection (SFFS and SFBS)

- **Sequential Floating Selection methods are an extension to the LRS algorithms with flexible backtracking capabilities**
  - Rather than fixing the values of 'l' and 'r', these Floating methods allow those values to be determined from the data:
    - The dimensionality of the subset during the search can be though to be "floating" up and down
- **There are two floating methods**
  - Sequential Floating Forward Selection (SFFS) starts from the empty set
    - After each forward step, SFFS performs backward steps as long as the objective function increases
  - Sequential Floating Backward Selection (SFBS) starts from the full set
    - After each backward step, SFBS performs forward steps as long as the objective function increases
- **SFFS Algorithm (SFBS is analogous)**

1. Start with the empty set $Y=\{\varnothing\}$
2. Select the best feature

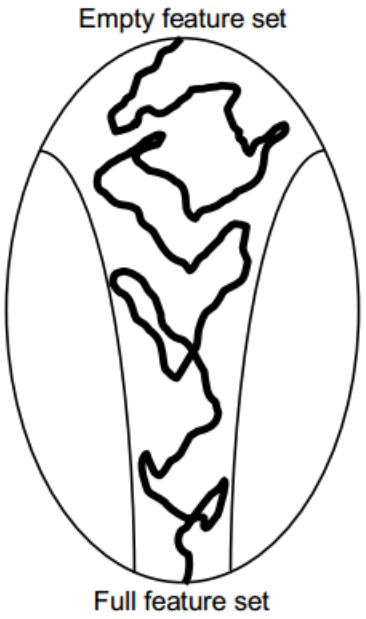
$$x^{+} = \underset{x \in X - Y_k}{\arg\max}\left[J(Y_k + x)\right]$$

$$Y_k = Y_k + x^{+}; \quad k = k + 1$$

3. Select the worst feature

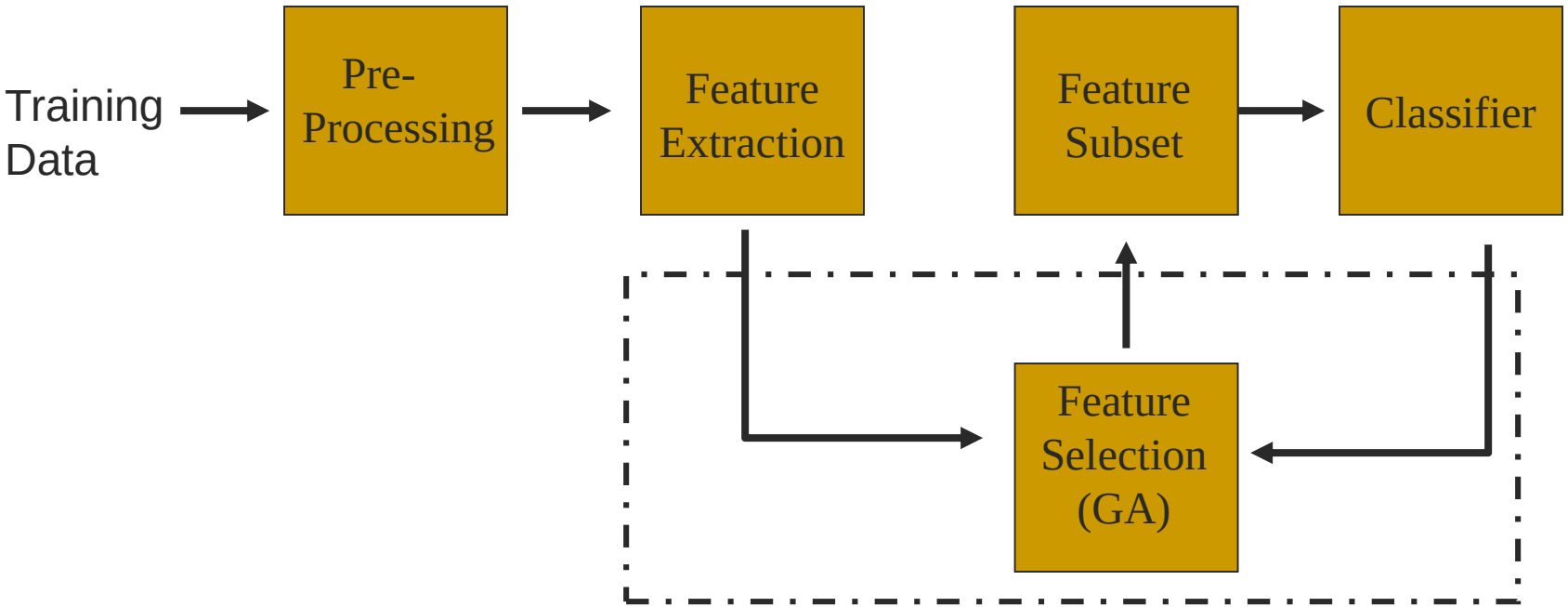
$$x^{-} = \underset{x \in Y_k}{\arg\max}\left[J(Y_k - x)\right]$$

4. If $J(Y_k - x^{-}) > J(Y_k)$ then
   $Y_{k+1} = Y_k - x$; $k = k+1$
   go to Step 3
   else
   go to Step 2



Feature Selection using Genetic Algorithms (GAs) (randomized search)

GAs provide a simple, general, and powerful framework for feature selection.



Genetic Algorithms (GAs)

- What is a GA?
  - An optimization technique for searching very large spaces.
  - Inspired by the biological mechanisms of natural selection and reproduction.
- Main characteristics of GAs
  - Randomized optimization technique.
  - Searches probabilistically using a population of structures, each properly encoded as a string of symbols.
  - Uses objective function information, not derivatives.

Encoding

- Encoding scheme
  - Transforms solutions in parameter space into finite length strings (chromosomes) over some finite set of symbols.

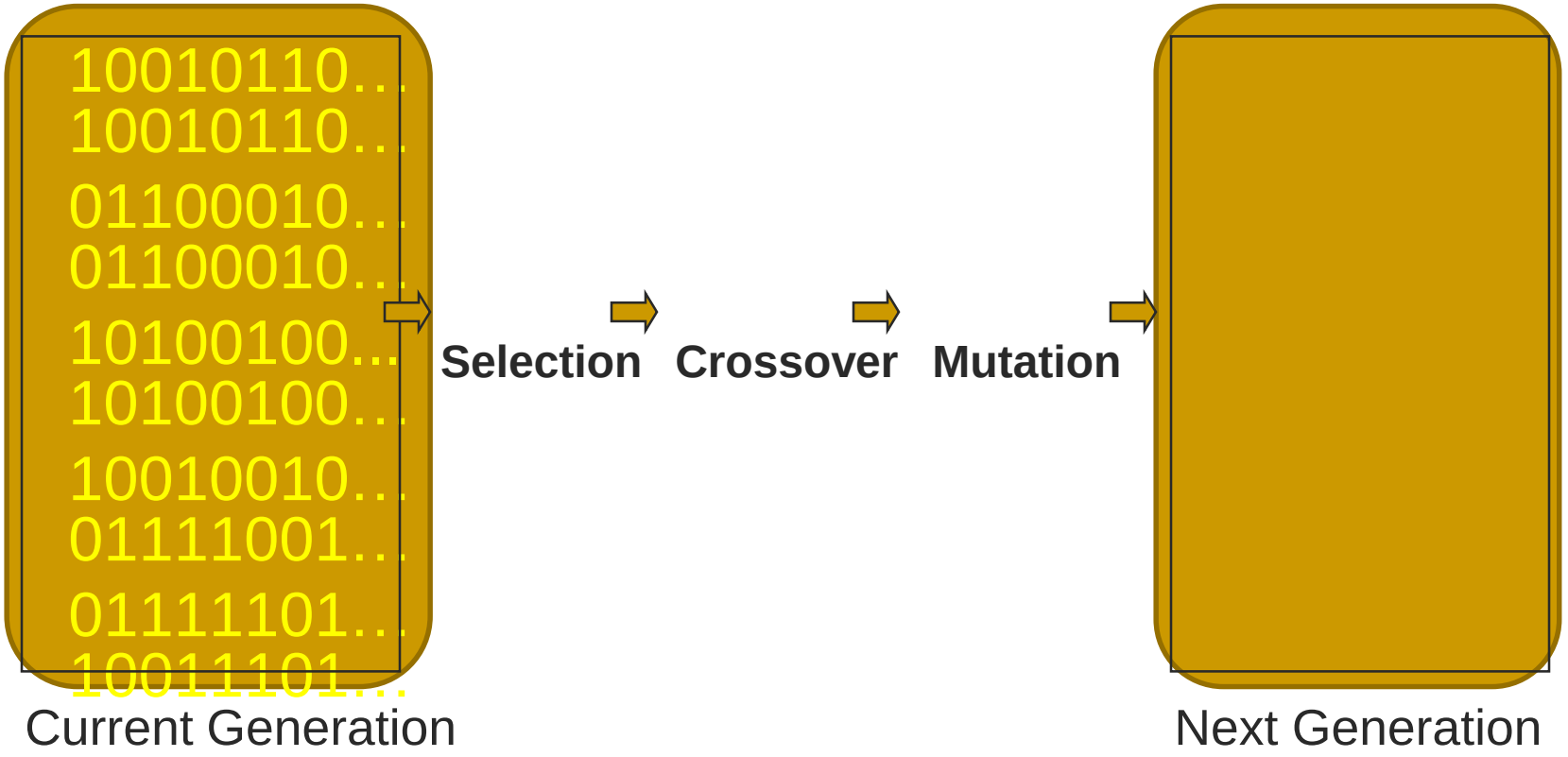
$$(11,6,9) \rightarrow (1011\_0110\_1001) \rightarrow (101101101001)$$

Fitness Evaluation

- Fitness function
  - Evaluates the goodness of a solution.

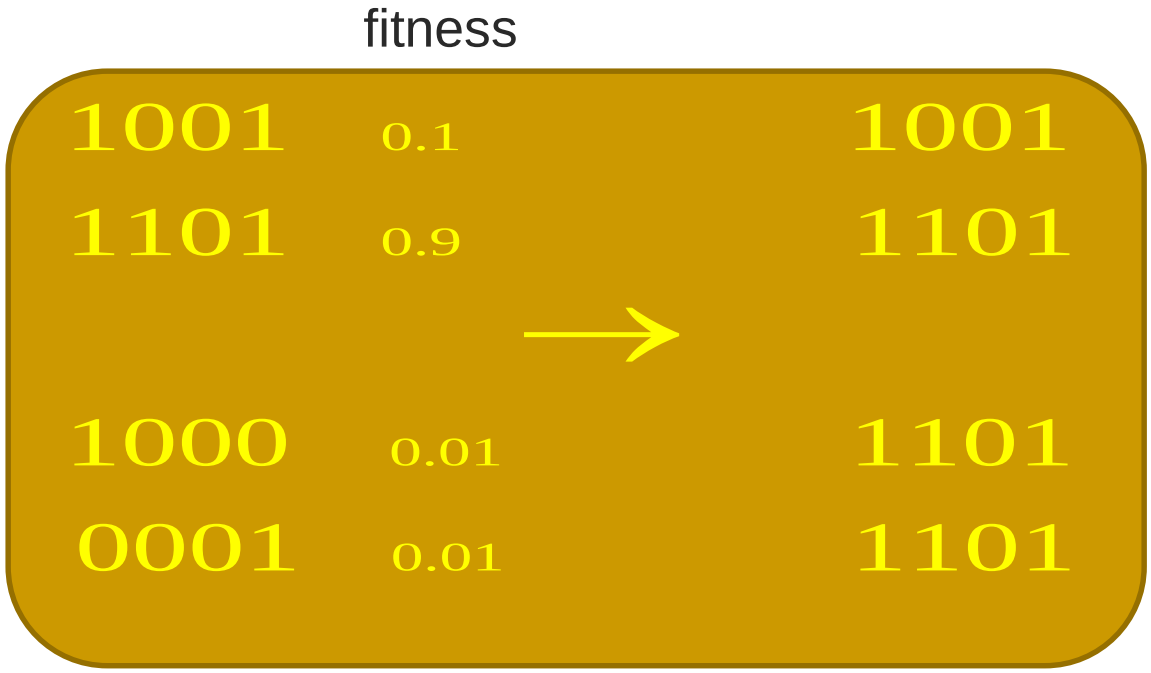
$$Fitness = f(decode(chromosome))$$

GA Search Process



Selection Operator

- Probabilistically filters out solutions that perform poorly, choosing high performance solutions to **exploit.**



Selection Operators

- Fitness proportionate selection (Roulette wheel selection)
- Reward-based selection (**The probability of being selected for an individual is proportional to the cumulative reward, obtained by the individual. The cumulative reward can be computed as a sum of the individual reward and the reward, inherited from parents.**)
- Stochastic universal sampling (SUS)
- Tournament selection (**Tournament selection involves running several "tournaments" among a few individuals (or "chromosomes") chosen at random from the population. The winner of each tournament (the one with the best fitness) is selected for crossover**)

http://www.geatbx.com/docu/algindex-02.html

Fitness proportionate selection (Roulette wheel)

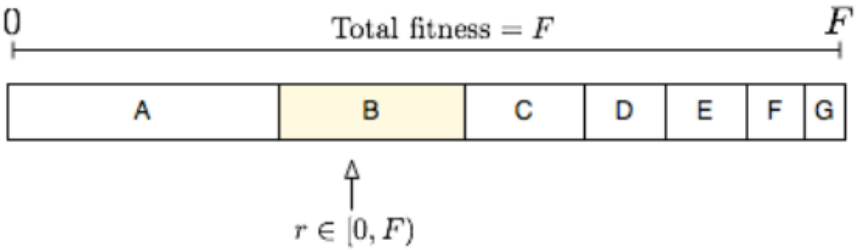
Fitness proportionate selection, also known as **roulette wheel selection**, is a genetic operator used in genetic algorithms for selecting potentially useful solutions for recombination.

In all selection methods, the fitness function assigns a fitness to possible solutions or chromosomes. This fitness level is used to associate a probability of selection with each individual chromosome. If f_i is the fitness of individual i in the population, its probability of being selected is:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

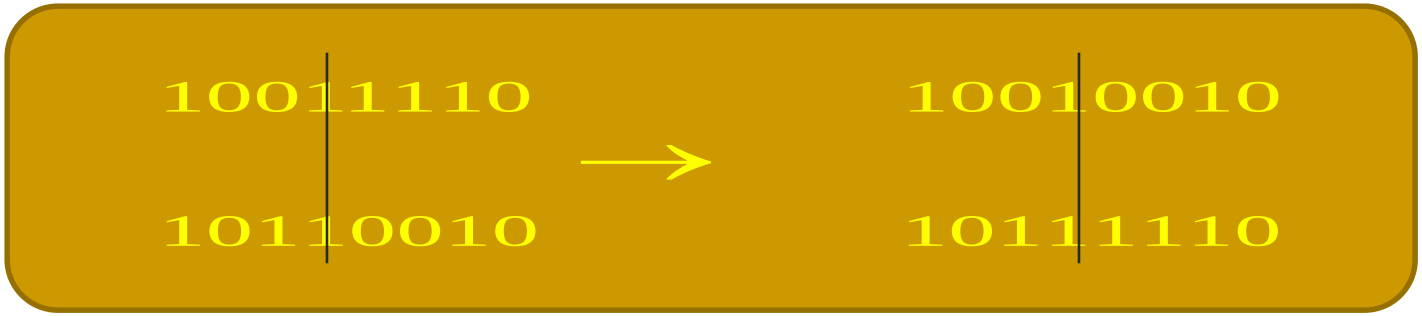
where N is the number of individuals in the population

Example of the selection of a single individual

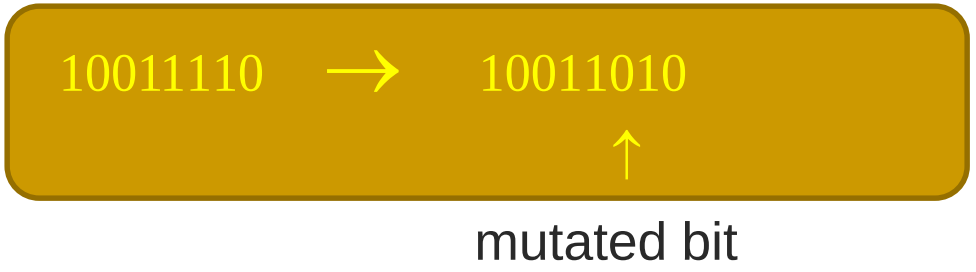


Crossover and Mutation Operators

- **Explore** new solutions.
- Crossover: information exchange between points.

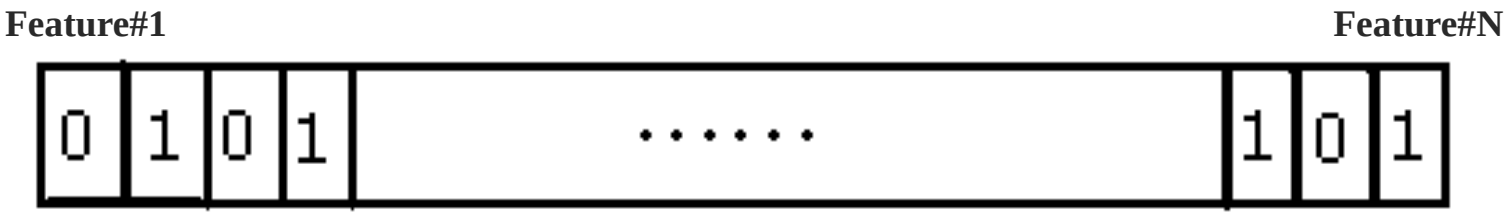


- Mutation: restore lost genetic material.



Genetic Feature Subset Selection

- Binary encoding



- Fitness evaluation

$$fitness = 10^4 \times accuracy + 0.4 \times \#zeros$$

accuracy from validation set

number of features

Class Separability Measures

The emphasis so far was on individually considered features. However, such an approach cannot take into account existing correlations among the features. That is, two features may be rich in information, but if they are highly correlated we need not consider both of them. To this end, in order to search for possible correlations, we consider features jointly as elements of vectors.

- Discard poor in information features, by means of a statistical test.

- Choose the maximum number of features to be used. This is dictated by the specific problem (e.g., the number, N, of available training patterns and the type of the classifier to be adopted).

Class Separability Measures

- Combine remaining features to search for the “best” combination. To this end:

  - Use different feature combinations to form the feature vector. Train the classifier, and choose the combination resulting in the best classifier performance.

    A major disadvantage of this approach is the high complexity. Also, local minima, may give misleading results.

  - Adopt a class separability measure and choose the best feature combination against this cost.

Class Separability Measures

- Let $\underline{x}$ be the current feature combination vector.

- Divergence. To see the rationale behind this cost, consider the two – class case. Obviously, if on the average the value of $\ln \frac{p(\underline{x}|\omega_1)}{p(\underline{x}|\omega_2)}$ is close to zero, then $\underline{x}$ should be a poor feature combination. Define:

  - $$D_{12} = \int_{-\infty}^{+\infty} p(\underline{x}|\omega_1) \ln \frac{p(\underline{x}|\omega_1)}{p(\underline{x}|\omega_2)} d\underline{x}$$

  Kullback-Leibler divergence of distribution of class 2 from class1 or it is also called the **relative Entropy** of distribution in class1 with respect to class2

  - $$D_{21} = \int_{-\infty}^{+\infty} p(\underline{x}|\omega_2) \ln \frac{p(\underline{x}|\omega_2)}{p(\underline{x}|\omega_1)} d\underline{x}$$

  - $$d_{12} = d_{21} = D_{12} + D_{21}$$

d_{12} is known as the divergence and can be used as a class separability measure.

Class Separability Measures

- For the multi-class case, define d_{ij} for every pair of classes ω_i, ω_j and the average divergence is defined as

$$d = \sum_{i=1}^{M} \sum_{j=1}^{M} P(\omega_i) P(\omega_j) d_{ij}$$

- Some properties:

$$d_{ij} \geq 0$$

$$d_{ij} = 0, \text{ if } i = j$$

$$d_{ij} = d_{ji}$$

- Large values of d are indicative of good feature combination.

Class Separability Measures

- Scatter Matrices. These are used as a measure of the way data are scattered in the respective feature space.
  - Within-class scatter matrix

$$S_w = \sum_{i=1}^{M} P_i S_i$$

where

$$S_i = E\left[(\underline{x} - \underline{\mu}_i)(\underline{x} - \underline{\mu}_i)^T\right]$$

and

$$P_i \equiv P(\omega_i) \approx \frac{n_i}{N}$$

n_i the number of training samples in ω_i.

Trace $\{S_w\}$ is a measure of the average variance of the features.

Class Separability Measures

- Between-class scatter matrix

$$S_b = \sum_{i=1}^{M} P_i \left(\underline{\mu}_i - \underline{\mu}_0\right)\left(\underline{\mu}_i - \underline{\mu}_0\right)^T$$

$$\underline{\mu}_0 = \sum_{i=1}^{M} P_i \underline{\mu}_i$$

Trace $\{S_b\}$ is a measure of the average distance of the mean of each class from the respective global one.

- Mixture scatter matrix

$$S_m = E\left[\left(\underline{x} - \underline{\mu}_0\right)\left(\underline{x} - \underline{\mu}_0\right)^T\right]$$

It turns out that:

$$S_m = S_w + S_b$$

Class Separability Measures

- Measures based on Scatter Matrices.

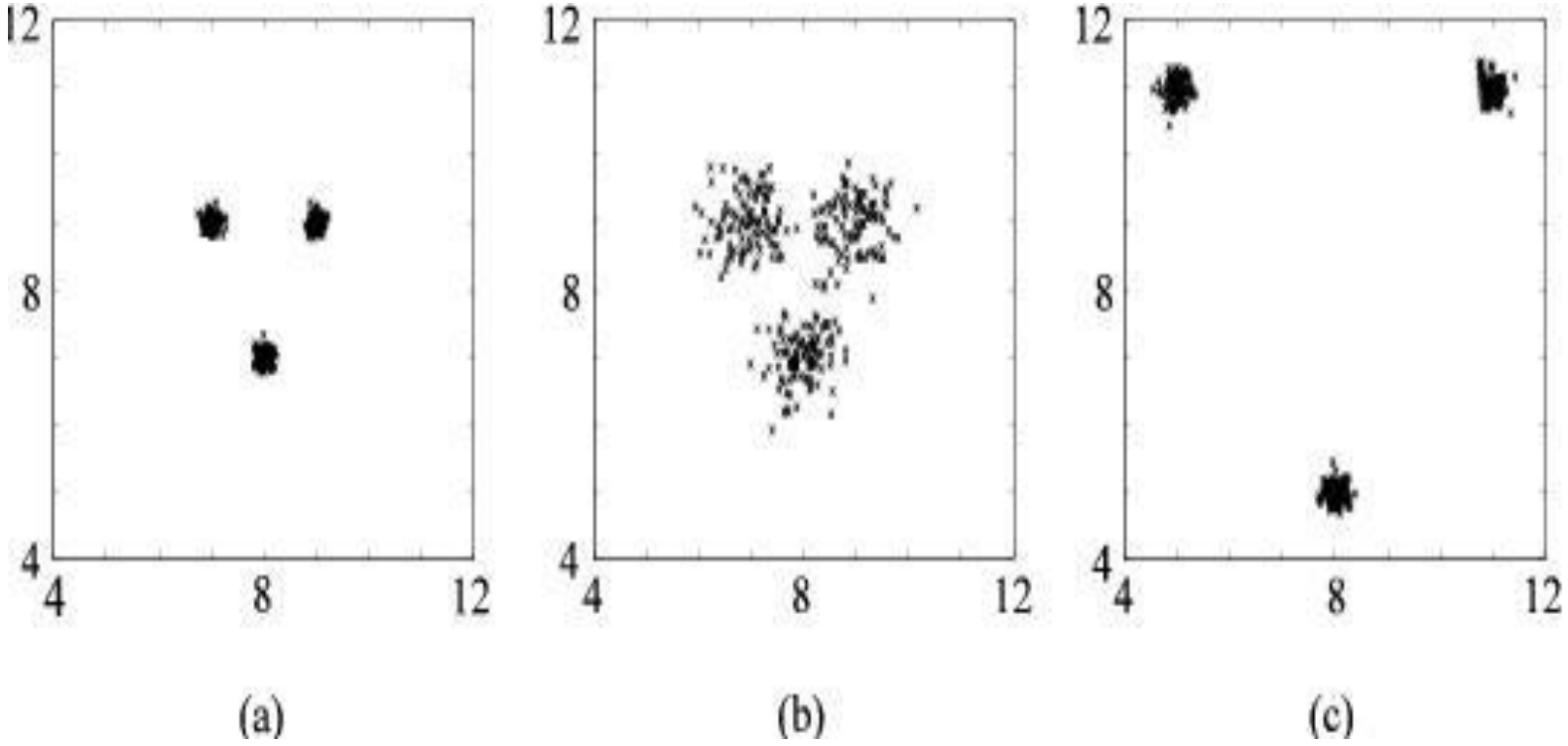
$$J_1 = \frac{\text{Trace}\{S_m\}}{\text{Trace}\{S_w\}} \qquad J_2 = \frac{|S_m|}{|S_w|} = |S_w^{-1} S_m| \qquad J_3 = \text{Trace}\{S_w^{-1} S_m\}$$

  - Other criteria are also possible, by using various combinations of S_m, S_b, S_w.

The above J_1, J_2, J_3 criteria take high values for the cases where:

- Data are clustered together within each class.
- The means of the various classes are far.

Class Separability Measures



Class Separability Measures

- Fisher's discriminant ratio. In one dimension and for two equiprobable classes the determinants become:

$$|S_w| \propto \sigma_1^2 + \sigma_2^2$$

$$|S_b| \propto (\mu_1 - \mu_2)^2$$

and

$$\frac{|S_b|}{|S_w|} = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$

known as Fischer's ratio.

Hints from Generalization Theory

Generalization theory aims at providing general bounds that relate the error performance of a classifier with the number of training points, N, on one hand, and some classifier dependent parameters, on the other. Up to now, the classifier dependent parameters that we considered were the number of its free parameters and the dimensionality, ℓ , of the subspace, in which the classifier operates. (ℓ also affects the number of free parameters).

- Definitions
  - Let the classifier be a binary one, i.e., $f : \Re^{\ell} \rightarrow \{0,1\}$
  - Let F be the set of all functions f that can be realized by the adopted classifier (e.g., changing the synapses of a given neural network different functions are implemented).

Hints from Generalization Theory

- The shatter coefficient $S(F,N)$ of the class F is defined as:

  the **maximum** number of dichotomies of N points that can be formed by the functions in $\boldsymbol{F}$.

  The maximum possible number of dichotomies is 2^N. However, NOT ALL dichotomies can be realized by the set of functions in $\boldsymbol{F}$.

- The Vapnik – Chernovenkis (VC) dimension of a class $\boldsymbol{F}$ is the largest integer k for which $S(F,k) = 2^k$. If $S(F,N)=2^N$, $\forall N$, we say that the VC dimension is infinite.
  - That is, VC is the integer for which the class of functions F can achieve all possible dichotomies, 2^k.
  - It is easily seen that the VC dimension of the single perceptron class, operating in the ℓ-dimensional space, is $\ell+1$.

Hints from Generalization Theory

- It can be shown that $S(F,N) \leq N^{V_c} + 1$

  V_c: the VC dimension of the class.

  That is, the shatter coefficient is either 2^N (the maximum possible number of dichotomies) or it is upper bounded, as suggested by the above inequality.

  In words, for finite V_c and large enough N, the **shatter coefficient is bounded by a polynomial growth**.

  - Note that in order to have a polynomial growth of the shatter coefficient, N must be larger than the V_c dimension.

- The V_c dimension can be considered as an **intrinsic capacity** of the classifier, and, as we will soon see, only if the number of training vectors **exceeds** this number sufficiently, we can expect good generalization performance.

Hints from Generalization Theory

- The V_c dimension may or may **not** be related to the dimension ℓ and the number of free parameters.
  - Perceptron: $V_c = \ell + 1$
  - Multilayer perceptron with hard limiting activation function

$$2\left\lfloor \frac{k_n^h}{2} \right\rfloor \ell \leq V_c \leq 2k_w \log_2(ek_n)$$

where k_n^h is the total number of hidden layer nodes, k_n the total number of nodes, and k_w the total number of weights.

  - Let $\underline{x}_i$ be a training data sample and assume that

$$\|\underline{x}_i\| \leq r, i = 1, 2, ..., N$$

Hints from Generalization Theory

Let also a hyperplane such that

$\|w\|^2 \leq c$ and $y_i\left(\underline{w}^T \underline{x}_i + b\right) \geq 1$

(i.e., the constraints we met in the SVM formulation). Then

$$V_c \leq \left(r^2 c, \ell\right)$$

That is, by controlling the constant c, the V_c of the linear classifier can be less than ℓ . In other words, V_c can be controlled independently of the dimension.

Thus, by minimizing $\|w\|^2$ in the SVM, one attempts to keep V_c as small as possible. Moreover, one can achieve **finite** V_c dimension, **even for infinite dimensional spaces**. This is an explanation of the potential for good generalization performance of the SVM's, as this is readily deduced from the following bounds.

Hints from Generalization Theory

- Generalization Performance
  - Let $P_e^N(f)$ be the error rate of classifier f, based on the N training points, also known as empirical error.
  - Let $P_e(f)$ be the true error probability of f (also known as generalization error), when f is confronted with data outside the finite training set.
  - Let P_e be the minimum error probability that can be attained over ALL functions in the set $\boldsymbol{F}$.

Hints from Generalization Theory

- Let f^* be the function resulting by minimizing the empirical (over the finite training set) error function.

- It can be shown that:

  - $$\text{prob}\left\{\max_{f\in F}\left(P_e^N(f)-P_e(f)\right)>\varepsilon\right\}\le 8S(F,N)\exp\left(-\frac{N\varepsilon^2}{32}\right)$$

  - $$\text{prob}\left\{P_e(f^*)-P_e>\varepsilon\right\}\le 8S(F,N)\exp\left(-\frac{N\varepsilon^2}{128}\right)$$

  - Taking into account that for finite V_c dimension, the growth of $S(F,N)$ is only polynomial, the above bounds tell us that for a large N :
    - $P_e^N(f)$ is close to $P_e(f)$, with high probability.
    - $P_e(f^*)$ is close to P_e , with high probability.

Hints from Generalization Theory

- Some more useful bounds
  - The minimum number of points, $N(\varepsilon,\rho)$, that guarantees, with high probability, a good generalization error performance is given by

$$N(\varepsilon,\rho)\leq \max\left\{\frac{k_1V_c}{\varepsilon^2}\ln\frac{k_2V_c}{\varepsilon^2},\frac{k_3}{\varepsilon^2}\ln\frac{8}{\rho}\right\}$$

That is, for any $N\geq N(\varepsilon,\rho)$

$$\text{prob}\{P_e(f)-P_e>\varepsilon\}\leq\rho$$

Where, k_1,k_2,k_3 constants. In words, for $N\geq N(\varepsilon,\rho)$ the performance of the classifier is guaranteed, with high probability, to be close to the optimal classifier in the class F. $N(\varepsilon,\rho)$ is known as the sample complexity.

Hints from Generalization Theory

- With a probability of at least $1-\rho$ the following bound holds:

$$P_e(f) \leq P_e^N(f) + \Phi\left(\frac{V_c}{N}\right)$$

where

$$\Phi\left(\frac{V_c}{N}\right) = \sqrt{\frac{V_c\left(\ln\left(\frac{2N}{V_c}+1\right) - \ln\left(\frac{\rho}{4}\right)\right)}{N}}$$

- Remark: Observe that all the bounds given so far are:
  - Dimension free
  - Distribution free

Hints from Generalization Theory

- Model Complexity vs Performance

This issue has already been touched in the form of overfitting in neural networks modeling and in the form of bias-variance dilemma. A different perspective of the issue is dealt below.

- Structural Risk Minimization (SRM)
  - Let P_B be he Bayesian error probability for a given task.
  - Let $P_e(f^*)$ be the true (generalization) error of an optimally design classifier f^*, from class F , given a finite training set.
  - $P_e(f^*) - P_B = \left(P_e(f^*) - P_e\right) + \left(P_e - P_B\right)$

    P_e is the minimum error attainable in F
    - If the class F is small, then the first term is expected to be small and the second term is expected to be large. The opposite is true when the class F is large

Hints from Generalization Theory

- Let $F^{(1)}, F^{(2)}, \ldots$ be a sequence of nested classes:

$$F^{(1)} \subset F^{(2)} \subset \ldots$$

with increasing, yet finite V_c dimensions $V_{c,F^{(1)}} \leq V_{c,F^{(2)}} \leq \ldots$

Also, let $\lim_{i \to \infty} \inf_{f \in F^{(i)}} P_e(f) = P_B$

For each N and class of functions $F^{(i)}$, i=1, 2, …, compute the optimum $f^*_{N,i}$, with respect to the empirical error. Then from all these classifiers choose the one than minimizes, over all i, the upper bound in:

That is,

$$P_e(f^*_{\mathrm{N},\iota}) \leq P_e^N(f^*_{\mathrm{N},\iota}) + \Phi\left(\frac{V_{c,F^{(i)}}}{N}\right) \qquad f^*_N = \arg\min_i \left[P_e^N(f^*_{\mathrm{N},\iota}) + \Phi\left(\frac{V_{c,F^{(i)}}}{N}\right) \right]$$

- Then, as $N \to \infty$

$$P_e(f_N^*) \to P_B$$

  - The term

$$\Phi\left(\frac{V_{c,F^{(i)}}}{N}\right)$$

    in the minimized bound is a complexity penalty term. If the classifier model is **simple** the penalty term is **small** but the empirical error term $P_e^N(f_{\mathrm{N},\iota}^*)$

    will be **large**. The opposite is true for complex models.

- The SRM criterion aims at achieving the best trade-off between performance and complexity.

Hints from Generalization Theory

- Bayesian Information Criterion (BIC)

  Let N the size of the training set, $\underline{\theta}_m$ the vector of the unknown parameters of the classifier, K_m the dimensionality of $\underline{\theta}_m$, and m runs over all possible models.

  - The BIC criterion chooses the model by minimizing:

$$BIC = -2L\left(\hat{\underline{\theta}}_m\right) + K_m \ln N$$

    - $L\left(\hat{\underline{\theta}}_m\right)$ is the log-likelihood computed at the ML estimate $\hat{\underline{\theta}}_m$, and it is the performance index.
    - $K_m \ln N$ is the model complexity term.
  - Akaike Information Criterion:

$$AIC = -2L\left(\hat{\underline{\theta}}_m\right) + 2K_m$$

Case Study 1: **Gender Classification**

Z. Sun, G. Bebis, X. Yuan, and S. Louis, "Genetic Feature Subset Selection for Gender Classification: A Comparison Study", **IEEE Workshop on Applications of Computer Vision**, pp. 165-170, Orlando, December 2002.

- Determine the gender of a subject from facial images.
  - Race, age, facial expression, hair style, etc.

- Ways to combine features:

Trying to form all possible combinations of ℓ features from an original set of m selected features is a computationally hard task. Thus, a number of suboptimal searching techniques have been derived.

  - Sequential forward selection. Let x_1, x_2, x_3, x_4 the available features (m=4). The procedure consists of the following steps:
    - Adopt a class separability criterion (could also be the error rate of the respective classifier). Compute its value for ALL features considered jointly $[x_1, x_2, x_3, x_4]^T$.
    - Eliminate one feature and for each of the possible resulting combinations, that is $[x_1, x_2, x_3]^T$, $[x_1, x_2, x_4]^T$, $[x_1, x_3, x_4]^T$, $[x_2, x_3, x_4]^T$, compute the class reparability criterion value C. Select the best combination, say $[x_1, x_2, x_3]^T$.

- From the above selected feature vector eliminate one feature and for each of the resulting combinations, $[x_1, x_2]^T$, $[x_2, x_3]^T$ $[x_1, x_3]^T$ compute and select the best combination.

The above selection procedure shows how one can start from features and end up with the "best" ones. Obviously, the choice is suboptimal. The number of required calculations is:

$$1+\frac{1}{2}\left((m+1)m-\ell(\ell+1)\right)$$

In contrast, a full search requires:

$$\binom{m}{\ell}=\frac{m!}{\ell!(m-\ell)!}$$

operations.

- Sequential backward selection. Here the reverse procedure is followed.
  - Compute C for each feature. Select the "best" one, say x_1
  - For all possible 2D combinations of x_1, i.e., $[x_1, x_2]$, $[x_1, x_3]$, $[x_1, x_4]$ compute C and choose the best, say $[x_1, x_3]$.
  - For all possible 3D combinations of $[x_1, x_3]$, e.g., $[x_1, x_3, x_2]$, etc., compute C and choose the best one.

The above procedure is repeated till the "best" vector with ℓ features has been formed. This is also a suboptimal technique, requiring:

$$\ell m - \frac{\ell(\ell-1)}{2}$$

operations.

Floating Search Methods

The above two procedures suffer from the nesting effect. Once a bad choice has been done, there is no way to reconsider it in the following steps.

In the floating search methods one is given the opportunity in reconsidering a previously discarded feature or to discard a feature that was previously chosen.

The method is still suboptimal, however it leads to improved performance, at the expense of complexity.

Remarks:

- Besides suboptimal techniques, some optimal searching techniques can also be used, provided that the optimizing cost has certain properties, e.g., monotonic.

- Instead of using a class separability measure (filter techniques) or using directly the classifier (wrapper techniques), one can modify the cost function of the classifier appropriately, so that to perform feature selection and classifier design in a single step (embedded) method.

- For the choice of the separability measure a multiplicity of costs have been proposed, including information theoretic costs.