L11: sequential feature selection

Feature extraction vs. feature selection

Search strategy and objective functions

Objective functions

- Filters
- Wrappers

Sequential search strategies

- Sequential forward selection
- Sequential backward selection
- Plus-l minus-r selection
- Bidirectional search
- Floating search

Feature extraction vs. feature selection

As discussed in L9, there are two general approaches to dim. reduction

- **Feature extraction**: Transform the existing features into a lower dimensional space
- **Feature selection**: Select a subset of the existing features without a transformation

$$\begin{bmatrix} x_1 \\ x_2 \\ \\ x_N \end{bmatrix} \rightarrow \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \\ x_{i_M} \end{bmatrix} \qquad \begin{bmatrix} x_1 \\ x_2 \\ \\ x_N \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \\ y_M \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \\ x_N \end{bmatrix} \right)$$

Feature extraction was covered in L9-10

- We derived the "optimal" linear features for two objective functions
  - Signal representation: PCA
  - Signal classification: LDA

Feature selection, also called feature subset selection (FSS) in the literature, will be the subject of the last two lectures

- Although FSS can be thought of as a special case of feature extraction (think of a sparse projection matrix with a few ones), in practice it is a quite different problem
- FSS looks at the issue of dimensionality reduction from a different perspective
- FSS has a unique set of methodologies

Feature subset selection

Definition

- Given a feature set $X = \{x_i \mid i = 1 \ldots N\}$, find a subset Y_M, with $M < N$, that maximizes an objective function $J(Y)$, ideally $P(correct)$

$$Y_M = \{x_{i1}, x_{i2}, \ldots, x_{iM}\} = \underset{M, i_M}{\arg\max}\, J\{x_i \mid i = 1..N\}$$

Why feature subset selection?

- Why not use the more general feature extraction methods, and simply project a high-dimensional feature vector onto a low-dimensional space?

Feature subset selection is necessary in a number of situations

- 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 project, the measurement units of your features (length, weight, etc.) are lost
- Features may not be numeric, a typical situation in machine learning

In addition, fewer features means fewer model parameters

- Improved the generalization capabilities
- Reduced complexity and run-time

Search strategy and objective function

FSS requires

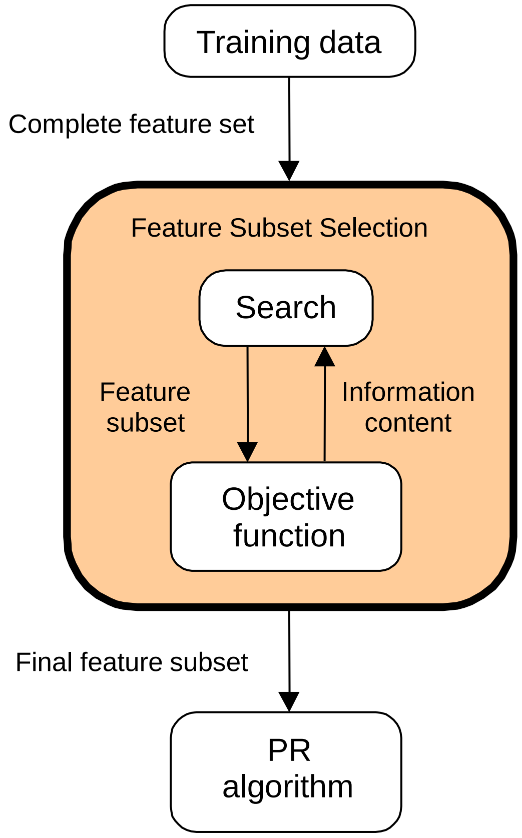
- A search strategy to select candidate subsets
- An objective function to evaluate these candidates

Search strategy

- Exhaustive evaluation of feature subsets involves $\binom{N}{M}$ combinations for a fixed value of M, and 2^N combinations if M must be optimized as well
  - This number of combinations is unfeasible, even for moderate values of M and N, so a search procedure must be used in practice
  - For example, exhaustive evaluation of 10 out of 20 features involves 184,756 feature subsets; exhaustive evaluation of 10 out of 100 involves more than 10^{13} feature subsets [Devijver and Kittler, 1982]
- A search strategy is therefore needed to direct the FSS process as it explores the space of all possible combination of features

Objective function

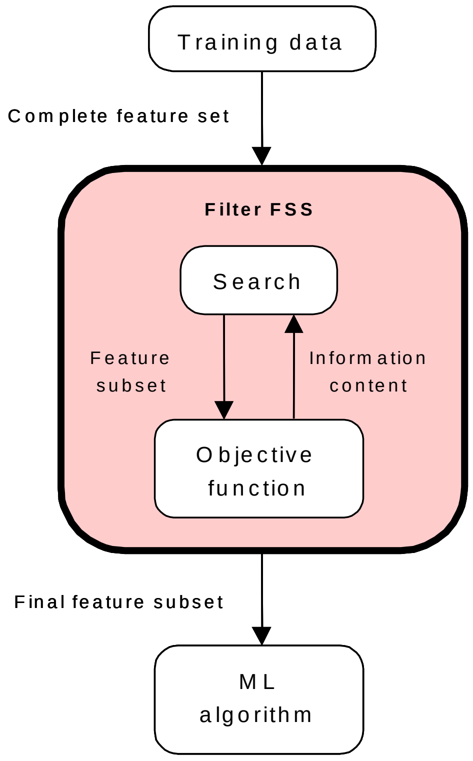
- The objective function evaluates candidate subsets and returns a measure of their "goodness", a feedback signal used by the search strategy to select new candidates

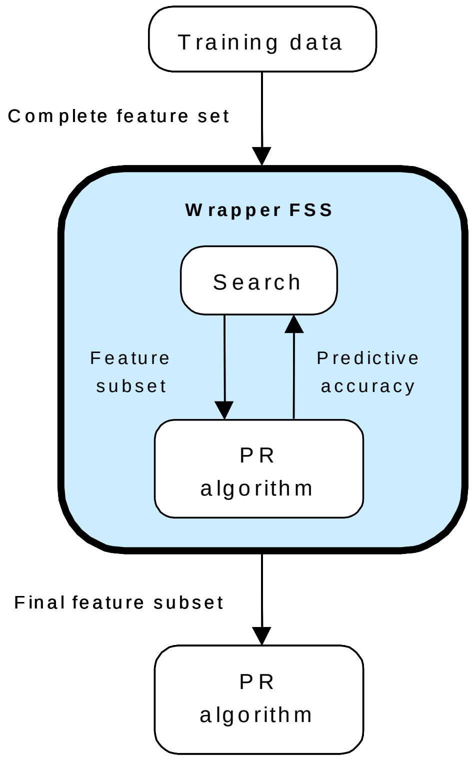


Objective function

Objective functions are divided in two groups

- **Filters**: evaluate subsets by their information content, e.g., interclass distance, statistical dependence or information-theoretic measures
- **Wrappers**: use a classifier to evaluate subsets by their predictive accuracy (on test data) by statistical resampling or cross-validation





Filter types

Distance or separability measures

- These methods measure class separability using metrics such as
  - Distance between classes: Euclidean, Mahalanobis, etc.
  - Determinant of $S_W^{-1}S_B$ (LDA eigenvalues)

Correlation and information-theoretic measures

- These methods are based on the rationale that good feature subsets contain features highly correlated with (predictive of) the class, yet uncorrelated with (not predictive of) each other
- **Linear relation measures**
  - Linear relationship between variables can be measured using the correlation coefficient

$$J(Y_M) = \frac{\sum_{i=1}^{M} \rho_{ic}}{\sum_{i=1}^{M} \sum_{j=i+1}^{M} \rho_{ij}}$$

  - Where ρ_{ic} is the correlation coefficient between feature i and the class label and ρ_{ij} is the correlation coefficient between features i and j

- **Non-linear relation measures**
  - Correlation is only capable of measuring linear dependence
  - A more powerful measure is the mutual information $I(Y_k; C)$

$$J(Y_M) = I(Y_M; C) = H(C) - H(C|Y_M) = \sum_{c=1}^{C} \int_{Y_M} p(Y_M, \omega_c) log \frac{p(Y_M, \omega_c)}{p(Y_M)P(\omega_c)} dx$$

  - The mutual information between the feature vector and the class label $I(Y_M; C)$ measures the amount by which the uncertainty in the class $H(C)$ is decreased by knowledge of the feature vector $H(C|Y_M)$, where $H(\cdot)$ is the entropy function
  - Note that mutual information requires the computation of the multivariate densities $p(Y_M)$ and $p(Y_M, \omega_c)$, which is ill-posed for high-dimensional spaces
  - In practice [Battiti, 1994], mutual information is replaced by a heuristic such as

$$J(Y_M) = \sum_{m=1}^{M} I\left(x_{i_m}; C\right) - \beta \sum_{m=1}^{M} \sum_{n=m+1}^{M} I\left(x_{i_m}; x_{i_n}\right)$$

Filters vs. wrappers

Filters

- **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
- **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

Wrappers

- **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
- **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

Search strategies

Exponential algorithms (Lecture 12)

- Evaluate a number of subsets that grows exponentially with the dimensionality of the search space
  - Exhaustive Search (already discussed)
  - Branch and Bound
  - Approximate Monotonicity with Branch and Bound
  - Beam Search

Sequential algorithms (Lecture 11)

- Add or remove features sequentially, but have a tendency to become trapped in local minima
  - Sequential Forward Selection
  - Sequential Backward Selection
  - Plus-l Minus-r Selection
  - Bidirectional Search
  - Sequential Floating Selection

Randomized algorithms (Lecture 12)

- Incorporate randomness into their search procedure to escape local minima
  - Random Generation plus Sequential Selection
  - Simulated Annealing
  - Genetic Algorithms

Naïve sequential feature selection

One may be tempted to evaluate each individual feature separately and select the best M features

– Unfortunately, this strategy RARELY works since it does not account for feature dependence

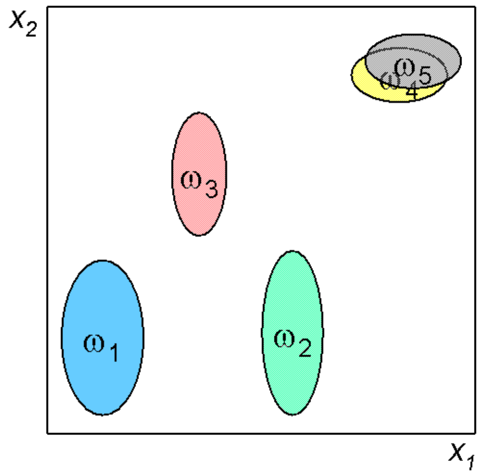
Example

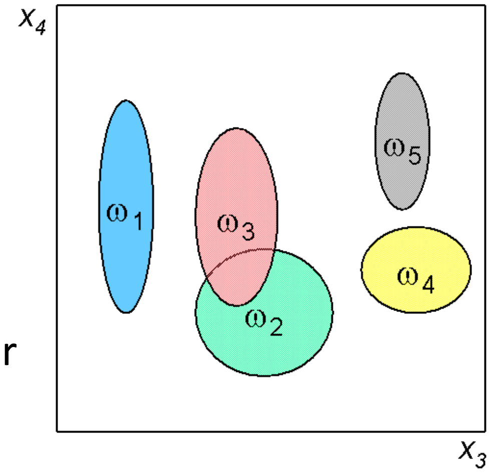
– The figures show a 4D problem with 5 classes

– 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 the best feature: it separates ω_1, ω_2, ω_3 and $\{\omega_4, \omega_5\}$
  - x_2 and x_3 are equivalent, and separate classes in three groups
  - x_4 is the worst feature: it can only separate ω_4 from ω_5

– 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 certainly pick x_1 and either x_2 or x_3, leaving classes ω_4 and ω_5 non separable
  - This naïve strategy fails because it does not consider features with complementary information



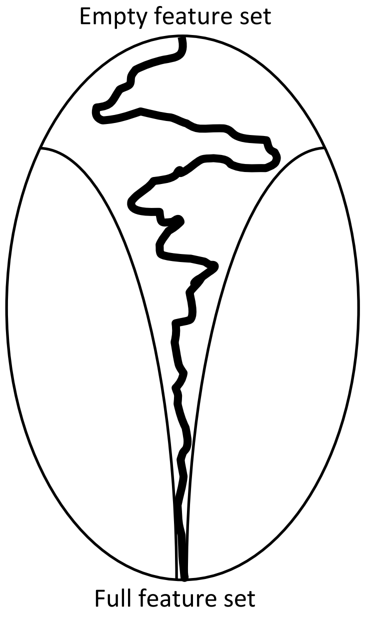


Sequential forward selection (SFS)

SFS is the simplest greedy search algorithm

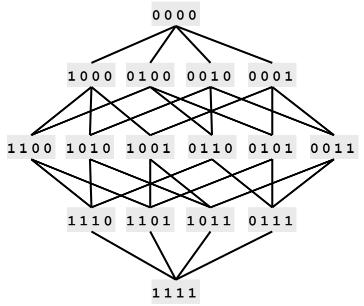
- Starting from the empty set, sequentially add the feature x^+ that maximizes $J(Y_k + x^+)$ when combined with the features Y_k that have already been selected

```
1. Start with the empty set $Y_0 = \{\emptyset\}$
2. Select the next best feature $x^+ = \arg\max_{x \notin Y_k} J(Y_k + x)$
3. Update $Y_{k+1} = Y_k + x^+;\ k = k + 1$
4. Go to 2
```



Notes

- SFS performs best when the optimal subset is small
  - 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 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
  - The main disadvantage of SFS is that it is unable to remove features that become obsolete after the addition of other features



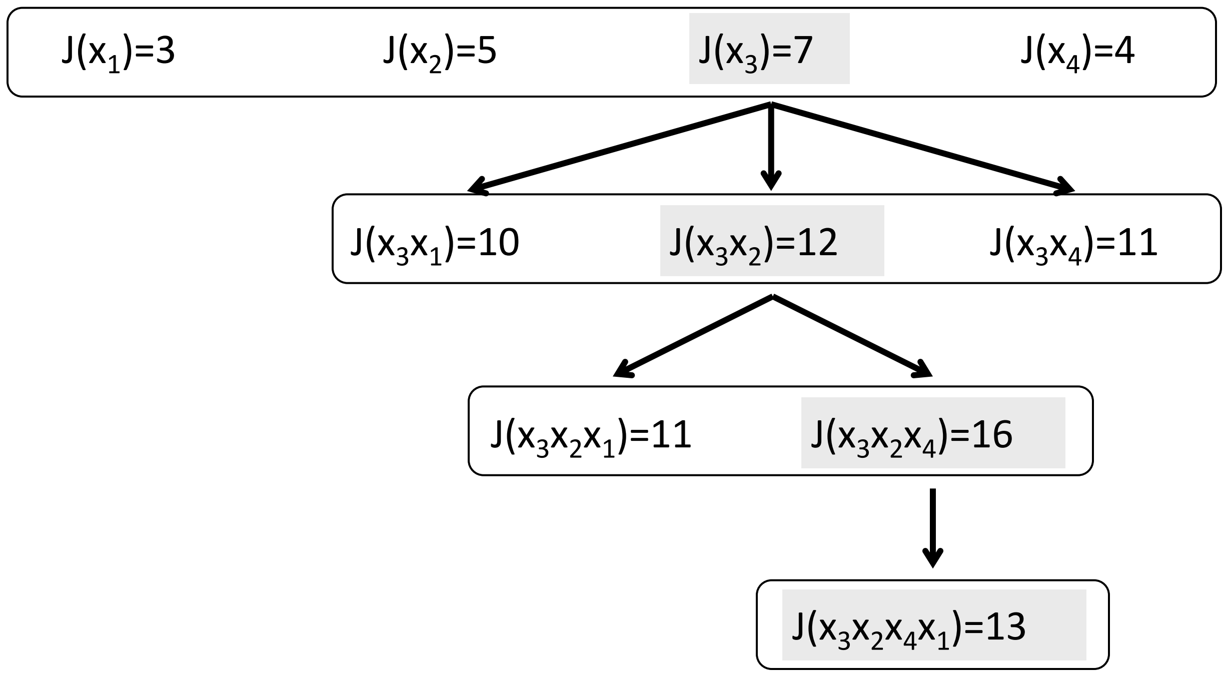
Example

Run SFS to completion for the following objective function

$$J(X) = -2x_1x_2 + 3x_1 + 5x_2 - 2x_1x_2x_3 + 7x_3 + 4x_4 - 2x_1x_2x_3x_4$$

- where x_k are indicator variables, which indicate whether the k^{th} feature has been selected $(x_k = 1)$ or not $(x_k = 0)$

Solution

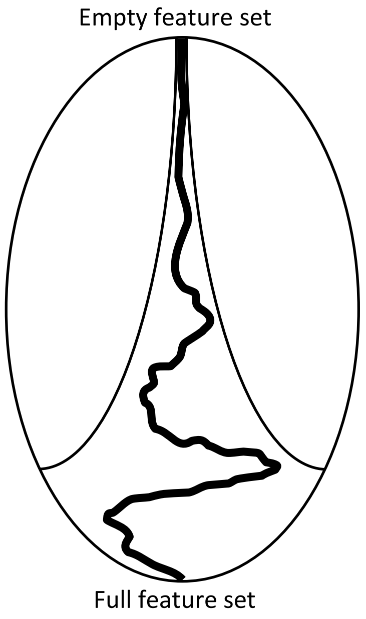


Sequential backward selection (SBS)

SBS works in the opposite direction of SFS

- Starting from the full set, sequentially remove the feature x^- that least reduces the value of the objective function $J(Y - x^-)$
  - Removing a feature may actually increase 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)

```
1. Start with the full set $Y_0 = X$
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 is large, 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

Plus-L minus-R selection (LRS)

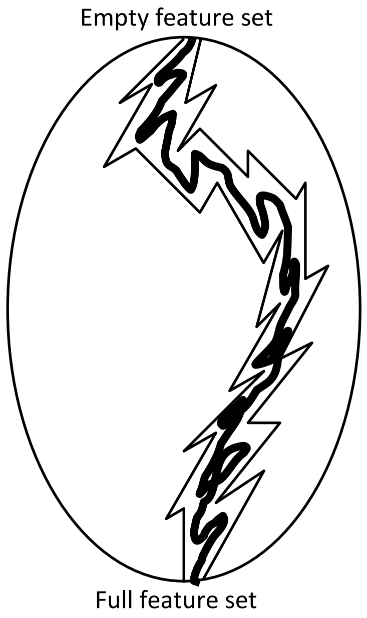
A generalization of SFS and SBS

- 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 followed by L additions

```
1. If L>R   then $Y_0 = \{\emptyset\}$
            else $Y_0 = X$; go to step 3
2. Repeat L times
            $x^+ = \arg\max_{x \notin Y_k} J(Y_k + x)$
            $Y_{k+1} = Y_k + x^+;\ k = k + 1$
3. Repeat R times
            $x^- = \arg\max_{x \in Y_k} J(Y_k - x)$
            $Y_{k+1} = Y_k - x^-;\ k = k + 1$
4. Go to 2
```

Notes

- LRS attempts to compensate for the weaknesses of SFS and SBS with some backtracking capabilities
- Its main limitation is the lack of a theory to help predict the optimal values of L and R

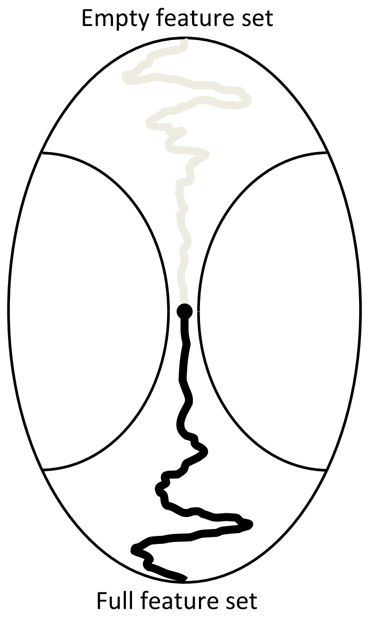


Bidirectional Search (BDS)

BDS 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
  - Features already selected by SFS are not removed by SBS
  - Features already removed by SBS are not selected by SFS

```
1. Start SFS with $Y_F = \{\emptyset\}$
2. Start SBS with $Y_B = X$
3. Select the best feature
        $x^+ = \underset{\substack{x \notin Y_{F_k} \\ x \in F_{B_k}}}{\arg\max} J\left(Y_{F_k} + x\right)$
        $Y_{F_{k+1}} = Y_{F_k} + x^+$
4. Remove the worst feature
        $x^- = \underset{\substack{x \in Y_{B_k} \\ x \notin Y_{F_{k+1}}}}{\arg\max} J\left(Y_{B_k} - x\right)$
        $Y_{B_{k+1}} = Y_{B_k} - x^-;\ k = k + 1$
5. Go to 2
```



Sequential floating selection (SFFS and SFBS)

An extension to LRS 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 thought to be “floating” up and down

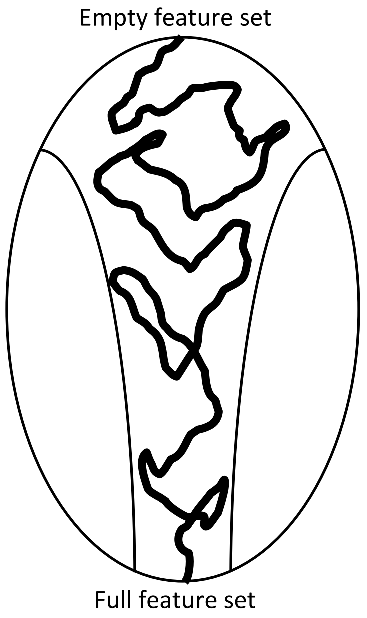
There are two floating methods

- 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. $Y = \{\emptyset\}$
2. Select the best feature
        $x^+ = \arg\max_{x \notin Y_k} J(Y_k + x)$
        $Y_k = Y_k + x^+; k = k + 1$
3. Select the worst feature*
        $x^- = \arg\max_{x \in Y_k} J(Y_k - x)$
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
```

*Notice that you'll need to do book-keeping to avoid infinite loops


Empty feature set
Full feature set