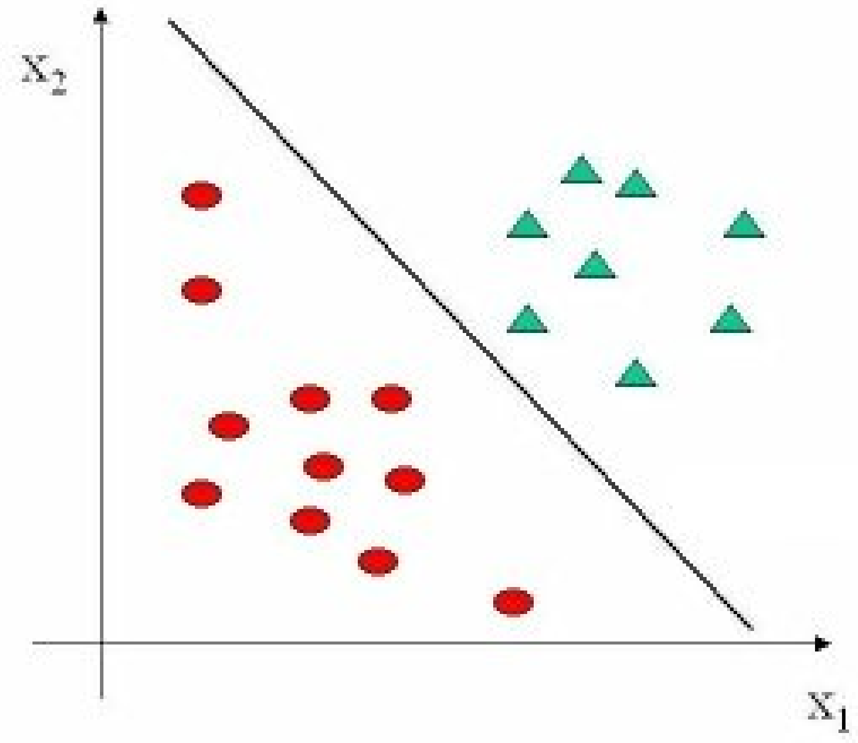
INF 5300 - Feature selection

- Basic classification principles (reminder)
- «Curse of dimensionality»
- Feature subset selection
  - Objective function
  - Search strategy

Sections 5.1, 5.2, 5.5 (5.5.1 and 5.5.2 not too detailed), 5.6 in ”Pattern Recognition” by S. Theodoridis and K. Koutroumbas (see link on course page for pdfs)

Reminder - Basic classification principles I/II



Reminder - Basic classification principles II/II

Classification task:

- Classify object $x = \{x_1, \ldots, x_n\}$ to one of K classes $\omega_1, \ldots \omega_M$
- *Decision rule* f(**x**)=ω_i divides the feature space into K disjoint subsets R_i, i=1,…K.
- The borders between subsets R_i, i=1,…K are defined by K scalar *discriminant functions* $g_1(\mathbf{x}), \ldots, g_K(\mathbf{x})$
- The pattern **x** will be classified to the class whose discriminant function gives a maximum:

  $$d(\mathbf{x})=\omega_i \Leftrightarrow g_i(\mathbf{x}) = \max_{j=1,\ldots K} g_j(\mathbf{x})$$

- Discriminant hypersurfaces are thus defined by

  $$g_i(\mathbf{x})-g_j(\mathbf{x})=0$$

- Training data vs test/unseen data

Key concept!

Reminder - Density-based classifiers I / II

Model/estimate $p(\mathbf{x}|c)$

Prior probability for class c

$$P(c|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|c)P(c)}{p(\mathbf{x}_i)}$$

Here this is our discriminant function

Bayes' rule

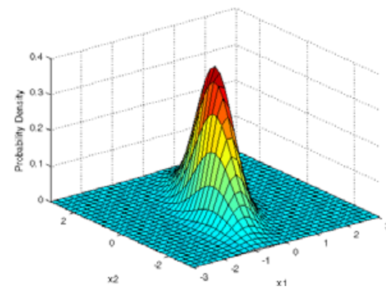
Our decision rule

$$f(\mathbf{x_i}) = \operatorname*{argmax}_c \; p(c|\mathbf{x_i})$$

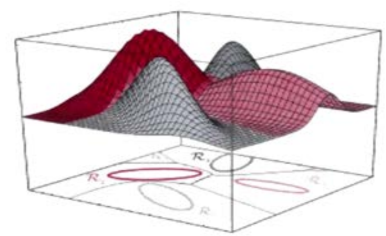
Reminder - Density-based classifiers II / II

$p(\mathbf{x}|c)$

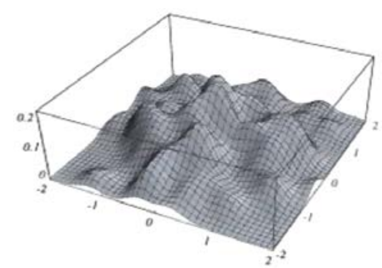
- Gaussian normal
  - Spherical (nearest mean)
  - Uncorrelated (naive Bayes)
  - Model-restrained
  - Equal-shaped classes
  - Full



- Mixture models (mixture of Gaussians)



- Kernel-based (e.g. Parzen-windows)



More complex

* Illustrations from Duda et al

Reminder - Classification with Gaussian distributions

- **x** normally distributed / Gaussian pdf:

$$p(x \mid \omega_s) = \frac{1}{(2\pi)^{d/2} \left| \Sigma_s \right|^{1/2}} \exp\left[-\frac{1}{2} (x - \mu_s)^t \Sigma_s^{-1} (x - \mu_s) \right]$$

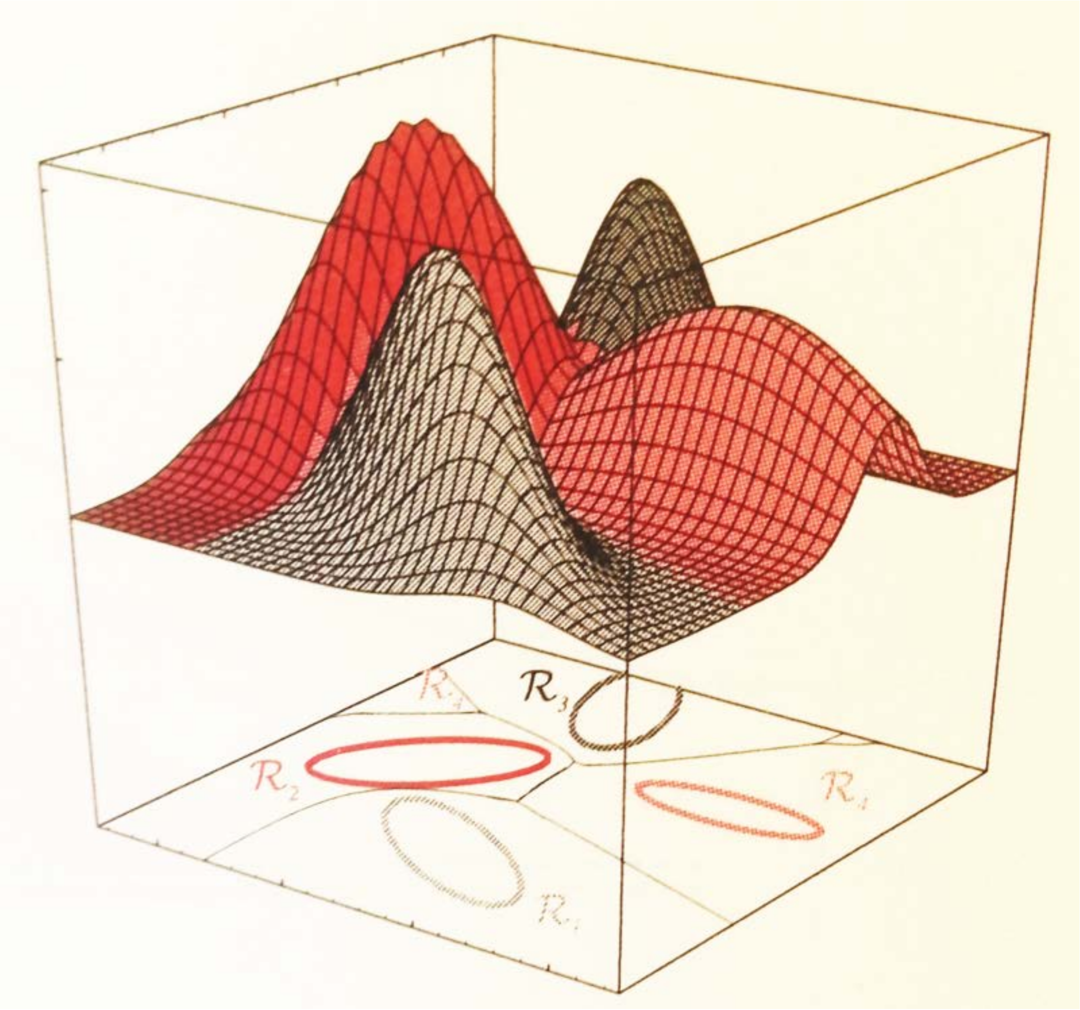
- μ_s and Σ_s are generally not known, often using sample mean and sample covariance:

$$\hat{\mu}_s = \frac{1}{M_s} \sum_{m=1}^{M_s} x_m,$$

$$\hat{\Sigma}_s = \frac{1}{M_s} \sum_{m=1}^{M_s} (x_m - \hat{\mu}_s)(x_m - \hat{\mu}_s)^t$$

where the sum is over all training samples belonging to class s

Reminder – Gaussian distributions | example



(Duda et.al 2000, fig 2.16)

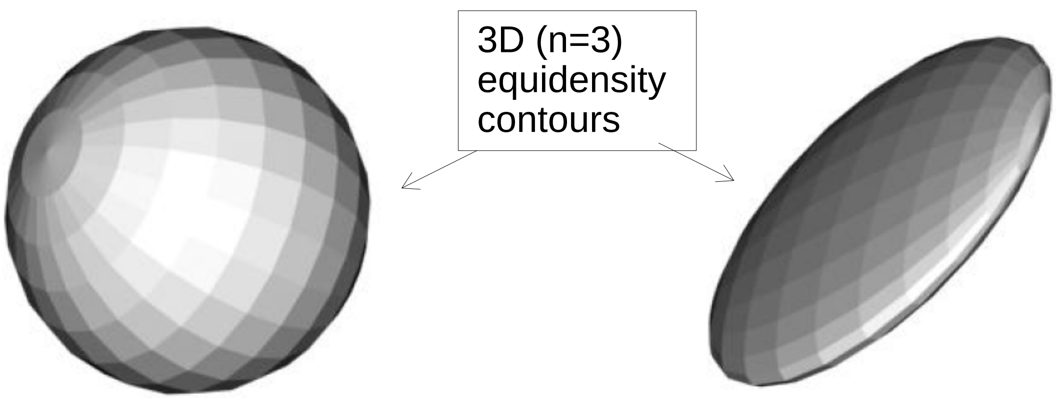
High dimensionality / low sample count

- Even the simple unimodal normal distribution can be too complex

$$g_c(\mathbf{x}) = -\frac{1}{2}\log|\Sigma_c| - \frac{1}{2}(\mathbf{x}-\mu_c)'\Sigma_c^{-1}(\mathbf{x}-\mu_c) + \log \pi_c$$

n elements

n(n+1)/2 elements

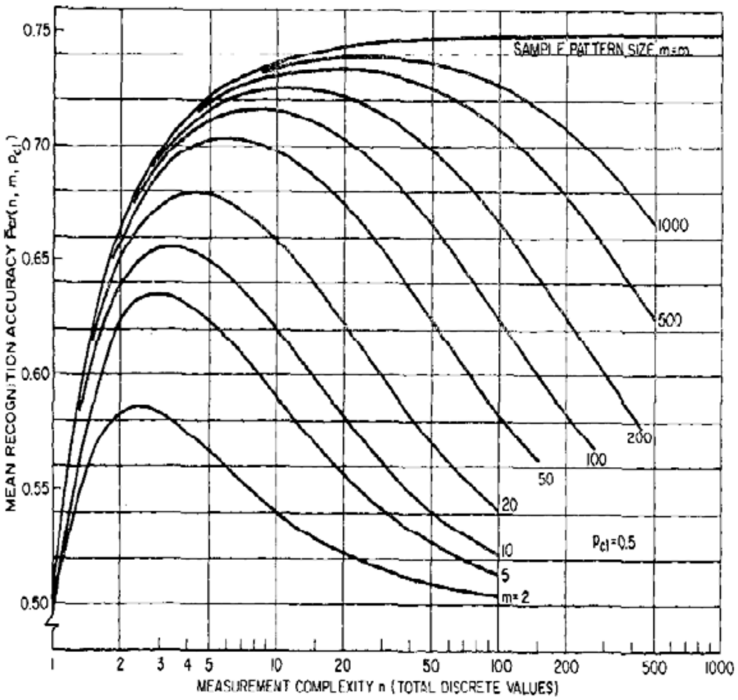


- *Overfits* easily causing poor generalization

Performance on “unseen” data

«Curse of dimensionality» I / II

- Peaking phenomenon
  - Finite number of training samples
  - Adding (even discriminative) features
  - => Eventually worse classification rate
- High dimension -> mostly empty space



Increased dimension of **x** →

* Illustration from Hughes

«Curse of dimensionality» II / II

- Handling the problem
  - Careful feature design to start with!
  - Dimension reduction
    - Feature selection
    - Feature extraction
  - Reduce classifier complexity
    - E.g. assuming diagonal Σ for Guassian-based classifiers
  - Biasing/regularization
    - E.g. diagonal loading for Guassian-based classifiers
  - Sometimes adding unlabeled samples might work (semi-supervised classification)

Today's topic

Cf. slide 5; moving upwards

Somewhat of a digression

Regularized discriminant analysis

Class-specific sample covariance matrix

Common-class sample covariance matrix

$$\hat{\Sigma}_c = (1-\alpha)\tilde{\Sigma}_c + \alpha\big((1-\gamma)\tilde{\Sigma} + \gamma\sigma^2 I\big)$$

Use this as covariance matrix for class *c*

α and γ guides transition between full quadratic classifier, linear classifier and nearest-mean classifier

Scaled identity matrix (think spherical distribution)

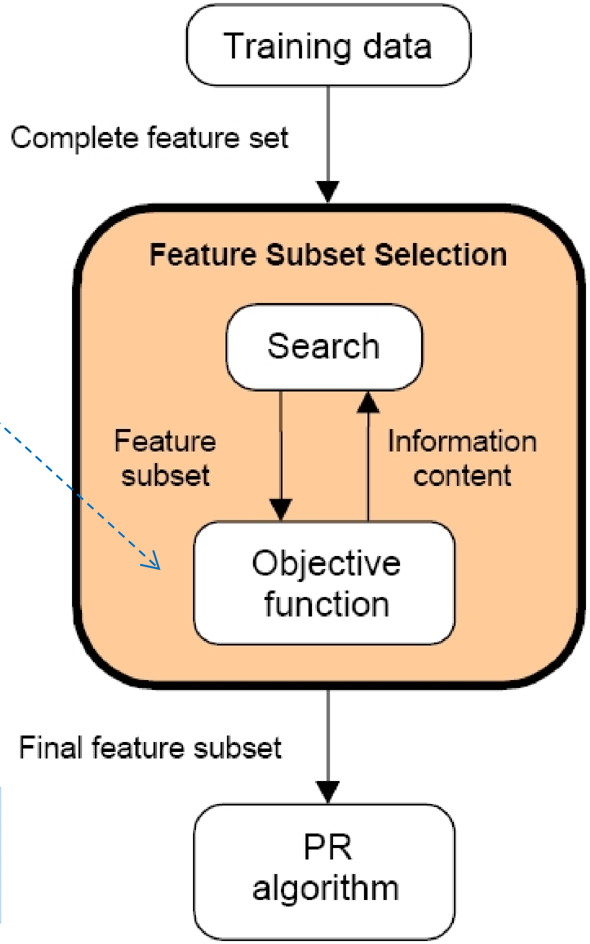
Feature (subset) selection intro

- Why?
  - Enhanced generalisation by reducing overfitting
  - Improved model interpretability
  - Computationally tractable dataset
- Three main approaches:
  - «Wrappers»
    - The optimization criterion is based on building and testing actual classifiers
  - «Filters»
    - Criterion is based on a (simplified) class-separability measure
  - «Embedded methods»
    - The classifier itself induces feature selection, e.g. decision threes

Feature selection

- Want the best m out of n feature subset
- Our search needs an objective function
  - ”Predict” classifier performance
  - Decides how good a subset is
  - Wrapper-based
    - Often good, often slow
    - Linked to specific classifier
  - Filter-based
    - A proxy measure
    - «Simple» function
    - Fast(er)
    - Might be more general

Remember separate training/vadiation set (cf. cross-validation)



Here non-wrapper-based!

Objective functions I/II

- Want a function that can predict good classifier performance
- E.g. for two classes:
  - Euclidean distance between class means $|\mu_1 - \mu_2|$
  - Mahalanobis distance between class means

$$\Delta = (\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)$$

  - Assume Gaussianity and calculate divergence (eq. 5.14 in Theodoridis)
  - Assume Gaussianity and calculate Bhattacharyya distance (linked to minimum attainable Bayes error rate)

$$B = \frac{1}{8}(\mu_r - \mu_s)^T \left(\frac{\Sigma_r + \Sigma_s}{2}\right)^{-1} (\mu_r - \mu_s) + \frac{1}{2}\ln \frac{\left|\frac{1}{2}(\Sigma_r + \Sigma_s)\right|}{\sqrt{|\Sigma_r||\Sigma_s|}}$$

Note!

«Gaussian + common Σ + divergence»

«Divergence»: Here a distance measure between pdfs

Objective functions II/II

- Multiple (>2) classes, e.g.:
  - Average objective-function value of all pairs of classes
  - Smallest value between a pair of classes
  - ...

Search strategies

- Want the best m out of n feature subset
- Exhaustive search implies $\binom{n}{m}$ evaluations if we fix *m*, and 2^n if we need to search all possible *m* as well
  - Choosing *10* out of *100* will result in $\sim 10^{13}$ queries
- Obviously we need to guide the search / use suboptimal search techniques

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

Method 1 - Individual feature selection

- Each feature is treated separately (no correlation/dependence between features is consideren)
  - Select a criterion/objective function, e.g. a distance measure
  - Calculate the objective function, C(k), for each feature k
  - Select the set of features with the best individual criterion value

- Multiclass situations:
  - Average objective functions over pairs of classes or
  - $C(k) = \min C_{i,j}(k)$, over all classes i and j ← Often used

- Advantage with individual selection: Computation time
- Disadvantage: Ignores feature dependence/comlementary information

Method 2 - Sequential backward selection

- Example: Want 2 out of 4 features x_1,x_2,x_3,x_4
  - Choose a criterion/objective function C
  - Eliminate one feature at a time by computing C for $[x_1,x_2,x_3]^T$, $[x_1,x_2,x_4]_T$, $[x_1,x_3,x_4]^T$ and $[x_2,x_3,x_4]^T$
  - Select the best (highest C) combination, say $[x_1,x_2,x_3]^T$.
  - From the selected 3-dimensional feature vector eliminate one more feature, and evaluate the criterion for $[x_1,x_2]^T$, $[x_1,x_3]_T$, $[x_2,x_3]^T$ and select the one with the best value.
- Number of combinations searched when selecting l out of m features

  $1+1/2((m+1)m-l(l+1))$
- Disadvantage:
  - Unable to remove features that become obsolete after including more feature
  - High starting dimensionality might put restrictions on objective function

Method 3: Sequential forward selection

- Compute the criterion value for each feature. Select the feature with the best value, say x_1.
- Form all possible combinations of features x_1 (the winner at the previous step) and a new feature, e.g. $[x_1,x_2]^T$, $[x_1,x_3]^T$, $[x_1,x_4]^T$, etc. Compute the criterion and select the best one, say $[x_1,x_3]^T$.
- Continue with adding a new feature.
- Number of combinations searched: $lm-l(l-1)/2$.
  - Backwards selection is faster if l is closer to m than to 1.
- Disadvantage: Discarded features might have been deemed useful at a later stage

Method 4: Plus-L Minus-R Selection (LRS)

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 feature additions

Algorithm

1. If $L > R$ then start with the empty set $Y = \emptyset$ else start with the full set $Y = X$ goto step 3
2. Repeat SFS step L times
3. Repeat SBS step R times
4. Goto step 2

LRS attempts to compensate for weaknesses in SFS and SBS by backtracking

How to decide on L and R?

Method 5: Floating search methods I/II

- Similar to plus-L minus-R selection, although with adaptive number of backtrackings
- The dimensionality (number of features) «floats»
- Both forward (SFFS) and backwards (SFBS) versions
- Basic idea for the forward version:
  - Repeat until desired features number of features is found:
    - Do a forward step by adding a feature
    - Continue deleting features as long as the results improve (for sets of equal size)
- Provides good results at an «affordable» computational cost

Method 5: Floating search methods II/II

SFFS Algorithm

Input:

$Y = \{y_j \mid j = 1, \ldots, D\}$ //available measurements//

Output:

$X_k = \{x_j \mid j = 1, \ldots, k, \;\; x_j \in Y\}, \;\; k = 0, 1, \ldots, D$

Initialisation:

$X_0 := \emptyset; \quad k := 0$

(in practice one can begin with $k = 2$ by applying SFS twice)

Termination:

Stop when k equals the number of features required

Step 1 *(Inclusion)*

$$x^+ := \arg \max_{x \in Y - X_k} J(X_k + x) \quad \begin{cases} \text{the most significant fea-} \\ \text{ture } \textit{with respect to } X_k \end{cases}$$

$X_{k+1} := X_k + x^+; \;\; k := k + 1$

Step 2 *(Conditional Exclusion)*

$$x^- := \arg \max_{x \in X_k} J(X_k - x) \quad \begin{cases} \text{the least significant fea-} \\ \text{ture } \textit{in } X_k \end{cases}$$

if $J(X_k - \{x^-\}) > J(X_{k-1})$ **then**

$X_{k-1} := X_k - x^-; \;\; k := k - 1$

go to Step 2

else

go to Step 1

Method 6: 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

Optimal searches and randomized methods

- If the criterion increases monotonically $J(x_{i1}) \leq J(x_{i1}, x_{i2}) \leq J(x_{i1}, x_{i2}, \ldots, x_{in})$, one can use graph-theoretic methods to perform effective subset searches. (I.e. branch and bound or dynamic programming)
- Randomized methods are also popular, examples would be sequential searching with random starting subsets, simulated annealing (a random subset permutation where the randomness cools off) or genetic algorithms.

Still limited to moderate l and m, though

Preprocessing

- Outlier detection
- Missing data
- Features may have different ranges
  - E.g. feature 1 has range $f1_{min}$-$f1_{max}$ while feature n has range fn_{min}-fn_{max}
  - This does seldomly reflect their significance in classification performance!
  - Example: minimum distance classifier uses Euclidean distance
    - Features with large absolute values will dominate the classifier

Feature normalization

- Make all features have similar ranges:
  - Data set with N objects and K features
  - Features x_{ik}, i=1…N, k=1,…K

Zero mean, unit variance:

$$\bar{x}_k = \frac{1}{N}\sum_{i=i}^{N} x_{ik}$$

$$\sigma_k^2 = \frac{1}{N-1}\sum_{i=i}^{N}(x_{ik} - \bar{x}_k)^2$$

$$\hat{x}_{ik} = \frac{x_{ik} - \bar{x}_k}{\sigma_k}$$

Softmax (non-linear)

$$y = \frac{x_{ik} - \bar{x}_k}{r\sigma_k}$$

$$\hat{x}_{ik} = \frac{1}{1 + \exp(-y)}$$

Note: Normalization may change your selected feature subset or the performance of your classifier in general

Summary

- Reminder on classification principles
- «Curse of dimensionality»
- Objective function
  - Wrapper-based
  - Filter-based
    - Distance measure
    - Divergence
- Search strategy
  - Exhaustive often not possible
  - Scalar/individual feature selection
  - SFS, SBS, Floating search methods, «randomized» methods, etc.