# PA196: Pattern Recognition

## 09. Feature selection and extraction

Dr. Vlad Popovici
popovici@recetox.muni.cz

RECETOX
Masaryk University, Brno

# Outline

# Outline

What?

- the vectors to be classified are elements of some $d$–dimensional space
- the problem is to identify those features that do not contribute to the classification task and eliminate them from classifier training
- thus, we seek the only $d_1 < d$ features that contribute to classification

Why?

- improve classification performance:
  - redundant features $\rightarrow$ unstable classifier, poor fit, etc.
  - *sparser* models have better generalization properties
- improve numerical stability
- reduce the required sample size
- reduce training time, classification time
- improve interpretability of the models

How?

- use some *optimality criterion* to select the optimal subset of features $\rightarrow$ search strategy
- optimality criterion:
  - single-variate or multi-variate
  - classifier-agnostic: *filtering methods*
  - use the classifier performance: *wrapper methods*
- can be implicit in some classifiers:
  - classification trees
  - AdaBoost with decision stumps
  - penalized methods ($L_1$ penalty)
  - etc
- hybrid: e.g. use a classification tree to select the features and another learner for final classifier

## WARNING

The feature selection should be included inside the cross-validation loop (or any other equivalent method) for performance estimation.

# Outline

# Filtering methods

Idea: replace the classifier as criterion with some other criterion simpler (faster) to compute and find a subset of features satisfying this objective.

- single-feature/variable methods
- feature-set methods
- do not guarantee that selected features are relevant for a particular choice of classifier
- usually, introduces a new *meta-parameter* of the modeling process: the number of features to keep → this needs to be optimized!

# Single-feature methods

- evaluate each variable independently
- may or may not take into account the class label
- suitable approximation for high dimensional cases
- does not avoid selecting correlated features
- fast and easily integrated in modeling pipeline
- the simplest form: discard features that are constant or have low variability

Using hypothesis testing for feature selection:

- use a statistical test (e.g. $t-$test) to test

$$H_0 \text{ :there is no difference between classes}$$
$$H_1 \text{ :there is a difference between classes}$$

- if $H_0$ is rejected, we infer that the variable/feature is of some importance (it bears information about the difference between classes)
- the process involves computing a statistic and comparing it with the expected values under $H_0$; if the value is too "unusual" then $H_0$ is rejected

- in practice, this results in a "p-value" that is compared with a predefined cut-off $\alpha$ (called $\alpha$–level, traditionally 0.05 or 0.01)
- p-value: the probability, under $H_0$, to obtain a statistic at least as extreme as the one from the data at hand
- $\alpha$–level: probability of falsely rejecting $H_0$

Strategies for selecting the features:

- select top $d_1$ features (rank the features by statistic or p-values): $d_1$ needs to be chosen somehow (e.g. by cross-validation)
- select all features with p-value below a selected $\alpha$–level
  - this requires *adjustment of p-values for multiple testing*
  - the easiest is to control *family-wise error rate* e.g. multiply each p-value by the number of tests (variables)
  - for indep. variables, the adjustment does not change the ordering of variables (except for the ties), only the number of variables with significant p-value
  - popular approach: control for false discovery rate (FDR)
  - more complex adjustments are possible - also to take into account the correlation structure

# Mutual information-based feature selection

Let $X$ and $Y$ be two random variables, then their *mutual information (MI)* is defined as

$$I(X;Y) = \int_Y \int_X p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \, dxdy$$

- provides a measure of dependency between 2 variables
- by considering $X$ as a feature and $Y$ the class to be predicted, $I(X;Y)$ gives the relevance of $X$ in predicting $Y$
- problem: estimation of MI? sample size?
- it can be extended to evaluate the relevance of a *set* of feature: $I(\{X_1, \ldots, X_k\}; Y)$
- can be extended to multi-class problems

Other similar measures:

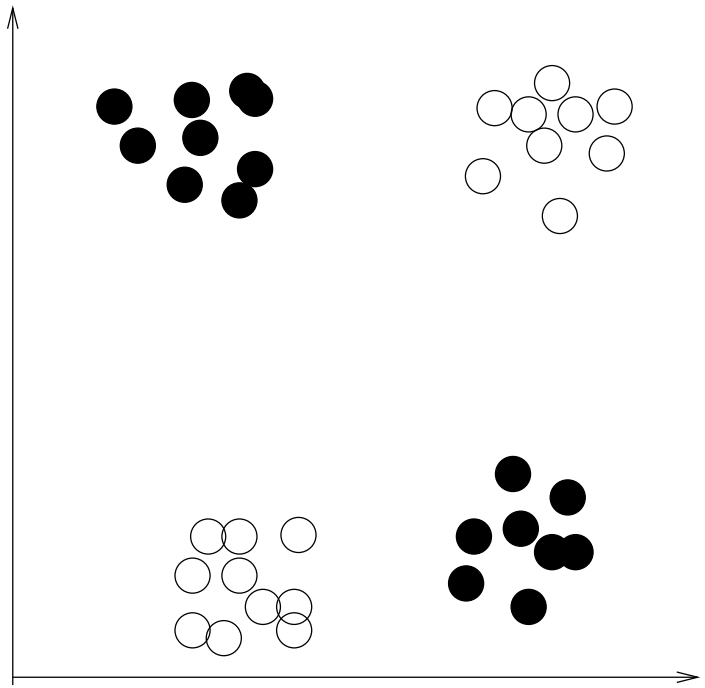
- Kullback-Leibler divergence,

$$KL(p||q) = \int p(x) \ln \frac{p(x)}{q(x)} \, dx$$

  which is usually used in a symmetrized version
- entropy
- Gini index (as in classification trees)

Issues with single-variable filtering:

- does not account for relationships between variables
- if two variables are not useful for classification when considered isolated, their combination might be useful



# Subset feature selection

- need a criterion to evaluate a set of features (e.g. MI)
- need a strategy to generate all or some of the possible sets of features

# Criteria

Let $J$ be the criterion function, which is to be maximized by the best set of features.

- previous probabilistic measures can be extended to sets of features (MI, KL, etc)
- some allow immediate extension to multiclass (e.g. MI) others require a pair-wise approach: $J(g_i, g_j)$ is computed for all pairs of classes $g_i, g_j$
- popular choices are based on variance estimates.

- Example:
  - $J = Tr(S_W^{-1} S_B)$ where $Tr(\cdot)$ is the trace of a matrix, and $S_B$ and $S_W$ are the between-class and within-class scatter matrices, respectively
  - $J = \frac{|\hat{\Sigma}|}{|S_W|}$ where $\hat{\Sigma}$ is the estimated pooled variance matrix
  - $J = \frac{Tr(S_B)}{Tr(S_W)}$
  - Mahalanobis cr. (for 2 classes):

$$J = (\mu_1 - \mu_2)^T \left( \frac{\Sigma_1 + \Sigma_2}{2} \right)^{-1} (\mu_1 - \mu_2)$$

- in general, the criterion is chosen to allow a recursive decomposition (to compute $J$ for $k+1$ feature sets, one uses the result for $k$ feature sets)
- also, to be monotone:

$$X \subseteq Y \Rightarrow J(X) \leq J(Y)$$

# Search strategies

- Exhaustive search:
  - affordable only if the total number of features is small
  - involves enumerating *all possible subsets*
  - can be implemented in a breadth-first or depth-first fashion
  - can either start from full set of features (top-down) or from an empty set of features (bottom-up)
- if $J$ is monotone, one can use a branch-and-bound procedure to avoid enumerating all sets; it will still produce a global optimum

Suboptimal search strategies:

- sequential forward selection (SFS): add sequentially features to the current set such that at each step the added feature produces the maximum increase in $J$
- generalized SFS (GSFS): instead of adding a single feature, add $k$ features
- sequential backward selection (SBS): start with the full set and remove at each step that feature that leads to minimum decrease in $J$
- generalized *SBS* (GSBS):...
- "plus $l$ minus $r$": add best $l$ and remove worse $r$ features, using sequential selection
- ...

Floating search methods:

- sequential forward/backward floating selection
- a generalization of "plus *l* minus *r*" selection in which *l* and *r* can vary
- for example, (sketch of) SFFS:
  1. start with empty set of features
  2. use SFS to obtain a set of 2 features
  3. add the feature that increases *J* the most and remove the one that decreases it the least; if it is the same feature, consider another feature not yet used
  4. continue removing features until *J* decreases or only 2 features are left; then go to step 3

# Outline

# Wrapper methods

- the optimality criterion $J$ is now the performance of the classifier to be trained
- any of the search strategies discussed before can be used in this case
- usually is it more computationally expensive than filtering methods
- produces features that are adapted to the intended classifier
- performance estimation is based on some re-sampling technique (e.g. cross-validation)

# Recursive feature elimination (RFE)

Idea: use the coefficients from the fitted model (classifier) to rank the features and eliminate those with low impact. Repeat the procedure until a convergence criterion is met.

- in LDA-like and (logistic) regression methods, the classifier has the form $h(\mathbf{x}) = \sum_i w_i x_i$ and $|w_i|$ can be used to judge the importance of $i$–th feature
- in the case of regression, one may compute p-values associated with the variables which, again, can be used for ranking them (smaller p-values correspond to more important variables)

# RFE for linear SVM

- use sensitivity analysis of some cost function: what is the influence of adding/removing a feature?
- the decision function is

$$h(\mathbf{x}) = \sum_{i \in \mathrm{SV}} y_i \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle = \left\langle \mathbf{x}, \sum_{i \in \mathrm{SV}} y_i \alpha_i \mathbf{x}_i \right\rangle$$

- then, $\nabla_{\mathbf{x}} h(\mathbf{x}) = \sum_{i \in \mathrm{SV}} y_i \alpha_i \mathbf{x}_i = \mathbf{w}$ and define the cost function as $J(\mathbf{w}) = \|\mathbf{w}\|$
- the smallest change is in the direction

$$w_j = \arg\min_j (|\mathbf{w}|)$$

  i.e. the smallest amplitude coefficient corresponds to the least significant feature
- RFE proceeds by iteratively eliminating the least significant feature(s) and retraining the classifiers on the updated data set

# RFE for non-linear SVM

- let $\mathbf{H} = [y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)]_{ij}$
- define the cost function $J = \boldsymbol{\alpha}^T H \boldsymbol{\alpha} - \langle \boldsymbol{\alpha}, \mathbf{e} \rangle$, $\mathbf{e} = [1, \ldots, 1] \in \mathbb{R}^d$
- trick: to compute the change in $J$ after removing feature $k$, assume no change in $\boldsymbol{\alpha}$ when retraining on reduced-feature set (if the feature is not important it would change (much) $\boldsymbol{\alpha}$)
- with this,

$$H_{ij}^{(-k)} = y_i y_j K(\mathbf{x}_i^{(-k)}, \mathbf{x}_j^{(-k)})$$

- the change in cost function is

$$\Delta J(k) = J - J^{(-k)} = \boldsymbol{\alpha}^T H \boldsymbol{\alpha} - \boldsymbol{\alpha}^T H^{(-k)} \boldsymbol{\alpha}$$

Algorithm: to obtain a ranking of features

1. let $\mathbf{X}^{\emptyset}$ be the original data set and let $S = \emptyset$ be the initial ordered list of features (from least to most significant), $X^{(S)}$ will denote the data set with features in $S$ removed
2. repeat 3-6 until no more features are left:
3. train the SVM on $X^{(S)}$ and let $\alpha$ be the solution
4. compute $\Delta J(i)$ for all features not yet removed ($i \notin S$)
5. select $k = \arg\min_i \Delta J(i)$
6. update $S = S \cup \{k\}$

# Outline

# FS via regularization

Idea: introduce some constraints on the model such that the number of effective features is decreased.

- regularization has not necessarily as ultimate goal feature selection
- usually, reg. is imposed by penalizing some complexity measure of the model or the loss functions for classification:

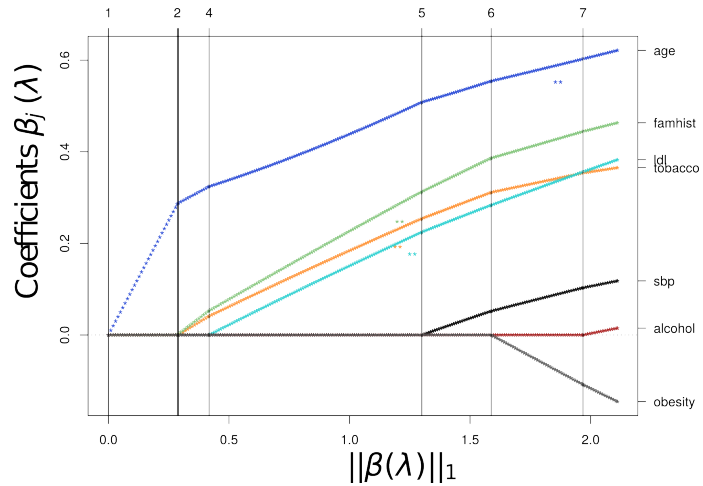
$$\text{Loss} + \lambda \text{Penalty}$$

- if $\mathbf{w}$ is the vector of coefficients, the classical penalties are $||\mathbf{w}||_2^2$ and $||\mathbf{w}||_1$
- what about $\lambda$?

Example: penalized logistic regression.
The new loss function is

$$L(\theta = \{\mathbf{w}, w_0\}) = \sum_{i=1}^{n} [y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + w_0) - \log(1 + \exp(\langle \mathbf{w}, \mathbf{x}_i \rangle + w_0)] + \lambda \sum_{j=1}^{d} |w_j|$$

From ESL:



- similar approaches exist for various classifiers
- there is a regularized version for classification trees
- the best choice for $\lambda$ is usually obtained from cross-validation

# Outline

# Feature extraction

Idea: transform the original feature space into a lower dimensional space where some important property (e.g. separability) of data is preserved or enhanced.

- the methods can be based on linear or non linear combinations of original variables
- some methods are variable-directed: they are primarily concerned with relations between variables
- while others are more individual-directed: the distances between data points are of main interest

# Outline

# Principal component analysis - PCA

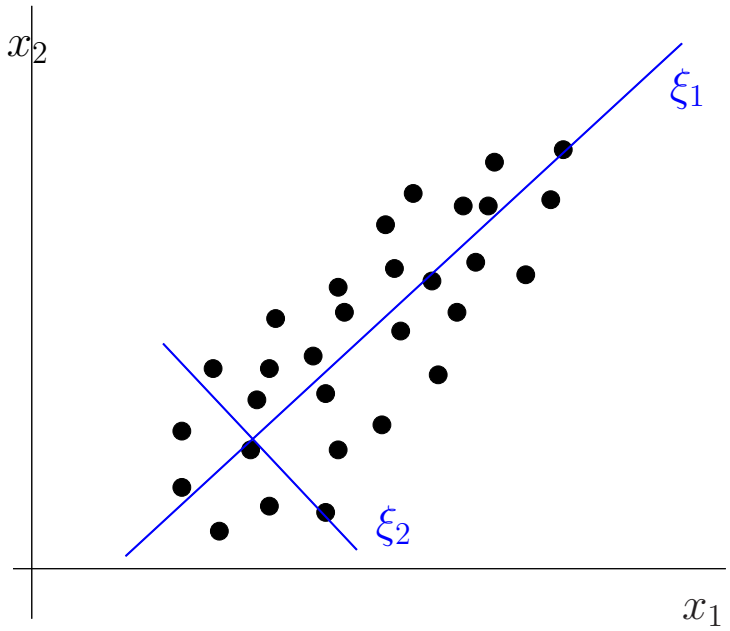
- the goal is to find a linear transformation of the original variables such that the resulting variables are decorrelated
- the hope is to find a smaller number of variables that explain the data
- it is a basic data exploration technique

Find an orthogonal transformation **A** of the input data, such that the new variables

$$\boldsymbol{\xi} = \mathbf{Ax}$$

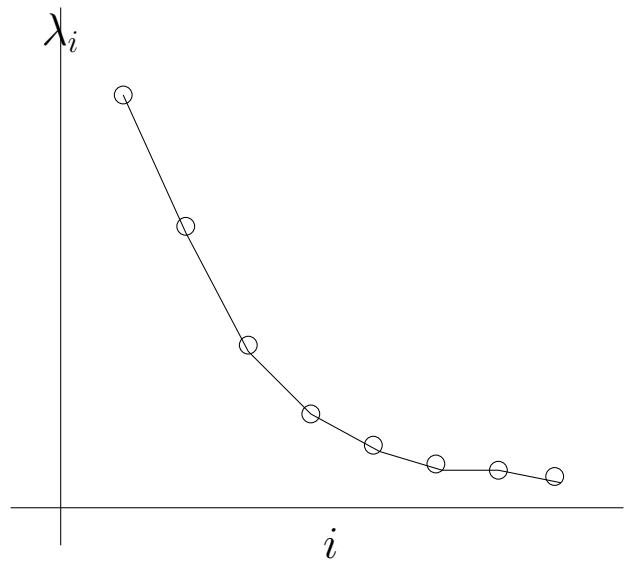
satisfy one of the following (equivalent) constraints:

- the variance of $\xi_j$ is stationary
- variables $\xi_j$ are decorrelated
- the new axes are the best fit (least square) of the data and are orthogonal



The solution is given by the *eigenvectors* of the covariance matrix of data vectors $\{\mathbf{x}_i\}$.

- let $\lambda_i$ be the *eigenvalues*: their ranking gives the order of the principal axes
- $\sum_{i=1}^{d} \lambda_i = \sum_{i=1}^{d} \mathrm{Var}(\xi_i)$
- dimensionality reduction: keep only the first $k$ principal components (e.g. to account for 80% of total variance)
- data approximation: reconstructing the original space from only $k$ PCs



# PCA in pattern recognition

- lots of applications, either alone or in combination with some classifier
- common combinations: PCA + LDA, PCA + SVM
- kernel PCA
- classical application in computer vision: Turk and Pentland's eigenfaces (1991)

# Eigenfaces

Problem: face recognition. (Turk, Pentland: Eigenfaces for recognition. 1991)

Face recognition procedure (sketch):

- collect a number of images, say 4 images, per identity
- construct the eigenfaces, an keep the top $k$
- represent each face in terms of its coordinates in eigenface space $\Omega = [\omega_1, \ldots, \omega_k]$
- for each identity, average the $\omega$–vectors to obtain a characteristic profile
- for a candidate face $\Omega^*$ compute the Euclidean distance to each for the $\Omega$–vectors representing the identities
- if the distance is too large, the candidate face does not represent the claimed identity

- the algorithm can be used for face detection as well
- it generated a lot of follow-up methods
- the method is easy to implement and can be easily be adapted to low-bandwidth environments

Other methods:

- PCA is a special case of Karhunen-Loève transform
- generalizations of PCA: common principal components, non-linear PCA
- independent component analysis: no longer an orthogonal transform
- factor analysis, etc

# Outline

# Multidimensional scaling (MDS)

- tries to find a set of points that would explain the given dissimilarities (distances)
- it is used as an exploratory technique and for data visualization
- the representation follows the idea: similar objects are represent close together, dissimilar ones, farther apart
- can work both with metric and non-metric data
- in the classical approach, only one (dis)similarity matrix is required; other versions may use several

Classical MDS:

- for two objects $i$ and $j$ the similarity is represented in terms of Euclidean distance between points $d_{ij}$
- $d_{ij}$ are related to the given similarity through a transformation $f$: $\delta_{ij} = f(d_{ij})$
- the problem is to find $f$ such that an error measure is minimized

- MDS is just an instance of larger class of methods: "manifold learning"
- depending on the problem, other transformations could be more useful for visualization or classification
- isomap, local linearly embedding spaces, etc etc

# Example: Roweis and Saul: Nonlinear Dimensionality Reduction by Locally Linear Embedding, Science 2000.

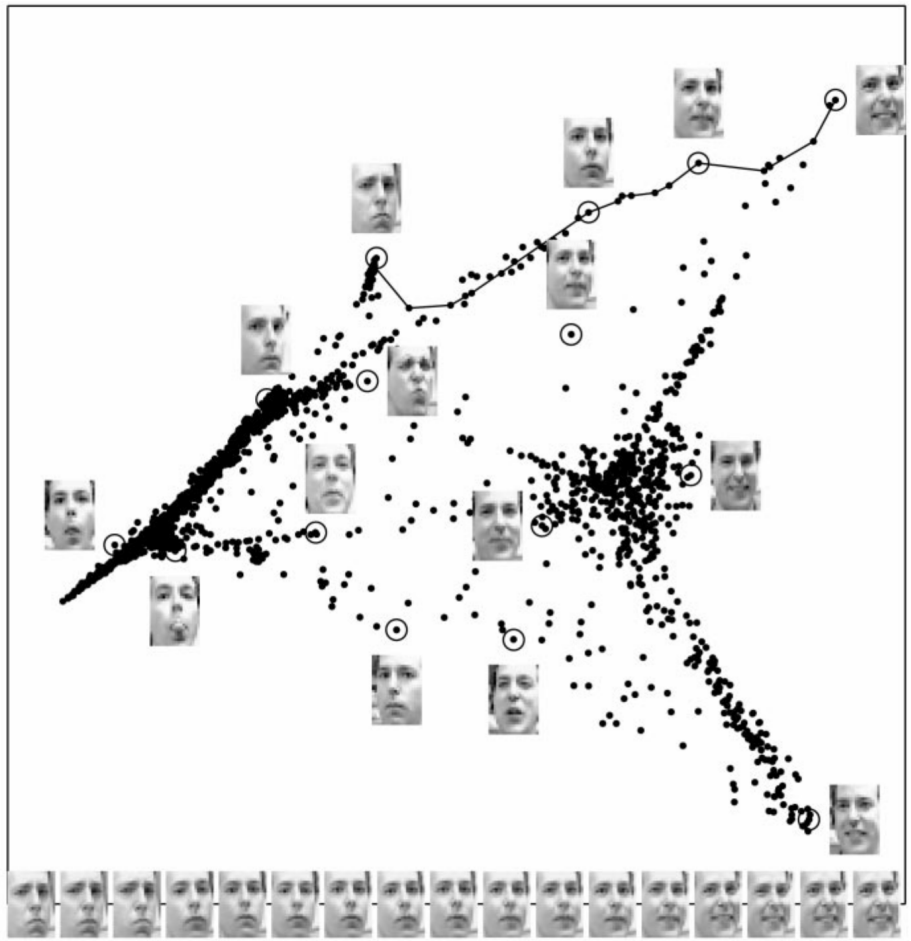


**Fig. 3.** Images of faces (*11*) mapped into the embedding space described by the first two coordinates of LLE. Representative faces are shown next to circled points in different parts of the space. The bottom images correspond to points along the top-right path (linked by solid line), illustrating one particular mode of variability in pose and expression.