MTTTS17
Dimensionality Reduction and Visualization

Spring 2020
Jaakko Peltonen

Lecture 2: Feature selection

Part 1: Preliminaries

How to avoid problems of high-dimensional data?

On the last lecture we saw that:

- High-dimensional spaces have **surprising properties** which make them behave differently than our expectations.
- Data is essentially always **sparse** in high-dimensional spaces
- High-dimensional models often have **many parameters**
- When we have sparse data compared to the number of parameters, **overfitting** often happens

How can we avoid the problems? Two approaches to avoid or at least attenuate the effects in the presence of high-dimensional data:

(1) Try to improve the separation between relevant and irrelevant variables

(2) Try to detect dependencies between the (relevant) variables, and remove unnecessary redundancies

Relevance of variables

- not all variables are necessarily related to the underlying information the user is interested in
- Irrelevant variables may be eliminated from the data
- supervised: subset of objects labeled by an oracle
  The relevance of an input is measured by computing correlations between the known input/output pairs

Dependencies between variables

- Even assuming that all variables are relevant the dimension of the observed data may still be larger than necessary
- 2 variables may be highly correlated→find a new set of transformed vars
- The new set should contain a smaller number of variables but preserve the interesting characteristics of the initial set
- Transformations or projections (linear, non-linear) should not alter these characteristics
- Dependencies result from imperfection of observation process: interesting variables not directly accessible

Goals of projection

(1) Reduce the number of variables - *dimensionality reduction*, eliminate redundancy (Example: feature selection, PCA)

(2) (more complex) Retrieve so-called *latent variables* that originate from observed ones but cannot be measured directly (for example: Blind Source Separation (BSS) in signal processing, Independent Component Analysis (ICA) in multivariate data analysis: *latent variable separation*)

Feature selection vs. feature extraction

- **feature selection**: choose k < d important features, ignore the remaining d - k

- **feature extraction** (more general): transform the original $\mathbf{x} \in \mathbb{R}^d$ to $\mathbf{z} \in \mathbb{R}^k$ (k < d).

  Example transformations:
  Linear transformations: principal component analysis (PCA), linear discriminant analysis (LDA), factor analysis.

  Some *metric learning* methods are similar to linear transformations.

- *Nonlinear* transformations: Self-Organizing Map (SOM), Multidimensional scaling (MDS), manifold embedding methods. Often based on assuming the data lies on a low-dimensional manifold.

- We will encounter both kinds of methods on the course

Feature selection

feature selection (also called variable selection): choose $k < d$ important features, ignore the remaining $d - k$

Objectives of feature selection:

- improve prediction performance of predictors trained for the set of remaining features
- make predictors faster to train and use & more cost-effective
- reduce measurement & storage requirements
- provide a better understanding of the underlying process that generated the data

Feature selection is needed both for **supervised tasks** (classification, regression, prediction) and for **unsupervised tasks** (density estimation, clustering, component analysis, visualization)

Feature selection, cont.

- example application: **microarray data analysis** (e.g. 100s of patients, activity measured for 60000 genes). Select genes to be analyzed: which gene activities are good predictors of an illness like cancer?

- example application: **classification of text documents**. After pruning noninteresting words, we may still have a vocabulary of e.g. 15000 words. If we count occurrences of different words in a document, which words are good predictors of a document category like ”news”, ”entertainment” or ”technical”?

Part 2: Feature selection for supervised tasks

Based on the presentation in
Guyon and Elisseef, JMLR 2003

Feature selection for supervised tasks

- We consider feature selection for prediction tasks, in particular classification and regression
- We focus on **selecting a subset** of features (alternative: rank all features)
- Simply ranking all features and picking the best ones in order is suboptimal: features may be **redundant**
- On the other hand, selecting a subset can leave out relevant (even if somewhat redundant) features
- More generally, a simple predictor may not be able to make use of complicated information in a feature even if it is relevant; best to choose features that are **most useful** for the predictor

- Two main approaches: **filter** methods and **wrapper** methods

Filter methods – variable ranking

- Simple filter methods select variables by ranking them with correlation coefficients – we'll come to more advanced filters later
- To build predictors, nested subsets with ever more variables of decreasing relevance are defined.
- Application example: in microarray analysis, a ranking criterion is used to find genes that discriminate between healthy and diseased patients. Proteins coded by such genes could be used as drugs, or as targets of drugs.

Variable ranking, properties

- Good properties: simplicity
- The ranking is independent of the choice of the predictor
- Under some assumptions, may still be optimal for a given predictor (ex. using Fisher's criterion for ranking is optimal for Fisher's linear discriminant classifier when class covariances are diagonal)
- Scalable: only needs computation of *n* scores, sorting the scores
- Robust against overfitting: introduces bias but may have much less variance

Variable ranking, notation

x = random vector value drawn from an unknown distribution
X_i = random variable corresponding to i^{th} component of **x**
Y = random variable of the outcome, realizations y
$\mathbf{x}_i$ = m-dim. vector, realizations of the i^{th} variable for training data
y = m-dim. vector containing target values for training data

m **examples** $\{\mathbf{x}^k, y^k\}$, $k = 1, \ldots m$. Each example has the values for n **input variables**, $x_{k,i}$, $i = 1, \ldots n$, and for one **output variable**, y^k.

Scoring function $S(i)$ computed from the values $x_{k,i}$ and y^k, $k = 1, \ldots m$. High score indicates a valuable variable; we sort variables in decreasing order of $S(i)$.

Ranking functions 1

- For continuous output variables: Pearson correlation coefficient

definition

$$\mathcal{R}(i) = \frac{cov(X_i, Y)}{\sqrt{var(X_i)var(Y)}}$$

estimator

$$R(i) = \frac{\sum_{k=1}^{m}(x_{k,i} - \bar{x_i})(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{m}(x_{k,i} - \bar{x_i})^2 \sum_{k=1}^{m}(y_k - \bar{y})^2}}$$

cov = covariance, var = variance, bar denotes mean over k

- Same as cosine between $\mathbf{x}_i$ and **y** after substracting their mean
- In linear regression, square of $R(i)$ is the fraction of variance (around $\bar{y}$) explained by a linear relation from $\mathbf{x}_i$ to **y**. Enforces ranking according to **goodness of linear fit** from each variable.
- More precisely: if X_i and Y are jointly normal,

$$p(y|x_i) = N\left(y; \mu_Y + \frac{\sigma_Y}{\sigma_{X_i}} R(i)(x_i - \mu_Y), (1 - R(i)^2)\sigma_Y^2\right)$$

Ranking functions 2

- For continuous output variables: Pearson correlation coefficient

definition

$$\mathcal{R}(i) = \frac{cov(X_i, Y)}{\sqrt{var(X_i)var(Y)}}$$

estimator

$$R(i) = \frac{\sum_{k=1}^{m}(x_{k,i} - \bar{x_i})(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{m}(x_{k,i} - \bar{x_i})^2 \sum_{k=1}^{m}(y_k - \bar{y})^2}}$$

cov = covariance, var = variance, bar denotes mean over k

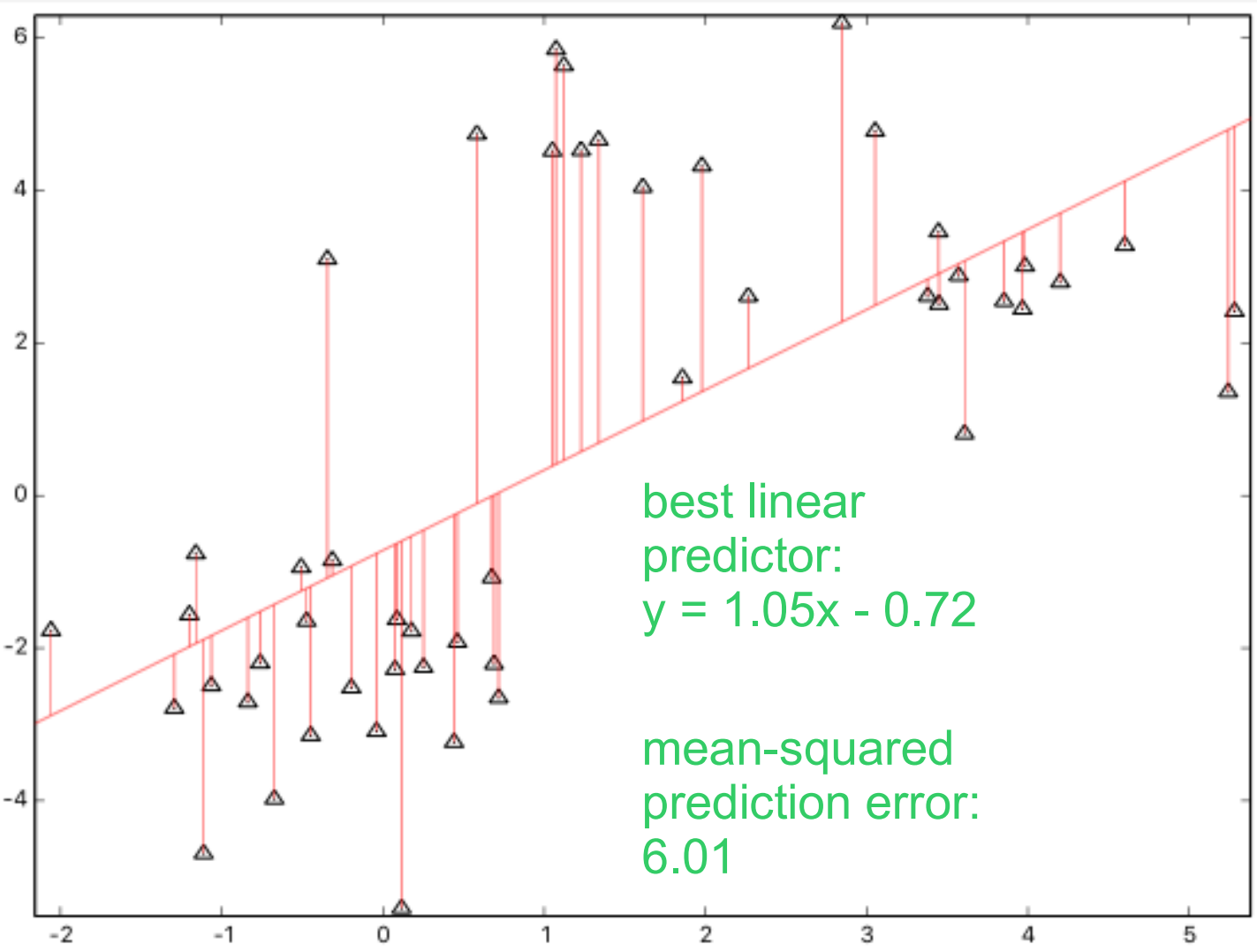
- $R(i)^2$ can be extended to two-class classification: use y=-1 and y=+1 for the two classes.
- Resulting $R(i)^2$ is related to Fisher's criterion
- Also related to t-test criterion: may be used as test statistic to assess significance of a variable

Ranking functions 3

- Correlation can only detect **linear dependencies** between variable and target.
- Simple alternative: make a non-linear fit of the target with single variables, rank by goodness of fit.
  Example: try to fit a **spline curve** between each input variable and the output variable, rank variables by the squared error of their spline fit.
- In classification, one can similarly rank variables by building a separate classifier with each single variable, and ranking according to performance of the classifier
- In two-class classification, the thresholded value of the variable (e.g. ”$x_i > 0.372$?”) can be used as the discriminant function

Ranking functions, example

- 50 points from a mixture of 4 Gaussians, predict vertical coordinate y from horizontal coordinate x



corr.
coefficient
R(i) = 0.64

Ranking functions 4

- Performance criteria in classification: **classification error rate**, more detailed measures for binary classification: various criteria involving false positive rate *fpr* and false negative rate *fnr.*
- E.g. the **receiver operating characteristics (ROC) curve** plots (1-*fpr*) vs. *fnr*, can compute **hit rate of break-even point** where fpr=fnr, or **area under the curve**.
- When many variables separate classes perfectly, these criteria cannot distinguish which of those variables is best —--> rank them by correlation coefficient or ”margin” (distance between closest different-class points)

Ranking functions 5

- Various **information-theoretic criteria** exist, often based on **mutual information** between variables and the target:

$$I(i) = \int_{x_i} \int_{y} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)} dxdy$$

definition for continuous variables

p() are densities over values of the respective variables

$$I(i) = \sum_{x_i} \sum_{y} P(X = x_i, Y = y) \log \frac{P(X = x_i, Y = y)}{P(X = x_i)P(Y = y)}$$

definition for discrete variables

probabilities P() can be estimated from occurrence counts.

- Estimation becomes harder when number of possible values increases.
- Continuous case is hardest. Possibilities: (1) discretize the variables and use the discrete definition; (2) use a density estimator: If **Gaussian density** is assumed, result depends on correlations; nonparametric estimators include e.g. Parzen windows (not on this course).

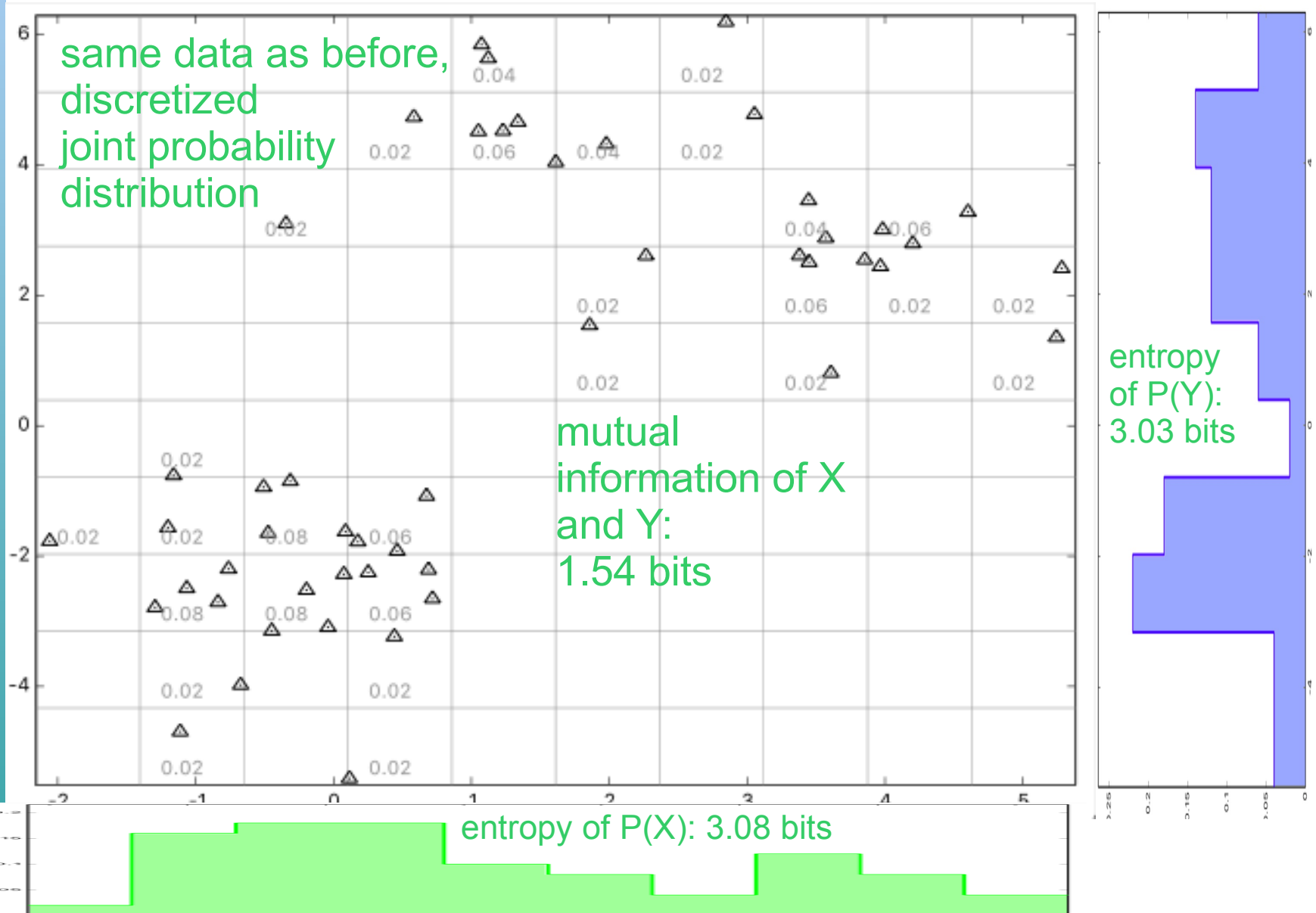
Ranking functions 6

- Entropy $-\sum_y P(Y=y)\log P(Y=y)$ of a (discrete) variable Y represents its uncertainty, or the average amount of information gained from observing it. Often measured in bits.
- **Mutual information means how much the uncertainty of Y is reduced if we know the other variable X**, that is, how much information X provides about Y.

$$I(i)=\sum_{x_i}\sum_y P(X_i=x_i,Y=y)\log\frac{P(Y=y|X=x_i)}{P(Y=y)}$$
$$=-\sum_y P(Y=y)\log P(Y=y)$$
$$+\sum_{x_i} P(X_i=x_i)\sum_y P(Y=y|X_i=x_i)\log P(Y=y|X_i=x_i)$$
$$=H(Y)-H(Y|X_i)$$

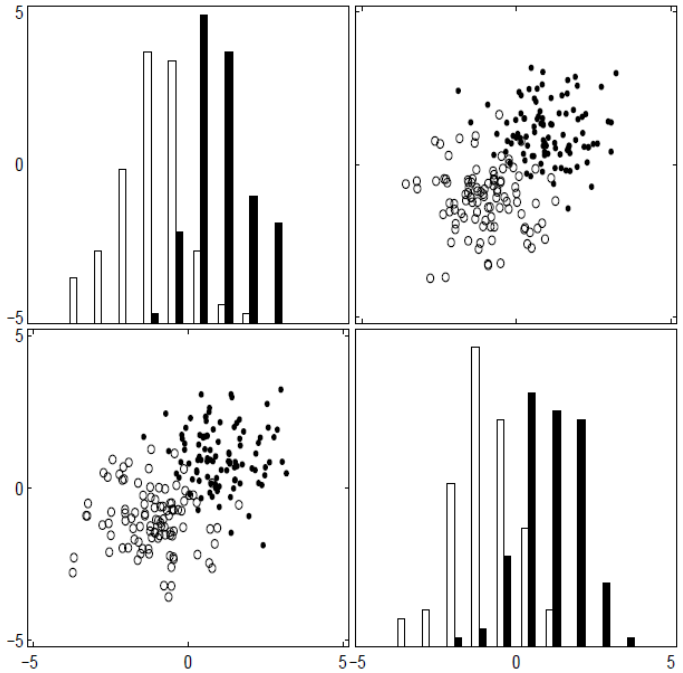
Ranking functions, example

- 50 points from a mixture of 4 Gaussians, predict vertical coordinate y from horizontal coordinate x

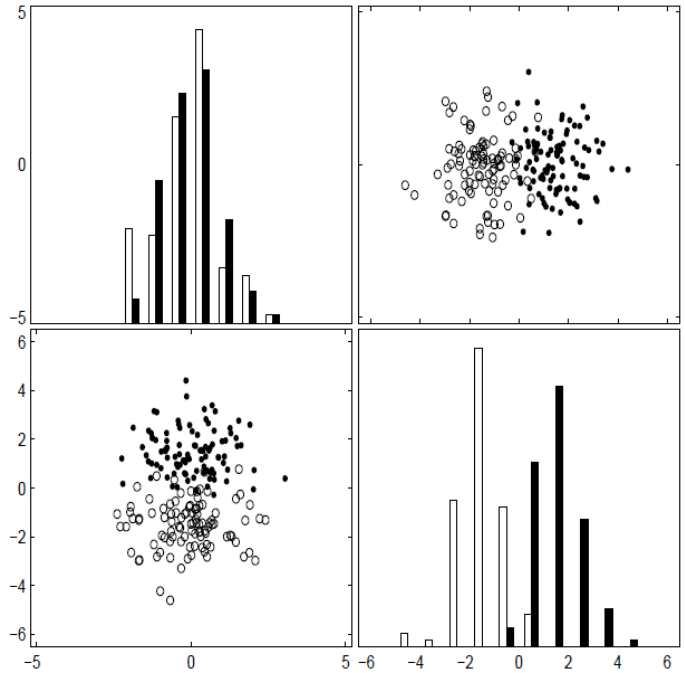


Ranking methods, redundancy

- Apparently redundant-looking variables could be non-redundant



The two variables have the same distribution->redundant?



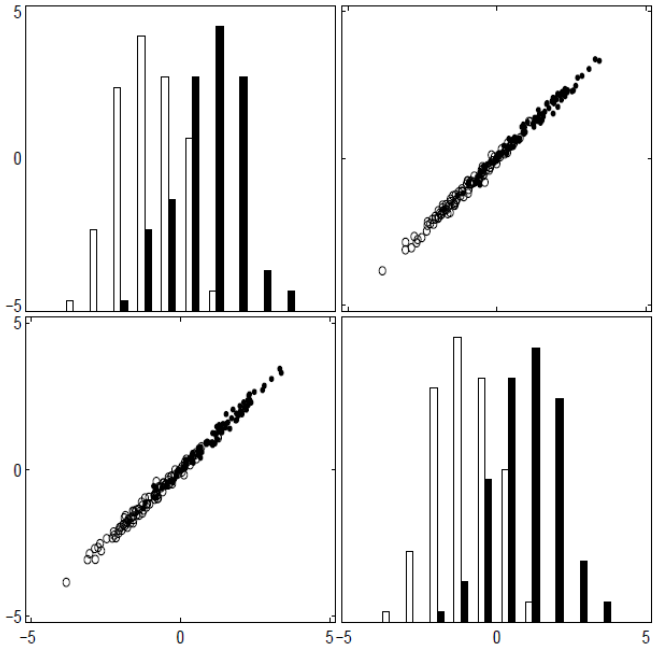
45-degree rotation: average of the variables->better class separation than either alone

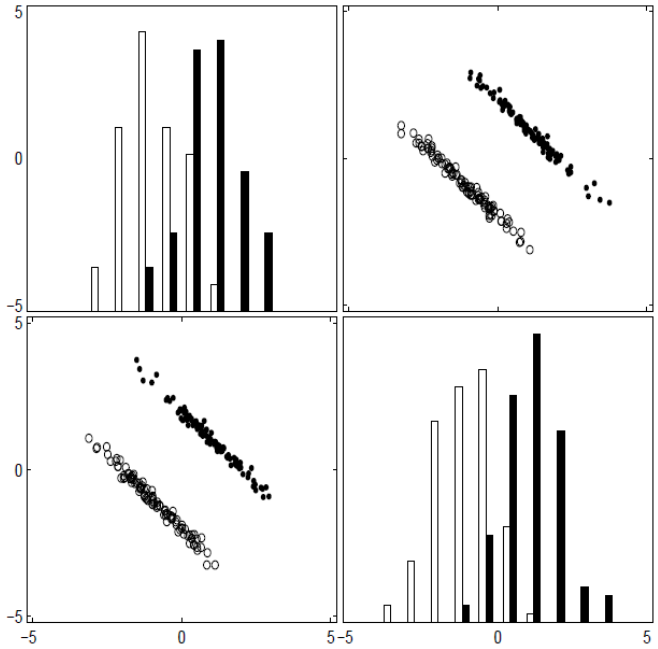
The variables have same class centers but indep. noise around classes. Averaging reduces noise → cleaner class separation

Pictures from Guyon and Elisseef, JMLR 2003

Ranking methods, redundancy

- Effect of within-class correlation on redundancy?





Pictures from Guyon and Elisseef, JMLR 2003

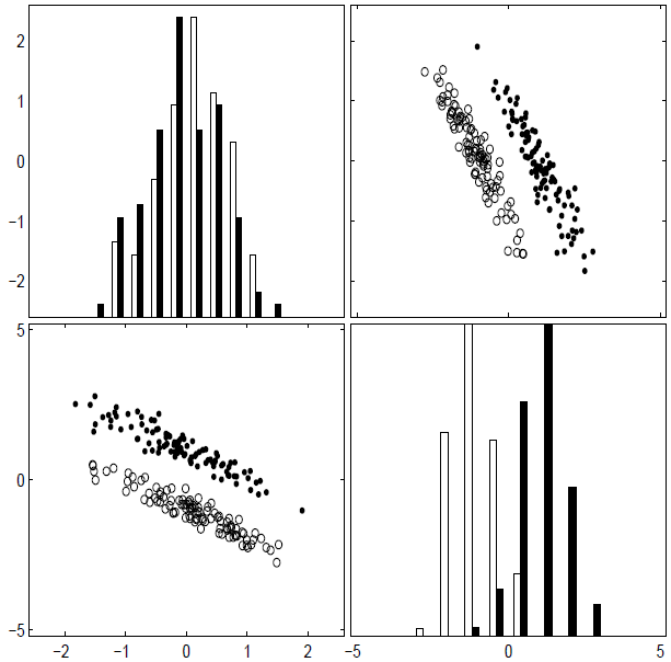
Variables highly correlated within class, correlation coincides with direction of class separation. Redundant.

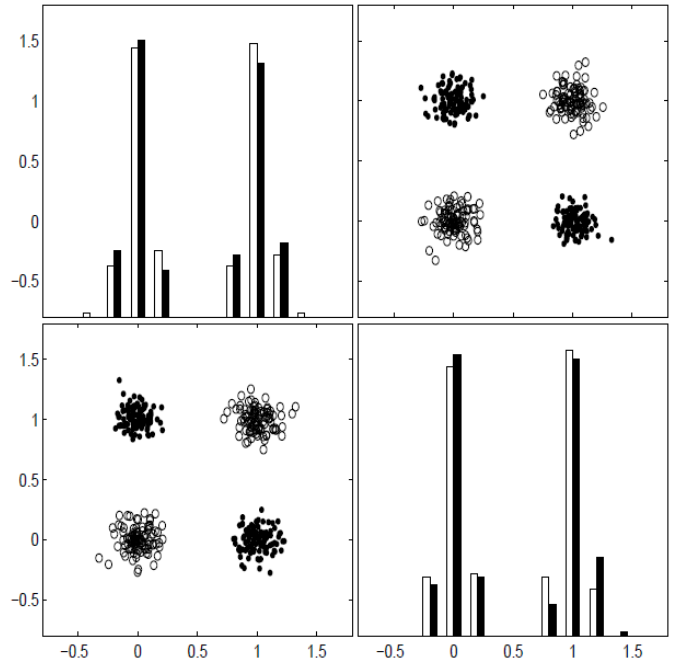
High within-class correlation, not along class separation direction. Variables are not redundant.

Perfectly correlated variables are redundant, but very highly correlated variables can still be complementary (non-redundant). Methods evaluating variables individually can't notice such effects.

Ranking methods, redundancy

- Using variable ranking to filter poor variables (to avoid overfitting to too many variables), also useful variables can be lost: a variable useless by itself can be useful with





Top-left variable is by itself unrelated to class (same marginal density for class 1&2) but together the variables give good separation

XOR (exclusive-or) problem: each variable by itself is useless (same marginal density for both classes) but together they give good separation

Pictures from Guyon and Elisseef, JMLR 2003

Wrapper methods

- Wrapper methods assess subsets of variables according to their usefulness to a given predictor
- The predictor (learning machine) is used as a **black box** that scores variable subsets by their predictive power
- Wrapper methods can be easily implemented on top of off-the-shelf predictor methods
- To use a wrapper method, define (1) how to search the space of feature subsets, (2) how to assess performance of a predictor, (3) which predictor to use.
- popular predictors include decision trees, naive Bayes classifiers, least-square linear predictors, and support vector machines
- Performance assessment often done on a validation set separate from the training set, to avoid overfitting

Wrapper methods, cont.

- Wrappers are often criticized because they can need a lot of computation. Brute force search is NP-hard. Many strategies: best-first, branch-and-bound, genetic algorithms, simulated annealing, …
- Coarse search strategies may sometimes reduce overfitting
- We will discuss two **greedy** strategies, forward selection and backward selection

- Some **embedded methods** use a similar idea as wrappers more efficiently, by optimizing a two-part objective function: a **goodness-of-fit term** plus a **penalty** for a large number of variables
- Advantages of embedded methods: better use of available data (no need to split training data into a training and validation set); reaching a solution faster – no retraining a predictor from scratch for every variable subset.
- Example embedded method: **decision trees**

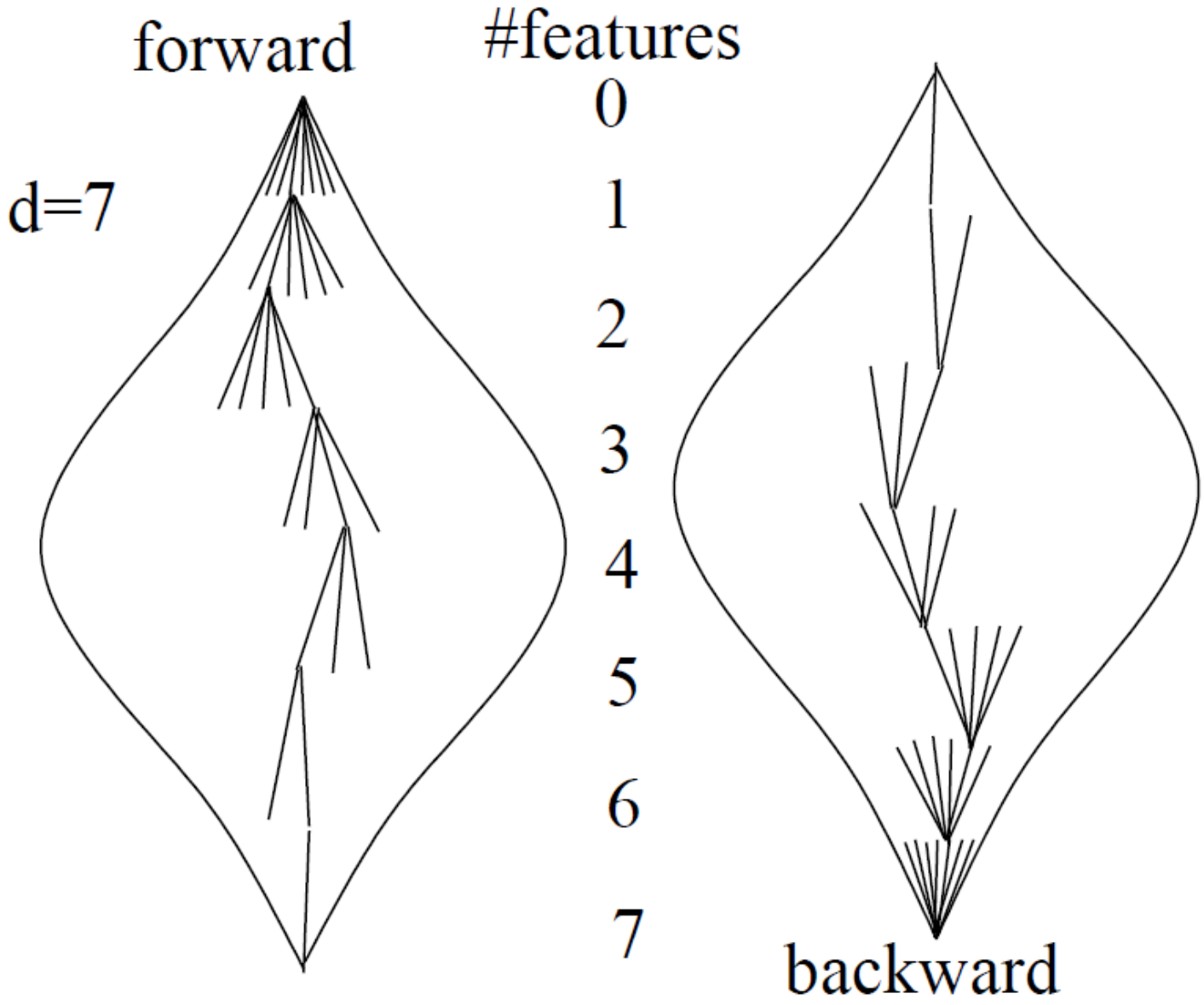
Forward selection and backward selection

- There are 2^d subsets of d features
- **Forward selection:** add the best feature at each step
  - Set of features F is initially empty
  - At each iteration, find the best new features

    $j = \text{argmin}_i \, E(F \cup x_i)$

    E( ) is some performance measure
  - Add x_j fo F if $E(F \cup x_j) < E(F)$
- Hill-climbing algorithm, $O(d^2)$ complexity
- **Backward selection**: Start with all features and remove one at a time, if possible. May choose different set than forward selection.
- **Floating search**: add k, remove l

Forward selection and backward selection – hypothesis space



Forward selection and backward selection – example

- Toy data set consists of 100 10-dimensional vectors from two classes (1 and 0)
- First two dimensions x_1^t and x_2^t: drawn from Gaussian with unit variance and mean of 1 or -1 for the classes 1 and 0 respectively.
- Remaining eight dimensions: drawn from Gaussian with zero mean and unit variance, that is, they contain no information of the class.
- Optimal classifier: if x_1+x_2 is positive the class is 1, otherwise the class is 0.
- Use nearest mean classifier.
- Split data in random into training set of 30+30 items and validation set of 20+20 items

Forward selection and backward selection – example, continued

Forward selection:

Features	E_{VALID}
$\emptyset$	0.500
1	0.175
1, **2**	**0.100**
1, 2, **4**	0.100
1, 2, 4, **5**	0.100
1, 2, 4, 5, **3**	0.075
1, 2, 4, 5, 3, **8**	**0.050**
1, 2, 4, 5, 4, 8, **6**	0.075
1, 2, 4, 5, 4, 8, 6, **7**	0.075
1, 2, 4, 5, 4, 8, 6, 7, **10**	0.100
1, 2, 4, 5, 4, 8, 6, 7, 10, **9**	0.150

Backward selection:

Features	E_{VALID}
9, 10, 4, 6, 7, 8, 3, 5, 2, 1	0.150
10, 4, 6, 7, 8, 3, 5, 2, 1	0.100
4, 6, 7, 8, 3, 5, 2, 1	0.075
6, 7, 8, 3, 5, 2, 1	0.075
7, 8, 3, 5, 2, 1	0.075
8, 3, 5, 2, 1	**0.050**
3, 5, 2, 1	0.075
5, 2, 1	0.100
2, 1	**0.100**
1	0.175
$\emptyset$	0.500

- Optimal selection consists of features 1 and 2!

Nested subset methods

- Nested subset methods guide search by estimating changes in objective function value from moves in variable subset space.
- Combined with greedy search (such as backward elimination or forward selection) they yield nested subsets of variables
- Objective function J(s) for s variables, predict changes using:
  - **Finite difference calculation:** compute difference between $J(s)$ and $J(s+1)$ or $J(s-1)$ for variables to be added/removed. The basic forward selection/backward elimination algorithms on the previous slides essentially do this because they compare objective values before/after adding/removing a variable.

    Sometimes exact differences can be computed without retraining a new models for each candidate variable.

Nested subset methods

- Objective function J(s) for s variables, predict changes using:
  - **Quadratic approximation of the cost function:** based on variable weights, can be used for backward selection by pruning variable weights. J(s) approximated by Taylor series approximation of J(s) with respect to the weight of of each variable. Allows estimation of change when removing a variable: $DJ_i = (1/2)\frac{\partial^2 J}{\partial w_i^2}(Dw_i)^2$, removal means changing the weight to zero ($Dw_i = w_i$) where w_i is current weight of variable i, Dw_i is the change in the weight and DJ_i is the change in the cost

  - **Sensitivity of J(s):** compute absolute value/square of the derivative of *J* with respect to x_i or weight w_i. If the derivative is small, the objective does not change much when the variable value or weight changes slightly; maybe the variable can be left out.
    Possible to also use leave-one-out cross-validation error.

Embedded methods

- Embedded methods directly optimize an objective function with two competing terms:
  - **goodness-of-fit** to be maximized
  - **number of variables** to be kept small
- Similar to objective functions with regularization penalty terms
- In some cases, penalties on variable weights end up leaving out some variables completely--> variable selection
- For example: linear predictors $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, Lp-norm penalty on weights. L0-penalty = number of variables (nonzero weights). In some settings iteration of L1/L2 based optimization and variable rescaling gives approximately the same results as L0.
- Penalty terms may correspond to priors on the model complexity

Filters for subset selection

- The previous filters we talked about ranked variables by their separate goodness
- Possibility for subset selection: use a wrapper (or embedded method) with a simple **linear predictor** as a filter. Train a **non-linear predictor** on the selected variables.
- Another alternative (not on this course): information theoretic filtering methods like estimating the **Markov blanket:**
  - The Markov blanket of a variable xi is a set of other variables (excluding xi) which make xi unnecessary. A variable eliminated by a Markov blanket remains unnecessary during further backward selection.

How many features to select?

- Sometimes e.g. using performance on a validation set may be sufficient to give a clear picture how many features are needed
- Significance testing could be used to test which features are really needed, but dangerous: overlapping training data sets in cross-validation, overlapping feature sets... must be careful about validity of the test to avoid biases.
- Other possiblity: use a ”probe”. Introduce known fake variables into the data, e.g. drawn from a Gaussian or by shuffling data of actual variables, use them in variable selection. Discard any variables whose estimated goodness is smaller than that of the fake variables.
- Or in forward selection, use ratio of fake to all variables as stopping criterion
- For some situations & models, possible to compute analytically rank of a probe for a given risk of accepting an irrelevant variable

”Checklist” for feature selection

- If you have **domain knowledge** from an expert, construct an improved set of ”ad hoc” features with the expert's help
- If your features are **not commensurate** (e.g. one is measured in meters, another in millimeters), it may be useful to normalize them
- If you think your features are **interdependent**, expand the feature set: create conjunctive features (”A and B”), products of features, up to available resources.
- If you need to **prune** data variables to improve cost, speed, or understanding: create disjunctive features (”A or B”) or weighted sums of features
- If you need to assess features **individually** to study their influence or to do a first filtering: use variable ranking. (Useful anyway as a baseline!)
- If you don't need a **predictor**, you can stop here.

”Checklist” for feature selection, cont.

- Is the data maybe ”**dirty**” (meaningless patterns, noisy outputs, wrong labels)? Detect outliers using the top-ranked features, check/discard outliers.
- Do you know **what to try**? If not, use linear prediction. Use forward-selection with ”probe” stopping criterion, or the l_0-norm embedded method. For comparison, using the feature-ranking on the previous page, construct several predictors using increasing feature subsets. Can you get equal/better performance with a smaller subset? If so, try non-linear prediction with that subset.
- If ideas, time, resources, and data samples permit: **compare several selection methods** such as your ideas, correlation coefficients, backward selection, embedded methods. Use linear and nonlinear predictors. Pick the best approach by model selection.
- If **solution stability** is needed: subsample the data, redo the analysis for several ”bootstraps”

Part 3: Feature selection for unsupervised tasks

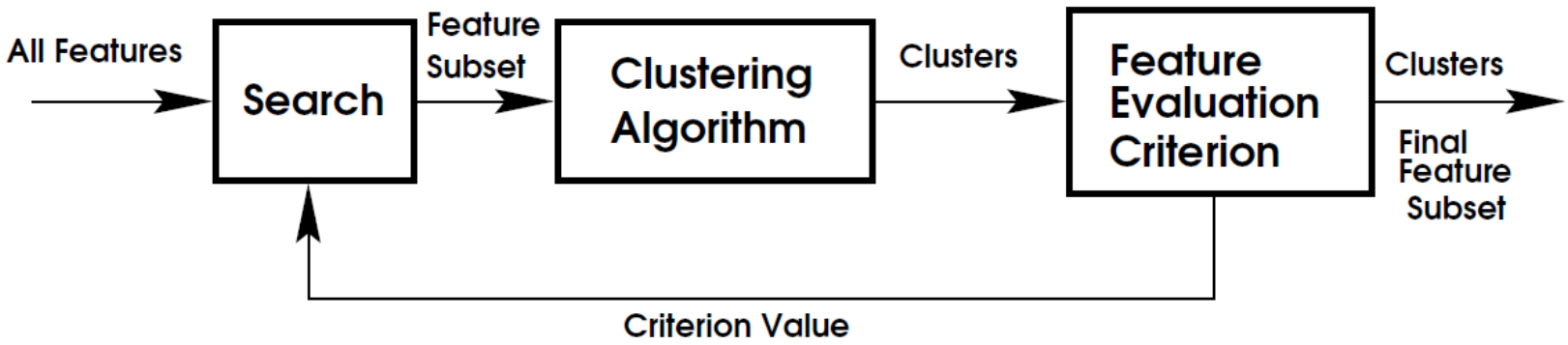
Based on the presentation in
Dy and Brodley, JMLR 2004

Feature selection for clustering

- We consider feature selection for unsupervised tasks, in particular unsupervised classification and clustering
- Goal of feature selection for unsupervised learning (at least for clustering): find smallest feature subset that best uncovers “interesting natural” groupings (clusters) from data by the chosen criterion
- There may be multiple solutions (best subsets), any one will do
- Must define ”interesting” and ”natural” by criterion functions

Feature selection for clustering

- We follow the wrapper approach



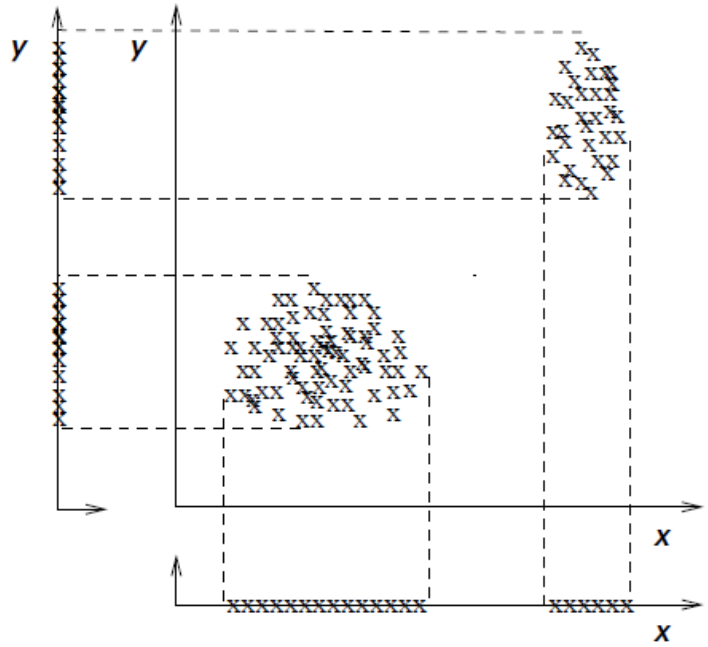
wrapper approach for unsupervised learning (clustering)

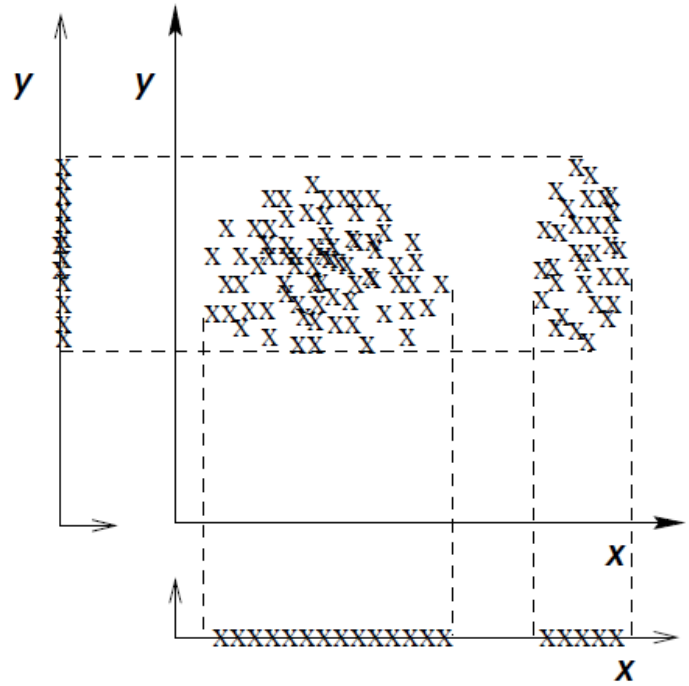
Feature selection for clustering

- What does ”Interestingness” mean: two general approaches.
  (1) same as the clustering criterion, (2) need not be the same

Feature selection for clustering

- Similar concepts of redundancy and irrelevance as before





x and y are redundant:
they yield the same information
for discriminating the clusters

y is irrelevant: without x
it would yield just one cluster
and it does not help x

Feature Selection for Clustering

- For each candidate subset, a clustering is fitted, the clusters and feature subset are then evaluated by a criterion
- Component tasks: (1) feature search, (2) clustering algorithm, and (3) feature subset evaluation
- Exhaustive feature search would again be intractable
- For feature search, forward selection or backward selection can be used; has $O(d^2)$ worst-case complexity
- For the clustering algorithm, we consider clustering by estimating a Gaussian mixture model with the expectation-maximization algorithm (=maximum likelihood fitting). Assumes each cluster is Gaussian.

Feature Selection for Clustering

- **Scatter separability criterion**: evaluates cluster separation
  (parametric version below, a nonparametric one exists)

$$S_w = \sum_{j=1}^{k} \pi_j E\{(X-\mu_j)(X-\mu_j)^T|\omega_j\} = \sum_{j=1}^{k} \pi_j \Sigma_j,$$

$$S_b = \sum_{j=1}^{k} \pi_j (\mu_j - M_o)(\mu_j - M_o)^T,$$

$$M_o = E\{X\} = \sum_{j=1}^{k} \pi_j \mu_j,$$

π_j =Prob. to belong to cluster j

μ_j =sample mean of cluster j

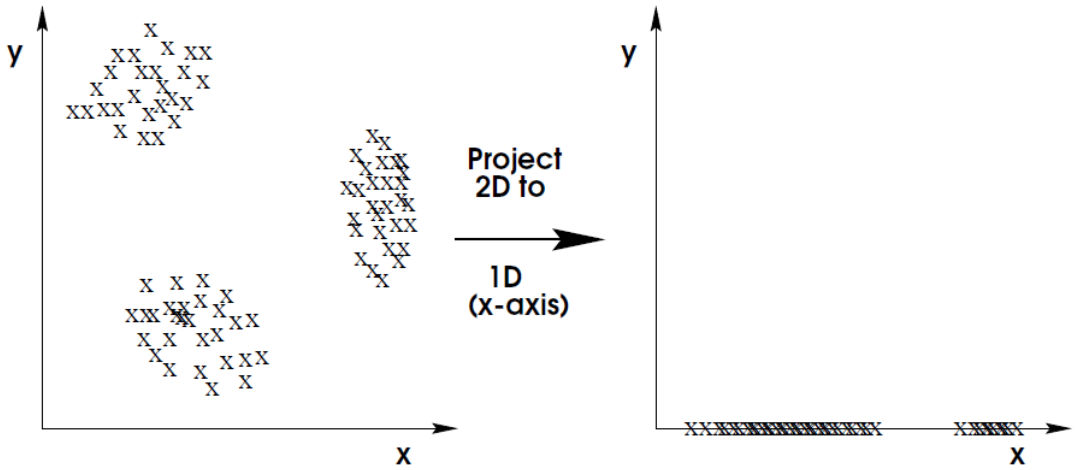
M_o =overall sample mean

Σ_j =sample cov.matrix of cluster j

- One possible criterion: $trace(S_w^{-1} S_b)$, the larger the better. Invariant under nonsingular linear transformation.
- **Maximum Likelihood (ML) criterion:** likelihood of data given by the model and parameters. With Gaussian mixture model: ”interesting” groupings are ”natural” Gaussian groupings

Feature Selection for Clustering

- Choosing the **number of clusters**: depends on dimensionality



2D vs. 1D have different apparent n. of clusters

- Given a dimensionality, some approaches add a penalty for the number of clusters to the log-likelihood, for example the **Bayesian Information Criterion** (BIC) term: $-L\log(N)$ for L free parameters and N data points.
- With no penalty ML would give each data point its own cluster
- Begin with many of clusters and merge; or begin with 1 and split; choose best cluster(s) to split/merge by optimizing the change in the criterion
- Many other ways to find the optimal number of clusters

Feature Selection for Clustering

- Biases: scatter separability criterion favors high dimensionality: value increases monotonically with added dimensions (assuming identical cluster assignments)
- In particular the trace criterion favors high dimensions because the sum is then over more terms, even if extra dimensions do not separate clusters
- Maximum likelihood criterion favors low dimensionality: because the more features are added, the more observed feature values the model must generate->more (negative) terms in log-likelihood
- Alternative possibility: always compute likelihood with all features, but model the irrelevant features as independent of clusters. If data is scarce, also model irrelevant features as independent of each other.

Feature Selection for Clustering

Normalizing the criterion-values by a cross-projection method:

- A typical normalization would divide criteria by the dimensionality, or by $\frac{1}{(2\pi e)^d}$, but they would not remove effect of growing covariance terms. Each criterion would need a ”magic normalization function”
- Projection approach: project clusters to subspaces being compared.
- Feature set S_1 yields clusters C_1, set S_2 yields clusters C_2.

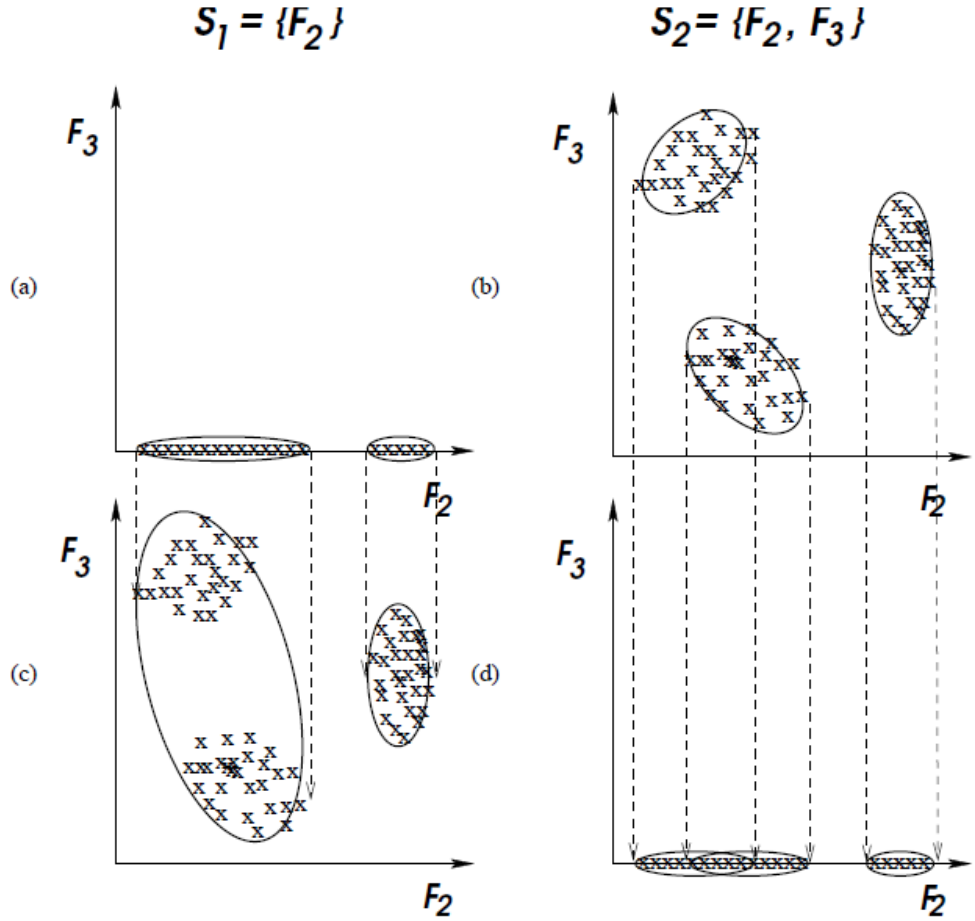
  CRIT = goodness criterion of a clustering on a feature set

$$normalizedValue(S_1, C_1) = CRIT(S_1, C_1) \cdot CRIT(S_2, C_1)$$
$$normalizedValue(S_2, C_2) = CRIT(S_2, C_2) \cdot CRIT(S_1, C_2)$$

  (goodness of clustering given by one feature subset, on both)
- Pick the subset yielding the better normalized score (or smaller dimensionality if scores are equal)
- If two subsets give the same clustering they get the same normalized score, as desired

Feature Selection for Clustering

- Example of the projection approach:



- Compares criteria in the same number of dimensions
- Assumes: clusters in new feature space should be consistent with the data structure in previous feature subset

References

Reading these scientific articles is not necessary for the course! They are simply additional material for a student interested in a particular topic of the course.

- Reference for part 2: Isabelle Guyon and André Elisseef. **An Introduction to Variable and Feature Selection.** *Journal of Machine Learning Research* 3, 1157-1182, 2003.
- Reference for part 3: Jennifer G. Dy and Carla E. Brodley. **Feature Selection for Unsupervised Learning.** *Journal of Machine Learning Research* 5, 845–889, 2004.