Knowledge Discovery in Databases II

Winter Term 2013/2014

Chapter 2: High-Dimensional Data

Lectures : PD Dr Matthias Schubert
Tutorials: Markus Mauder, Sebastian Hollizeck

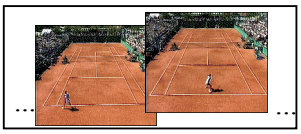
http://www.dbs.ifi.lmu.de/cms/Knowledge_Discovery_in_Databases_II_(KDD_II)

Chapter Overview

1. Introduciton and Challenges
2. Feature Selection
3. Feature Reduction and Metric Learning
4. Clustering in High-Dimensional Data

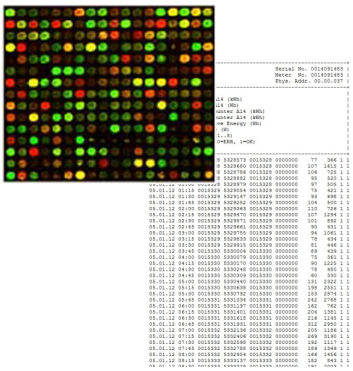
Examples for High-Dimensional Data

- **Image Data**
  - low-level image descriptors (color histograms, pyramids of gradients, textures,..)
  - Regional descriptors
  - Betwewn 16 and 1000 features



- **Metabolome data**
  - feature = concentration of one metabolite
  - between 50 and 2000 features

- **Micro-Array Data**
  - Features correspond to genes
  - Up to 20,000 features
- **Text**
  - Features correspond to words or terms
  - between 5000 and 20000 features



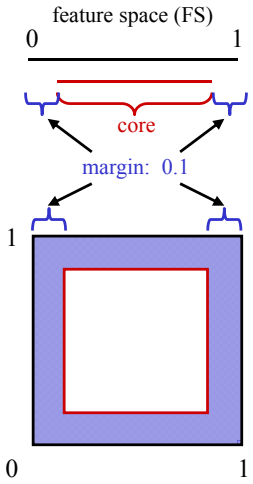
Challenges

Curse of Dimensionality

- Distance to the nearest and the farthest neighbor converge

$$\frac{\text{nearestNeighborDist}}{\text{farthestNeighborDist}} \approx 1$$

- The likelihood that a data object is located on the margin of the data space exponentially increases with the dimensionality



1D: $\mathcal{P}_{FR} = 1^1 = 1$, $\mathcal{P}_{core} = 0.8^1 = 0.8$, $\mathcal{P}_{margin} = 1 - 0.8 = 0.2$

2D: $\mathcal{P}_{FR} = 1^2 = 1$, $\mathcal{P}_{core} = 0.8^2 = 0.64$, $\mathcal{P}_{margin} = 1 - 0.64 = 0.36$

3D: $\mathcal{P}_{FR} = 1^3 = 1$, $\mathcal{P}_{core} = 0.8^3 = 0.512$, $\mathcal{P}_{margin} = 1 - 0.512 = 0.488$

10D: $\mathcal{P}_{FR} = 1^{10} = 1$, $\mathcal{P}_{core} = 0.8^{10} = 0.107$, $\mathcal{P}_{margin} = 1\text{-}0.107 = 0.893$

Challenges

Further Explanation of the *Curse of Dimensionality*:

- Consider the space of *d* relevant features for a given application
  => truely similar objects display small distances in most features
- Now add *d*x* additional features being independent of the *d* above
- With increasing x the distance in the independent subspace will dominate the distance in the complete feature space

⇒ How many relevant features must by similar to indicate object similarity?

⇒ How many relevant features must be dissimilar to indicate dissimilarity?

⇒ With increasing dimensionality the likelihood that two objects are similar in every respect gets smaller.

Further challenges

- Patterns and models on high-dimensional data are often hard to interprete
  ⇒ Long decision rules
- Efficiency in high-dimensional spaces is often limited
  ⇒ index structures degenerate
  ⇒ distance computations are much more expensive
- Pattern might only be observable in subspaces or projected spaces
- Cliques of correlated feature dimensions dominate the object description

2. Feature Selection

Idea: Not all features are necessary to represent the object:

- Features might be useful for the given task
- Correlated features add redundant information

Deleting some of the features can improve the efficiency as well as the quality of the found methods and patterns.

Solution:

Delete all useless features from the original feature space.

Feature Selection

input: Vector space $F = D_1 \times .. \times D_n$ *with* $D = \{D_1, .., D_n\}$.

output: a minimal subspace M over $D` \subseteq D$ which is optimal for a giving data mining task.

=> Minimality increases the efficiency and reduces the effects of the curse of dimensionality

Challenges:

- Optimality depends on the given task
- There are 2^d possible solution spaces (exponential search space)
- There is often no monotonicity in the quality of subspace (Features might only be useful in combination with certain other features)

$\Rightarrow$ For many popular criteria, feature selection is an exponential problem

$\Rightarrow$ Most algorithms employ search heuristics

Selected methods in this course

1. Forward Selection and Feature Ranking
   Information Gain , χ^2-Statistik, Mutual Information
2. Backward Elimination and Random Subspace Selection
   Nearest Neighbor Criterion, Modelbased Search
3. Branch and Bound Search based on Inconsistency
4. Genetic Algorithms for Subspace Search
5. Feature Clustering for Unsupervized Problems

Forward Selektion and Feature Ranking

Input: a supervised learning task

- *Target variable C*
- Training set of labeled feature vectors

Approach

- Compute the quality $q(D_i, C)$ for each dimension in $D' \in \{D_1,..,D_n\}$ to predict the correlation to *C*
- Sort the dimensions $\{D_1,..,D_n\}$ w.r.t. $q(D_i, C)$
- Select the k-best dimension

Assumption:

Features are only correlated via their connection to *C*

=> it is sufficient to evaluate the connection between each single feature *D* and the target variable *C*

Statistic quality measures for features

How suitable is *D* to predict the value of *C*?

Statistical measures :

- Rely on distributions over feature values and target values.
  - For discrete values: determine probabilities all value pairs.
  - For real valued features:
    - Discretize the value space (reduction to the case above)
    - Use probability density functions (e.g. uniform, Gaussian,..)
- How strong is the correlation between both value distributions?
- How good does splitting the values in the feature space seperate values in the target dimension?

Information Gain

Idea: Evaluate the class discrimination in each dimension.
(compare decision tree)

Divide the training set w.r.t. a feature/subspace into two subsets
(either by values or by a split criterion).

The entropie of the training set *T* is defined as

$$entropy(T) = -\sum_{i=1}^{k} p_i \cdot \log p_i$$ (p_i : relative frequency of *class i* in *T*)

entropy(*T*) = 0, falls $p_i = 1$ for any class *i*

entropy(*T*) = 1 for *k* = 2 class having $p_i = 1/2$

$$InformationGain(T, a) = entropy(T) - \sum_{j=1}^{m} \frac{|T(a)_j|}{|T|} \cdot entropy(T(a)_j)$$

where $T(a)_j$ is a subset of *T*, a is an attribute having the j^{th} value.

Real valued attributes: Determine a splitting position in the value set.

χ2-Statistics

Idea: Measure for the independency of a variable from the class variable.
Divide data based on a split value s or based on discrete values

$A = \left|\{o \mid x \le s \wedge Class(o) = C_j\}\right|$ Objects in C_j where $x \le s$

$B = \left|\bigcup_{l \ne j} \{o \mid x \le s \wedge Class(o) = C_l\}\right|$ Objects of other classes with $x \le s$

$C = \left|\{o \mid x > s \wedge Class(o) = C_j\}\right|$ Objects in C_j where $x > s$

$D = \left|\bigcup_{l \ne j} \{o \mid x > s \wedge Class(o) = C_l\}\right|$ Objects of other classes where x > s

χ2-Statistics is defined as: $\chi^2(t, C_j) = \frac{|DB|(AD - CB)^2}{(A+C)(B+D)(A+B)(C+D)}$

larger the maximum/average value over all classes correspond to better features:

$\chi^2_{\max}(a) = \max_{i=1}^{m}\{\chi^2(a, C_i)\}$ or $\chi^2_{avg}(a) = \sum_{i=1}^{m} \Pr(C_i)\chi^2(a, C_i)$

Mutual Information (MI)

Measures the dependency between random variables.

Here: Measure the dependency between the class variable and the features.

(compare features among theself to measure redundancy)

1. Discrete case:

$$I(X,Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

2. Real-valued case:

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

rel. Frequency of the value pair (x,y)

Term for describing the independency of x,y:
In case of statistical independency:
p(x,y)= p(x)p(y) => log(1) = 0

Forward Selektion and Feature Ranking

Advantages:

- Efficient: compares *d* features to C instead of $\binom{d}{k}$ subspaces
- Training suffices with rather small sample sets

Disadvantages:

- Independency assumptions: Classes and feature values must display a direct correlation.
- In case of correlated features: Always selects the features having the strongest direct correlation to the class variable, even if the features are strongly correlated between each other.
  (features might even have an identical meaning)

Backward Elimination

Idea: Start with the complete feature space and delete redundant features

Approach: Greedy Backward Elimination

1. Generate the subspaces *R* of the feature space *F*
2. Evaluate subspaces *R* with the quality measure q*(R)*
3. Select the best subspace R* w.r.t. q(R)
4. If R* has the wanted dimensionality , terminate
   else start backward elimination on R*.

Applications:

- Useful in supervized and unsupervized setting
  in unsupervised cases: q(R) measures structural characteristics
- Greedy search if there is no monotonicity on q(R)
  => for monotonous q(R) employ branch and bound search

Distance-based Subspace Quality

Idea: Subspace quality can be evaluated by the distance between the within-class nearest neighbor and the between class nearest neighbor

Quality criterion:

For all o ∈ DB compute the closest object having the same class $NN_C(o)$ (within-class nearest neighbor) and the closet object belonging to another class $NN_{K \neq C}(o)$ (between-class nearest neighbor) where $C=Class(o)$.

Quality of subspace U: $$q(U) = \frac{1}{|DB|} \cdot \sum_{o \in DB} \frac{NN^U_{K \neq C}(o)}{NN^U_C(o)}$$

Remark: q(U) is not monotonous.

=> By deleting a dimension the quality can increase or decrease.

Model-based Approach

Idea: Directly employ the data mining function to evaluate the subspace.

Example: Evaluate each subspace by training a Naive Bayes classifier and 10-facher and evaluate the classifier in the subspace R by employing cross validation on a moderate sample set.

Practical aspects:

- Success of the data mining algorithm must be measurable (e.g. class accuracy)
- Runtime for training and applying the classifier should be low
- The classifier parameterization should not be of great importance
- Test set should have a moderate number of instances

Backward Elemination

Advantages:

- Considers complete subspaces
  (multiple dependencies are used)
- Can recognize and eliminate redundant features

Disadvantages:

- Tests w.r.t. subspace quality usually require much more effort
- All solutions employ heuristic greedy search which does not need to find the optimal result space

Branch and Bound Search

Given: A classification task over the feature space *F*.

Aim: Select the *k* best dimensions to learn the classifier.

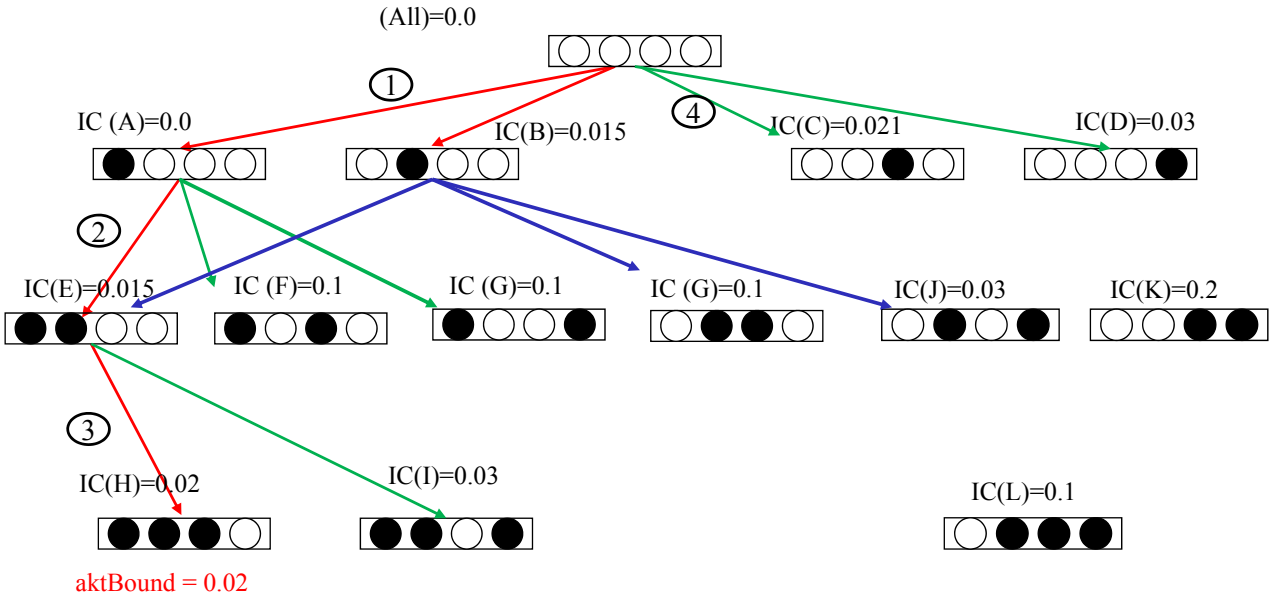
Backward-Elimination based in Branch and Bound:

```
FUNCTION BranchAndBound(Featurespace F, int k)
        queue.init(ASCENDING);
        queue.add(F, quality(F))
        curBound:= INFINITY;
        WHILE queue.NotEmpty() and aktBound > queue.top() DO
                curSubSpace:= queue.top();
                FOR ALL Subspaces U of curSubSpace DO
                        IF U.dimensionality() = k THEN
                             IF quality(U)< curBound THEN
                                  curBound  := quality(U);
                                  BestSubSpace := U;
                        ELSE
                             queue.add( U, quality(U));
        RETURN BestSubSpace
```

Example: Branch and Bound Search

Example: Target dimension k = 1.

○ selected feature ● deleted feature



Subspace Consistency

Idea: Having identical vectors u, v ($v_i = u_i$ $1 \le i \le d$) in subspace U for the class labels are different ($C(u) \neq C(v)$).

=> the subspace displays an inconsistent *labeling*

Measuring the inconsistency of a subspace *U*:

- $X_U(A)$: Amount of all identical vectors *A* in *U*
- $X^c_U(A)$: Amount of all identical vectors in *U* having class label *C*
- $IC_U(A)$: inconsistency w.r.t. *A* in *U*

$$IC_U(A) = X_U(A) - \max_{c \in C} X^c_U(A)$$

Inconsistency of U: $$IC(U) = \frac{\sum_{A \in DB} IC_U(A)}{|DB|}$$

Monotonicity: $$U_1 \subset U_2 \Rightarrow IC(U_1) \geq IC(U_2)$$

Branch and Bound based on inconsistency

Advantage:

- Monotonicity allows efficient search for optimal solutions
- Well-suited for binary or discrete data
  (identical vectors are very likely with decreasing dimensionality)

Disadvantages:

- Useless without groups of identical features (real-valued vectors)
- Worst-Case runtime complexity remains exponential in d

K-dimensional Projections and Model-Based Evalution

Idea: Select *n* random Subspace having the target dimensionality k out of the $\binom{d}{k}$ possible subspaces and evaluate each.

Application:

- Needs quality measures for complete subspaces
- Trade-off between quality and effort depends on k.

Disadvantages:

- No directed search for combining well-suited and non-redundant features.
- Computational effort and result strongly depend on the used quality measure and the sample site.

Genetic Algorithms for Feature Selection

Idea: Employ sophisticated randomized search schemes on the set of k-dimensional subspaces.

Genetic Algorithms:

- **Population of solutions** := set of *k*-dimensional subspaces
- **Fitness criterion**: quality measure for a subspace
- **Rules and likelihood for mutation**:
  dimension D_i in U is replaced by dimension D_j with a likelihood of x%
- **Reproduction**: combination of 2 subspaces *U1* and *U2*:
  - *Unite the features sets of U1 and U2.*
  - *Delete random dimension until dimensionality is k*
- **Selection**: All subspaces having at least a quality of y% of the best fitness in the current generation are copied to the next generation.
- **Free tickets**: Additionally each subspace is copied to the next generation with a probability of *u%* into the next generation.

Genetic Algorithm: Schema

```
Generate initial population
WHILE Max_Fitness > Old_Fitness  DO
        Mutate current population
        WHILE nextGeneration < PopulationSize DO
            Generate new  candidate from pairs of old subspaces
            IF K has a free ticket or K is fit enough THEN
                    copy K to the next generation
RETURN fittest subspace
```

Genetic Algorithm

Remarks:

- Here: only basic algorithmic scheme (multiple variants)
- Efficient convergence by „Simulated Annealing“
  (Likelihood of free tickets decreases with the iterations)

Advantages:

- Can escape from local extreme values during the search
- Often good approximations for optimal solutions

Disadvantages:

- Runtime is not bounded can become rather inefficient
- Configuration depends on many parameters which have to be tuned to achieve good quality results in efficient time

Feature-Clustering based on correlation

Given: A feature space F and an unsupervized data mining task.

Target: Reduce F to subspace to k dimension while reducing redundancy.

Idea: Cluster the features in the space of objects and select one representative feature for each of the clusters.
(This is equivalent to clustering in a transposed data matrix)

Requires a measure for the redundancy of features:

Pearson correlation: $COR(X,Y) = \frac{COV(X,Y)}{\sqrt{VAR(X)}\sqrt{VAR(Y)}}$

Feature-Clustering based on correlation

Algorithmic scheme:

- Cluster features with a *k*-medoid clustering method based on correlation
- Select the medoids to span the target data space

Remark:

- For group/cluster of dependent features there is one representative feature
- Other clustering algorithms could be used as well.
- For large dimensionalities, approximate clustering methods are used due to there linear runtime (c.f. BIRCH chapter 3.)

DATABASE SYSTEMS GROUP

Feature-Clustering based on correlation

LMU

Advantages:

- Depending on the clustering algorithm quite efficient
- Does not need any type of examples

Disadvantages:

- Results are usually not deterministic (partitioning clustering)
- Representatives are usually unstable for different clustering methods and parameters.
- Based on pairwise correlation and dependencies
  => multiple dependencies are not considered

Summary of Feature Selection Approaches

- *Forward-Selection*: Examines each dimension D‘ $\in\{D_1,..,D_d\}$. and selects the k-best to span the target space.
  - Greedy Selection based on Information Gain, $\chi 2$ Statistics or Mutual Information
- *Backward-Elimination*: Start with the complete feature space and successively remove the worst dimensions.
  - Greedy Elimination with model-based and nearest-neighbor based approaches
  - Branch and Bound Search based on inconsistency
- *k-dimensional Projections*: Directly search in the set of k-dimensional subspaces for the best suited
  - Genetic algorithms (quality measures as with backward elemination)
  - Feature clustering based on correlation

Feature Selection

Discussion:

- Many algorithms based on different heuristics
- There are two reason to delete features:
  - Redundancy: Features can be expressed by other features.
  - Missing Correlation to the target variable
- Often even approximate results are capable of increasing efficiency and quality in a data mining task
- **Caution**: Selected features need not have a causal connection to the target variable, but both observation might depend on the same mechanisms in the data space. (hidden variables)

What you should know by now

- Reasons for using feature selection
- Curse-of-Dimensionality
- Forward selection and feature ranking
  - Information Gain
  - $\chi 2$ Statistics
  - Mutual Information
- Backward Elimination
  - Model-based subspace quality
  - Nearest neighbor-based subspace quality
  - Inconsistency and Branch & Bound search
- k-dimensional projections
  - Genetic algorithms
  - Feature clustering

Literature

- A.Blum and P. Langley: *Selection of Relevant Features and Examples in Machine Learning*, Artificial Intelligence (97),1997
- H. Liu and L. Yu: *Feature Selection for Data Mining* (WWW), 2002
- L.C. Molina, L. Belanche, Â. Nebot: Feature Selection Algorithms: *A Survey and Experimental Evaluations*, ICDM 2002, Maebashi City, Japan
- P. Mitra, C.A. Murthy and S.K. Pal: *Unsupervised Feature Selection using Feature Similarity*, IEEE Transacitons on pattern analysis and Machicne intelligence, Vol. 24. No. 3, 2004
- J. Dy, C. Brodley: *Feature Selection for Unsupervised Learning*, Journal of Machine Learning Research 5, 2004
- I. Guyon, A. Elisseeff: An Introduction to Variable and Feature Selection, Journal of Machine Learning Research 3, 2003
- M. Dash, H. Liu, H. Motoda*: Consistency Based Feature Selection*, 4th Pacific-Asia Conference, PADKK 2000, Kyoto, Japan, 2000