Table of Contents

1. Introduction to Feature Spaces
2. Challenges of High Dimensional Data
3. Supervised Feature Selection
   3.1 Forward Selection and Feature Ranking
   3.2 Backward Elimination and Random Subspace Selection
   3.3 Subspace Projections
4. Feature Reduction and Metric Learning
   4.1 Reference Point Embedding
   4.2 Principle Component Analysis (PCA)
   4.3 Singular Value Decomposition (SVD)
   4.4 Kernel PCA
   4.5 Further Measures
5. Clustering High Dimensional Data
   5.1 Challenges
1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data
Feature Transform

- Consider the following spaces:
  - $\mathbb{U}$ denotes the universe of data objects
  - $\mathbb{F} \subseteq \mathbb{R}^n$ denotes an $n$-dimensional feature space
- A feature transformation is a mapping $f : \mathbb{U} \rightarrow \mathbb{R}^n$ of objects from $\mathbb{U}$ to the feature space $\mathbb{F}$.

Similarity Model

- A similarity model $S : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}$ is defined for all objects $p, q \in \mathbb{U}$ as
  $$S(p, q) = sim(f(p), f(q))$$
  where $sim : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ is a similarity measure or a dissimilarity (distance) measure in $\mathbb{F}$.
Similarity versus Dissimilarity

Comments:

- Often, dissimilarity (distance) is measured instead of similarity
- This is a small but important difference!
  - A similarity measure \((sim)\) assigns high values to similar objects
  - A dissimilarity measure \((dist)\) assigns low values to similar objects
- The design of \(f\) and the definition of \(sim/dist\) are important assumptions about the patterns we want to find later in the data
- As explained before, \(f\) and \(sim/dist\) can be derived manually (explicit transformation and coding versus implicit Kernels) or automatically (representation learning)
Dissimilarity

- Dissimilarity measures follow the idea of the geometric approach
  - objects are defined by their perceptual representations in a perceptual space
  - perceptual space = psychological space
  - geometric distance between the perceptual representations defines the (dis)similarity of objects

- Within the scope of Feature-based similarity
  - perceptual space = feature space $\mathbb{F}$ or feature representation space $\mathbb{R}^n$
  - geometric distance = distance function
Distance Functions

• The distance measure $dist$ is a distance function if it is reflexive, non-negative, and symmetric.

• A distance function $dist$ is a metric if it additionally satisfies the triangle inequality.

• Comments:
  • Sound mathematical interpretation
  • Allow domain experts to model their notion of dissimilarity
  • Metric distances allow to tune efficiency of data mining approaches
  • Long-lasting discussion of whether the distance properties and in particular the metric properties reflect the perceived dissimilarity correctly, see the following contradicting example:
Similarity versus Dissimilarity (again)

- Transformation
  - Let $\mathbb{F}$ be a feature space and $\text{dist} : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{R}$ be a distance function.
  - Any monotonically decreasing function $f : \mathbb{R} \rightarrow \mathbb{R}$ defines a similarity function $s : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{R}$ as follows:

$$\forall x, y \in \mathbb{F} : s(x, y) = f(\text{dist}(x, y))$$

- Some prominent similarity functions $(x, y \in \mathbb{F})$:
  - exponential:
    $$s(x, y) = e^{-\text{dist}(x, y)}$$
  - logarithmic:
    $$s(x, y) = 1 - \log(1 + \text{dist}(x, y))$$
  - linear: $s(x, y) = 1 - \text{dist}(x, y)$
Similarities: Examples (only very few)

- Dot-Product \((x, y \in \mathbb{F} \subseteq \mathbb{R}^d)\)
  \[
x \cdot y^T = \sum_{i=1}^{d} x_i \cdot y_i = \|x\| \cdot \|y\| \cdot \cos \angle(x, y)
  \]

- Cosine \((x, y \in \mathbb{F} \subseteq \mathbb{R}^d)\)
  \[
  \frac{x \cdot y^T}{\|x\| \cdot \|y\|}
  \]

- Pearson Correlation \((x, y \in \mathbb{F} \subseteq \mathbb{R}^d)\)
  \[
  \frac{\sum_{i=1}^{d} (x_i - \bar{x}_i) \cdot (y_i - \bar{y}_i)}{\sqrt{\sum_{i=1}^{d} (x_i - \bar{x}_i)^2} \cdot \sqrt{\sum_{i=1}^{d} (y_i - \bar{y}_i)^2}}
  \]
  where \(\bar{z}_i\) denotes the mean in attribute \(i\) over all data points

- Random-Walk Kernel (for graphs \(x, y\))
  - Count common (random) walks in \(x\) and \(y\)
  - Walks are sequences of nodes (connected by edges)
Distances: Examples (only very few)

- \( L_p \)-norm (aka Minkowski metric) \( (x, y \in F \subseteq \mathbb{R}^d) \)

\[
    L_p(x, y) = \sqrt[p]{\sum_{i=1}^{d} |x_i - y_i|^p}
\]

where
- \( p < 1 \): fractional Minkowski distance
- \( p = 1 \): Manhattan distance
- \( p = 2 \): Euclidean distance
- \( p = \infty \): Chebyshev/Maximum distance

- Malahanobis distance

- Hamming distance \( \text{HammingDist}(x, y) = \sum_{i=1}^{d} \left\{ \begin{array}{ll} 1 & : x_i \neq y_i \\ 0 & : \text{else} \end{array} \right. \)
Kapitel 2: Challenges

1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data
A Motivating Example

- Let’s play the baby shapes game (truly motivating for students ...): Group the items!!!

- What about grouping based on both shape and color?
- Lesson to learn: there may be different semantic concepts (and their corresponding patterns) hidden in the data (here: shape and color)
The good old days of data mining . . .

• Data generation and, to some extend, data storage was costly (hard to imagine but those were the days ...)
• Domain experts carefully considered which features/variables to measure before designing experiments/a feature transform/...
• Consequence: also data sets were well designed and potentially contained only a small number of relevant features
Nowadays, data science is also about integrating everything

- Generating and storing data is easy and cheap
- People tend to measure everything they can and even more (including even more complex feature transformations)
- The Data Science mantra is often interpreted as “we can analyze data from as many sources as (technically) possible, just record anything you can”
- Consequence: data sets are high-dimensional containing a large number of features but the relevancy of each feature for the analysis goal is not clear a priori
High-dimensional Data is NOT a Myth

• Example: Image data
  • Low-level image descriptors (color histograms, textures, shape information ...)
  • Regional descriptors: between 16 and 1,000 features
  • ...

• Example: Metabolome data
  • Feature = concentration of one metabolite (intermediates/results of metabolism)
  • Bavaria newborn screening (for each baby, the blood concentrations of 43 metabolites are measured in the first 48 hours after birth)
  • between 50 and 2,000 features
More High-dimensional Data

• Example: Microarray data (deprecated)
  • Features correspond to genes
  • Up to 20,000 features
  • Dimensionality is much higher than the sample size

• Example: Text data
  • Term frequency: features correspond to words/terms
  • Between 5,000 and 20,000 features (and even more)
  • Often, esp. in social media: abbreviations, colloquial language, special words

Excerpt from LMU website:
http://tinyurl.com/qhq6byz
Overview:

- Distances grow
- Contrast of distances diminish (concentration problem)
- Meaning of “neighborhood” concept
- Growing data space
- Growing hypothesis space
- Empty spaces and importance tails
- Different semantic layers
- ...

So let us have a closer look on these problems ...
Distances Grow

The following example uses the Euclidean distance but holds for most distance measures:

- Consider 2D vectors $a = (1, 2)$ and $b = (4, 3)$
- The Euclidean distance between $a$ and $b$ is

$$L_2(a, b) = L_2((1, 2), (4, 4)) = \sqrt{(1 - 4)^2 + (2 - 3)^2} = \sqrt{10}$$

which corresponds to the norm of the difference vector $c = (3, 1)$:

$$\|c\|_2 = \sqrt{3^2 + 1^2}$$
With increasing dimensionality, distances grow, too:

- Example: $L_2((1,2),(4,3)) = \sqrt{10}$
- Now double the feature vector length (double the original features): $L_2((1,2,1,2),(4,3,4,3)) = \sqrt{(3^2 + 1^2 + 3^2 + 1^2)} = \sqrt{20}$
- Effect seems not so important, values might be only in a larger scale?
- NOPE:

  **Contrast of distances is lost in high dimensional data since distances grow more and more alike!**

This is know as the Concentration of Distances problem (see next)
Concentration Phenomenon

- As dimensionality grows, distance values grow, too, such that the (numerical) contrast provided by usual measures decreases or even diminishes.
- In other words, the distribution of norms in a given distribution of points tends to concentrate.

- Example: Euclidean norm of vectors consisting of several variables that are (assumed to be) independent and identically distributed:

\[ \|y\|_2 = \sqrt{y_1^2 + y_2^2 + \ldots + y_d^2} \]

- In high dimensional spaces this norm behaves unexpectedly . . .
Concentration of Distances

Theorem: Concentration of Distances

• Let $y$ be a $d$-dimensional vector $(y_1, \ldots, y_d)$ where all components $y_i (1 \leq i \leq d)$ are independent and identically distributed.

• Then the mean and the variance of the Euclidean norm are:

\[
\mu_{\|y\|} = \sqrt{a \cdot d - b} + O(d^{-1}) \quad \text{and} \quad \sigma_{\|y\|} = b + O(d^{-1/2})
\]

where $a$ and $b$ are parameters depending only on the central moments of order 1, 2, 3, 4.

Interpretation:

• The norm grows proportionally to $\sqrt{d}$, but the variance remains approx. constant for large $d$ (because $\lim_{d \to \infty} d^{-\text{const}} = 0$).

• With growing dimensionality, the relative error made by taking $\mu_{\|y\|}$ instead of $\|y\|$ becomes negligible.

Implications from the concentration of distances:

• A lot of data mining methods use distances and neighborhoods to define patterns (e.g. $k$NN classifier, density-based clustering, distance-based outlier detection, ...)

• Using neighborhoods is based on a key assumption:
  • Objects that are similar to an object $o$ are in its neighborhood
  • Object that are dissimilar to $o$ are not in its neighborhood

• What if all objects are in the same neighborhood?
  • Consider the above effect on distances: $k$NN distances are almost equal to each other, i.e., the $k$ nearest neighbors are random objects
Neighborhood Concept Become Meaningless

**Definition: Unstable Neighborhood**

- A NN-query is unstable for a given $\varepsilon$ if the distance from the query point to most data points is less than $(1 + \varepsilon)$ times the distance from the query point to its nearest neighbor.

- It can be shown that with growing dimensionality, the probability that a query is unstable converges to 1.
Neighborhood Concept Become Meaningless

• Consider a $d$-dimensional query point $q$ and $n$ $d$-dimensional sample points $x_1, \ldots, x_n$ (independent and identically distributed)

• We define:

$\text{DMIN}_d = \min \{ L_2(x_i, q) | 1 \leq i \leq n \}$ (dist to next neighbor)

$\text{DMAX}_d = \max \{ L_2(x_i, q) | 1 \leq i \leq n \}$ (dist to farthest neighbor)

Theorem

• If $\lim_{d \to \infty} \left( \frac{\text{VAR}_{L_2(x_i, q)}}{\mu_{L_2(x_i, q)}} \right) = 0$

• Then $\forall \varepsilon > 0 : \lim_{d \to \infty} \mathbb{P}(\text{DMAX}_d \leq (1 + \varepsilon) \text{DMIN}_d) = 1$

In other words: if the precondition holds, all points converge to the same distance from the query!

---

Visually: Pairwise distances of a sample of 105 instances drawn from a uniform \([0, 1]\) distribution, normalized \((1/\sqrt{d})\).
Neighborhood Concept Become Meaningless

• Be clear about the precondition of the Theorem!!!
• Consider the feature space of $d$ relevant features for a given application (i.e., truly similar objects display small distances in most features)
• Now add $d \cdot c$ additional features being independent of the initial feature space
• With increasing $c$ the distance in the independent subspace will dominate the distance in the complete feature space
• So the question is:
  How many relevant features must be similar to indicate object similarity?
  (or: 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.
Growing Data Space

- OK, the data space grows with increasing dimensionality
- But what are the problems?
- In low dimensional spaces we have some (intuitive) assumptions on the behavior of volumes (sphere, cube, etc.) and on the distribution of data objects
- However, basic assumptions do not hold in high dimensional spaces:
  - Spaces become sparse or even empty and the probability of one object inside a fixed range tends to become zero
  - Distribution of data has a strange behavior e.g. a normal distribution has only few objects in its center and the tails of distributions become more important

We will have a closer look on these issues ...
Growing Hypotheses Space

• The more features, the larger the hypothesis space
• The lower the hypothesis space is,
  • the easier it is to find the correct hypothesis
  • the less examples you need to properly test hypothesis
• Consider $f$ a unit multivariate normal distribution and normal kernel (KDE)
• The aim is to find an estimate $\hat{f}$ of $f$ at the point $0$
• The relative mean square error should be fairly small, e.g.
\[
\frac{\mu^2_{\hat{f}(0) - f(0)}}{f(0)^2} < 0.1
\]

<table>
<thead>
<tr>
<th>Dim.</th>
<th>Req. sample size to achieve 0.1 error estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>768</td>
</tr>
<tr>
<td>8</td>
<td>43.700</td>
</tr>
<tr>
<td>10</td>
<td>842.000</td>
</tr>
</tbody>
</table>

Even with only 10 dimensions, we need nearly a million observations to estimate a distribution with an error less than 0.1!!!

---

Empty Spaces and Tails

- Consider a $d$-dimensional space with partitions of constant size $1/m$
- The number of cells $N$ increases exponentially in $d$: $N = m^d$
- Suppose $x$ points are randomly placed in this space
- In low-dimensional spaces there are few empty partitions and many points per partitions
- In high-dimensional spaces there are far more partitions than points there are many empty partitions

$d = 1$
$$N = 4$$

$d = 2$
$$N = 4^2 = 16$$

$d = 3$
$$N = 4^3 = 64$$
Analogously:

- Consider a simple partitioning scheme, which splits the data in each dimension in 2 halves.
- For $d$ dimensions we obtain $2^d$ partitions.
- Consider $n = 10^6$ samples in this space.
- For $d \leq 10$ such a partition may make sense.
- For $d = 100$ there are around $10^{30}$ partitions, so most partitions are empty (given the above $10^6$ points).
Empty Spaces and Tails

- Consider a hyper-cube range query with length $s$ in all dimensions, placed arbitrarily in the data space $[0, 1]^d$
- $E$ is the event that an arbitrary point lies within the query cube
- The probability for $E$ is $P(E) = s^d$

⇒ with increasing dimensionality, even very large hyper-cube range queries are not likely to contain a point
Empty Spaces and Tails

• The same holds of course for a spherical range query (instead of a cubical range query)

• Consequence: with increasing dimensionality the center of the hyper-cube (or more generally: of the data space) becomes less important and the volume of the data space concentrates in its corners (i.e. randomly distributed points tend to be on the border of the data space . . .)

• This seems to be a distortion of space compared to our 3D way of thinking — and that is actually what it is ...
And that also means, that the tails of a distribution become extremely important

- Consider standard density function \( f \)
- Consider \( \hat{f} \) with
  \[
  \hat{f}(x) = \begin{cases} 
  0 & f(x) < 0.01 \\
  f(x) & \text{else}
  \end{cases}
  \]

- Rescaling \( \hat{f} \) to a density function will make very little difference in 1D, since very few data points occur in regions where \( f \) is very small
Empty Spaces and Tails

But for high dimensional data:

• More than half of the data has less than 1/100 of the maximum density $f(0)$ (for $\mu = 0$)

• Example: 10-dimensional Gaussian distribution $X$:

$$\frac{f(X)}{f(0)} = e^{-\frac{1}{2}X^TX} \approx e^{-\frac{1}{2}\chi^2_{10}}$$

since the median of the $\chi^2_{10}$ distribution is 9.34, the median of $\frac{f(X)}{f(0)}$ is $e^{-\frac{9.34}{2}} = 0.0094$

• Thus, most objects occur at the tails of the distribution

• In other words, in contrast to the low dimensional case, regions of relatively very low density can be extremely important parts
Empty Spaces and Tails

But for high dimensional data:

- More than half of the data has less than $1/100$ of the maximum density $f(0)$ (for $\mu = 0$)
- Example: 10-dimensional Gaussian distribution $X$:

$$\frac{f(X)}{f(0)} = e^{-\frac{1}{2}X^T X} \approx e^{-\frac{1}{2} \chi^2_{10}}$$

since the median of the $\chi^2_{10}$ distribution is 9.34, the median of $\frac{f(X)}{f(0)}$ is $e^{-\frac{9.34}{2}} = 0.0094$

- Thus, most objects occur at the tails of the distribution
- In other words, in contrast to the low dimensional case, regions of relatively very low density can be extremely important parts
Example: $(\mu = 0, \sigma = 1)$

- 1D: 90% of the mass of the distribution lies between $-1.6$ and $1.6$
- 10D: 99% of the mass of the distribution is at points whose distance from the origin is greater than $1.6$
- Thus, it is difficult to estimate the density, except for enormous samples because in very high dimensions virtually the entire sample will be in the tails
Further Problems

- Patterns and models on high-dimensional data are often hard to interpret, e.g. long decision rules
- Efficiency in high-dimensional spaces is often limited because e.g. index structures degenerate and distance computations are much more expensive
- There may be different semantic layers so pattern might only be observable in subspaces or projected spaces (cf. the baby shape game)
- Cliques of correlated features dominate the object description
The Case Kröger versus Tresp

- Summarizing: the higher the dimensionality, the worse is the expected outcome of the mining algorithm (i.e., dimensionality is a curse, says Kröger)

- Well, not in general, the Kernel trick shows the opposite: through the extension of the data space with new attributes, the mining algorithm (e.g. a SVM classifier) gets more accurate (i.e., dimensionality is a blessing, says Tresp in his ML course)

- So: Who is right????????? – Both – What????
• Look at what we assumed for the curse: attributes are independent (and often even uniformly distributed)
• These attributes are likely to be irrelevant for the mining task
• And the blessing: a Kernel (if it works) adds relevant attributes (even more relevant than the original ones)
• Message: high-dimensional data is tricky and the curse can come by as several problems
  • Some are due to irrelevant attributes, so try to get rid of irrelevant attributes and keep the relevant ones
  • Some are instead of relevant attributes, so among the relevant attributes, try to get rid of redundant ones
1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection
   3.1 Forward Selection and Feature Ranking
   3.2 Backward Elimination and Random Subspace Selection
   3.3 Subspace Projections

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data
Feature Selection

- A task to remove irrelevant and/or redundant features
  - Irrelevant features:
    - Not useful for a given task
    - Probably decrease accuracy
  - Redundant features:
    - Strongly correlated with another relevant feature
    - Does not drop the accuracy, but may drop efficiency, explainability, etc.

- Deleting irrelevant and redundant features can improve the quality as well as the efficiency of the methods and the found patterns.

- New feature space: Delete all useless features from the original feature space.

**Keep in mind...**

Feature selection ≠ Dimensionality reduction
Feature selection ≠ Feature extraction
Irrelevance

Feature $y$ is irrelevant, because if we omit $x$, we have only one cluster, which is uninteresting.

Redundancy

Features $x$ and $y$ are redundant, because $x$ provides (appr.) the same information as feature $y$ with regard to discriminating the two clusters.

Irrelevant and Redundant Features (Supervised Case)

Irrelevance

Feature $y$ separates well the two classes. Feature $x$ is irrelevant. Its addition “destroys” the class separation.

Redundancy

Features $x_1$ and $x_2$ are redundant.

Individually irrelevant together relevant

Source: http://www.kdnuggets.com/2014/03/machine-learning-7-pictures.html
Problem Definition

- **Input:** Vector space $F = d_1 \times \cdots \times d_n$, dimensions $D = \{d_1, \ldots, d_n\}$.
- **Output:** a minimal subspace $M$ over dimensions $D' \subseteq D$ which is optimal for a given data mining task.
  - Minimality increases the efficiency, reduces the effects of the curse of dimensionality and increases interpretability.

**Challenges:**

- Optimality depends on the given task.
- There are $2^d$ possible solution spaces (exponential complexity)
- This search space is similar to the frequent itemset mining problem, but:
  - There is often no monotonicity in the quality of subspace (which is important for efficient searching)
  - Features might only be useful in combination with other certain features.

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

$\Rightarrow$ Most algorithms employ search heuristics.
Two Main Components (Steps)

1. Feature subset generation
   - Single dimensions
   - Combinations of dimensions (subspaces)

2. Feature subset evaluation
   - Importance scores like information gain, $\chi^2$
   - Performance of a learning algorithm

⇒ How to select/evaluate features? How to traverse the search space?
Feature Selection/Evaluation Methods

1. Filter methods
   – Explores the general characteristics of the data, independent of the learning algorithm.

2. Wrapper methods
   – The learning algorithm is used for the evaluation of the subspace.

3. Embedded methods
   – The feature selection is part of the learning algorithm.
• Filter methods
  – Basic idea: assign an “importance” score to each feature to filter out useless ones
  – Examples: information gain, $\chi^2$-statistic, TF-IDF for text...
  – Disconnected from the learning algorithm.
  – Pros:
    ○ Fast and generic
    ○ Simple to apply
  – Cons:
    ○ Doesn’t take into account interactions between features
    ○ Individually irrelevant features, might be relevant together
    ○ Too generic?
• Wrapper methods
  – A learning algorithm is employed and its performance is used to determine the quality of selected features.
  – Pros:
    ○ take feature dependencies into account
    ○ interaction between feature subset search and model selection
  – Cons:
    ○ higher risk of overfitting than filter techniques
    ○ very computationally intensive, especially if building the classifier has a high computational cost.
• Embedded methods
  – Such methods integrate the feature selection in model building
  – Example: decision tree induction algorithm: at each decision node, a feature has to be selected.
  – Pros:
    ○ less computationally intensive than wrapper methods.
  – Cons:
    ○ specific to a learning method
Search Strategies in the Feature Space

- **Forward selection**
  - Start with an empty feature space and add relevant features

- **Backward selection**
  - Start with all features and remove irrelevant features

- **Branch-and-bound**
  - Find the optimal subspace under the monotonicity assumption

- **Randomized**
  - Randomized search for a $k$ dimensional subspace

- ...
1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection
   3.1 Forward Selection and Feature Ranking
   3.2 Backward Elimination and Random Subspace Selection
   3.3 Subspace Projections

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data
General Idea

Input

• Target dimensionality \( k \leq d \)
• Training set of \( n \)-dimensional feature vectors with features \( d_1, d_2, \ldots, d_n \) and target variable \( C \)

General Approach

• Compute the quality \( q(d_i, C) \) for each dimension \( d_i \in \{d_1, \ldots, d_n\} \) to predict the correlation to \( C \)
• Sort the dimensions \( d_1, \ldots, d_n \) w.r.t. \( q(d_i, C) \)
• Select the best \( k \) dimensions

Basic Assumption

• Attribute independence (no correlations between features)
General Idea

Key Concept

• Quality of feature $d_i$: How suitable is the feature for predicting the value of class attribute $C$?
• Statistical measures
  • Rely on distributions over feature values and target values
  • How strong is the correlation between both value distributions?
  • How good does splitting the values in the feature space separate values in the target dimension?
Quality of Features

How to measure the distribution?

• For discrete values: determine probabilities for all value pairs.

• For real valued features:
  • Discretize the value space (reduction to the case above)
  • Use probability density functions (e.g. uniform, Gaussian,..)

• Example quality measures:
  • Information Gain
  • Chi-square $\chi^2$-statistics
  • Mutual Information
Quality of Features: Entropy

- Idea: Evaluate class discrimination in each dimension (Used in ID3 algorithm for decision trees)
- It uses entropy, a measure of pureness of the data set $S$ w.r.t. the class labels $c_i \in C$

$$\text{Entropy}(S) = \sum_{c_i \in C} -p_{c_i} \cdot \log_2(p_{c_i})$$

where $p_{c_i}$ is the relative frequency of class $c_i$ in $S$
Quality of Features: Entropy

Example

- Let $S$ be a collection of positive and negative examples for a binary classification problem, i.e., $C = \{+, -\}$
- Then $Entropy(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$
  - $p_+$ is the percentage of positive examples in $S$
  - $p_-$ is the percentage of negative examples in $S$
- Example splits:
  - Let $S : [9+, 5-]$: $Entropy(S) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log\left(\frac{5}{14}\right) = 0.940$
  - Let $S : [7+, 7-]$: $Entropy(S) = -\frac{7}{14} \log_2\left(\frac{7}{14}\right) - \frac{7}{14} \log\left(\frac{7}{14}\right) = 1$
  - Let $S : [14+, 0-]$: $Entropy(S) = -\frac{14}{14} \log_2\left(\frac{14}{14}\right) - \frac{0}{14} \log\left(\frac{0}{14}\right) = 0$
- Obviously: Entropy is 0, when all samples belong to the same class while Entropy is 1, when there is an equal number of samples in all splits
Quality of Features: Information Gain

- The information gain $Gain(S, d_i)$ of a feature $d_i$ relative to a training set $S$ measures the gain reduction in $S$ due to splitting on $d_i$, i.e., the entropy of the data set $S$ before splitting minus the weighted sum of the entropies of all splits $S_j$ in a given feature $d_i$:

$$Gain(S, d_i) = Entropy(S) - \sum_{S_j} \frac{|S_j|}{|S|} \cdot Entropy(S_j)$$

- For nominal attributes: use attribute values for splitting, i.e. each possible value $v_j$ in $d_i$ defines one split and $S_j$ contains all objects having $v_j$ in $d_i$

- For real valued attributes: Determine a splitting position $v$ in the value set and split e.g. into $S_1$ containing all objects with values $\leq v$ and $S_2$ containing all objects with values $> v$ in $d_i$
Example

• Which dimension, “Humidity” or “Wind”, is better?

• Larger values are better!
Quality of Features: Chi-square Statistics

- Idea: Measures the independence of a feature $d$ from the class variable $C$
- Contingency table: divide data based on a split value $s$ or based on discrete values
- Example: Does “liking science fiction movies” imply “playing chess”?

<table>
<thead>
<tr>
<th>Predictor attribute</th>
<th>Play chess</th>
<th>Not play chess</th>
<th>Sum (row)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Like science fiction</td>
<td>250</td>
<td>200</td>
<td>450</td>
</tr>
<tr>
<td>Not like science fiction</td>
<td>50</td>
<td>1000</td>
<td>1050</td>
</tr>
<tr>
<td>Sum(col.)</td>
<td>300</td>
<td>1200</td>
<td>1500</td>
</tr>
</tbody>
</table>

Chi-square $\chi^2$ test

$$\chi^2 = \sum_{i=1}^{|C|} \sum_{j=1}^{\text{Values}(d)} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

- $o_{ij}$: observed freq. of value $j$ in class $i$
- $e_{ij}$: expected freq. of value $j$ in class $i$
Example

- Compute the $\chi^2$ values for the following table (numbers in parenthesis are expected counts calculated based on the data distribution in the two categories)

<table>
<thead>
<tr>
<th>Class attribute</th>
<th>Play chess</th>
<th>Not play chess</th>
<th>Sum (row)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Like science fiction</td>
<td>250 (90)</td>
<td>200 (360)</td>
<td>450</td>
</tr>
<tr>
<td>Not like science fiction</td>
<td>50 (210)</td>
<td>1000 (840)</td>
<td>1050</td>
</tr>
<tr>
<td>Sum(col.)</td>
<td>300</td>
<td>1200</td>
<td>1500</td>
</tr>
</tbody>
</table>

\[
\chi^2 = \frac{(250 - 90)^2}{90} + \frac{(50 - 210)^2}{210} + \frac{(200 - 360)^2}{360} + \frac{(1000 - 840)^2}{840} = 507.93
\]

- Smaller values are better!
Quality of Features: Mutual Information

• In general, the Mutual Information (MI) between two variables $x$ and $y$ measures how much knowing one of these variables reduces uncertainty about the other.

• In our case, it measures how much information a feature contributes to making the correct classification decision, i.e., $x$ is the dimension $d_i$ we want to evaluate and $y$ is the class variable $C$.

• MI is based on probability distributions:
  • $p(x)$ and $p(y)$ are the marginal probability distributions of $x$ and $y$, respectively.
  • $p(x, y)$ is the joint probability distribution function.
Quality of Features: Mutual Information

• Discrete case

\[ MI(x, y) = \sum_{x_i \in x} \sum_{y_i \in y} p(x_i, y_i) \cdot \log \frac{p(x_i, y_i)}{p(x_i)p(y_i)} \]

• Continuous case

\[ MI(x, y) = \int_x \int_y p(x, y) \cdot \log \frac{p(x, y)}{p(x)p(y)} \, dx \, dy \]

• Interpretation: if \( x \) and \( y \) are statistically independent, then

- \( p(x, y) = p(x) \cdot p(y) \) and, thus, \( \log(1) = 0 \)
- Or in other words: knowing \( x \) does not reveal anything about \( y \)
Forward Selection and Feature Ranking: Discussion

**Advantages**

- Efficiency: it compares each feature \( \{d_1, d_2, \ldots, d_n\} \) separately to the class attribute \( C \) (and takes the best \( k \)) instead of testing \( \binom{n}{k} \) subspaces
- Works already for rather small sample sizes

**Limitations**

- Independency assumption: Classes and features 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 with each other
Kapitel 3: Feature Selection

1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection
   3.1 Forward Selection and Feature Ranking
   3.2 Backward Elimination and Random Subspace Selection
   3.3 Subspace Projections

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data
Backward Elimination: General Idea

General Approach

- Start with the complete feature space and delete redundant features
- 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 target dimensionality, terminate else start backward elimination on $R^*$.

Remarks

- Useful in supervised and unsupervised setting (in the latter scenario, $q(R)$ measures structural characteristics)
- Greedy search if there is no monotonicity on $q(R)$; for monotonous measures, branch and bound can be employed
Supervised Quality Measure: Distance-based

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

- Quality criterion: For each object $o$ from the data set $S$, compute distance to the closest object having the same class $\text{NN}_{c_i=C(o)}(o)$ (within-class nearest neighbor distance) in subspace $R$, and to the closest object belonging to another class $\text{NN}_{c_j\neq C(o)}(o)$ (between-classes nearest neighbor distance), where $C(o)$ denotes the class label of object $o$ in subspace $R$:

$$q(R) = \frac{1}{S} \cdot \sum_{o \in S} \frac{\text{NN}_{c_j\neq C(o)}(o)}{\text{NN}_{c_i=C(o)}(o)}$$

- Remark: $q(R)$ is not monotonous: by deleting a dimension, the quality can increase or decrease.
Supervised Quality Measure: Model-based

• Idea: Directly employ the data mining algorithm to evaluate the subspace, e.g. by training a Naive Bayes classifier

• 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 Elimination: Discussion

Advantages

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

Limitations

• Tests w.r.t. subspace quality usually requires much more effort
• All solutions employ heuristic greedy search which do not necessarily find the optimal feature space
Branch and Bound: General Idea

General Approach

• Given: A classification task over the feature space $F$
• Aim: Select the $k$ best dimensions to learn the classifier
• Backward elimination approach “Branch and Bound” is guaranteed to find the optimal feature subset under the monotonicity assumption
• The monotonicity assumption states that for two feature subsets $X, Y \in F$ and a feature selection criterion $J$, if $X \subset Y$ then
  • $J(X) \leq J(Y)$ if $J$ is maximized
  • $J(X) \geq J(Y)$ if $J$ is minimized
• Branch and Bound starts from the full set $F$ and removes features using a depth-first strategy
• Nodes whose objective function are smaller (greater) than the current best are not explored since the monotonicity assumption ensures that their children will not contain a better solution
Example: Original dimensionality 4, \(\langle A, B, C, D \rangle\). Target dimensionality \(d = 1\).

- selected feature
- removed feature

\((\text{All})=1.0\)

A  B  C  D

- **Selected feature**
- **Removed feature**

Diagram:
- $J(\text{BCD}) = 1.0$
- $J(\text{ACD}) = 0.715$
- $J(\text{ABD}) = 0.421$
- $J(\text{ABC}) = 0.603$
- $(\text{All}) = 1.0$

- blue: selected feature
- gray: removed feature

(All) = 1.0

$J(BCD) = 1.0$

$J(ACD) = 0.715$

$J(ABD) = 0.421$

$J(ABC) = 0.603$

$J(CD) = 0.815$

$J(BD) = 0.5$

$J(BC) = 0.5$

- Selected feature
- Removed feature

Example:

- \( J(BCD) = 1.0 \)
- \( J(ACD) = 0.715 \)
- \( J(ABD) = 0.421 \)
- \( J(ABC) = 0.603 \)

- \( J(CD) = 0.815 \)
- \( J(BD) = 0.5 \)
- \( J(BC) = 0.5 \)
- \( J(D) = 0.62 \)
- \( J(C) = 0.43 \)

\( \text{aktBound} = 0.62 \)

- **selected feature**
- **removed feature**

```
Example:
A
B
C
D

J(BCD)=1.0
J(ACD)=0.715
J(ABD)=0.421
J(ABC)=0.603

J(CD)=0.815
J(BD)=0.5
J(BC)=0.5
J(D)=0.62
J(C)=0.43

aktBound = 0.62
```

Prune with aktBound = 0.62
Example: Original dimensionality 4, <A,B,C,D>. Target dimensionality \( d = 1 \).

- **Selected feature**: blue circle
- **Removed feature**: gray circle

```
(All)=1.0

J(BCD)=1.0

J(ACD)=0.715

J(ABD)=0.421

J(ABC)=0.603

J(CD)=0.815

J(BD)=0.5

J(BC)=0.5

J(AD)=0.5

J(AC)=0.32

J(D)=0.62

J(C)=0.43

aktBound = 0.62
```
Subspace Inconsistency (IC)

- Given a data set $S$ (works best for categorical data)
- Idea: Having identical vectors $u, v$ ($u_i = v_i, 1 \leq i \leq d$) in subspace $R$ but the class labels are different ($C(u) \neq C(v)$), this subspace displays an inconsistent labeling
- Measuring the inconsistency of a subspace $R$
  - $X_R(u)$: Amount of all identical vectors $u$ in $R$
  - $X_R^c(u)$: Amount of all identical vectors $u$ in $R$ having class label $c \in C$
  - Inconsistency of $u$ in $R$: $IC_R(u) = X_R(u) - \max_{c \in C} X_R^c(u)$

Then, inconsistency of subspace $R$ is

$$IC(R) = \frac{\sum_{u \in S} IC_R(u)}{|S|}$$

- Monotonicity: $R_1 \subset R_2 \Rightarrow IC(R_1) \geq IC(R_2)$
Backward Elimination: Discussion

**Advantages**

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

**Limitations**

- Useless without groups of identical features (real-valued vectors)
- Worse-case runtime complexity remains exponential in the number of features $d$
Random Subspace Selection: General Idea

General Approach

• Idea: Select $n$ random subspaces having the target dimensionality $k$ out of the $\binom{d}{k}$ many possible subspaces and evaluate each of them.
• Needs quality measures for complete subspaces.
• Trade-off between quality and effort depends on $n$.
• Good alternative to forward selection if quality measure is not monotonic.

• Different randomization approaches exist (see next subsection):
  • Genetic algorithms
  • $k$-medoids feature clustering
  • ...

1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection
   3.1 Forward Selection and Feature Ranking
   3.2 Backward Elimination and Random Subspace Selection
   3.3 Subspace Projections

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data
Genetic Algorithms: General Idea

General Approach

- Idea: Randomized search through genetic algorithms
- Genetic Algorithms encode individual states in the search space as bit-strings
- Population (of current solutions) is a subset of all possible \( k \)-dimensional subspaces
- Fitness function: quality measure for a subspace
- Algorithmic schema to find the best solution in the search space by mixing/changing the population in each iteration (stops e.g. if the best solution of the current population is less fit than the best solution in the previous population)
- Each iteration manages a specific population from which the next population is obtained
Genetic Algorithms: Population Generation

- Operators on the population (\(k\)-dim subspaces) to create candidates for the next population:
  - Mutation: dimension \(d_i\) in subspace \(R\) is replaced by dimension \(d_j\) with a likelihood of \(x\%\)
  - Crossover: combine two subspaces \(R_1\) and \(R_2\), i.e., unite the features sets of \(R_1\) and \(R_2\) and delete random dimensions until dimensionality is \(k\) again
- Selection for next population: 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 into the next generation with a probability of \(u\%\)
- Remark: Many variants on the basic algorithmic schema, e.g. different operations, efficient convergence by “Simulated Annealing” (likelihood of free tickets decreases with the iterations), ...
Advantages

- Can escape from local optima during the search
- Often good approximations of the optimal solutions

Limitations

- Runtime (is not bounded (in the original schema)
- Configuration depends on many parameters which have to be tuned to achieve good quality results in efficient time
Feature Clustering: General Idea

General Approach

- Given: A feature space $F$ and an unsupervised data mining task
- Target: Reduce $F$ to a subspace of $k$ (original) dimensions 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)
- Problem: often many more samples than features so transposed data matrix has many more features than samples
Feature Clustering: Example

- Typical example: item-based collaborative filtering
- E.g. features 3 and 4 are similar over all persons so they could be “merged” to one feature

<table>
<thead>
<tr>
<th></th>
<th>1 (Titanic)</th>
<th>2 (Braveheart)</th>
<th>3 (Matrix)</th>
<th>4 (Inception)</th>
<th>5 (Hobbit)</th>
<th>6 (300)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Susan</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Bill</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Jenny</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Tim</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Thomas</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
Feature Clustering: Example

• Work around for the “many features” problem: specialized feature similarity measures, e.g.
  • Cosine similarity
  • Pearson correlation

• Algorithmic schema
  • 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, e.g. approximate clustering methods for performance reasons
Feature Clustering: Discussion

Advantages

• Depending on the clustering algorithm quite efficient
• Unsupervised method

Limitations

• Results are usually not deterministic (partitioning clustering results depend on initialization)
• Representatives are usually unstable for different clustering methods and parameters
• Method captures pairwise correlations and dependencies among features but multiple dependencies are not considered
Summary: Feature Selection

- Forward-Selection examines each dimension separately 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 (monotonicity required!) based on inconsistency
- $k$-dimensional Projections directly search in the set of $k$-dimensional subspaces for the best suited
  - Genetic algorithms (any quality measures possible, e.g. those from backward elimination)
  - Feature clustering based on correlation
Discussion: Feature Selection

- Many algorithms based on different heuristics
- There are two reasons 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 to have a causal connection to the target variable, but both might depend on the same mechanisms in the data space (hidden variables)
- Different indicators to consider in the comparison of before and after selection performance, e.g., model performance, time, dimensionality, ...


• A. Blum and P. Langley: Selection of Relevant Features and Examples in Machine Learning, Artificial Intelligence (97), 1997.


1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning
   4.1 Reference Point Embedding
   4.2 Principle Component Analysis (PCA)
   4.3 Singular Value Decomposition (SVD)
   4.4 Kernel PCA
   4.5 Further Measures
5. Clustering High Dimensional Data
• Idea: Instead of removing features, try to find a low dimensional feature space generating the original space as accurate as possible:
  • Redundant features are summarized
  • Irrelevant features are weighted by small values or are “erased” (in the best case of course, the new feature space should contain no irrelevant features anymore)

• Some sample methods (among lots of others):
  • Reference point embedding
  • Principal component analysis (PCA)
  • Singular value decomposition (SVD)
  • Fischer-Faces (FF) and Relevant Component Analysis (RCA)
  • Large Margin Nearest Neighbor (LMNN)
• **Goal**: Describe data with fewer features (reduce number of columns)
• Be clear: (like in feature selection) there will always be an information loss

![Diagram showing feature reduction](image)

• There are supervised and unsupervised methods
1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning
   4.1 Reference Point Embedding
   4.2 Principle Component Analysis (PCA)
   4.3 Singular Value Decomposition (SVD)
   4.4 Kernel PCA
   4.5 Further Measures

5. Clustering High Dimensional Data
General Approach

• Idea: Describe the position of each object by their distances to a set of reference points

• Given: Vector space $F = D_1 \times ... \times D_n$ where $D = \{D_1, ..., D_n\}$

• Target: A $k$-dimensional space $R$ which yields optimal solutions for a given data mining task

• Method: For each reference point $R = \{r_1, ..., r_k\}$ and a distance measure $dist$, transform vector $x \in F$ as follows:

$$ r_R(x) = \begin{pmatrix} dist(r_1, x) \\ \vdots \\ dist(r_k, x) \end{pmatrix} $$
Diskussion

- Distance measure is usually determined by the application
- Selection of reference points can be important (use centroids of the classes or cluster-centroids, points on the margin of the data space, use random samples, ...)

Advantages

- Simple approach which is easy to implement
- The transformed vectors yields lower and upper bounds of the exact distances (What the hell is that good for???)

Disadvantages

- Even using $d$ reference points does not reproduce a $d$-dimensional feature space
- Selecting good reference points is important but very difficult
1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning
   4.1 Reference Point Embedding
   4.2 Principle Component Analysis (PCA)
   4.3 Singular Value Decomposition (SVD)
   4.4 Kernel PCA
   4.5 Further Measures

5. Clustering High Dimensional Data
Motivation

- Consider the grades of students in Physics and Statistics
- If we want to compare among the students, which grade should be more discriminative? Statistics or Physics?

Answer: Physics because the variation along that axis is larger

Source: [http://astrostatistics.psu.edu/su09/lecturenotes/pca.html](http://astrostatistics.psu.edu/su09/lecturenotes/pca.html)
Introduction

Motivation

• Suppose now the plot looks as below
• What is the best way to compare students now?

Answer:
We should take a linear combination of the two grades (that represents the direction of highest variance) to get the best results

Source: http://astrostatistics.psu.edu/su09/lecturenotes/pca.html
Motivation

- PCA returns two principal components
- The first gives the direction of the maximum spread of the data.
- The second gives the direction of maximum spread perpendicular to the first

Source: http://astrostatistics.psu.edu/su09/lecturenotes/pca.html
A feature $X$ can be normalized by subtracting its values with the mean $\bar{X}$ and dividing by the standard deviation $s_X$, e.g. $\tilde{X} = \frac{X - \bar{X}}{s_X}$.

**Example:**

Consider the following body heights measured in different units:

<table>
<thead>
<tr>
<th></th>
<th>Person A</th>
<th>Person B</th>
<th>Person C</th>
<th>mean</th>
<th>sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>body height (cm)</td>
<td>180.00</td>
<td>172.00</td>
<td>175.00</td>
<td>175.67</td>
<td>4.04</td>
</tr>
<tr>
<td>body height (m)</td>
<td>1.80</td>
<td>1.72</td>
<td>1.75</td>
<td>1.76</td>
<td>0.04</td>
</tr>
<tr>
<td>body height (feet)</td>
<td>5.91</td>
<td>5.64</td>
<td>5.74</td>
<td>5.76</td>
<td>0.13</td>
</tr>
</tbody>
</table>

After normalizing, we always obtain the normalized body height (no matter which unit we used):

<table>
<thead>
<tr>
<th></th>
<th>Person A</th>
<th>Person B</th>
<th>Person C</th>
<th>mean</th>
<th>sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>normalized body height</td>
<td>1.07</td>
<td>-0.91</td>
<td>-0.16</td>
<td>0.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Normalizing all features in a data set, can have several advantages:

- It puts all features into *comparable* units, i.e., we make sure that all normalized features have mean 0 and standard deviation of 1.
- It can avoid numerical instabilities in several algorithms, e.g. if a feature has very low / high values.
- It helps in computing meaningful *distances* between observations.
- Finally, if we want to find directions of highest variances, it might be better to do this on normalized data.
• Sure, there are many ways to do normalization
• here, we will use the notion common in Statistics, where the
  **variance** of a normalized feature is always 1, its mean is always 0
• The **covariance** of two normalized features $\tilde{X} = \frac{X - \bar{X}}{s_X}$ and $\tilde{Y} = \frac{Y - \bar{Y}}{s_Y}$ is the same as the **correlation** of the non-normalized features $X$ and $Y$.
• One can proof this with the help of

\[
s_{\tilde{X}\tilde{Y}} = \frac{1}{n-1} \sum_{i=1}^{n} (\tilde{x}_i - \bar{\tilde{x}})(\tilde{y}_i - \bar{\tilde{y}}) = \ldots = \frac{1}{n-1} \sum_{i=1}^{n} \frac{(x_i - \bar{x})}{s_X} \frac{(y_i - \bar{y})}{s_Y} = r_{XY}.
\]
**Example I:**

- Feature $x_1$ explains most of the variation
- Feature $x_2$ has a lower variance than $x_1$
- If we disregard $x_2$ and project the points into the 1-dimensional space of $x_1$, we do not lose much information w.r.t. variability
Example II:

- $x_1$ and $x_2$ are correlated and have similar variances.
- Find a new orthogonal axes (e.g. PC1 and PC2), where PC1 explains most of the variation
- Rotate the points and consider PC1 and PC2 as new coordinate system (situation as in the previous example)
- We can now project points onto PC1 and disregard PC2 (hopefully without losing much information)
PCA Intuition

• PCA finds the optimal rotation such that the transformed data explains the variability of the data best

• The new axis are the principal components (also called “eigenvectors” because PCA is technically an Eigen-Decomposition); for a $d$-dimensional data set we always get $d$ principal components

• The variance along each eigenvector (called “eigenvalue”) is decreasing, i.e. the first eigenvector has the highest eigenvalue, while the $d$-th eigenvector has the smallest eigenvalue

• This can be used for dimensionality reduction: if we pick the $k$-th first eigenvectors as new axes and transform the $d$-dimensional data into the new $k$-dimensional space, this transformation is optimal w.r.t. loss of total variance
General procedure

1. Rotate the original $p$-dimensional coordinate system until the first PC that explains most of the variation is found.

2. Fix the first PC and proceed with rotating the remaining $p - 1$ coordinates until the second PC (which is orthogonal to the first PC) is found that explains most of the *remaining* variation, etc.

3. We can reduce the dimensions by projecting the points onto the first, say $k < p$, PC.
PCA Intuition: Find first PC
Variance of projected points: 0.87
Variance of projected points: 0.25

- rotated coordinate system
- original coordinate system
- projected points
PCA Intuition: Animation

Variance of projected points: 0.08

rotated coordinate system
original coordinate system
projected points
PCA Intuition: Animation

Variance of projected points: 0.38

- rotated coordinate system
- original coordinate system
- projected points

---

Prof. Dr. Peer Kröger: KDD2 (SoSe 2019) — Lecture 2 – High Dimensional Data — 4. Feature Reduction and Metric Learning
Variance of projected points: 0.84

- rotated coordinate system
- original coordinate system
- projected points
PCA Intuition: Animation

Variance of projected points: 1.31

rotated coordinate system
original coordinate system
projected points
Variance of projected points: 1.63

rotated coordinate system
original coordinate system
projected points
PCA Intuition: Reduce dimensionality

Rotate the points and use PC1 and PC2 as new coordinate system.

Here, the PC1 axis explains most of the variance:
Dimensionality can be reduced by projecting the points onto the PC1 (and by disregarding PC2). The hope is that we won’t lose much information this way.
Idea: Transform an original set of correlated metric features to a new set of uncorrelated (orthogonal) metric features, called principal components (PC), that explain the variability in the data.

- The objective is to investigate if only a few PC account for most of the variability in the original data.
- If the objective is fulfilled, we can use fewer PCs to reduce the dimensionality.
- The PCs remove collinearity of the input variables as they are orthogonal to each other.
PCA Intuition: Final Remarks

• PCA is used for dimensionality reduction by disregarding dimensions with lower variability.

• There is always an information loss, especially for other criteria.

• **Attention:** dimensionality reduction can worsen the classification accuracy when the task is to classify two groups:
Aim: Find a new set of features (PC scores, eigenvectors) $\mathbf{pc}_1, \ldots, \mathbf{pc}_p$ based on the original data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_p]$ so that

- each PC score $\mathbf{pc}_1, \ldots, \mathbf{pc}_p$ is a linear combination of the original metric features with coefficient weights (so-called loading vectors) $a_1, \ldots, a_p$, i.e.

$$\mathbf{pc}_j = a_{j1}\mathbf{x}_1 + a_{j2}\mathbf{x}_2 + \ldots + a_{jp}\mathbf{x}_p = \mathbf{X}a_j.$$ 

- the set is mutually uncorrelated: $\text{Cov}(\mathbf{pc}_j, \mathbf{pc}_k) = 0$, $\forall j \neq k$.

- the variances (eigenvalues) of the PC scores decrease:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_p,$$

where $\lambda_k := \text{Var}(\mathbf{pc}_k)$. 


We look for the loading vector $\mathbf{a}_1 = (a_{11}, a_{21}, \ldots, a_{p1})^\top$ that maximizes the variance of $\mathbf{pc}_1$:

$$\max_{\mathbf{a}_1} \text{Var}(\mathbf{pc}_1) = \text{Var}(\mathbf{Xa}_1) = \mathbf{a}_1^\top \Sigma \mathbf{a}_1$$

subject to the normalization constraint $\mathbf{a}_1^\top \mathbf{a}_1 = \sum_{k=1}^{p} a_{k1}^2 = 1$.

The constraint is required for identifiability reasons, otherwise we could maximize the variance by just increasing the values in $\mathbf{a}_1$.

Repeat this maximization step for the other PCs and additionally use the orthogonality constraint, i.e. for the second PC:

$$\mathbf{a}_2^\top \mathbf{a}_1 = 0.$$
The heptathlon data set (e.g. available in the R package HSAUR3) contains the competition results of 25 athletes in 7 disciplines for the Olympics held in Seoul in 1988.

- **Aim**: Rank the athletes according to their overall performance in all 7 disciplines.
- **Idea**: Use PCA to reduce the dimensionality (i.e., reduce the results of the 7 disciplines to one dimension) and compare the scores of the first PC with the official scores.
Example: The Olympic Heptathlon Data

Features of the heptathlon data:

- **hurdles**: results 100m hurdles (in seconds).
- **highjump**: results high jump (in m).
- **shot**: results shot putt (in m).
- **run200m**: results 200m race (in seconds).
- **longjump**: results long jump (in m).
- **javelin**: results javelin (in m).
- **run800m**: results 800m race (in seconds).
- **score**: total score of the official scoring system.
Example: The Olympic Heptathlon Data

The features hurdles, run200m and run800m are time measurements, i.e. low values are better. For all other features high values are better.

Results of the best and worst participant:

<table>
<thead>
<tr>
<th></th>
<th>hurdles</th>
<th>highjump</th>
<th>shot</th>
<th>run200m</th>
<th>longjump</th>
<th>javelin</th>
<th>run800m</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joyner-Kersee (USA)</td>
<td>12.7</td>
<td>1.86</td>
<td>15.8</td>
<td>22.6</td>
<td>7.27</td>
<td>45.7</td>
<td>129</td>
<td>7291</td>
</tr>
<tr>
<td>Launa (PNG)</td>
<td>16.4</td>
<td>1.50</td>
<td>11.8</td>
<td>26.2</td>
<td>4.88</td>
<td>46.4</td>
<td>163</td>
<td>4566</td>
</tr>
</tbody>
</table>

We use negative time measurements so that higher values are better and therefore all features have the same direction:

<table>
<thead>
<tr>
<th></th>
<th>hurdles</th>
<th>highjump</th>
<th>shot</th>
<th>run200m</th>
<th>longjump</th>
<th>javelin</th>
<th>run800m</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joyner-Kersee (USA)</td>
<td>-12.7</td>
<td>1.86</td>
<td>15.8</td>
<td>-22.6</td>
<td>7.27</td>
<td>45.7</td>
<td>-129</td>
<td>7291</td>
</tr>
<tr>
<td>Launa (PNG)</td>
<td>-16.4</td>
<td>1.50</td>
<td>11.8</td>
<td>-26.2</td>
<td>4.88</td>
<td>46.4</td>
<td>-163</td>
<td>4566</td>
</tr>
</tbody>
</table>
Scatter Plot Matrix

Prof. Dr. Peer Kröger: KDD2 (SoSe 2019) — Lecture 2 – High Dimensional Data — 4. Feature Reduction and Metric Learning
Perform PCA

- If features are on very different scales, PCA should be carried out on the correlation matrix (which is equivalent to the correlation matrix if normalized features are used).

- As the features of the heptathlon data are on different scales, we perform the PCA based on the correlation matrix.

- Alternatively, we could also perform the PCA based on the covariance matrix but on the normalized heptathlon data.

- The result contains:
  - The loadings $a_1, \ldots, a_p$,
  - The PC scores $pc_1, \ldots, pc_p$ and
  - The variance $\lambda_1, \ldots, \lambda_p$ (or standard deviation) of the PC scores.
The total variance of the $p$ PC scores is equal the total variance of the original features, i.e.,

$$\sum_{j=1}^{p} \lambda_j = s_1^2 + s_2^2 + \cdots + s_p^2,$$

where $\lambda_j$ is the variance of the $j$th PC and $s_j^2$ is the sample variance of variable $x_j$.

The proportion of explained variance of the $j$-th PC is

$$\frac{\lambda_j}{\sum_{j=1}^{p} \lambda_j}.$$

The first $k$ PCs account for a proportion

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{p} \lambda_j}.$$
Choosing the Number of PCs

Two simple rules of thumb for choosing the number of PCs:

1. Retain the first $k$ components, which explain a large proportion of the total variation, e.g., 80-90%.

2. Use a scree plot: Plot the component variances vs. the component number and look for an *elbow*. For components after the *elbow*, the variance decreases more slowly.
The first PC explains 63.72% of the variation of the heptathlon, the loadings of the first PC are:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Loading</th>
</tr>
</thead>
<tbody>
<tr>
<td>hurdles</td>
<td>0.4529</td>
</tr>
<tr>
<td>highjump</td>
<td>0.3772</td>
</tr>
<tr>
<td>shot</td>
<td>0.3631</td>
</tr>
<tr>
<td>run200m</td>
<td>0.4079</td>
</tr>
<tr>
<td>longjump</td>
<td>0.4562</td>
</tr>
<tr>
<td>javelin</td>
<td>0.0754</td>
</tr>
<tr>
<td>run800m</td>
<td>0.3750</td>
</tr>
</tbody>
</table>

Dimensionality reduction:

- Project all 8 features onto the first PC.
- Compare the scores of the first PC with the official scores used to rank the athletes.
The scores of the first PC $\text{pc}_1$ have a similar ranking as the scores of the official scoring system:
Discussion

Advantage

• Considers arbitrary correlations between features
• Selected subspace is optimal w.r.t. loss of variance

Disadvantage

• Assumption: components with high variance are useful to discover the desired patterns
• Considers only linear correlations (work-around: Kernel-PCA, see later)
Kapitel 4: Feature Reduction and Metric Learning

1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning
   4.1 Reference Point Embedding
   4.2 Principle Component Analysis (PCA)
   4.3 Singular Value Decomposition (SVD)
   4.4 Kernel PCA
   4.5 Further Measures

5. Clustering High Dimensional Data
Idea

• PCA is an eigenvalue decomposition of the $d \times d$ covariance matrix \( \Sigma = D^T D \) of the (normalized) data matrix \( D \):

\[
\Sigma = VEV^T
\]

such that

• \( V = (pc_1, ..., pc_d) \), is a $d \times d$ matrix whose columns are the pairwise independent unit vectors, the eigenvectors

\[
V = \begin{pmatrix}
\lambda_1 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \lambda_d
\end{pmatrix}
\]

• \( E = \begin{pmatrix}
\lambda_1 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \lambda_d
\end{pmatrix} \) is a $d \times d$ diagonal matrix, the diagonal elements are the eigenvalues of the corresponding eigenvectors

• The decomposition can be found e.g. based on numerical algorithms
Compute the SVD

- SVD is a generalization of the eigenvalue decomposition
- Let $D$ be the $n \times d$ data matrix ($n$ objects, $d$ dimensions) and let $k$ be its rank (max number of independent rows/ columns)
- We can decompose $D$ into matrices $O, S, A$ with $D = OSA^T$ or

\[
\begin{pmatrix}
  x_{1,1} & \ldots & x_{1,d} \\
  \vdots & \ddots & \vdots \\
  x_{n,1} & \ldots & x_{n,d}
\end{pmatrix}
= \begin{pmatrix}
  o_{1,1} & \ldots & o_{1,k} \\
  \vdots & \ddots & \vdots \\
  o_{n,1} & \ldots & o_{n,k}
\end{pmatrix}
\cdot \begin{pmatrix}
  \lambda_1 & \ldots & 0 \\
  \vdots & \ddots & \vdots \\
  0 & \ldots & \lambda_k
\end{pmatrix}
\cdot \begin{pmatrix}
  a_{1,1} & \ldots & a_{1,d} \\
  \vdots & \ddots & \vdots \\
  a_{k,1} & \ldots & a_{k,d}
\end{pmatrix}
\]

such that
- $O$ is a $n \times k$ column-orthonormal matrix (each of its columns is a unit vector and the dot product of any two columns is 0)
- $S$ is a diagonal $k \times k$ matrix
- $A$ is a $k \times d$ column-orthonormal matrix. Note that we always use $A$ in its transposed form, so it is the rows of $A^T$ that are orthonormal
SVD: Example I

- $D$ contains movie ratings by users
  - The corresponding SVD shows two concepts “science fiction” and “romance”
- $S$ shows the strength of these concepts
- $A$ relates movies to concepts

$$D = \begin{pmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 0 & 0 & 2 & 2 \\
\end{pmatrix} \quad \begin{pmatrix}
\cdot14 & 0 \\
\cdot42 & 0 \\
\cdot56 & 0 \\
\cdot70 & 0 \\
0 & \cdot60 \\
0 & \cdot75 \\
0 & \cdot30 \\
\end{pmatrix} = \begin{pmatrix}
12.4 & 0 \\
0 & 9.5 \\
\end{pmatrix} \begin{pmatrix}
\cdot58 & \cdot58 & \cdot58 & 0 & 0 \\
0 & 0 & 0 & \cdot71 & \cdot71 \\
\end{pmatrix}$$

(Source: http://infolab.stanford.edu/~ullman/mmds/ch11.pdf)
SVD: Example II

- Now a slightly different $D$
  - The corresponding SVD shows three concepts “science fiction” and “romance” and ???

\[
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{pmatrix}
= 
\begin{pmatrix}
.13 & .02 & -.01 \\
.41 & .07 & -.03 \\
.55 & -.09 & -.04 \\
.68 & .11 & -.05 \\
.15 & -.59 & .65 \\
.07 & -.73 & -.67 \\
.07 & -.29 & .32
\end{pmatrix}
\cdot
\begin{pmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3
\end{pmatrix}
\cdot
\begin{pmatrix}
.56 & .59 & .56 & .09 & .09 \\
.12 & -.02 & .12 & -.69 & -.69 \\
.40 & -.80 & .40 & .09 & .09
\end{pmatrix}
\]

• To reduce dimensionality, we can set the smallest singular values to 0 in $S$ and eliminate the corresponding columns in $O$ and rows in $A^T$ (check previous examples)

• How Many Singular Values Should We Retain?
  • Rule of thumb: retain enough singular values to make up 90% of the energy in $S$
  • Energy is defined in terms of the singular values (matrix $S$)
  • In the previous example, the total energy is:
    
    $$(12.4)^2 + (9.5)^2 + (1.3)^2 = 245.70$$

  • The retained energy is: $(12.4)^2 + (9.5)^2 = 244.01 > 99\%$
Connection between SVD and PCA

• PCA is applying SVD on the covariance matrix $\Sigma = D^T D$
• SVD means: $D = OSA^T$
• Thus:
  $$\Sigma = D^T D = (OSA^T)^T OSA^T = AS^T (O^T O) SA^T$$
• Since $O$ is an orthonormal matrix, $O^T O$ is the identity:
  $$AS^T (O^T O) SA^T = A(S^T S) A^T$$
• $S$ is a diagonal matrix, so transposing has no effect:
  $$A(S^T S) A^T = AS^2 A^T = A \begin{pmatrix} \lambda_1^2 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \lambda_k^2 \end{pmatrix} A^T$$
Connection between SVD and PCA

• Here: $A$ is a matrix of eigenvectors
• Eigenvalues of the covariance matrix = squared singular values of $D$
• Conclusion: Eigenvalues and eigenvectors of the covariance matrix $S$ can be determined by the SVD of the data matrix $D$ (or in other words: SVD is a method to perform PCA)
• SVD is sometimes a better way to perform PCA (Large dimensionalities e.g., text data)
• SVD can cope with dependent dimensions ($k < d$ is an ordinary case in SVD)
Kapitel 4: Feature Reduction and Metric Learning

1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning
   4.1 Reference Point Embedding
   4.2 Principle Component Analysis (PCA)
   4.3 Singular Value Decomposition (SVD)
   4.4 Kernel PCA
   4.5 Further Measures

5. Clustering High Dimensional Data
Consider the following scenarios:

- PCA will be effective since data is linearly correlated
- PCA may find the orange line as the first component
Basic Idea

Recall: the solution of linear classifiers (e.g. SVMs) for non-linear problems is “make them linear!” using a suitable feature mapping

- No linear separation of classes possible
- Mapping $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ with $(x_1, x_2) \rightarrow (x_1, x_2, x_1^2 + x_2^2)$
Kernel Trick

• Since a high-dimensional mapping can still have negative impact, the Kernel trick is used whenever possible (see KDD I lecture)
• Given the intended mapping $\Phi$, the Kernel is usually defined as $K(x, y) = \Phi(x)^T \Phi(y)$
• Example: Degree-$d$ polynomials: $K(x, y) = (x^T y + c)^d$ with an arbitrary constant $c$, e.g. for $d = 2$:

\[
\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3
\]
\[
(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)
\]

(Image source: http://i.stack.imgur.com/qZV3s.png)
Kernel PCA Using SVD

- Recall the SVD $D = OSA^T$
- $A$ is a $k$-dimensional basis of the eigenvectors of $DD^T$ (originally $d \times d$)
- Analogously, $O$ is a $k$-dimensional basis of eigenvectors of $DD^T$
- $DD^T$ is a Kernel matrix for the linear Kernel (i.e., no mapping made - cf. KDD I) or any other Kernel
- $A$ and $O$ are related as follows:

$$D = OSA^T \Rightarrow O^TD = O^TOSA^T = SA^T \Rightarrow S^{-1}O^TD = A^T$$

i.e. each $d$-dimensional eigenvector in $A$ is a linear combination of vectors in $D$ (original or mapped!) and the $nk$ $k$-dimensional eigenvectors in $O^T$ ($O$ is $n \times k$)
Kernel PCA Using SVD

- Let $K(x, y) = \Phi(x)^T \Phi(y)$ be a kernel for the non-linear transformation $\Phi$
- Assume: $K(x, y)$ is known, but $\Phi(x)$ is not explicitly given
- Let $K$ be the Kernel matrix of $D$ w.r.t. $K(x, y)$, i.e.

$$K = \begin{pmatrix}
K(x_1, x_1) & \cdots & K(x_1, x_n) \\
\vdots & \ddots & \vdots \\
K(x_n, x_1) & \cdots & K(x_n, x_n)
\end{pmatrix}$$

- The eigenvalue decomposition of $K$ is $K = VSV^T$ where $V$ is a $n$-dimensional basis from eigenvectors of $K$
- Dimensionality Reduction through mapping of $y \in D$ w.r.t $V$ to

$$\hat{y} = \begin{pmatrix}
\Phi(y)^T (\sum_{i=1}^{n} v_{i,1} \Phi(x_i)) \\
\vdots \\
\Phi(y)^T (\sum_{i=1}^{n} v_{i,k} \Phi(x_i))
\end{pmatrix} = \begin{pmatrix}
\sum_{i=1}^{n} v_{i,1} (\Phi(y)^T \Phi(x_i)) \\
\vdots \\
\sum_{i=1}^{n} v_{i,k} (\Phi(y)^T \Phi(x_i))
\end{pmatrix} = \begin{pmatrix}
\sum_{i=1}^{n} v_{i,1} K(y, x_i) \\
\vdots \\
\sum_{i=1}^{n} v_{i,k} K(y, x_i)
\end{pmatrix}$$
Matrix Factorization as Optimization Task

- BTW, SVD (and, thus PCA) is a matrix decomposition that can be formalized as optimization task

\[ D = \text{OSA}^T = \left( \begin{array}{cccccc|ccc} \sqrt{\lambda_1} & \cdots & 0 & | & | & | & | & | & | \\ \vdots & \ddots & \vdots & | & | & | & | & | & | \\ 0 & \cdots & \sqrt{\lambda_k} & | & | & | & | & | & | \end{array} \right) \left( \begin{array}{ccc|ccc} \sqrt{\lambda_1} & \cdots & 0 & | & | & | \\ \vdots & \ddots & \vdots & | & | & | \\ 0 & \cdots & \sqrt{\lambda_k} & | & | & | \end{array} \right) \text{A}^T = UV^T \]

- As an optimization problem:

\[ L(U, V) = \| D - UV^T \|_f^2 \]

subject to \( \forall i \neq j : \langle v_i, v_j \rangle = 0 \land \langle u_i, u_j \rangle \)

using the squared Frobenius Norm of an \( n \times m \) matrix \( M \):

\[ \| M \|_f^2 = \sum_{i=1}^n \sum_{j=1}^m |m_{i,j}|^2 \]
1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning
   4.1 Reference Point Embedding
   4.2 Principle Component Analysis (PCA)
   4.3 Singular Value Decomposition (SVD)
   4.4 Kernel PCA
   4.5 Further Measures

5. Clustering High Dimensional Data
Fisher Faces

• Idea: Use examples from a training set (supervised!) to increase the discriminative power of the target space

• Minimize the similarity between objects from different classes (between class scatter matrix: \( \sigma_b \))
  Use covariance matrix of the class centroids for \( \Sigma_b \)

• Maximize similarity between objects belonging to the same class (within class scatter matrix \( \Sigma_w \))
  Use average covariance matrix of all classes for \( \Sigma_w \)

• Determine new basis vectors \( b_i \) by maximizing
  \[
  \frac{b_i^T \Sigma_b b_i}{b_i^T \Sigma_w b_i}
  \]
  subject to \( \forall i \neq j : \langle b_i, b_j \rangle = 0 \)
Remarks on Fisher Faces

- The vector having the largest eigenvalue corresponds to the normal vector of the separating hyper plane in linear discriminant analysis or Fisher’s discriminant analysis. (cf. KDD I)
- Fischer Faces are limited due to the assumption of mono-modal classes: each class is assumed to follow one multivariate Gaussian
- Multi-modal or non-Gaussian distributions are not modeled well
- Many variants (e.g. Relevant Component Analysis (RCA), Large Margin Nearest Neighbor (LMNN))
• Linear basis transformation yield a rich framework to optimize feature spaces
• Unsupervised methods delete low variant dimensions (PCA und SVD)
• Kernel PCA allows to compute PCA in non-linear kernel spaces
• Basic assumption: direction of highest variance bear the most relevant information
• Supervised methods try to minimize the within class distances while maximizing between class distances (Fischer Faces and variants)


1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data
   5.1 Challenges
   5.2 Subspace Clustering
   5.3 Projected Clustering
   5.4 Correlation Clustering
1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data

5.1 Challenges

5.2 Subspace Clustering

5.3 Projected Clustering

5.4 Correlation Clustering
Motivating Example

Customer Recommendation / Target Marketing

• Data: customer ratings for given products
  • Rows (objects): customers (millions?)
  • Columns (features): products (hundreds to thousands)
  • Value $x_{ij}$ in the data matrix is the rating of product $i$ by user $j$

• Task: Cluster customers to find groups of persons that share similar preferences or disfavor (e.g. to do personalized target marketing)

• Challenge: customers may be grouped differently according to different preferences/disfavors, i.e. different subsets of products (See: baby shapes game)
Curse of Dimensionality (Revisited)

Relevant and irrelevant attributes

• Not all features, but a subset of the features may be relevant for clustering
• Groups of similar (e.g. “dense”) points may be identified when considering only these features
• Different subsets of attributes may be relevant for different clusters
Effect on clustering

- Traditional distance functions give equal weight to all dimensions
- However, not all dimensions are of equal importance
- Adding irrelevant dimensions ruins any clustering based on a distance function that equally weights all dimensions
Curse of Dimensionality (Revisited)

• Example: different attributes are relevant for different clusters
It can even be a little bit more complex ...

- Task: Cluster test persons to find groups of individuals with similar correlation among the concentrations of metabolites indicating homogeneous metabolic behavior (e.g. disorder)

- Challenge: different metabolic disorders appear through different correlations of (subsets of) metabolites
Curse of Dimensionality (Revisited)

Correlation among attributes

- A subset of features may be correlated
- Groups of similar (e.g. “dense”) points may be identified when considering this correlation of features only
- Different correlations of attributes may be relevant for different clusters
Why not feature selection/reduction

• (Unsupervised) feature selection or feature reduction (e.g. PCA) is global, i.e., it transforms the original feature space into one new representation

• We face a local feature relevance/correlation: some features (or combinations of them) may be relevant for one cluster, but may be irrelevant for a second one, i.e., we need multiple representations
Example: use PCA (target dim = 1) before clustering
Example: cluster first, then find correlations (with PCA)
Problem Summary

- Feature relevance and correlation
  - Usually, no clusters in the full dimensional space
  - Often, clusters are hidden in subspaces of the data, i.e. only a subset of features is relevant for the clustering
  - E.g. a group of genes play a common role in a subset of experimental conditions but may behave completely different in other conditions

- Local feature relevance/correlation
  - For each cluster, a different subset of features or a different correlation of features may be relevant
  - E.g. different genes are responsible for different phenotypes other conditions

- Overlapping clusters (different semantic concepts)
  - Clusters may overlap, i.e. an object may be clustered differently in varying subspaces
  - E.g. a gene plays different functional roles depending on the environment other conditions
General Problem Setting

Search for clusters in (in general arbitrarily oriented) subspaces of the original feature space

Challenges

- Find the correct subspace of each cluster (Search space virtually infinite: all possible arbitrarily oriented subspaces of a feature space)
- Find the correct cluster in each relevant subspace (Search space depends on clustering algorithm)

- Even worse: both challenges depend on each other:
  - In order to determine the correct subspace of a cluster, we need to know (at least some) cluster members
  - In order to determine the correct cluster memberships, we need to know the subspaces of all clusters
General Problem Setting

• Hmm, it is really not so easy, especially in practice ...

• Even if we found the relevant subspace, we still might have a hard time to find the correct cluster(s)

• Here, clusters (yellow, red, green, blue) and noise (gray) are not separable in the “correct” subspace ...

• Rather, we would need a new cluster model (recall: cluster = group of similar objects)

• What is the concept of similarity that all members of a cluster share in this example (and does not include members from other clusters/noise)???
### Subspace Clustering (restricted to axis-parallel subspace)
- Find all clusters in all subspaces (allow overlaps)
- Usually bottom-up subspace search

### Projected Clustering (restricted to axis-parallel subspace)
- Each point is assigned to one subspace cluster or noise
- Usually top-down subspace search

### Correlation Clustering (explores arbitrarily oriented subspaces)
- Each point is assigned to one subspace cluster or noise
- Bottom-up and top-down subspace search approaches
Kapitel 5: Clustering High-dim Data

1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data
   5.1 Challenges
   5.2 Subspace Clustering
   5.3 Projected Clustering
   5.4 Correlation Clustering
Bottom-up Subspace Cluster Search

• Similar to Branch-and-Bound feature selection: Start with 1-D subspaces or subspace clusters and merge them to compute higher dimensional ones
• Most approaches transfer this problem into a frequent item set mining problem
• In this case, the cluster criterion must implement the downward closure (monotonicity) property:
  • If the criterion holds for a $k$-dimensional subspace $S$, then it also holds for any $(k-1)$-dimensional projection of $S$
  • Use the reverse implication for pruning: If the criterion does not hold for a $(k-1)$-dimensional projection of $S$, then the criterion also does not hold for $S$
• Some approaches use other search heuristics (especially if monotonicity does not hold) like best-first-search, greedy-search, ...
Downward-closure Property: Example

- Consider a simple cluster criterion (density of grid cells): If a cell \( C \) of side length \( s \) contains more than \( m \) points, it represents a cluster.

- Monotonicity: if \( C \) contains more than \( m \) points in subspace \( S \) then \( C \) also contains more than \( m \) points in any subspace \( T \subset S \).

Cell \( C \) contains more than \( m=5 \) points in subspace „AB“

\[ \Rightarrow \text{Also in subspaces „A“}\subset „AB“ \text{ and } „B“\subset „AB“ \]

Cell \( C \) contains less than \( m=5 \) points in subspace „A“

\[ \Rightarrow \text{Also in subspace „AB“} \]
CLIQUE: Idea

- Probably the first bottom-up algorithm
- It uses a density-grid-based cluster model (similar to previous slide)

- Clusters are “dense regions” in the feature space
- Partition the feature space into $\xi$ equal sized parts in each dimension
- A unit is the intersection of one interval from each dimension
- Unit $u$ is dense if it contains more than $\tau$ objects

- Clusters are maximal sets of connected dense units (e.g., $A \cup B$)
- Two-step approach (1. subspace search, 2. clustering)
Step 1: Find subspaces with dense units

- Explore Downward Closure property of dense cells (APRIORI-style search)

- Candidate generation
  - Based on $D_{k-1}$, the set of $(k-1)$-dimensional dense units, generate candidate set $C_k$ by self joining $D_{k-1}$
  - Join condition: units share first $k-2$ dimensions
  - Discard those candidates which have a $k-1$ projection not included in $D_{k-1}$
  - For the remaining candidates: check density
Step 2: Find clusters as maximal sets of connected dense units

- Given: a set of dense units $D$ in the same $k$-dimensional subspace $S$
- Output: A partition of $D$ into clusters $D_1, \ldots, D_k$ of connected dense units
- The problem is equivalent to finding connected components in a graph
  - Nodes: dense units
  - Edge between two nodes if the corresponding dense units have a common face (neighboring units)
  - Depth-first search algorithm: Start with a unit $u$ in $D$, assign it to a new cluster ID and find all the units it is connected to
  - Repeat if there are nodes not yet visited
• Input parameters: $\xi$ and $\tau$ specifying the density threshold
• Output: all clusters in all subspaces, clusters may overlap
• Simple but efficient cluster model
• Uses a fixed density threshold for all subspaces (in order to ensure the downward closure property)
  • To represent a cluster, a unit in 10D must contain as many points (or more) as in 2D . . .
  • For a cluster $C$ in subspace $S$, all clusters in all projections of $S$ are also reported as clusters (extremely high redundancy)
  • Which of the redundant information is more interesting? ($D$ in $S$ or $D'$ in $T \subset S$; $D'$ may contain more units/points)
• Worst case runtime?
There are different variations of CLIQUE varying mainly in terms of ...

- ... different density definition, e.g. using Entropy for subspaces rather than simple counts of units
- ... different grid construction methods, e.g. adaptive intervals (but then, no downward closure property is given anymore)

Drawbacks of a grid-based clustering model:

- Positioning of the grid influences the clustering
- Selection of $\xi$ and $\tau$ is very sensitive
- Example: Either $C_2$ and $C_1$ are found as clusters or none of them
Use DBSCAN model (maximal density-connected sets): density is defined w.r.t. the location of points not w.r.t. the data space:

### Core Points

- Points finding more than $MinPts$ other points in its $\epsilon$-neighborhood

### Density connectivity

- Core points may have core points in their $\epsilon$-neighborhood
- Build transitive chains of such core points to find connected sets
• Richer cluster model: detects clusters of arbitrary shapes and locations (in the corresponding subspaces)
• Naive approach: Apply DBSCAN in all possible subspaces (exponential runtime!)
• Idea: Exploit clustering information from previous step (subspaces)
• Density-connected clusters are not monotonic, but density connected sets are (see next slide)
  • If \( C \) is a density connected set in subspace \( S \) then \( C \) is a density connected set in any subspace \( T \subset S \)
  • But, if \( C \) is a cluster in \( S \), it need not to be a cluster in \( T \subset S \) because maximality might be violated, i.e., in \( T \) there may be additional points density connected to the points in \( C \)
  • A cluster in a higher-dimensional subspace \( S \) will be a subset of a cluster in the a projection \( T \)
• Thus, APRIORI-style subspace search is possible
Example (circles indicate $\varepsilon$-neighborhood):

$p$ and $q$ density connected in $\{A, B\}$, thus, they are also density connected in $\{A\}$ and $\{B\}$

$p$ and $q$ not density connected in $\{B\}$, thus, they are not density connected in $\{A, B\}$, although they are density connected in $\{A\}$
• Algorithm
  • All subspaces that contain any density-connected set are computed using the bottom-up approach (similar to CLIQUE/APRIORI)
  • Density-connected clusters are computed using a specialized DBSCAN run in the resulting subspace to generate the subspace clusters
• Discussion
  • Input: $\epsilon$ and $MinPts$ specifying the density threshold
  • Output: all clusters in all subspaces, clusters may overlap
  • Uses a fixed density threshold for all subspaces
  • Advanced but costly cluster model
Bottom-up Variants: FIRES

Uses a different bottom-up heuristic for subspace search:

- Starts with 1-dimensional clusters called base clusters (generated by applying any traditional clustering algorithm to each 1-dimensional subspace)
- Merges these clusters to generate subspace cluster approximations by applying a clustering of the base clusters using a variant of DBSCAN (similarity between two clusters $C_1$ and $C_2$ is defined by $|C_1 \cap C_2|$) in order to “jump” to maximal dimensional subspaces
- Refines the resulting cluster approximations using any traditional clustering algorithm on the points within the approximations
Similar idea to FIRES

- Cluster cores are hyper-rectangular approximations of subspace clusters
- Subspace search is APRIORI-style: cluster cores are computed bottom-up from “significant” 1D intervals
- Significant 1D intervals are determined using a hypothesis test:
  - Hypothesis: no clusters, i.e. points are randomly distributed
  - If an interval contains significantly more points than expected, the hypothesis is rejected (significant 1D interval)
- Significant 1D intervals follow the downward closure property
- Postprocessing: Cluster cores initialize an EM fuzzy clustering of all data points
Find subspace cluster hierarchies (lower dimensional clusters embedded in higher dimensional ones):

- Integrate a proper distance function into hierarchical clustering
- Learns distance function instance-based bottom-up
Summary: Subspace Clustering

Several different variants exist that all suffer from a key limitation: global density thresholds

- In order to ensure the downward closure property, the density threshold must be fixed globally
- Consequence: the points in an e.g. 20-dimensional subspace cluster must be as dense as in an e.g. 2-dimensional cluster
- This is a rather optimistic assumption since the data space grows exponentially with increasing dimensionality (see “curse” discussion)
- Consequences:
  - A strict threshold will most likely produce only lower dimensional clusters
  - A loose threshold will most likely produce higher dimensional clusters but also a huge amount of (potentially meaningless) low dimensional clusters
Kapitel 5: Clustering High-dim Data

1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data
   5.1 Challenges
   5.2 Subspace Clustering
   5.3 Projected Clustering
   5.4 Correlation Clustering
Kapitel 5: Clustering High-dim Data

1. Introduction to Feature Spaces

2. Challenges of High Dimensional Data

3. Supervised Feature Selection

4. Feature Reduction and Metric Learning

5. Clustering High Dimensional Data
   5.1 Challenges
   5.2 Subspace Clustering
   5.3 Projected Clustering
   5.4 Correlation Clustering