Tomáš Horváth

BUSINESS ANALYTICS

Lecture 3

Data Pre-processing

Information Systems and Machine Learning Lab

University of Hildesheim

Germany

Overview

The aim of this lecture is to describe some data pre-processing approaches.

- What to have in mind when checking the quality of data
- Value editing
  - aggregation, missing value completion, noise handling, normalization, discretization, value transformations, ...
- Feature selection
- Sampling
- Dimensionality reduction

Data quality factors

The quality of data is influenced by

- Noise
  - sometimes can be ignored, depending on the context
- Outliers
  - since these are legitimate data objects (a kind of anomalies in data), can sometimes be of interests
- Missing values
- Inconsistent values
- Redundant data
- Application related issues
  - timeliness, relevance, ...
- Knowledge about data

Missing values

- Ignoring the tuple
  - When does it works fine and when doesn't?
- Fill the values manually
  - When would You use this method?
- Replace missing values by a constant, e.g. "unknown", $-\infty$
  - What is the drawback here?
- Fill by a computed value
  - Use the attribute mean
  - Use the attribute mean of objects belonging to the same class
  - Use the most probably value for an attribute derived by some learning algorithm from other attributes

Note, that a missing value is not necessary an error in data (e.g. one has no driver licence number to fill in a questionnaire).

Noisy data

- Binning
  - Distribute sorted values to equal-width or equal-frequency **bins** and smooth individual values in each bin by a mean, median or the boundaries of the given bin.
- Regression[1]
  - Fit the data to a function using an other attribute and smooth the values by the values of the fitted function.
- Clustering[2]
  - Organize similar objects (values) to groups and use a representative value of each cluster for smoothing.
- Concept hierarchies
  - Smooth the data values by a more general value from the concept hierarchy, if known (e.g. map price to ranges)

Some smoothing methods can also be used for data reduction.

[1] A machine learning technique used to fit the data to a (most often linear) function.

[2] Will be discussed in the following lecture.

Binarization

- Uniquely assign[1] each of the m values to an integer in $[0, m-1]$ and convert it to a binary number
- Use one binary attribute for each categorical value

Categorical value	Integer value	x_1	x_2	x_3	x_1	x_2	x_3	x_4	x_5
awful	0	0	0	0	1	0	0	0	0
poor	1	0	0	1	0	1	0	0	0
ok	2	0	1	0	0	0	1	0	0
good	3	0	1	1	0	0	0	1	0
great	4	1	0	0	0	0	0	0	1

[1] In case of ordinal values the order must be kept.

Discretization

Transformation of a continuous attribute to categorical attribute with two steps:

- deciding the number of categories
- determining how to map the values to categories

Visual

- Sometimes, visually inspecting the data can be an effective approach, at least to decide on the number of categories.

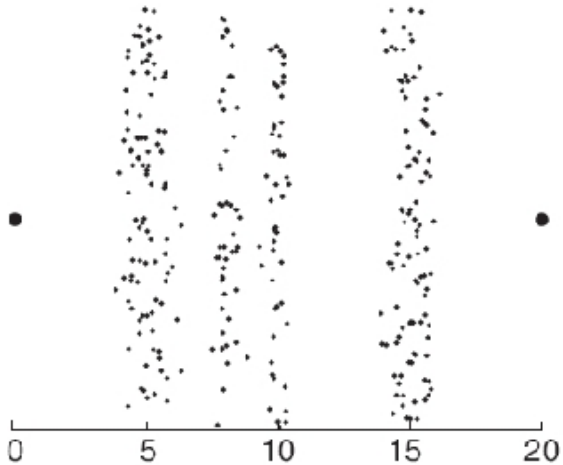
Unsupervised

- class information is not used
  - equal width, equal frequency
  - k-means

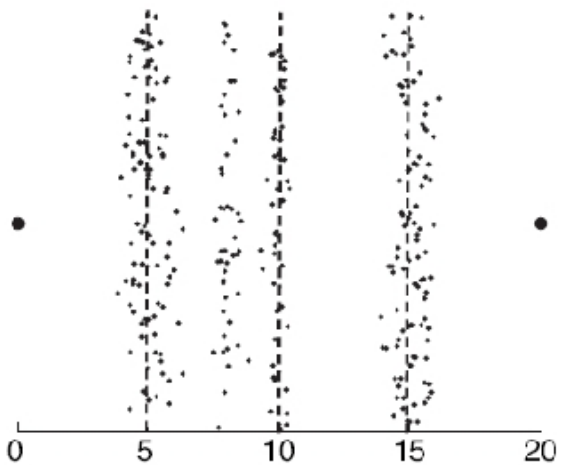
Supervised

- with the use of class information

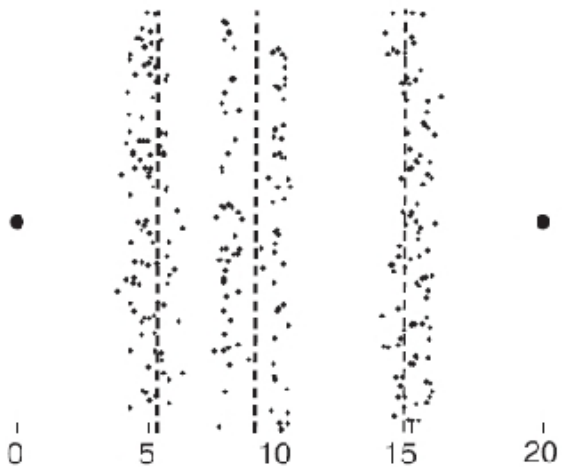
Unsupervised discretization



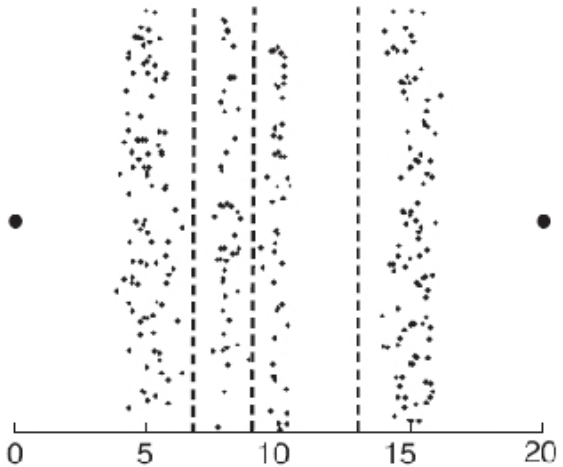
(a) Original data.



(b) Equal width discretization.



(c) Equal frequency discretization.



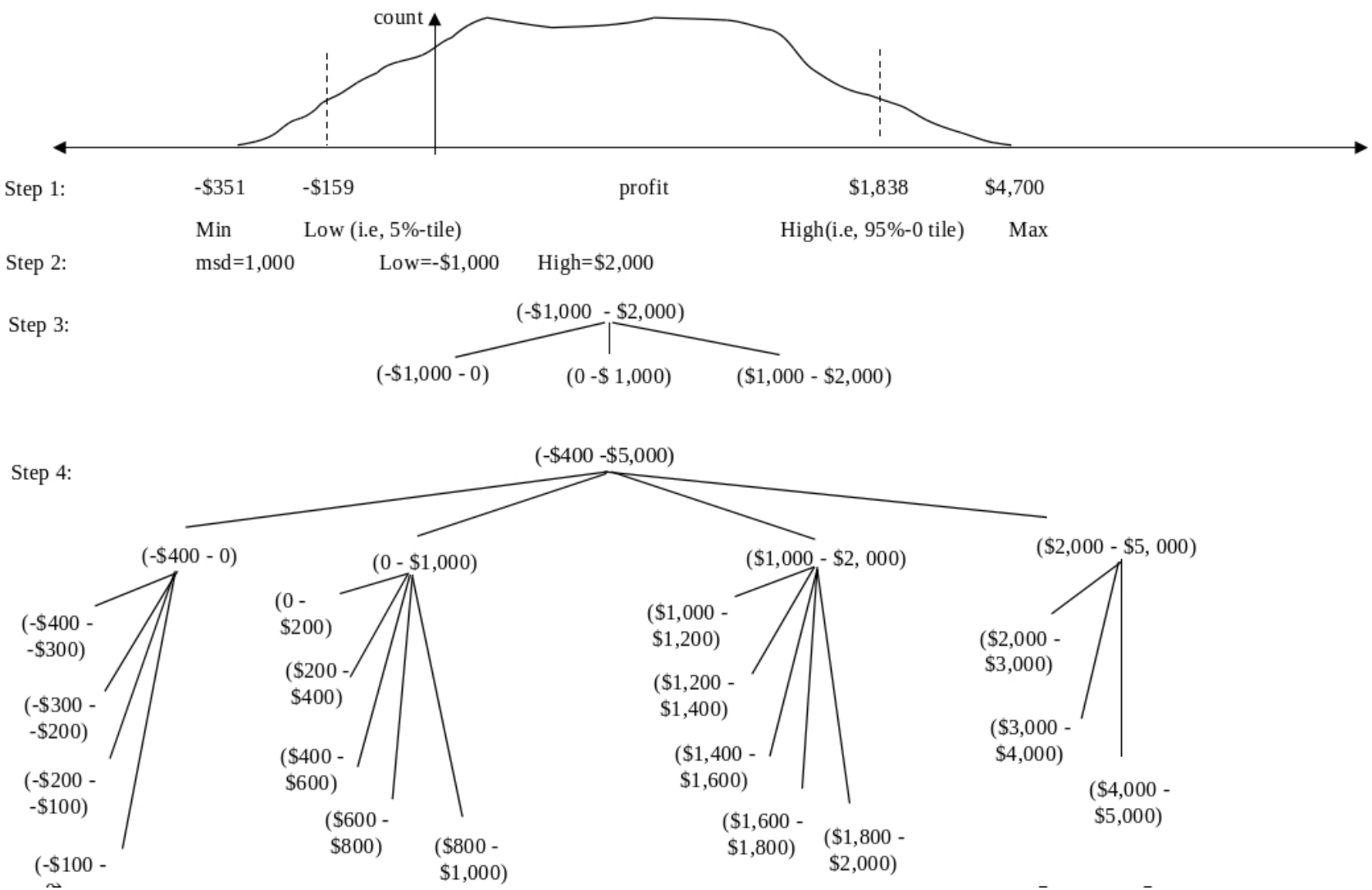
(d) K-means discretization.

Unsupervised discretization: the 3-4-5 rule (1)

Used to segment data into relatively uniform natural intervals using the most significant digit.

- If an interval covers 3, 6, 7 or 9 distinct values at the most significant digit, partition the range into 3 equal-sized intervals.
- If an interval covers 2, 4 or 8 distinct values at the most significant digit, partition the range into 4 equal-sized intervals.
- If an interval covers 1, 5 or 10 distinct values at the most significant digit, partition the range into 5 equal-sized intervals.

Unsupervised discretization: the 3-4-5 rule (2)



Unsupervised discretization using concept hierarchies

If the knowledge about the hierarchies of concepts in data is present, we can discretize the values at lower levels to values at highest levels

- e.g. street – city – state – country

Entropy-based supervised discretization (1)

Let's have k different class labels, n_i values in the i-th interval, n_{ij} values of class j in the i-th interval and m the number of intervals.

- The **entropy**[1] of the i-th interval is defined as

$$e_i = \sum_{j=1}^{k} \frac{n_{ij}}{n_i} log_2 \frac{n_{ij}}{n_i}$$

- The total entropy is

$$e = \sum_{i=1}^{m} \frac{n_i}{n} e_i$$

The basic idea is to split initial values to m intervals such that the total entropy is minimal.

- How to choose m?

[1] measure of "purity".

Entropy-based supervised discretization (2)

A simple iterative approach to find splitting points

1. $j = 0, \quad \mathcal{X}^0 = \mathcal{X}, \quad S = \emptyset$
2. $l^j = min\{x | x \in \mathcal{X}^j\}, \quad r^j = max\{x | x \in \mathcal{X}^j\}$
3. find $X_1 = \langle l^j, s \rangle, X_2 = (s, r^j\rangle$ such that $\frac{|X_1|}{n} e_{X_1} + \frac{|X_2|}{n} e_{X_2}$ is minimal, where e_{X_1}, e_{X_2} are the entropies for X_1, X_2.
4. $j = j + 1, \quad S = S \bigcup \{s\}$
5. if $e_{X_1} > e_{X_2}$ then $\mathcal{X}^j = X_1$ else $\mathcal{X}^j = X_2$
6. if stopping criteria not fulfilled then goto the step 2.
7. return S

Value transformation

Sometimes, transformed values are more convenient as the original ones, e.g.

- if only the magnitudes of values are important, then we can take the absolute value
- if the values are the numbers of data bytes in a session ranging from 1 to billion, it is more convenient to use a log_{10} transformation.
- transform the data to have a normal distribution

Popular transformation functions are $|x|, x^k, logx, e^x, \sqrt{x}, \frac{1}{x}, \ldots$

The transformation change the nature of data!

- using $\frac{1}{x}$ to $\{1, 2, 3\}$ results in a changed ordering $\{1, \frac{1}{2}, \frac{1}{3}\}$

Normalization

- Normalization by decimal scaling

$$x' = \frac{x}{10^j}, \quad \text{such that} \quad max\{|v'|\} < 1$$

- Min-max normalization

$$x' = \frac{x - min_{\mathcal{X}}}{max_{\mathcal{X}} - min_{\mathcal{X}}}$$

- z-score (zero-mean) normalization

$$x' = \frac{x - \overline{x}}{\sigma_x}$$

  - where $\overline{x}, \sigma_x$ are the mean and standard deviation of the values
  - the median or the absolute standard deviation can be also used

Feature subset selection

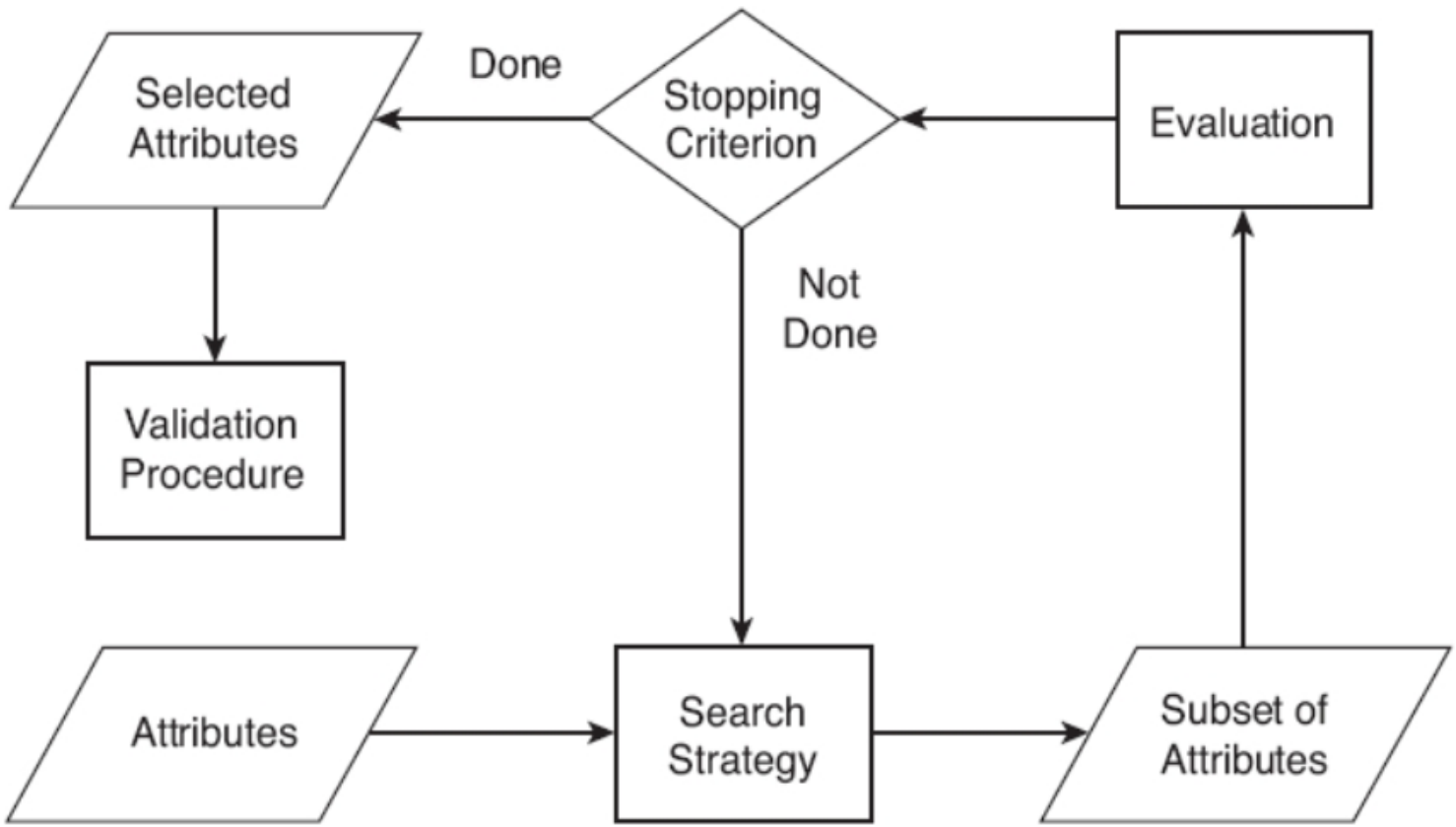
The aim is to get rid of **redundant** and **irrelevant** features.

- ideal approach is to try all possible features as input to the data mining algorithm used
  - intractable in most of the cases
- alternative strategies
  - Embedded approaches
    - feature selection is a part of the data mining algorithm which decides by itself on which attribute to use (e.g. decision trees)
  - Filter approaches
    - features are selected before running the data mining algorithm by an approach independent of the data mining task (e.g. select attributes with low pairwise correlations)
  - Wrapper approaches
    - use the data mining algorithm as a black box to find the best subset of attributes without enumerating all possible subsets

Feature subset selection: An architecture

Encompassing filter and wrapper methods in one architecture

- they differ only in the way in which they evaluate a subset of features



Sampling

The goal is to find a sample of the dataset having approximately the same properties as the original dataset.

- A good sampling scheme guaranties a high probability of getting a representative sample.
  - This involves choosing the approppriate sample size as well as the sampling technique.

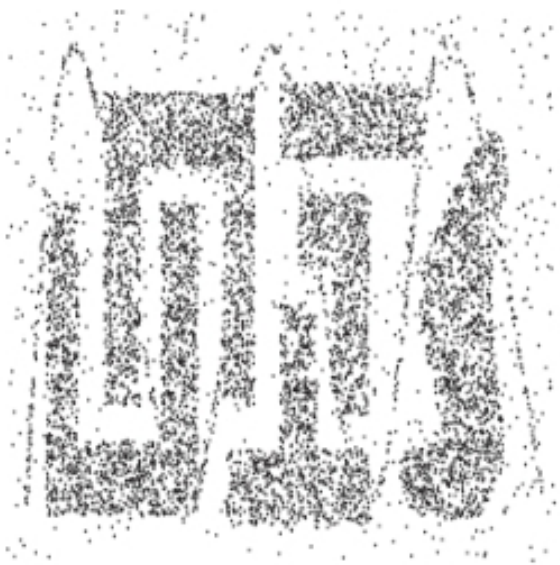
Simple random sampling

- an equal probability of selecting any particular object
  - without replacement
  - with replacement

Stratified sampling

- accomodating different frequencies for items
  - draw equal number of objects from each group
  - drawn objects from groups proportional to the sizes of groups

Sampling: Sample size vs. the loss



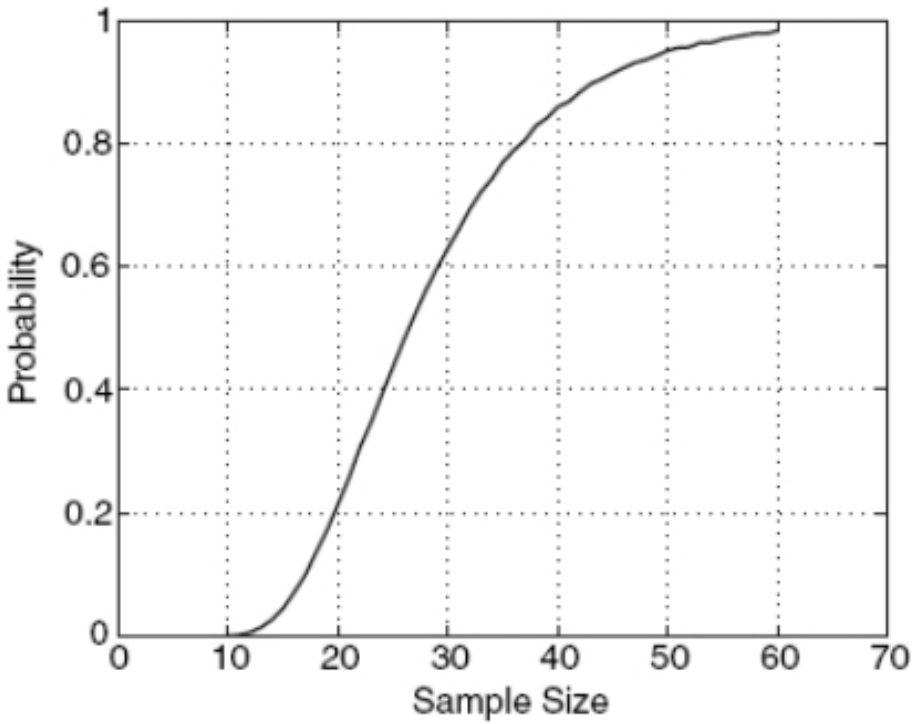
(a) 8000 points

(b) 2000 points

(c) 500 points

Sampling: Determining the sample size

(a) Ten groups of points.

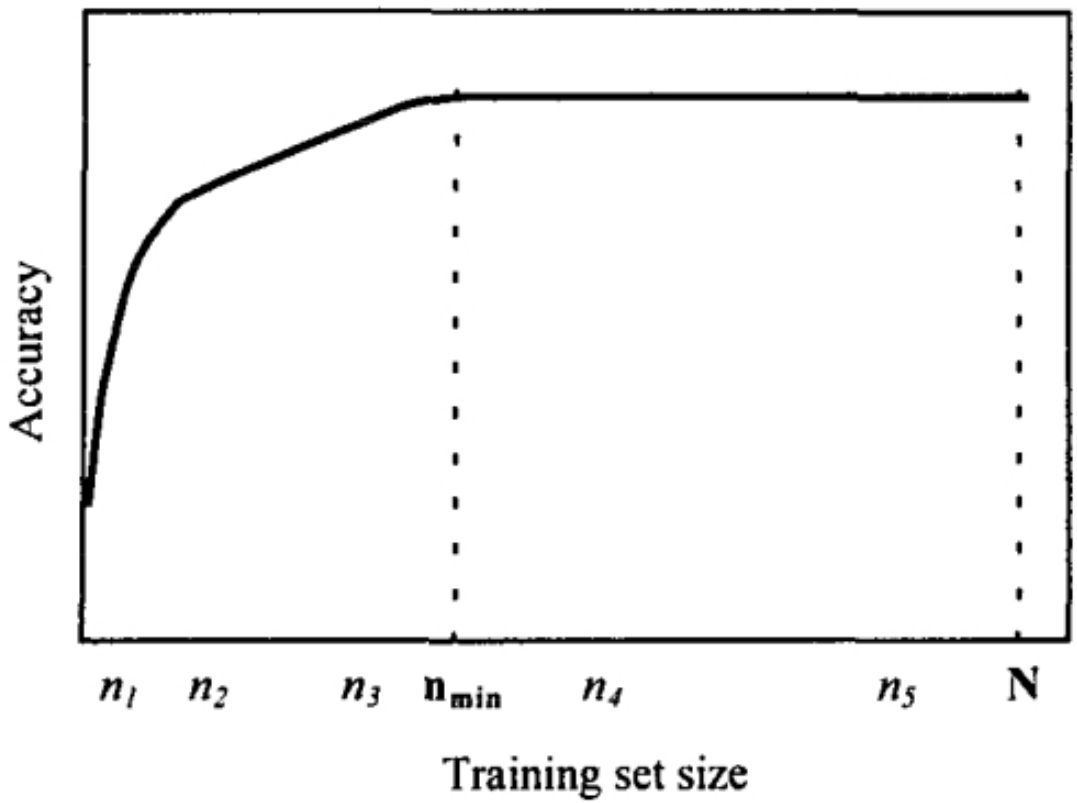


(b) Probability a sample contains points from each of 10 groups.

Progressive Sampling

What is the size of the smallest sufficient training set?

- n_{min} is hard to determine from theory, however an empirical approximation $\hat{n}_{min}$ is possible.



[1]

[1] Image source: F. Provost, D. Jensen, T. Oates: Efficient Progressive Sampling, KDD-99, San Diego CA USA, ACM 1999.

PS: Algorithm

- Compute schedule $S = \{n_0, n_1, \ldots, n_k\}$ of sample sizes
- $n \leftarrow n_0$
- $M \leftarrow$ model induced from n instances
- while not *converged*
  - recompute S if necessary
  - $n \leftarrow$ next element of S larger than n
  - $M \leftarrow$ model induced from n instances
- end while
- return M

PS: Schedules

Basic schedules

- All instances: $S_N = \{N\}$
- Omniscient oracle: $S_O = \{n_{min}\}$

Static sampling

- computes $\hat{n}_{min}$ according to a subsample's statistical similarity to the entire sample: $S_S = \{\hat{n}_{min}\}$

Arithmetic sampling

- $S_A = \{n_0, n_0 + n_\delta, n_0 + 2 \cdot n_\delta, \ldots, n_0 + k \cdot n_\delta\}$
- more accurate models than S_S since n_{min} depends on the relationship between data and the specific learning algorithm

Geometric sampling

- $S_G = \{n_0, a \cdot n_0, a^2 \cdot n_0, \ldots, a^k \cdot n_0\}$
- robust schedule

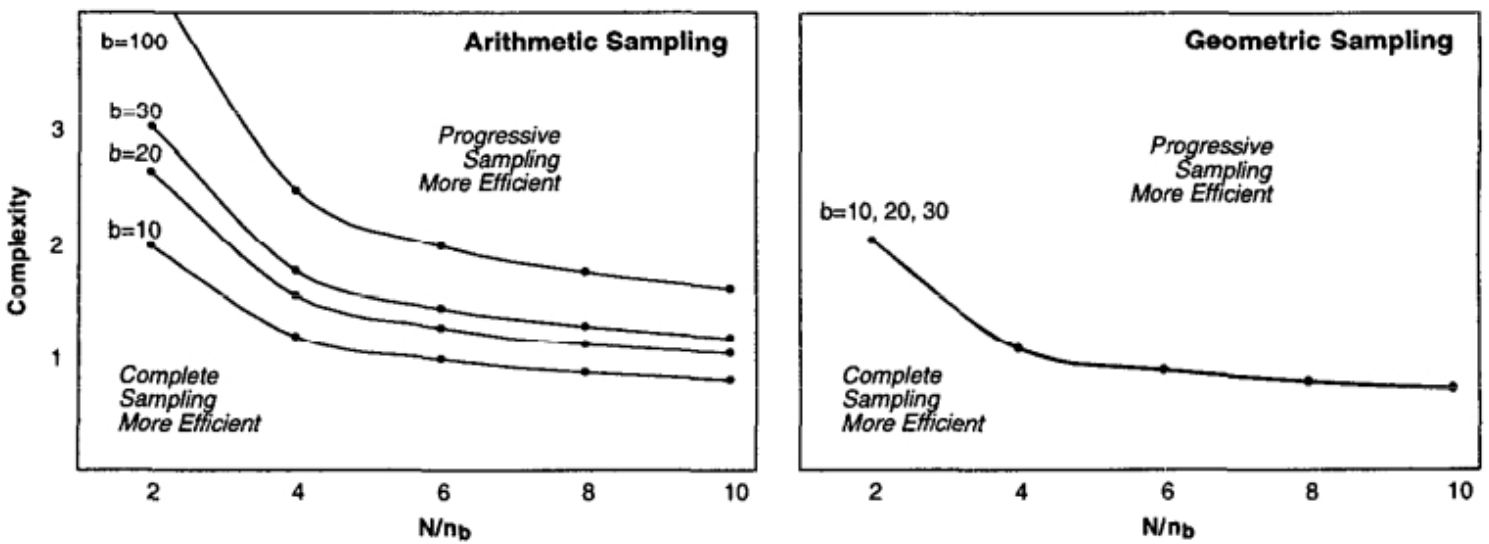
PS: Efficiency

Consider

- an induction algorithm with a polynomial complexity $O(n^c)$
- n_b as the first schedule point prior to the point of the convergence

The condition under which the computational cost of progressive sampling is equal to the cost of using all instances is

$$N^c = n_0^c + n_1^c + n_2^c + \cdots + n_b^c$$



PS: Asymptotic optimality (1)

How do simple schedules compare to $S_O = \{n_{min}\}$?

Theorem: *For induction algorithms with polynomial time complexity* $\Theta(f(n))$*, no better than* $O(n)$*, if confergence also can be detected in* $O(f(n))$*, then geometric progressive sampling is asymptotically optimal among progressive sampling methods in terms of run time.*

Proof: Given n_{min} the size of the smallest sufficient training set, the run-time complexity of S_O is $\Theta(f(n_{min}))$.
Geometric progressive sampling runs the induction algorithm on subsets of sizes $a^i \cdot n_0$ for $i = 0, 1, \ldots, b$, before the convergence is detected.

PS: Asymptotic optimality (2)

Assuming that the convergence is well detected, we have

$$a^{b-1} \cdot n_0 < n_{min} \leq a^b \cdot n_0 < a \cdot n_{min}$$

which means that

$$a^i \cdot n_0 < \frac{a^i}{a^{b-1}} \cdot n_{min}$$

for $i = 0, 1, \ldots, b$. Since $O(f(\cdot))$ is at best linear, the run time of S_G is

$$O\big(f(\sum_{i=0}^{b} \frac{a^i}{a^{b-1}} \cdot n_{min})\big) = O\big(f(a \cdot n_{min} \cdot \underbrace{(1 + \frac{1}{a} + \frac{1}{a^2} + \cdots + \frac{1}{a^b})}_{\rightsquigarrow const, \text{ since } a>1})\big)$$

Therefore the run time of S_G is asymptotically no worse than the run time of S_O. $\square$

PS: Expectation-based optimality (1)

How can optimal schedules be constructed given expectations between the two extremes S_O and S_N?

- probability $\Phi(n)$ that convergence requires more than n instances
  - $\Phi(n) = (N - n)/N$ – if no prior information are available

The expected cost of convergence by $S = \{n_0 = 0, n_1, \ldots, n_k\}$ is

$$C^S = \sum_{i=1}^{k} \Phi(n_{i-1}) f(n_i)$$

Example: 10 instances, uniform prior, $f(n) = n^2$

$S_1 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$	$C^{S_1} = 121$
$S_2 = \{10\}$	$C^{S_2} = 100$
$S_3 = \{2, 6, 10\}$	$C^{S_3} = 72.8$

$$C^{S_3} = \Phi(0)f(2) + \Phi(2)f(6) + \Phi(6)f(10) = 1 \cdot 4 + \tfrac{8}{10}36 + \tfrac{4}{10}100 = 72.8$$

PS: Expectation-based optimality (2)

For each value of n, a model can either be built or not

- We have 2^N possible schedules! How to find an optimal one?

Optimal schedules are composed of optimal sub-schedules.

- use of dynamic programming
  - $m[i, j]$ is the cost of minimum expected cost-schedule of all samples in the size range $[i, j]$
  - $m[0, N]$ is the cost of the optimal schedule given a dataset containing N instances computed by the following recurrence

$$m[i,j] = min \begin{cases} \Phi(i)f(j) \\ min_{i<k<j}\, m[i,k] + m[k,j] \end{cases}$$

  - $O(N^3)$ time-complexity, which is still high
    - we can use some heuristics, e.g. looking only at multiplies of 100 or 1000 instances, however, we sacrifice the precision

PS: Example of schedule computation (1)

Let's have: $N = 5, \quad \Phi(i) = \frac{N-i}{N}, \quad f(n) = n^2$

Two tables $M = m[i,j]$ and $S = s[i,j]$, for saving

- the computed *minimal costs* $m[i,j]$ for the different $i < j$
- the *schedules* corresponding to minimal costs, i.e. the way the given costs $m[i,j]$ we've computed
  - if $\Phi(i)f(j) \leq min_{i<k<j}\, m[i,k] + m[k,j]$, then $s[i,j] = \{i,j\}$[1]
  - if $\Phi(i)f(j) > min_{i<k<j}\, m[i,k] + m[k,j]$, then $s[i,j] = s[i,k] \bigcup s[k,j]$

First, we fill out the cells $m[i.j]$, for which "recursion" is not needed, i.e. there is no k between i and j

Then, in each iteration (until we don't reach the right-upper corners of the tables) we use the already computed costs and schedules. Finally,

the schedule $s[0,5] = s[0,3] \bigcup s[3,5] = a[0,1] \bigcup s[1,3] \bigcup s[3,5] = \{1\} \bigcup \{1,3\} \bigcup \{3,5\} = \{1,3,5\}$ is optimal with $C^{\{1,3,5\}} = 18.2$.

[1] If $i = 0$ then $s[i,j] = \{j\}$, since having 0 in a schedule makes no sense.

PS: Example of schedule computation (2)

$i \setminus j$	1	2	3	4	5
0	1				
1	–	3.2			
2	–	–	5.4		
3	–	–	–	6.4	
4	–	–	–	–	5
5	–	–	–	–	–

$i \setminus j$	1	2	3	4	5
0	{1}				
1	–	{1,2}			
2	–	–	{2,3}		
3	–	–	–	{3,4}	
4	–	–	–	–	{4,5}
5	–	–	–	–	–

PS: Example of schedule computation (3)

$i \setminus j$	1	2	3	4	5
0	1	4			
1	–	3.2	7.2		
2	–	–	5.4	9.6	
3	–	–	–	6.4	10
4	–	–	–	–	5
5	–	–	–	–	–

$i \setminus j$	1	2	3	4	5
0	{1}	{2}			
1	–	{1,2}	{1,3}		
2	–	–	{2,3}	{2,4}	
3	–	–	–	{3,4}	{3,5}
4	–	–	–	–	{4,5}
5	–	–	–	–	–

PS: Example of schedule computation (4)

$i \setminus j$	1	2	3	4	5
0	1	4	8.2		
1	–	3.2	7.2	12.8	
2	–	–	5.4	9.6	14.6
3	–	–	–	6.4	10
4	–	–	–	–	5
5	–	–	–	–	–

$i \setminus j$	1	2	3	4	5
0	{1}	{2}	$s[0,1] \bigcup s[1,3]$		
1	–	{1,2}	{1,3}	{1,4}	
2	–	–	{2,3}	{2,4}	$s[2,4] \bigcup s[4,5]$
3	–	–	–	{3,4}	{3,5}
4	–	–	–	–	{4,5}
5	–	–	–	–	–

PS: Example of schedule computation (5)

$i \setminus j$	1	2	3	4	5
0	1	4	8.2	13.6	
1	–	3.2	7.2	12.8	17.2
2	–	–	5.4	9.6	14.6
3	–	–	–	6.4	10
4	–	–	–	–	5
5	–	–	–	–	–

$i \setminus j$	1	2	3	4	5
0	{1}	{2}	$s[0,1] \bigcup s[1,3]$	$s[0,2] \bigcup s[2,4]$	
1	–	{1,2}	{1,3}	{1,4}	$s[1,3] \bigcup s[3,5]$
2	–	–	{2,3}	{2,4}	$s[2,4] \bigcup s[4,5]$
3	–	–	–	{3,4}	{3,5}
4	–	–	–	–	{4,5}
5	–	–	–	–	–

PS: Example of schedule computation (6)

$i \setminus j$	1	2	3	4	5
0	1	4	8.2	13.6	18.2
1	–	3.2	7.2	12.8	17.2
2	–	–	5.4	9.6	14.6
3	–	–	–	6.4	10
4	–	–	–	–	5
5	–	–	–	–	–

$i \setminus j$	1	2	3	4	5
0	{1}	{2}	$s[0,1] \bigcup s[1,3]$	$s[0,2] \bigcup s[2,4]$	$s[0,3] \bigcup s[3,5]$
1	–	{1,2}	{1,3}	{1,4}	$s[1,3] \bigcup s[3,5]$
2	–	–	{2,3}	{2,4}	$s[2,4] \bigcup s[4,5]$
3	–	–	–	{3,4}	{3,5}
4	–	–	–	–	{4,5}
5	–	–	–	–	–

PS: Convergence & Actual run-time complexity

Efficient and accurate **convergence detection** plays a key role

- *model the learning curve* as sampling progresses
  - three regions: the primary and the secondary *rise*, and the *plateau*
    - simple functional form usually cannot capture all three regions
    - adding more points to the schedule would be beneficial for more precise modeling but it impairs efficiency
- linear regression with local sampling
  - sample l additional points in the local neighborhood of the latest scheduled sample size n_i
  - estimate a linear regression line and compare its slope to zero
    - if the slope is "close" to zero, convergence is detected

An **actual run-time complexity** of the underlying induction algorithm is more required as the worst-case complexity

- assuming that $f(n) = const \cdot n^c$, we have $log\, f(n) = log\, const + c \cdot log\, n$
- sample $f(n)$ for som n, take their logarithm and use linear regression to estimate c

Dimensionality Reduction

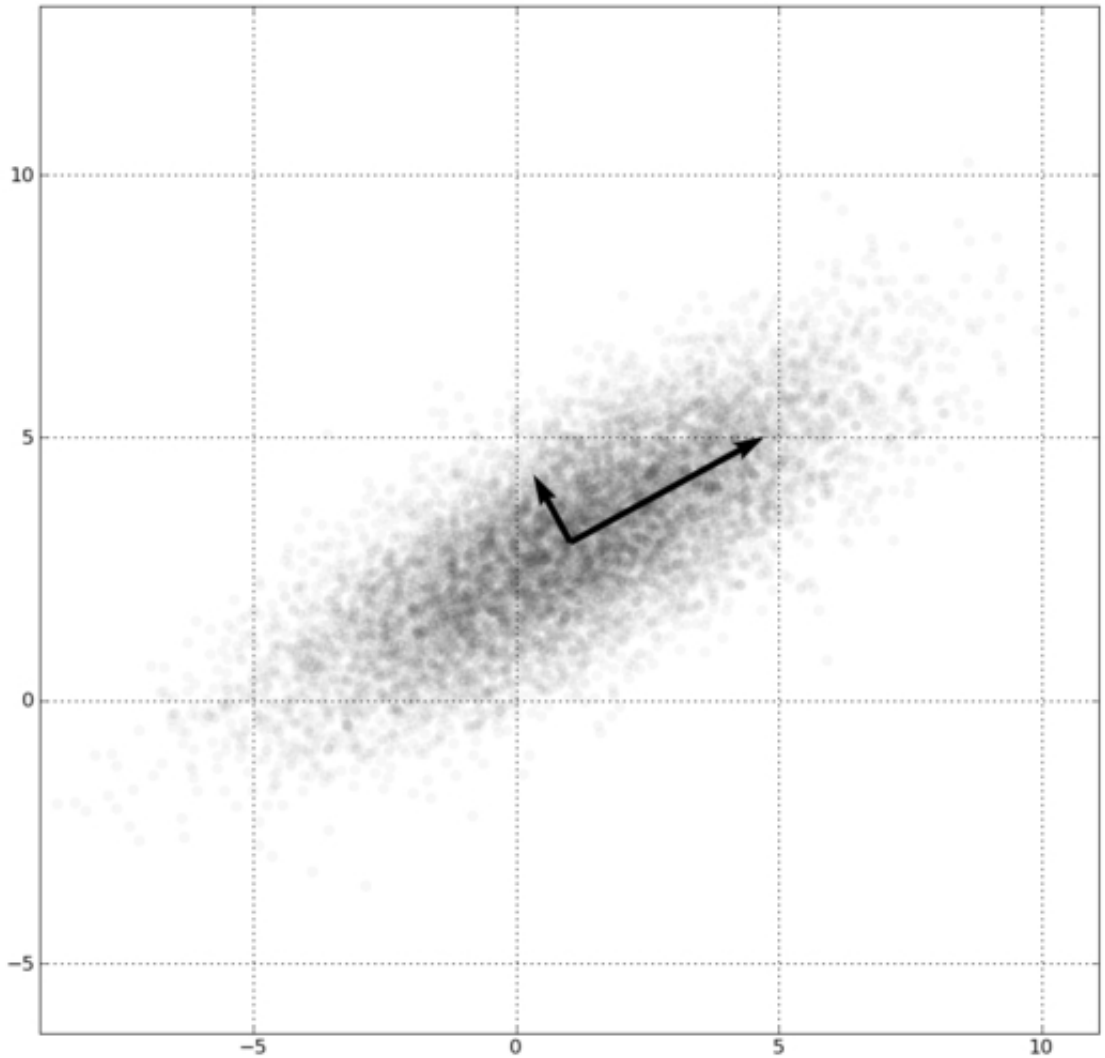
The goal is to obtain a reduced (compressed) representation of the data.

- *different from feature subset selection* where a subset of "suitable" attributes were only retained in the initial data
- leads to loss of information

Principal Component Analysis is one of the most popular methods

- describe a dataset of d variables by new, k variables which are linear combinations of the original variables
  - search for k d-dimensional, pairwise *orthogonal* and *uncorrelated* vectors ($k < d$) which represent the data in a best way
    - i.e. find "directions" with the *largest variance* in the data, which are the most important – in other words, most *principal*
- minimizes the loss of information

PCA & Multivariate Normal Distribution



[2]

[2] Figure from http://en.wikipedia.org/wiki/Multivariate_normal_distribution

Multivariate Normal Distribution (1)

notation: $\mathbf{x} = [x_1, \ldots, x_d]^T \sim \mathcal{N}_d(\boldsymbol{\mu}, \Sigma)$

mean

- $\mathcal{E}\{\mathbf{x}\} = \boldsymbol{\mu} = [\mu_1, \ldots, \mu_d]^T$

covariance

- $\sigma_{ij} = Cov(X_i, X_j) = \mathcal{E}\{(X_i - \mu_i)(X_j - \mu_j)\} = \mathcal{E}\{X_i X_j\} - \mu_i \mu_j$
- σ_i^2 (σ_{ii}) – variance of x_i

covariance matrix

- $\Sigma \equiv Cov(\boldsymbol{X}) = \mathcal{E}\{(\boldsymbol{X} - \boldsymbol{\mu})(\boldsymbol{X} - \boldsymbol{\mu})^T\} = \mathcal{E}\{\boldsymbol{X}\boldsymbol{X}^T\} - \boldsymbol{\mu}\boldsymbol{\mu}^T$

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \ldots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \ldots & \sigma_{2d} \\ \vdots & & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \ldots & \sigma_d^2 \end{bmatrix}$$

Multivariate Normal Distribution (2)

The projection of a d-dimensional normal distribution to a vector $\mathbf{w} \in \mathbb{R}^d$ results in a univariate normal distribution.

- $\mathcal{E}\{\mathbf{w}^T\mathbf{x}\} = \mathbf{w}^T\mathcal{E}\{\mathbf{x}\} = \mathbf{w}^T\boldsymbol{\mu}$
- $Var(\mathbf{w}^T\mathbf{x}) = \mathcal{E}\{(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\boldsymbol{\mu})^2\} = \mathcal{E}\{(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\boldsymbol{\mu})(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\boldsymbol{\mu})\} = \mathcal{E}\{\mathbf{w}^T(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T\mathbf{w}\} = \mathbf{w}^T\mathcal{E}\{(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T\}\mathbf{w} = \mathbf{w}^T\Sigma\mathbf{w}$
- $\mathbf{w}^T\mathbf{x} = w_1x_1 + \cdots + w_dx_d \sim \mathcal{N}(\mathbf{w}^T\boldsymbol{\mu}, \mathbf{w}^T\Sigma\mathbf{w})$

The projection of a d-dimensional normal distribution to a k-dimensional space results in a k-dimensional normal distribution.

- $\mathbf{W}^T\mathbf{x} \sim \mathcal{N}(\mathbf{W}^T\boldsymbol{\mu}, \mathbf{W}^T\Sigma\mathbf{W})$, where $\mathbf{W}$ is a $d \times k$ matrix with rank[1] $k < d$.

[1] The number of linearly independent rows or columns.

PCA: The first principal component

The principal component of the data is $\mathbf{w}_1$, i.e. the data, after projection to $\mathbf{w}_1$ are spreaded out the most.

- to make the direction important, $\|\mathbf{w}_1\| = 1$
- let $z_1 = \mathbf{w}_1^T \mathbf{x}$ be the projection of $\mathbf{x}$ to $\mathbf{w}_1$

$Var(z_1) = \mathbf{w}_1^T \Sigma \mathbf{w}_1$

- find $\mathbf{w}_1$ which maximizes $Var(z_1)$, subject to $\mathbf{w}_1^T \mathbf{w}_1 = 1$
- converting this problem to a Lagrange[1] problem we get $\max_{\mathbf{w}_1} \mathbf{w}_1^T \Sigma \mathbf{w}_1 - \alpha_1(\mathbf{w}_1^T \mathbf{w}_1 - 1)$
  - from $2\Sigma \mathbf{w}_1 - 2\alpha_1 \mathbf{w}_1 = 0$ we get $\Sigma \mathbf{w}_1 = \alpha_1 \mathbf{w}_1$
    - holds if $\mathbf{w}_1$ is an *eigenvector*[2] of Σ with the *eigenvalue* α_1
  - Since $\mathbf{w}_1^T \Sigma \mathbf{w}_1 = \alpha_1 \mathbf{w}_1^T \mathbf{w}_1 = \alpha_1$, the principal component will be the eigenvector $\mathbf{w}_1$ with the largest eigenvalue α_1.

[1] Shortly, to find an extreme of a function $f(x)$ subject to a constraint $g(x)$, one should define and extremise a new, Lagrangian function $F(x, \alpha) = f(x) - \alpha g(x)$ to get the solution.

[2] There can be more eigenvectors (and corresponding eigenvalues) for a square matrix.

PCA: The second principal component

The second principal component $\mathbf{w}_2$ should be of unit length, maximize the variance $Var(z_2) = \mathbf{w}_2^T \Sigma \mathbf{w}_2$ and be orthogonal[1] to $\mathbf{w}_1$.

$$\max_{\mathbf{w}_2} \mathbf{w}_2^T \Sigma \mathbf{w}_2 - \alpha_2(\mathbf{w}_2^T \mathbf{w}_2 - 1) - \beta(\mathbf{w}_2^T \mathbf{w}_1 - 0)$$

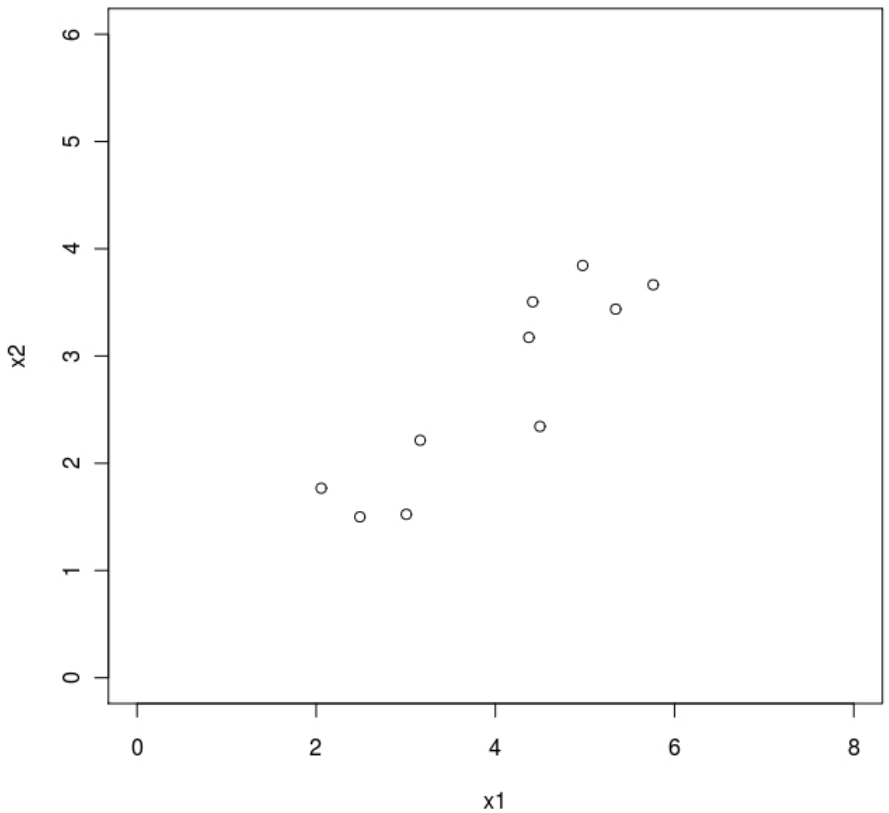
- derivate with respect to $\mathbf{w}_2$ and setting it equal to 0 we get $2\Sigma\mathbf{w}_2 - 2\alpha_2\mathbf{w}_2 - \beta\mathbf{w}_1 = 0$
  - $2\mathbf{w}_1^T\Sigma\mathbf{w}_2 - 2\alpha_2\mathbf{w}_1^T\mathbf{w}_2 - \beta\mathbf{w}_1^T\mathbf{w}_1 = 0$
    - (1) $\mathbf{w}_1^T\mathbf{w}_2 = 0 \implies 2\alpha_2\mathbf{w}_1^T\mathbf{w}_2 = 0$
    - (2) $\Sigma\mathbf{w}_1 = \alpha_1\mathbf{w}_1 \implies \mathbf{w}_1^T\Sigma\mathbf{w}_2 = \mathbf{w}_2^T\Sigma\mathbf{w}_1 = \alpha_1\mathbf{w}_2^T\mathbf{w}_1 = 0$
    - (1) $\wedge$ (2) $\implies \beta = 0$
- from $\beta = 0$ we get $\Sigma\mathbf{w}_2 = \alpha_2\mathbf{w}_2$
  - Since $\mathbf{w}_2^T\Sigma\mathbf{w}_2 = \alpha_2\mathbf{w}_2^T\mathbf{w}_2 = \alpha_2$, the second principal component will be the eigenvector $\mathbf{w}_2$ with the second largest eigenvalue α_2.

Similarly, the other principal components can be found.

[1] Because $z_2 = \mathbf{w}_2^T\mathbf{x}$ should be uncorrelated with z_1.

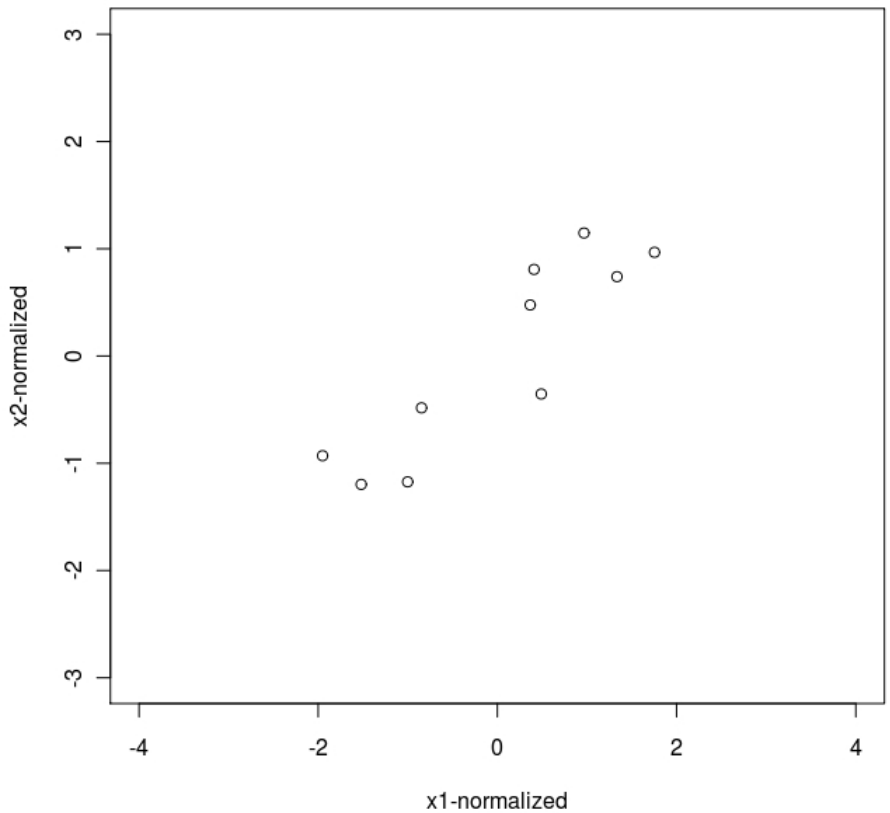
PCA howto: 1. Getting the data

n	x_1	x_2
1	3.160724	2.214081
2	4.974025	3.844295
3	5.342253	3.437068
4	5.761281	3.664205
5	4.373575	3.173270
6	2.056070	1.767576
7	3.005617	1.523987
8	2.487312	1.500183
9	4.497077	2.343299
10	4.416509	3.504797
$\overline{\mathbf{x}}$	4.007444	2.697276



PCA howto: 2. Subtract the mean

n	x_1	x_2
1	-0.8467200	-0.4831947
2	0.9665806	1.1470194
3	1.3348088	0.7397920
4	1.7538371	0.9669287
5	0.366130	0.4759943
6	-1.9513746	-0.9297005
7	-1.0018272	-1.1732889
8	-1.5201328	-1.1970934
9	0.4896325	-0.3539773
10	0.4090651	0.8075204



PCA howto: 3. Compute Σ, $\mathbf{W}$ and $\boldsymbol{\alpha}$

Covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} 1.574702 & 1.0379763 \\ 1.0379763 & 0.8565907 \end{bmatrix}$$

Eigenvalues

$$\boldsymbol{\alpha} = [2.3139707, 0.1173224]$$

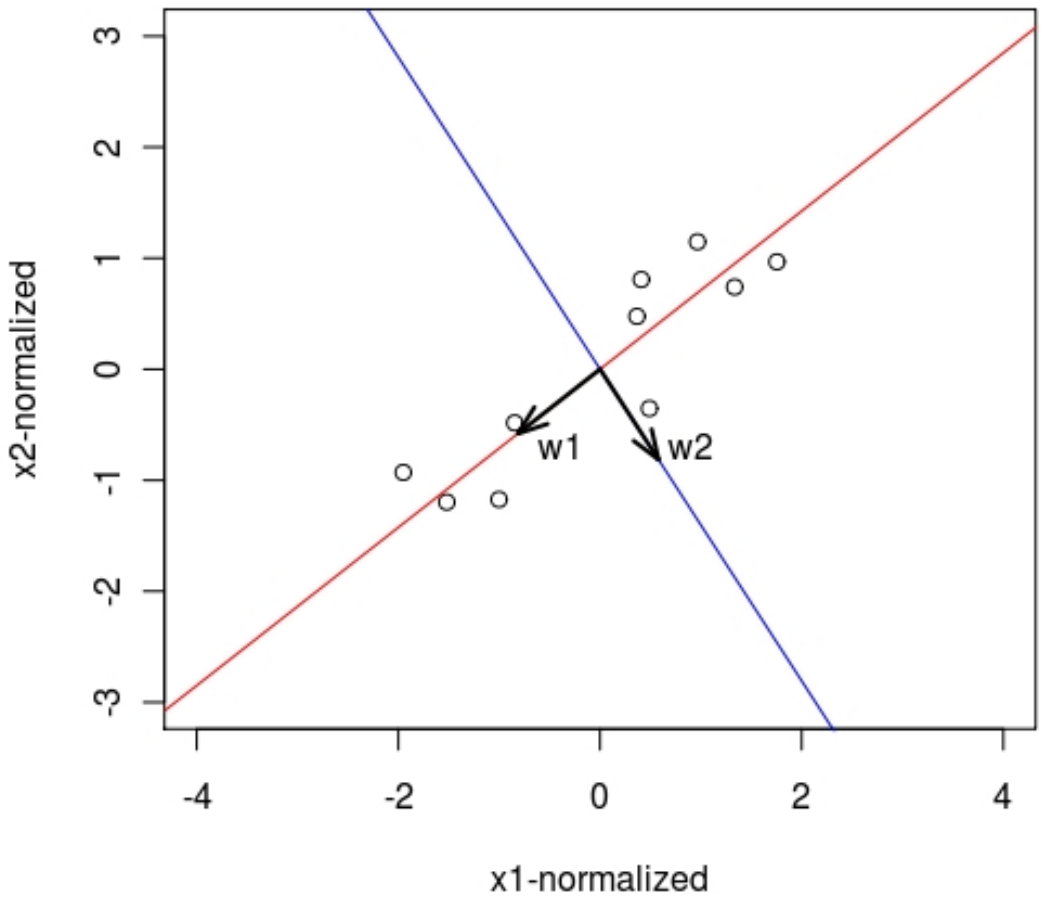
- $\alpha_1 = 2.3139707$
- $\alpha_2 = 0.1173224$

Eigenvectors

$$\mathbf{W} = \begin{bmatrix} -0.8145282 & 0.5801239 \\ -0.5801239 & -0.8145282 \end{bmatrix}$$

- $\mathbf{w}_1 = [-0.8145282, -0.5801239]$
- $\mathbf{w}_2 = [0.5801239, -0.8145282]$

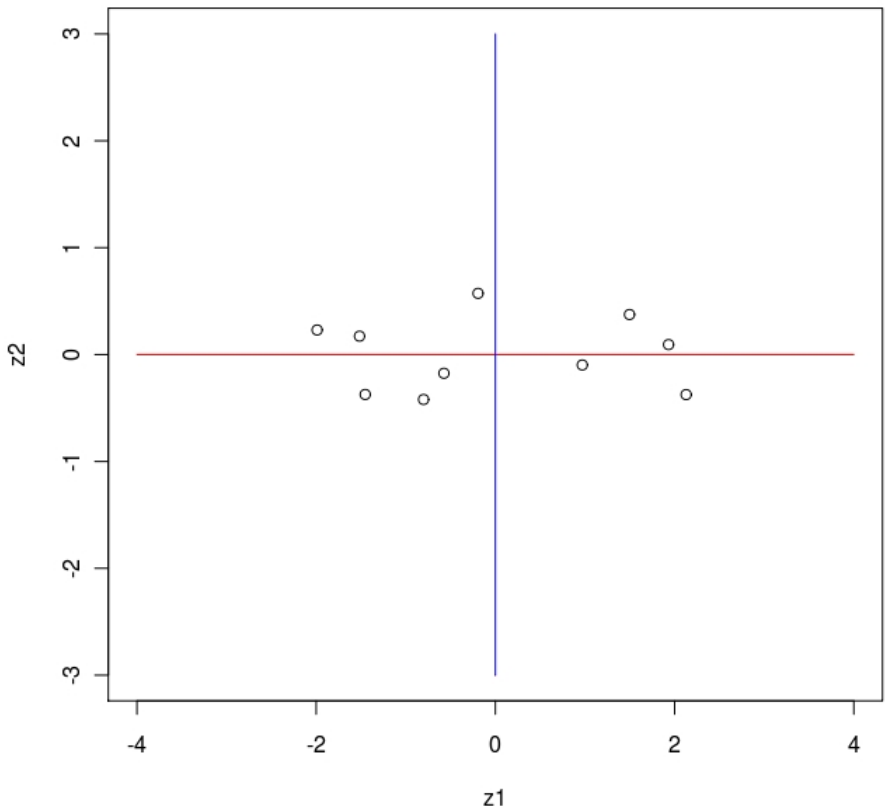
PCA: Result



PCA: Data transformation (1)

$$\mathbf{z} = \mathbf{W}^T \mathbf{x}$$

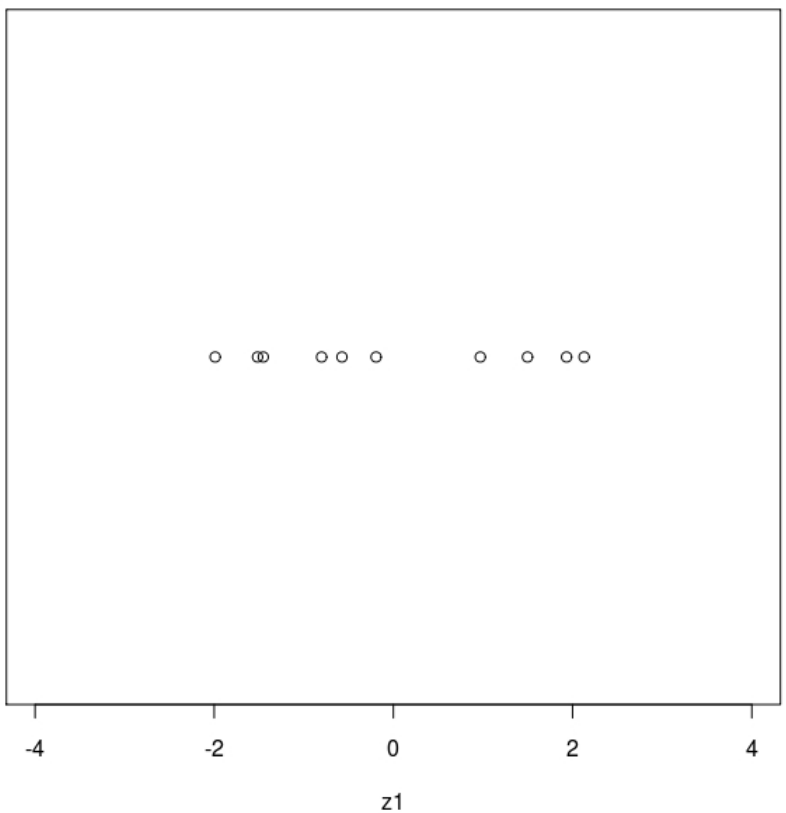
n	x_1	x_2
1	0.9699901	-0.09762680
2	-1.4527205	-0.37354311
3	-1.5164105	0.17177305
4	-1.9894882	0.22985217
5	-0.5743592	-0.17530976
6	2.1287911	-0.37477183
7	1.4966695	0.37449302
8	-1.9326535	0.09320095
9	-0.1934688	0.57237203
10	-0.8016570	-0.42043972

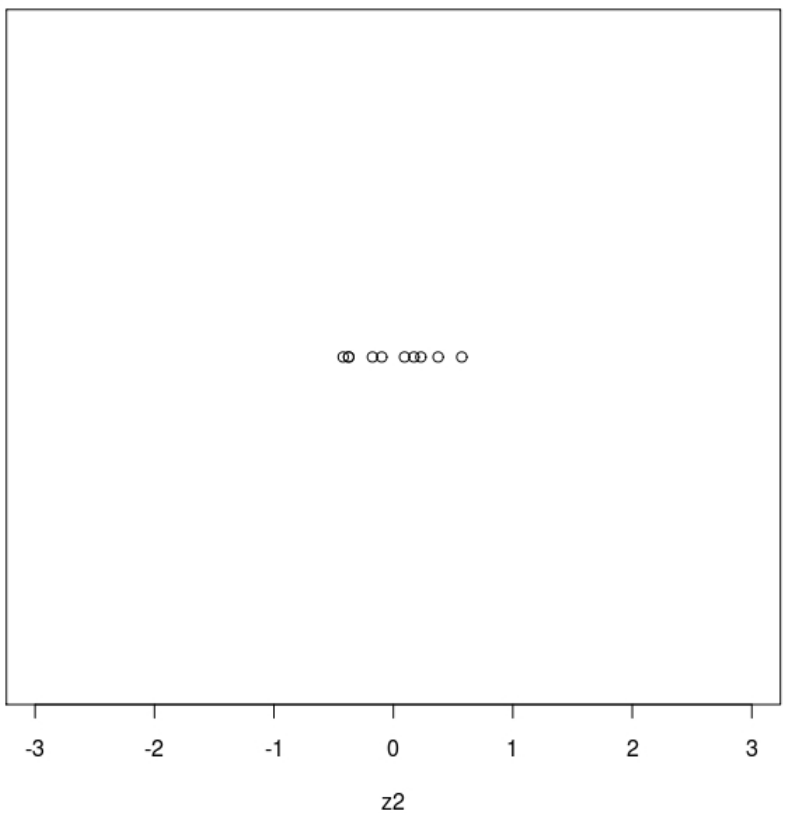


PCA: Data transformation (2)

$$\mathbf{z}_1 = \mathbf{w}_1^T \mathbf{x}$$

$$\mathbf{z}_2 = \mathbf{w}_2^T \mathbf{x}$$





Summary

Value editing

- aggregation, missing value completion, noise handling, normalization, discretization, value transformations, ...
- A good backround/domain knowledge can be useful!

Feature Selection and Dimensionality reduction

- While some attributes are entirely removed in feature selection, dimensionality reduction aims at compressing the data with the lowest loss

Sampling

- Important factors are the sampling technique used and the sample size.

Lot of other techniques can be found in the literature.

Recommended Reading

The slides were created based on the following literature:

- Pang-Ning Tan, Michael Steinbach and Vipin Kumar: Introduction to Data Mining. Addison-Wesley, 2006, ISBN-13: 978-0-321-32136-7, 769pp.
- Jiawei Han and Micheline Kamber: Data Mining: Concepts and Techniques (2nd edition). Morgan Kaufmann Publishers, Elsevier Inc., 2006, ISBN 13: 978-1-55860-901-3, 743pp.
- Ethem Alpaydin: Introduction to Machine Learning. The MIT Press, 2004, ISBN: 0-262-01211-1, 415pp.
- F. Provost, D. Jensen, T. Oates: Efficient Progressive Sampling, KDD-99, San Diego CA USA, ACM 1999.
- Lindsay I Smith: A tutorial on principal components analysis, 2002.

Thanks for Your attention!

Questions?

horvath@ismll.de