# Statistisches Data Mining (StDM)
# Woche 3

Zurich University
of Applied Sciences

*Oliver Dürr*
Institut für Datenanalyse und Prozessdesign
Zürcher Hochschule für Angewandte Wissenschaften

oliver.duerr@zhaw.ch
Winterthur, 4 Oktober 2016

# No laptops, no phones, no problems



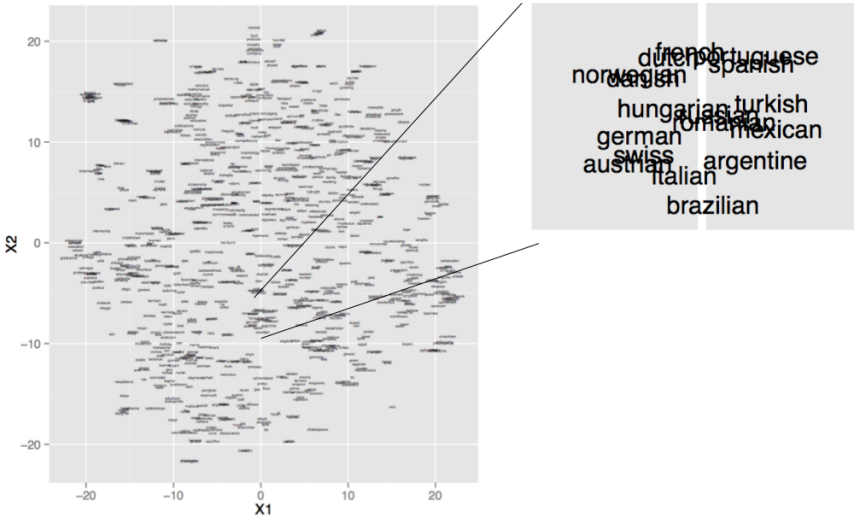
**Multitasking senkt Lerneffizienz:**

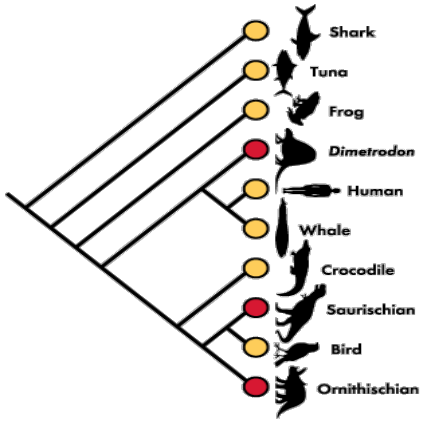
- **Keine Laptops im Theorie-Unterricht Deckel zu oder fast zu (Sleep modus)**

# Overview of the semester

**Part I (Unsupervised Learning)**

- Dimension Reduction
  - PCA
- Similarities, Distance between objects
  - Euclidian, L-Norms, Gower,…
- Visualizing Similarities (in 2D)
  - MDS, t-SNE
- Clustering
  - K-Means
  - Hierarchical Clustering



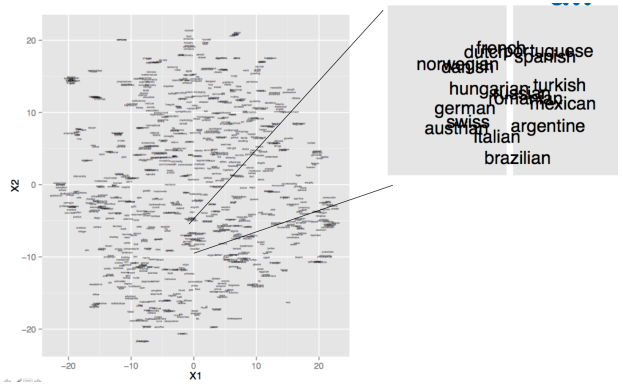



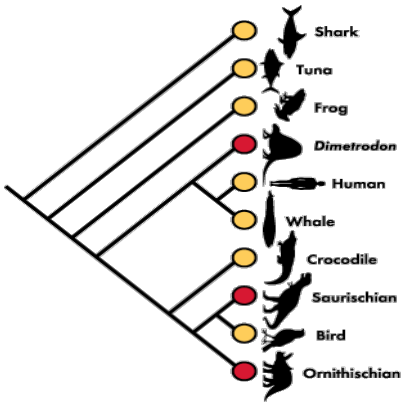

**Part II (Supervised Learning)**

- …

# Overview: Unsupervised learning

- Methods to visualize Data (dimension reduction in metric spaces)
  - PCA
- Distances
  - **Definition of distances**
  - **Euclidean and Minkowski Distance**
  - Binary Data
  - Categorical Data
  - Mixed data types
- Methods to visualize distance (in 2D)
  - Multidimensional Scaling (MDS)
  - Linear Metric MDS
  - Non-Linear Metric MDS
  - [isoMDS]
  - t-SNE
- Clustering approaches
  - Grouping of data
- Skript Andreas Ruckstuhl



# (Dis-)similarities / Distance

Pairs of Objects:

| | |
|---|---|
| Similarity | (large ⇒ similar), vague definition |
| **Dissimilarity** | (small ⇒ similar), Rules 1-3 |
| **Distance,Metric** | (small ⇒ similar), Rule 4 in addition |

Rules

1. $d(x, y) \geq 0$ (*non-negativity*, or separation axiom)
2. $d(x, y) = 0$ if and only if $x = y$ (*identity of indiscernibles*, or coincidence axiom)
3. $d(x, y) = d(y, x)$ (*symmetry*)
4. $d(x, z) \leq d(x, y) + d(y, z)$ (*subadditivity / triangle inequality*).

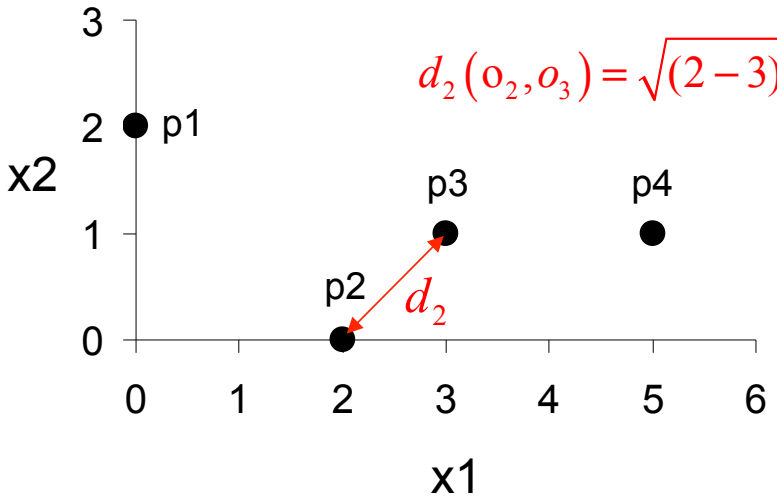
Examples of metrics (more follow with the examples)

- Euclidian and other Lp-Metrics
- Jaccard-Distance ( 1 - Jaccard Index)
- Graph Distance (shortest-path)

# Euclidean Distance and its Generalization

2D example
(2 feature per observation)

| obs | x1 | x2 |
|---|---|---|
| o1 | 0 | 2 |
| o2 | 2 | 0 |
| o3 | 3 | 1 |
| o4 | 5 | 1 |




$$d_2(o_2, o_3) = \sqrt{(2-3)^2 + (0-1)^2} = \sqrt{2}$$

- Distance between observations $\mathbf{o}_i$, $\mathbf{o}_j$ p features describing each observation
- **Eucledian Distance** for 2 observations $\mathbf{o}_i$, $\mathbf{o}_j$, described by n numeric feature:

$$d_2(o_i, o_j) = \sqrt{\sum_{k=1}^{p} (o_{ik} - o_{jk})^2}$$

- **Minkowski Distance** as generalization

$$d_r(o_i, o_j) = \left( \sum_{k=1}^{p} | o_{ik} - o_{jk} |^r \right)^{\frac{1}{r}}$$

# Distance matrix

As discussed on the last couple of slides there are different possibilities to determine the pair-wise distance between two observations $o_i$ and $o_j$.

$$d\left(o_i, o_j\right) = d_{ij}$$

We can collect all these pair-wise distance $d_{ij}$ in a distance matrix:

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & . & . & d_{1n} \\ d_{21} & d_{22} & . & . & d_{2n} \\ . & & . & & . \\ . & & & . & . \\ d_{n1} & d_{n2} & . & . & d_{nn} \end{bmatrix}$$

Symmetry:

$$d_{ij} = d\left(o_i, o_j\right) = d\left(o_j, o_i\right) = d_{ji}$$

All diagonal elements are 0!

$$d\left(o_k, o_k\right) = d_{kk} = 0$$

# Principle Idea (it's all about compromise)

- Have data in high dimensional space with distance, (e.g. 99 features)

**or (➔)**

- Have distances / dissimilarities $d_{ij}$ between many objects (e.g. 100 Objects)

- Draw this in low dimensional space (2, 3)

$$d_{ij} \rightarrow d^*_{ij} = \left|\left|\vec{y}_i - \vec{y}_j\right|\right|^2$$

High Dimensional Space

Low 2,3 Dimensional Space (Euklidean)

- The distances in (low-D) $d^*_{ij}$ should match the original ones $d_{ij}$ (high-D) as "**good as possible"**

# Classical Metric Scaling MDS

- Classical MDS. Formulation as minimisation of a cost function.
- In R: cmdscale()

$$\text{Cost} = \sum_{i<j} (d_{ij} - d^*_{ij})$$

$$d_{ij} = \| x_i - x_j \|^2$$

$$d^*_{ij} = \| y_i - y_j \|^2$$

Euclidean Distances also in high-D

Remarks

- Fast, “based on linear algebra”
- Only distances are needed as input (as all MDS methods)
- The formulation as a cost function is valid for Euclidian distances only (internally Eigenvalues are used)
- If other distances (besides Euclidian) are taken but nothing is guaranteed. Usually works if they are „mildly non-euclidean“, i.e. air-distances between cities on a Swiss map (small country, curvature of earth plays a minor role)
- Non-Euclidean distances -> negative Eigenvalues
- For Euclidean distances, classical MDS is equivalent to PCA (but conceptually different)

# End of Recap

# Evaluation: how good is the reduction

- Explained variance
  - If original distances are Euclidean, then Eigenvalues λ are positive
  - If Eigenvalues are too negative other methods might be better (see below)
  - Goodness of Fit using m-dimensions

$$P_m^{(1)} = \frac{\sum_{i=1}^{m} |\lambda_i|}{\sum_{i=1}^{n} |\lambda_i|}$$

Similar to PCA (explained variance) but absolute values
Values above 0.8 are good

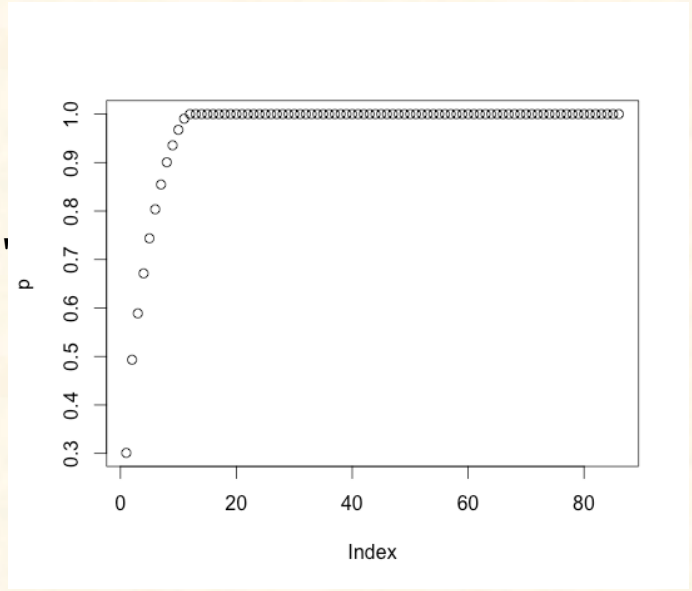
- Shepard Diagram
  - Simply plot distances between points in High Dim, vs Low Dim
  - Works also for other methods

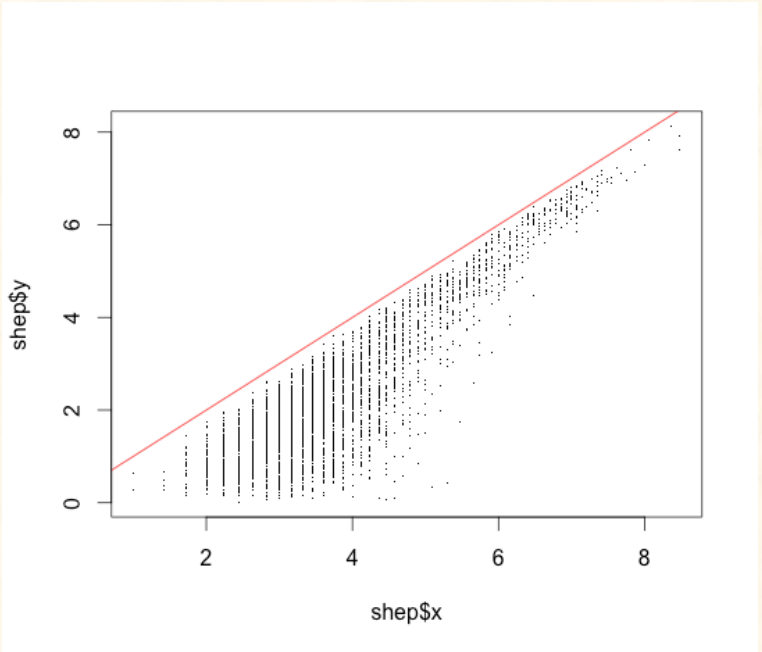
# Evaluation: in R

```
d <- dist(whiskies.f, method = 'man') #method = '
res = cmdscale(d)
plot(res, pch='.')
text(res, rownames(whiskies.f), cex=0.5)

# Plotting the eigenvalue
r = cmdscale(d, eig = TRUE)
min(r$eig) #-1.649809e-14, -164(euklidian, manhattan)
p = (cumsum(abs(r$eig)) / sum(abs(r$eig)))
plot(p)
abline(v=12)
# Why is p[12] == 1 for Euklidean
p[11:14]

# Shepard Diagram (Princip)
library(MASS)
shep <- Shepard(d, r$points) #MASS
plot(shep, pch='.')
abline(a=0,b=1, col='red')
# **REPEAT** and change to manhattan
```



# Take home message (Metrical MDS / PCA)

- PCA and metrical MDS are equivalent, if original distances are taken in Euclidean Space
- Metric MDS needs only distances
- Metric MDS OK (kind of) for “mild” non-Euclidean distances

- What if you have non-Euclidean distances?

# Sammon Mapping

- Sammon Mapping. Formulation as minimisation of a cost function.
- In R: MASS:sammon()

$$\text{Cost} = \frac{1}{\sum_{i<j} d_{ij}} \sum_{i<j} \frac{(d_{ij} - d^*_{ij})^2}{d^*_{ij}}$$

Just for normalization

$d_{ij}$ distance in orginial

$d^*_{ij}$ distance (euklidian) in low-D

Remarks

- More importance on small distances
- Slower, numerical optimization
- The formulation as a cost function is defining quantity (no need that $d_{ij}$ are Euclidean)
- Arbitrary distances as input

For comparison (cmdscale)

$$\text{Cost} = \sum_{i<j} (d_{ij} - d^*_{ij})$$

# Goodness of fit for Sammon mapping

- The cost function is also called stress

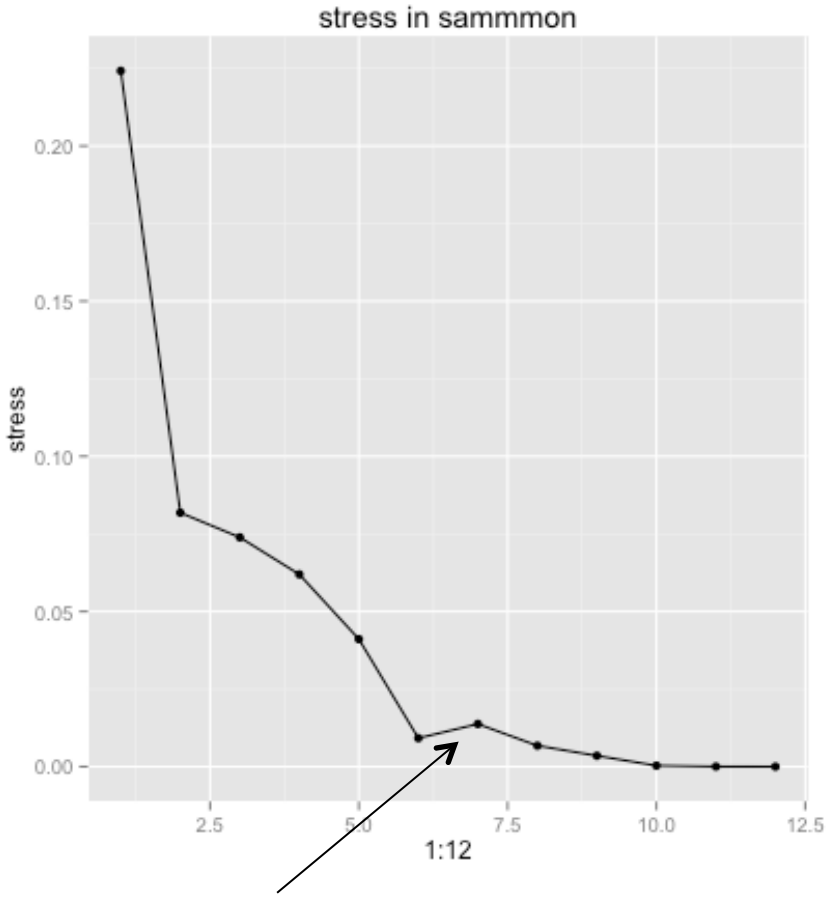
$$\text{Cost} = \frac{1}{\sum_{i<j} d_{ij}} \sum_{i<j} \frac{(d_{ij} - d^*_{ij})^2}{d^*_{ij}}$$

$d_{ij}$ distance in orginial

$d^*_{ij}$ distance (euklidian) in low-D

```
stress = NULL
for (i in 1:12) {
  res <- sammon(d, k = i)
  stress[i] = res$stress
}
```

Stress as a function of dimension




Numerical instability

# Kruskal's Non-metric Multidimensional (isoMDS)

- Formulation as minimisation of a cost function.
- In R: MASS:isoMDS() do not confuse with isoMAP

$$\text{Cost} = \frac{1}{\sum_{i<j} d^{*}{}_{ij}{}^{2}} \sum_{i<j} (\theta(d_{ij}) - d^{*}{}_{ij})^{2}$$

Just for normalization

$\theta(d_{ij})$ monoton trafos of dissimilarities in orginial

$d^{*}{}_{ij}$ distance (euclidian) in low-D

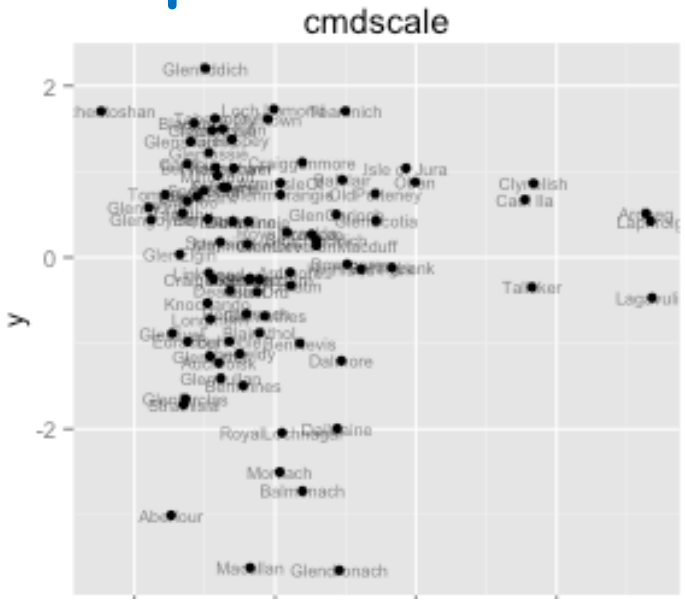
$$d_{ij} \leq d_{k\ell} \Longrightarrow \hat{d}_{ij} \leq \hat{d}_{k\ell}$$

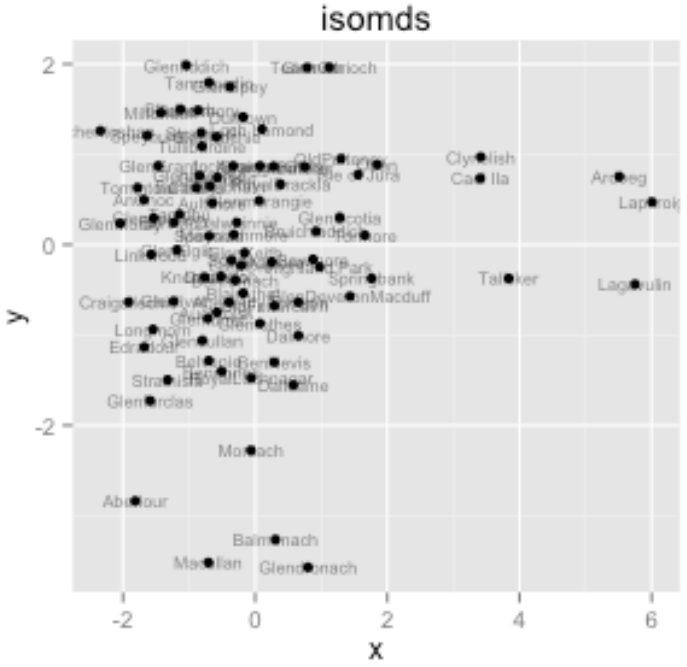
Remarks

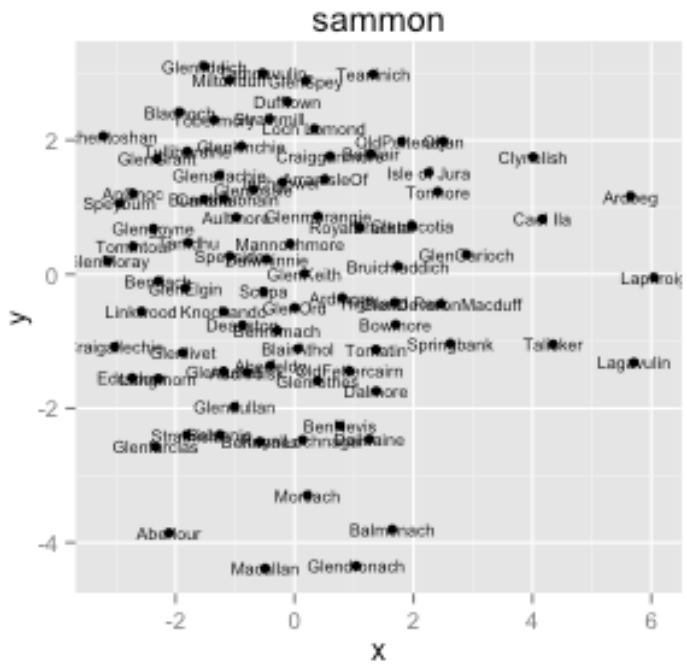
- Slower, numerical optimization
- The formulation as a cost function is defining quantity (no need that $d_{ij}$ are Euclidean)
- Arbitrary distances / dissimilarities as input

Try to best preserves the **rank** ordering of the distances between observation, good for ordinal dissimilarities.

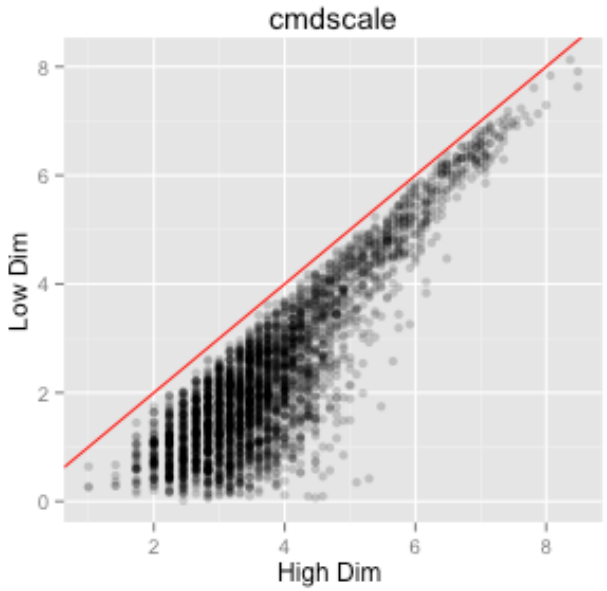
# Comparison

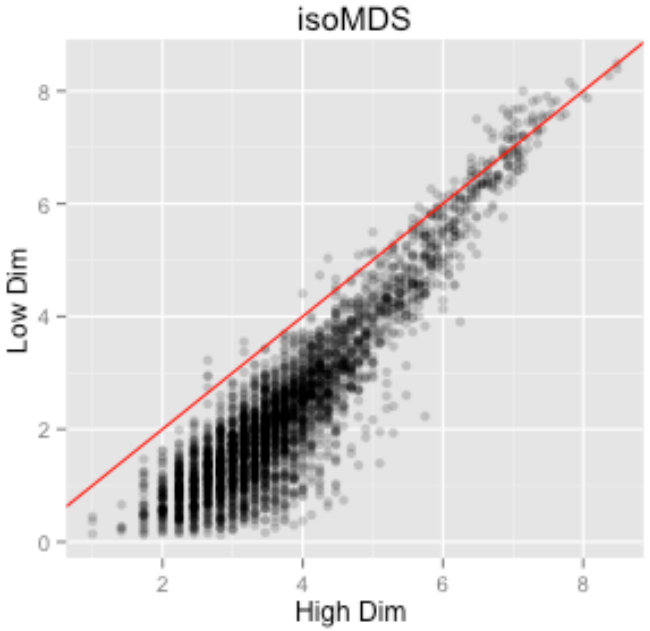


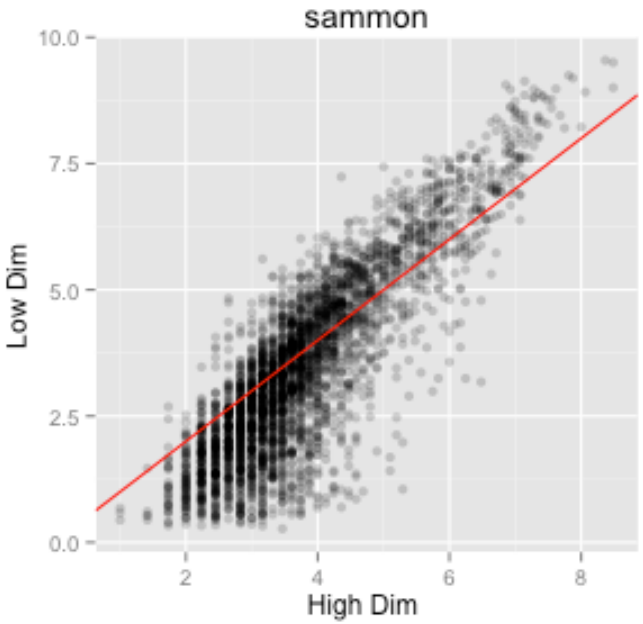



# Comparison




- Below Diagonal (low dimensional space more compact)
- Sammon puts more weights on short distances

# Take Home Message

- All methods can be understood by minimizing a cost function
  - isoMDS and sammon are iterative procedures. cmdscale use fast linear algebra.

- For isoMDS (not for Sammon) only the order matters. Taking an order preserving transformation of the distances as e.g. the log leads to no changes.

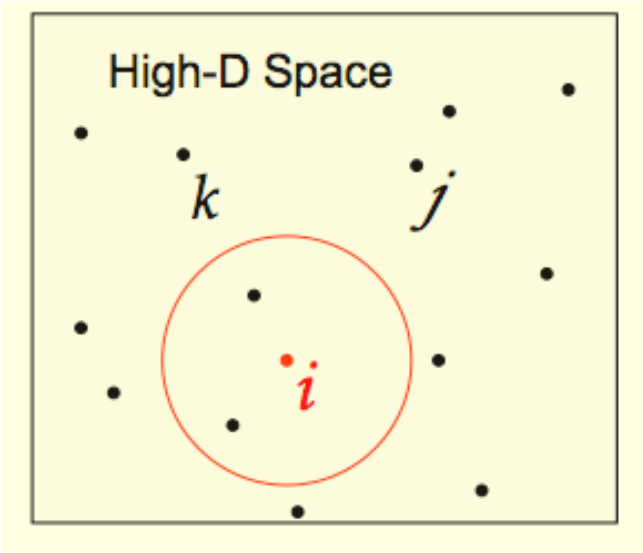
- Evaluation of performance
  - Shepard diagram and/or
  - Stress (isoMDS and sammon) or Explained Variance (cmdscale)
  - **Practical advice, use all methods and choose the best performing one.**

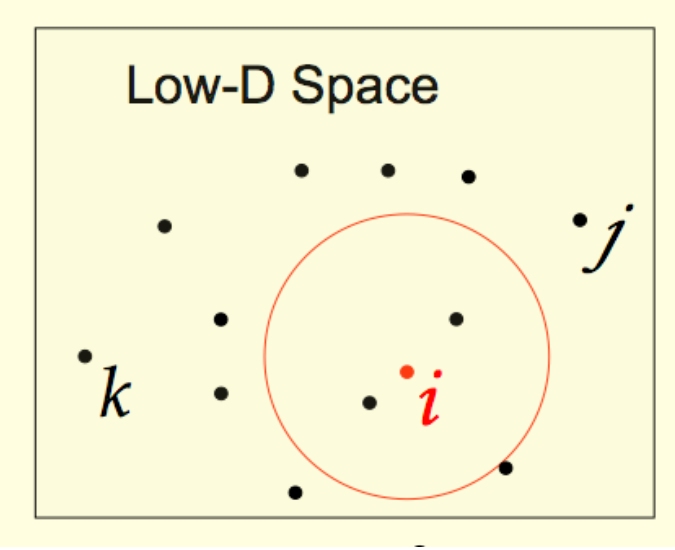
# Stochastic Neighbour Embedding (SNE)
# The new kid on the block

# tSNE: Line of thought

- L.J.P. van der Maaten and G.E. Hinton (2008)
  - Theory a bit shacky but works well in practice
  - Standard plot for the Machine Learning guys
- The exact structure of the data in the high-dimension space cannot be transferred to the low dimensional embedding
- It's sufficient to model the distance / dissimilarities (as in MDS)
- Proxy similarity by (conditional) probabilities of one point being the neighbour of another point in high and low dim space. These probabilities are parameterized using a Gaussian (high dim) and t-Distribution (low dimensional) space
- Maximize similarity of both distributions (Kullback-Leibler divergence)
- More information: http://lvdmaaten.github.io/tsne/

# SNE: Details




$$p_{j|i} = \frac{e^{-d_{ij}^2 / 2\sigma_i^2}}{\sum_k e^{-d_{ik}^2 / 2\sigma_i^2}}$$

probability of picking j given that you start at i

$\sigma_i$ is selected so that the number of neighbours is about constant (parameter perplecity)

$$q_{j|i} = \frac{e^{-d^{*2}_{ij}}}{\sum_k e^{-d^{*2}_{ik}}}$$

probability of picking j given that you start at i

In tSNE this is t-distribution

# Cost function

For every point i, we get two conditional distributions P_i (in the high-dimensional space) and Q_i (in the low-dimensional space) over all other points.

$$Cost = \sum_i KL(P_i \,||\, Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

- KL is standard, when comparing two distributions
- Solution placement in low-D space so that Cost is minimal

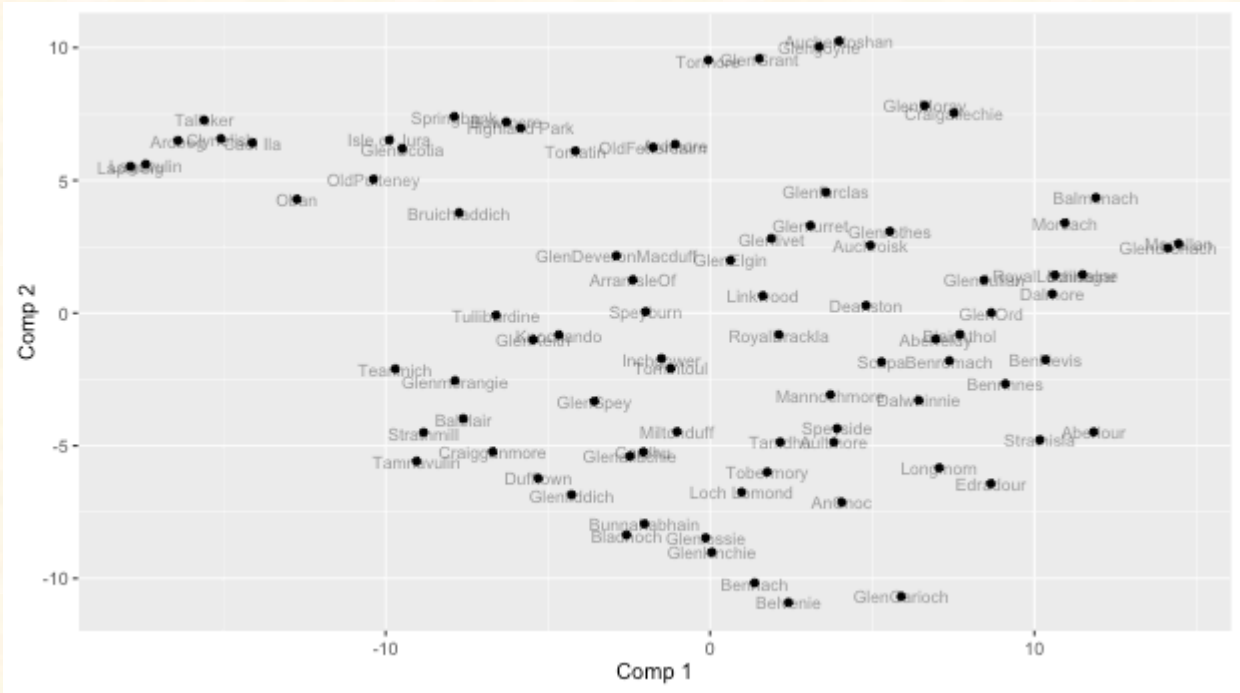
- For points where $p_{ij}$ is large and $q_{ij}$ is small we lose a lot.
  - Nearby points in high-D really want to be nearby in low-D
  - But not so extreme as in Sammon MDS

- For points where $q_{ij}$ is large and $p_{ij}$ is small we lose a little because we waste some of the probability mass in the Qi distribution.
  - Widely separated points in high-D have a mild preference for being widely separated in low-D

# tSNE in R

```
# Much faster due to Barnes Hut
library(Rtsne)
resTSNE <- Rtsne(d, perplexity = 10)
x = resTSNE$Y[,1]
y = resTSNE$Y[,2]

# Slower implementation but allows callbacks
library(tsne)
resTSNE <- tsne(d, perplexity = 10)
```

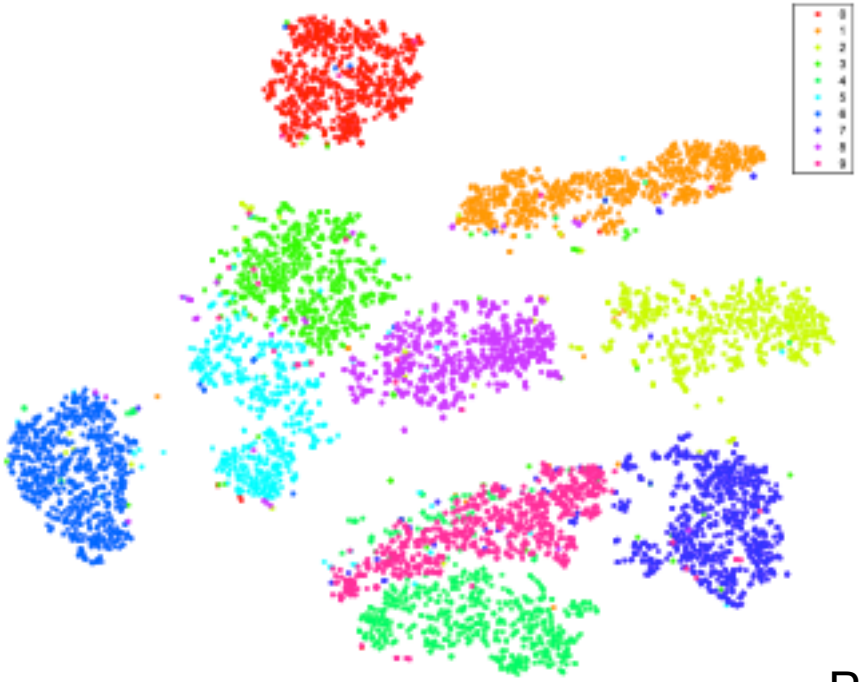


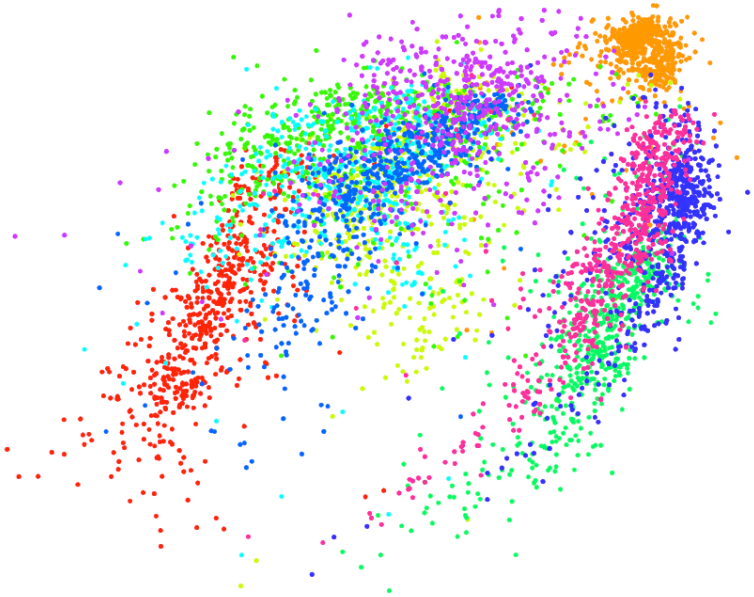
# Animations

- Nice blog post http://colah.github.io/posts/2014-10-Visualizing-MNIST/
  - Read as homework!

- PCA, Sammon's Mapping and tSNE

# Results of tSNE vs isoMAP on MNIST

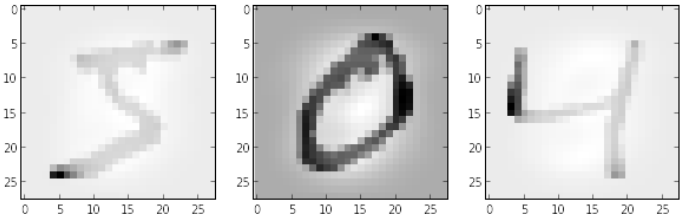




Plots are done as follows.

1. PCA to reduce dimension to 30 ➔!
2. Calculate Euclidean Distance

See http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf



Examples from the MNIST data set

# Back to dissimilarities, now for non-continuous data

# Recap

- So far we had Euclidean and the Minkowski Generalizations for continuous data

$$d_r\left(o_i, o_j\right) = \left( \sum_{k=1}^{p} | o_{ik} - o_{jk} |^r \right)^{\frac{1}{r}}$$

Rest of this lecture

| $X_1$ | $X_2$ | $X_3$ |
|---|---|---|
| 2.5 | 3.4 | 1.6 |
| 4.3 | 5.3 | 5.3 |
| 6.3 | 9.4 | 8.9 |

Use correlation

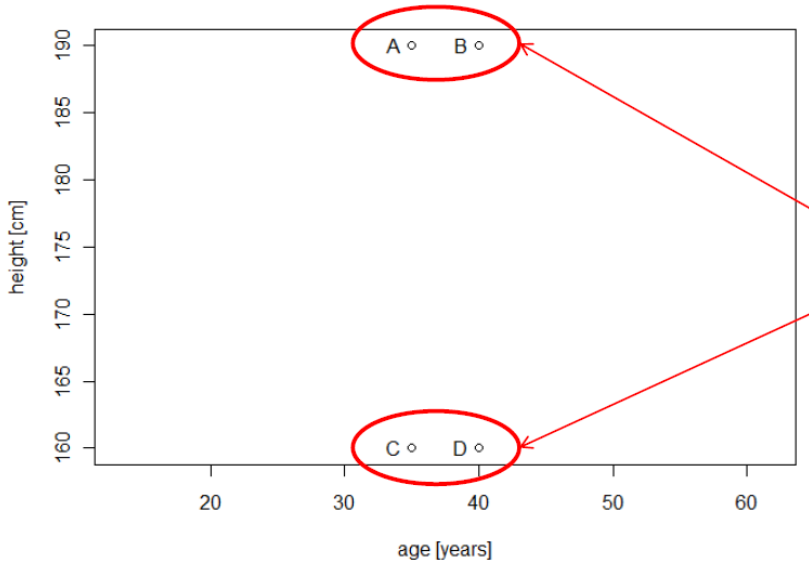
$$d(X_i, X_j) = \frac{1 - Cor(X_i, X_j)}{2}$$

Slide, taken from M. Kalish ETH, and also the following ones.

# Distances depend on scaling

unscaled

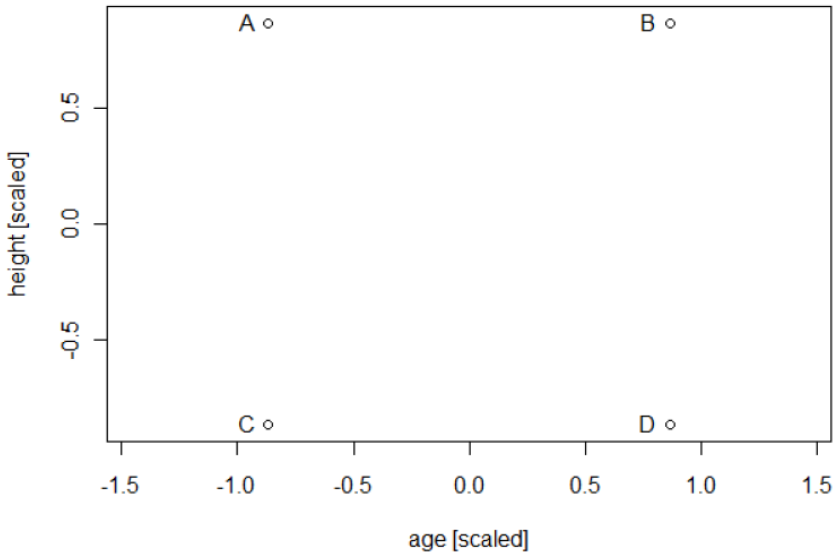
| Person | Age [years] | Height [cm] |
|---|---|---|
| A | 35 | 190 |
| B | 40 | 190 |
| C | 35 | 160 |
| D | 40 | 160 |




2 subgroups
(A,B) , (C,D)

scaled

| Person | Age [scaled] | Height [scaled] |
|---|---|---|
| A | -0.87 | 0.87 |
| B | 0.87 | 0.87 |
| C | -0.87 | -0.87 |
| D | 0.87 | -0.87 |




No subgroups
anymore

Source: M. Kalisch

# Categorical Data

# Similarity measures between binary vectors

$$o_1 = (o_{11}, o_{12}, ...., o_{1p}) \text{ and } o_2 = (o_{21}, o_{22}, ...., o_{2p})$$

Common situation is that objects, $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$, have only binary attributes, like for example gender (f/M), driving license (yes/no), Nobel price holder (yes/no). We code them with 0 and 1.

We distinguish between symmetric and asymmetric binary variables.

In symmetric binary variable both levels have roughly comparable frequencies
example: gender

In asymmetric binary variable both levels have very different frequencies
many 0s few 1s.
Example: Nobel price holder = 1. It's nothing discriminative to be not a noble price winner. Therefore 0 don't count.

# Matching Coefficient
# Similarity measures for "symmetric" binary vectors

$$o_1 = (o_{11}, o_{12}, ...., o_{1p}) \text{ and } o_2 = (o_{21}, o_{22}, ...., o_{2p})$$

The objects, $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$, have only binary attributes

Compute for the symmetric binary variable (could be only a subset of the p binary variables) the **S**imple **M**atching **C**oefficients:

**SMC = # matches / # attributes**

Corresponding to the proportion of matching features over **all** features

$$\mathbf{SMC} = (\mathbf{M_{11}} + \mathbf{M_{00}}) / (\mathbf{M_{01}} + \mathbf{M_{10}} + \mathbf{M_{11}} + \mathbf{M_{00}})$$

$M_{01}$ = the number of attributes where $o_{1i}$ was 0 and $o_{2i}$ was 1
$M_{10}$ = the number of attributes where $o_{1i}$ was 1 and $o_{2i}$ was 0
$M_{00}$ = the number of attributes where $o_{1i}$ was 0 and $o_{2i}$ was 0
$M_{11}$ = the number of attributes where $o_{1i}$ was 1 and $o_{2i}$ was 1

1 – SMC is a distance

# Matching Coefficient
# Similarity measures for "asymmetric" binary vectors

$$o_1 = (o_{11}, o_{12}, ...., o_{1p}) \text{ and } o_2 = (o_{21}, o_{22}, ...., o_{2p})$$

The objects, $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$, have only binary attributes

Compute for the asymmetric binary variable (could be only a subset of the p binary variables) the **J**accard Coefficient:

**J = # both-1-matches / # of not-both-zero attributes values**

Corresponding to the proportion of matching features over **those** features which are 1 in at least one of both observations. "0's don't count'

$$\mathbf{J = (M_{11}) / (M_{01} + M_{10} + M_{11})}$$

$M_{01}$ = the number of attributes where $o_{1i}$ was 0 and $o_{2i}$ was 1
$M_{10}$ = the number of attributes where $o_{1i}$ was 1 and $o_{2i}$ was 0
$M_{11}$ = the number of attributes where $o_{1i}$ was 1 and $o_{2i}$ was 1

1 – J is a distance (Jaccard Distance)

# Example: How similar are two given binary vectors?

$p$ = 1 0 0 0 0 0 0 0 0 0
$q$ = 0 0 0 0 0 0 1 0 0 1

$M_{01}$ = 2 (the number of attributes where p was 0 and q was 1)
$M_{10}$ = 1 (the number of attributes where p was 1 and q was 0)
$M_{00}$ = 7 (the number of attributes where p was 0 and q was 0)
$M_{11}$ = 0 (the number of attributes where p was 1 and q was 1)

$SMC = (M_{11} + M_{00})/(M_{01} + M_{10} + M_{11} + M_{00})$

$SMC = (M_{11} + M_{00})/(M_{01} + M_{10} + M_{11} + M_{00}) = (0+7) / (2+1+0+7) = 0.7$

$J = (M_{11}) / (M_{01} + M_{10} + M_{11})$

$J = (M_{11}) / (M_{01} + M_{10} + M_{11}) = 0 / (2 + 1 + 0) = 0$

# More than 2 levels (Nominal Data)

- Simple matching coefficient
- mm: Number of variables in which object i and j mismatch
- p number of mismatch

$$d(i,j) = \frac{mm}{p}$$

Proportion of variables, in which people disagree

- Character strings can be understood as nominal data.
- Also called Hamming-Distance if with strings of same length (sometimes w/ o dividing by p)
- What is the Hamming-Distance between:
  - `HOUSE`
  - `MOUSE`

# Gower's dissimilarity for mixed data types

Idea: Use distance measure $d_{ij}$ between 0 and 1 for each variable or feature:

- $k_{th}$ variable is binary, nominal: Use discussed methods, e.g.

$$d_{ij}^{(k)} = \frac{M_{11} + M_{00}}{M_{01} + M_{10} + M_{11} + M_{00}}$$

- $k_{th}$ variable is numeric:

  $x_{ik}$: Value for object i in variable k
  $R_k$: Range of variable k for all objects

$$d_{ij}^{(k)} = \frac{|x_{ik} - x_{jk}|}{R_k}$$

- Ordinal: Use ranks, then same like with numeric variables

Aggregate distance measures over all variables/feature/dimensions:

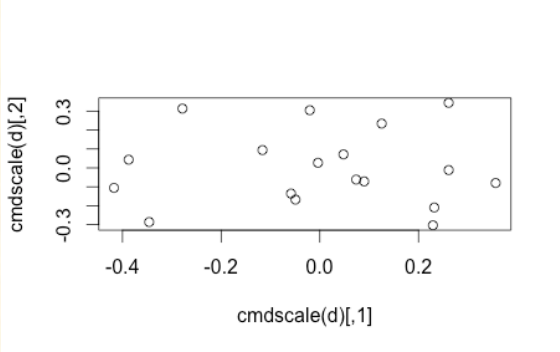
$$d_{ij} = \frac{1}{p} \sum_{k=1}^{p} d_{ij}^{(k)}$$

# Dissimilarity for mixed data types with R-function "daisy" calculating Gower's dissimilarity

```
> str(flower)
'data.frame':  18 obs. of  8 variables:
 $ V1: Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 2 2 ...
 $ V2: Factor w/ 2 levels "0","1": 2 1 2 1 2 2 1 1 2 2 ...
 $ V3: Factor w/ 2 levels "0","1": 2 1 1 2 1 1 1 2 1 1 ...
 $ V4: Factor w/ 5 levels "1","2","3",..: 4 2 3 4 5 4 4 2 3 5 ...
 $ V5: Ord.factor w/ 3 levels "1"<"2"<"3": 3 1 3 2 2 3 3 2 1 2 ...
 $ V6: Ord.factor w/ 18 levels "1"<"2"<"3"<..: 15 3 1 16 2 12 ...
 $ V7: num  25 150 150 125 20 50 40 100 25 100 ...
 $ V8: num  15 50 50 50 15 40 20 15 15 60 ...

> dist=daisy(flower, type=list(asymm=c(1, 3), symm=2, ordratio=7))
> str(dist)
Classes 'dissimilarity', 'dist'  atomic [1:153] 0.901 0.618 ...
  ..- attr(*, "Size")= int 18
  ..- attr(*, "Metric")= chr "mixed"
  ..- attr(*, "Types")= chr [1:8] "A" "S" "A" "N" ...

> plot(cmdscale(d))
```



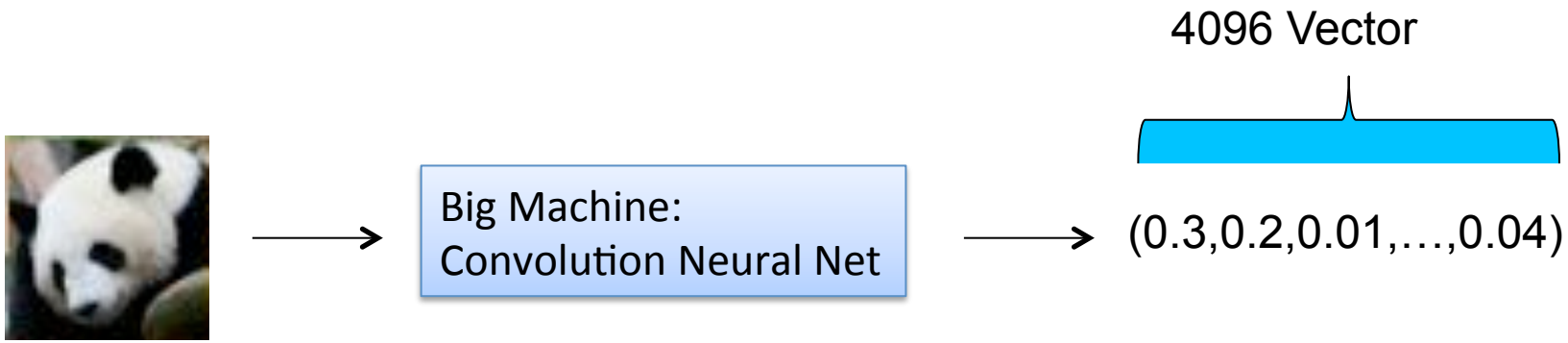
# Special Distances

# String Distances

- Package stringdist
  - Hamming Distance (see before)
    - Just count the number of mismatches
  - More distances
    - Longest common substring
    - …
- Many distances used in bioinformatics comparing DNAs
- Or use embeddings like word2vec
  - String (complicated procedure) → 100 (or so) dimensional vector
  - Calculate Euclidean Distance

# Distances between images [just for completeness]

- Dump approach use Euclidean distance between pixel values directly.
- Current state of the art, for image similarities: convolutional neural network




See: http://cs.stanford.edu/people/karpathy/cnnembed/

**Semantic similarities not pixel wise!**

See: http://cs.stanford.edu/people/karpathy/cnnembed/ for more images