

Machine Learning

6 – Dimension Reduction

SS 2018

Gunther Heidemann

I. Basics of dimension reduction

1. Introduction: How high dimensionality arises, features of high dimensionality
2. Principal component analysis (PCA)
3. Nonlinear extension of PCA

II. Dimension reduction for visualization

1. Basic visualizations of high dimensional data: Scatterplot, glyphs
2. Projection pursuit
3. Multidimensional scaling

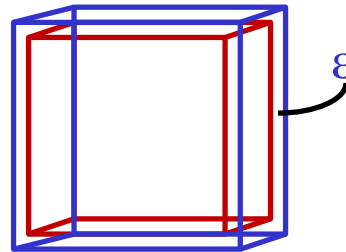
- Data mining and pattern recognition often lead to high dimensionality:
 - HD-image: $1920 \times 1080 \times 3 = 6220800$ data dimensions.
 - Medical diagnosis from 10^1 to 10^3 parameters.
- A problem sufficiently described by few variables (low dimensionality) may be turned into a problem of high dimension by adding
 - redundant variables (duplicates or combinations of the original ones),
 - variables containing unnecessary information,
 - variables containing noise.

- Combinatorial explosion: For d variables with n values each, there are n^d combinations (= volume of the resulting space).
- n^d points are necessary to sample a d -dimensional space of continuous variables with n sampling points on each axis. Thus, dense sampling becomes impossible for large d .
- By the same argument, any real data set will not be able to “fill” a space of high dimension.
- Pairs of vectors chosen at random from a space of high dimension are likely to
 - have similar distances,
 - be close to perpendicular to each other.
- Example: Bertillonage employs 11 anthropometric measures to achieve close to perfect discrimination of humans.

Surface layer of unit cube in \mathbb{R}^d :

Blue: unit cube

Red: (unit $- 2\varepsilon$) cube



Surface layer has thickness ε .

Volume ratio of red and blue cube: $(1 - 2\varepsilon)^d / 1^d = (1 - 2\varepsilon)^d$.

Examples for $\varepsilon = 1\%$:

$d = 20 \rightarrow (1 - 2\varepsilon)^d = 67\%$ of volume is inside the inner cube,

$d = 300 \rightarrow (1 - 2\varepsilon)^d = 0,23\%$ of volume is inside the inner cube.

For high dimension, almost the entire volume is in a thin surface layer.

This effect is often illustrated by comparing a unit sphere to a unit cube:

There is almost no volume in the sphere compared to the cube because the entire volume is “in the corners”.

Example:

Answers to a questionnaire given by a test person are represented as numerical values and combined to a vector.

The more questions in the questionnaire, the more likely it becomes that each test person gives an extreme answer to at least one question.

Example:

Make a full HD 24 bit RGB movie at 25 fps, starting at the big bang until today. You get

$$25 \text{ frames/sec} \cdot 60 \text{ sec/min} \cdot 60 \text{ min/h} \cdot 24 \text{ h/d} \cdot 365 \text{ d/y} \cdot 13,7 \cdot 10^9 \text{ y} \approx \underline{1,08 \cdot 10^{19} \text{ frames}}.$$

Assume all frames are different, then how much of the image space do these frames cover?

$$\text{Volume of image space: } (2^{3 \cdot 8})^{1920 \cdot 1080} = 10^{3 \cdot 8 \cdot 1920 \cdot 1080 \cdot \log_{10} 2} \approx \underline{10^{14981179}}.$$

$$\text{Percentage covered} \approx \underline{10^{-14\,981\,158} \%}.$$

This applies to random images. As images are similar in many aspects, they cover only a **tiny subspace** of the image space.

Descriptive dimensionality:

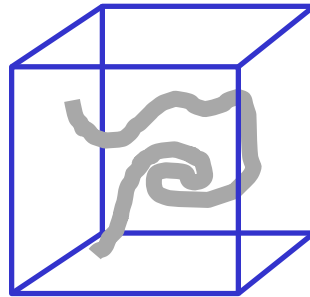
- # parameters used in unprocessed data set.
- Depends on the type of representation (e.g., 640x480 pixels vs. 1920x1080 pixels).

Intrinsic dimensionality:

- # independent parameters necessary to describe the data.
- Depends on problem, not the data representation (e.g., parameters to describe the trajectory of a car in a video are independent of video resolution).

Note some authors use *data dimensionality* in the sense of *intrinsic dimensionality*, others as *descriptive dimensionality*.

- Presence of hidden structure can be recognized from $d_{\text{intrinsic}} \ll d_{\text{description}}$.
- Data concentrate along a manifold of dimensionality much lower than the embedding representation space (manifold \approx space made from distorted pieces of an euclidean \mathbb{R}^n , manifold \neq subspace).



Example:

- Human hand has approx. 20 joints (description dimensionality = 20).
- The description space of a posture is thus 20-dimensional (one posture is a point in this space).
- Assume 3 different angles can be realized with each joint $\rightarrow 3^{20} \approx 3 \cdot 10^9$ different hand postures (human lifespan $\approx 2 \cdot 10^9$ seconds).
- Only a tiny subset of postures is realized since joint angles are highly correlated.
- This subset of postures is not randomly distributed in the joint angle space.
- Rather, postures are carried out on continuous trajectories in joint angle space.
- As a consequence, postures are on a continuous manifold.

Example:

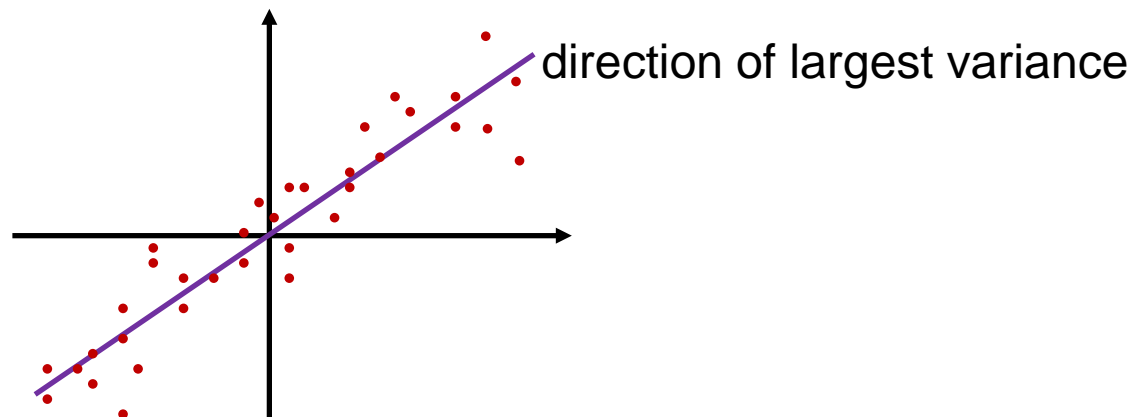
- Object under fixed illumination and with fixed background,
- viewed by a camera that can be moved freely around the object.
- 3 position + 3 pose parameters of the camera characterize an image uniquely.
- These 6 parameters control all pixel values of the image.
- Generation of the high dimensional data set is easy (move camera).
- Information (6 parameters) “spreads out” over all pixels.
- This makes parameter recovery from the pixel values alone very difficult.

- Find local dimensionalities of the data manifold.
- Find new coordinate system to represent the data manifold.
- Project data onto the new coordinate system.
- By this means, get rid of the empty parts of the space of description parameters.
- The new parameters may be more meaningful than the original description parameters.
- Example: Parameter of progress on trajectory of a closing hand instead of joint angles.

- Standard method of dimension reduction.
- Idea: Find the subspace that captures most of the data variance.
- Unsupervised learning.
- Given: Data set $D = \{\vec{x}_1, \vec{x}_2, \dots\}$, $\vec{x}_i \in \mathbb{R}^d$, with zero mean:

$$\langle \vec{x} \rangle = \sum_{i=1 \dots |D|} \vec{x}_i = 0.$$

- **PCA** finds $m < d$ orthonormal vectors $\vec{p}_1 \dots \vec{p}_m$ such that the \vec{p}_i are in the **directions of largest variance** of D .



[H]

- The vectors \vec{p}_i are the m eigenvectors of the autocorrelation matrix

$$C = (C_{ij}) = (\langle \vec{x} \vec{x}^T \rangle_{ij})$$

with largest eigenvalues λ_i :

$$C \vec{p}_i = \lambda_i \vec{p}_i$$

- Since C is symmetric and real, the λ_i are real and there is an orthonormal basis of eigenvectors $\vec{p}_i \vec{p}_j^T = \delta_{ij}$.
- Expansion of \vec{x}_i using of *all* eigenvectors would yield \vec{x}_i without error (provided all $\lambda_i \neq 0$):

$$\vec{x}_i = \sum_{j=1 \dots d} a_{ij} \vec{p}_j \quad \text{with}$$

$$a_{ij} = \vec{x}_i \cdot \vec{p}_j.$$

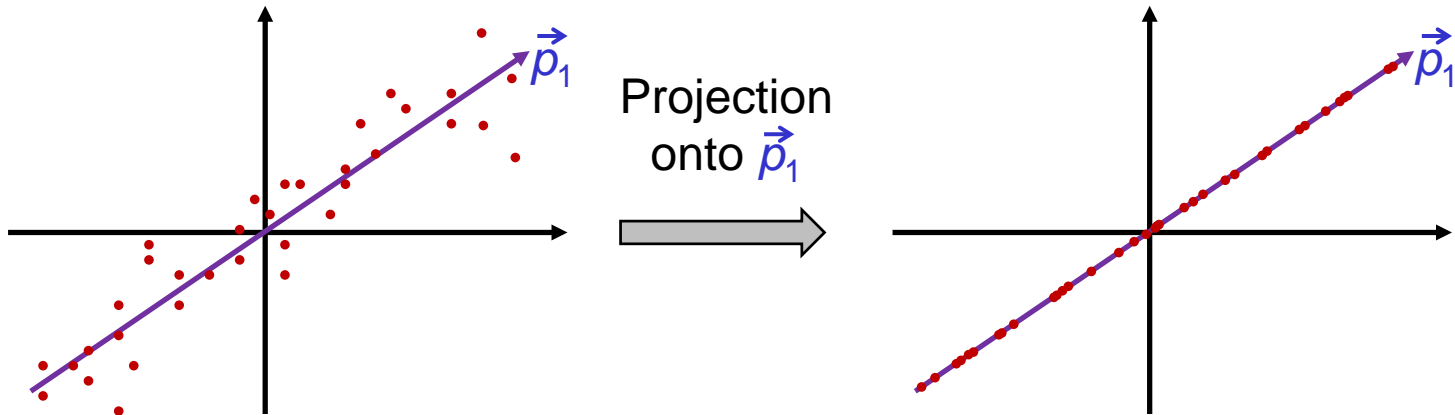
- To obtain an approximation \vec{z}_i of \vec{x}_i , only $m < d$ basis vectors are used (Karhunen-Loeve expansion):

$$\vec{x}_i \approx \vec{z}_i = \sum_{j=1 \dots m} a_{ij} \vec{p}_j$$

provided the eigenvectors are ordered such that $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$.

- The eigenvectors are called the *principal components*.
- They can be interpreted as *features* (pattern recognition), *receptive fields* (Neuroinformatics) or *filter kernels* (computer vision).
- Expansion after the $m < d$ eigenvectors of largest eigenvalues is the optimal linear method to minimize the mean square reconstruction error

$$E = \sum_{i=1 \dots |D|} (\vec{z}_i - \vec{x}_i)^2.$$



[H]

The coefficients

$$a_{ij} = \vec{x}_i \cdot \vec{p}_j$$

are pairwise uncorrelated and the eigenvalues are the variances:

$$\begin{aligned}
 1/|D| \sum_{k=1 \dots |D|} a_{ki} a_{kj} &= 1/|D| \sum_{k=1 \dots |D|} (\vec{p}_i^T \vec{x}_k) (\vec{p}_j^T \vec{x}_k) \\
 &= 1/|D| \sum_{k=1 \dots |D|} \vec{p}_i^T (\vec{x}_k \vec{x}_k^T) \vec{p}_j \\
 &= \vec{p}_i^T C \vec{p}_j \\
 &= \vec{p}_i^T \lambda_j \vec{p}_j \\
 &= \delta_{ij} \lambda_j.
 \end{aligned}$$

The approximation

$$\vec{z}_i = \sum_{j=1 \dots m} a_{ij} \vec{p}_j$$

is the projection of \vec{x}_i onto the m -dimensional subspace $\text{span}\{\vec{p}_1 \dots \vec{p}_m\}$ of \mathbb{R}^d .

The residuum is

$$\delta \vec{x}_i = \sum_{j=m+1 \dots d} a_{ij} \vec{p}_j.$$

The variance of the residuum over all data is

$$\begin{aligned} \sigma^2(\delta \vec{x}) &= 1/|D| \sum_{k=1 \dots |D|} (\delta \vec{x}_k)^2 \\ &= 1/|D| \sum_{k=1 \dots |D|} \sum_{i=m+1 \dots d} \sum_{j=m+1 \dots d} a_{ki} \vec{p}_i a_{kj} \vec{p}_j \\ &= \sum_{i=m+1 \dots d} \sum_{j=m+1 \dots d} \delta_{ij} \lambda_j \vec{p}_i \vec{p}_j \\ &= \sum_{i=m+1 \dots d} \lambda_i, \end{aligned}$$

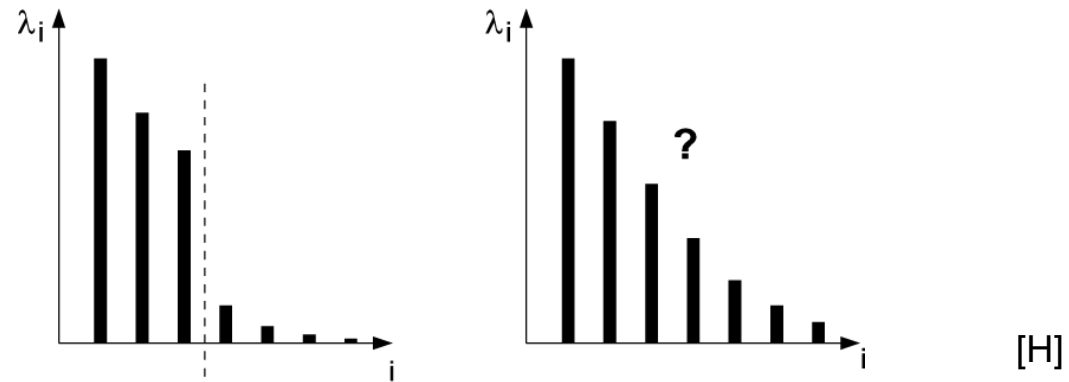
i.e., the mean approximation error is the sum of the left out eigenvalues.

Principal component analysis (PCA)

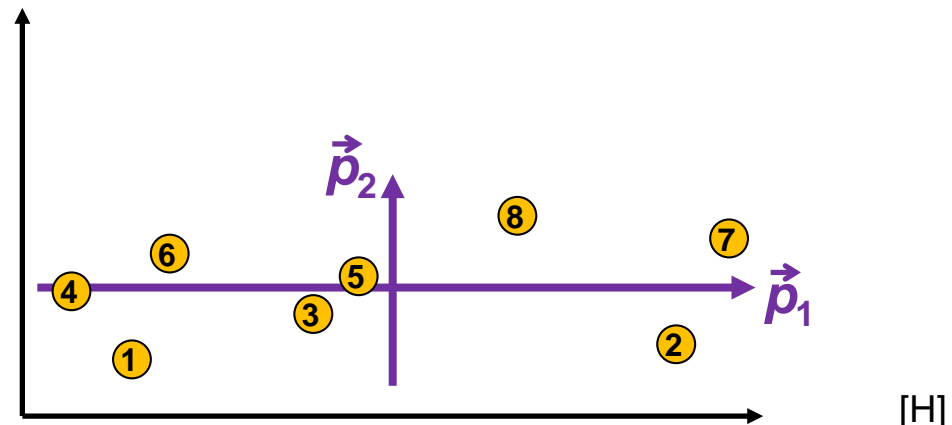
- How can we find a suitable number m of eigenvectors?

Have a look at the spectrum of eigenvalues !

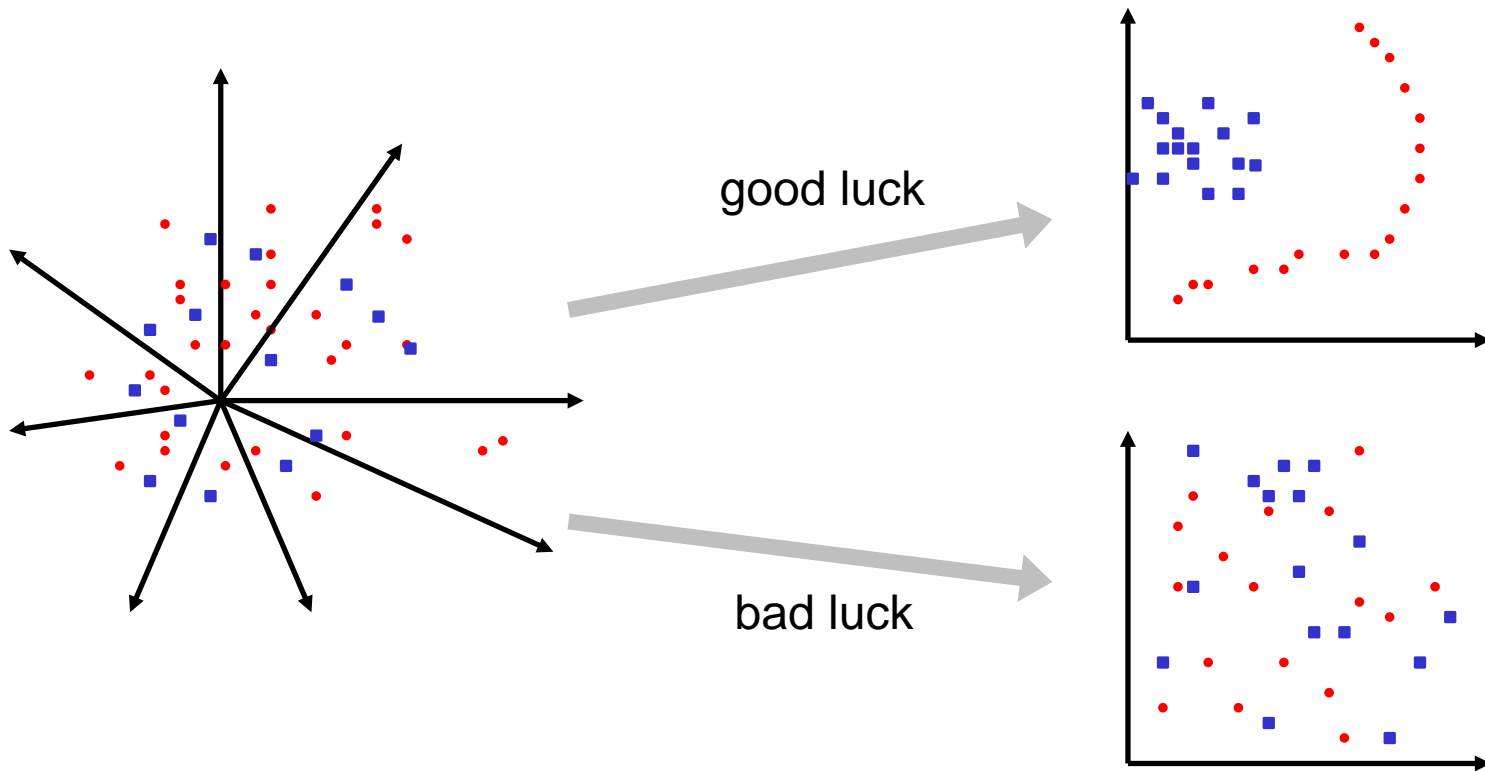
- We hope to find a distinct jump in the spectrum, indicating a suitable cut off number.



- But: A large eigenvalue may also be due to noise and/or inappropriate scaling !



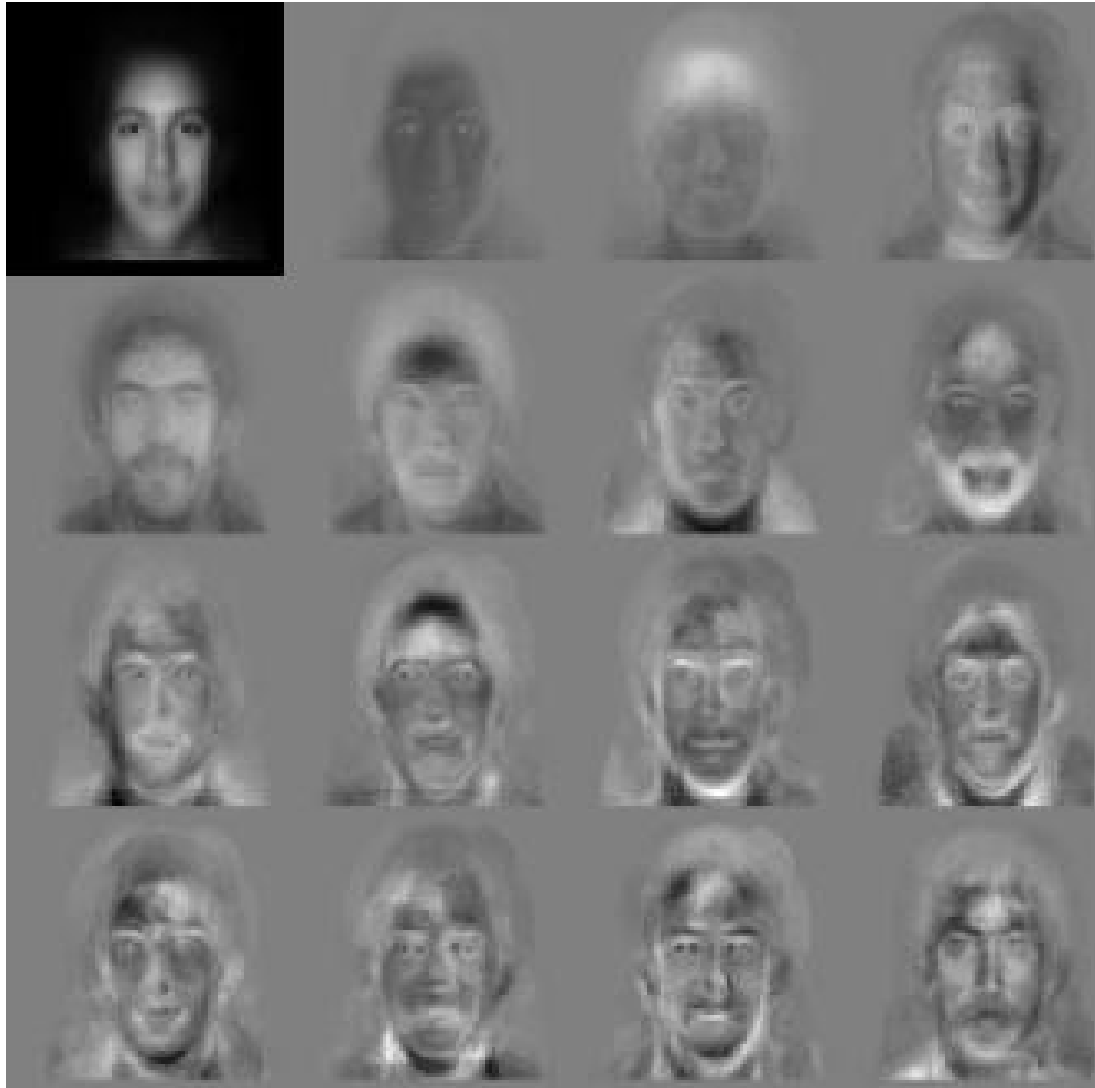
We also hope to obtain structure within the data in case there are classes (or other semantics):



[H]

Note PCA does not generate structure but only makes existing structure accessible.

- Classification of faces: Turk & Pentland, 1991, „Eigenfaces for Recognition“ [TP].
- Images of faces under normalized conditions: Fixed position, pose, illumination.
- Data vector \vec{x}_i is the collection of all pixel gray values of image i .
- Average face image is subtracted to obtain zero mean data vectors.
- Visualizing the eigenvectors of C in image space shows they have a face like structure → “eigenfaces”



The eigenvectors of a face database, ordered by decreasing eigenvalues.

Upper left:
The average face.

[TP]

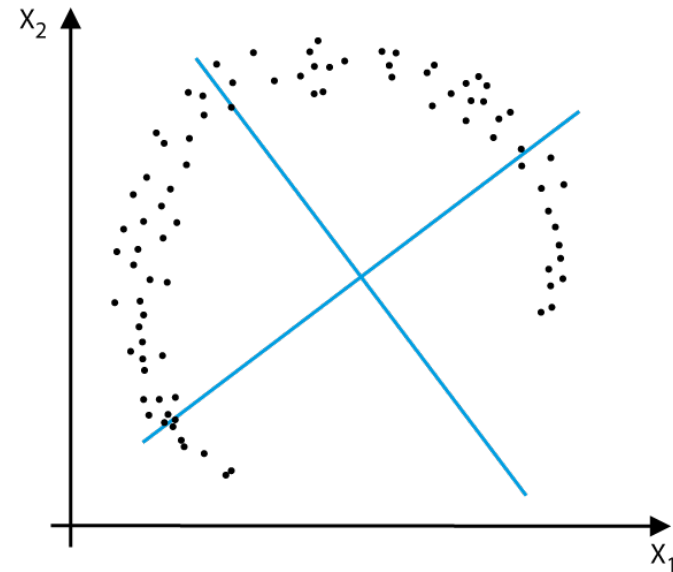
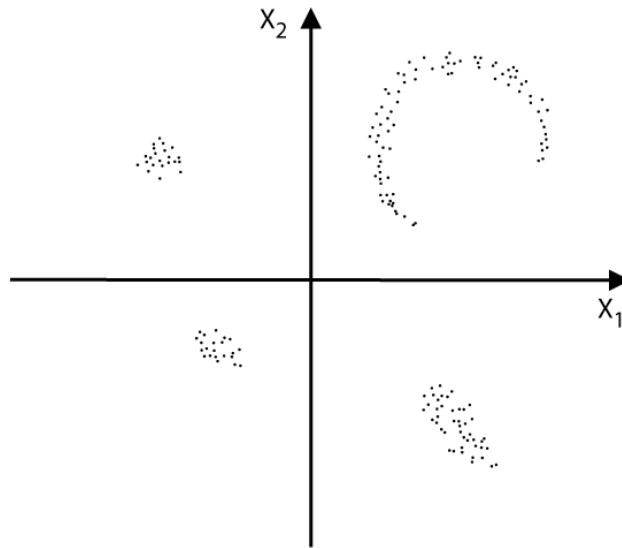
Recognition of a new image:

- Subtract average face image.
- Project onto principal components (eigenfaces):
 - Tremendous dimension reduction
 - Coefficients are robust features
- Compare coefficients with stored coefficients of training faces or feed to classifier.
- For recognition, faces must be recorded under the same fixed position, size (distance), viewpoint and illumination.

- PCA is most widespread and simple method for linear dimension reduction.
- Easy computation from eigenvectors of covariance matrix, high dimensionality may require special / approximative methods.
- Purely variance based method.
- Semantics of the eigenvectors (if any) must be found in subsequent analysis.
- Linear dimension reduction is inappropriate for
 - nonlinear distributions,
 - clustered distributions.

Examples:

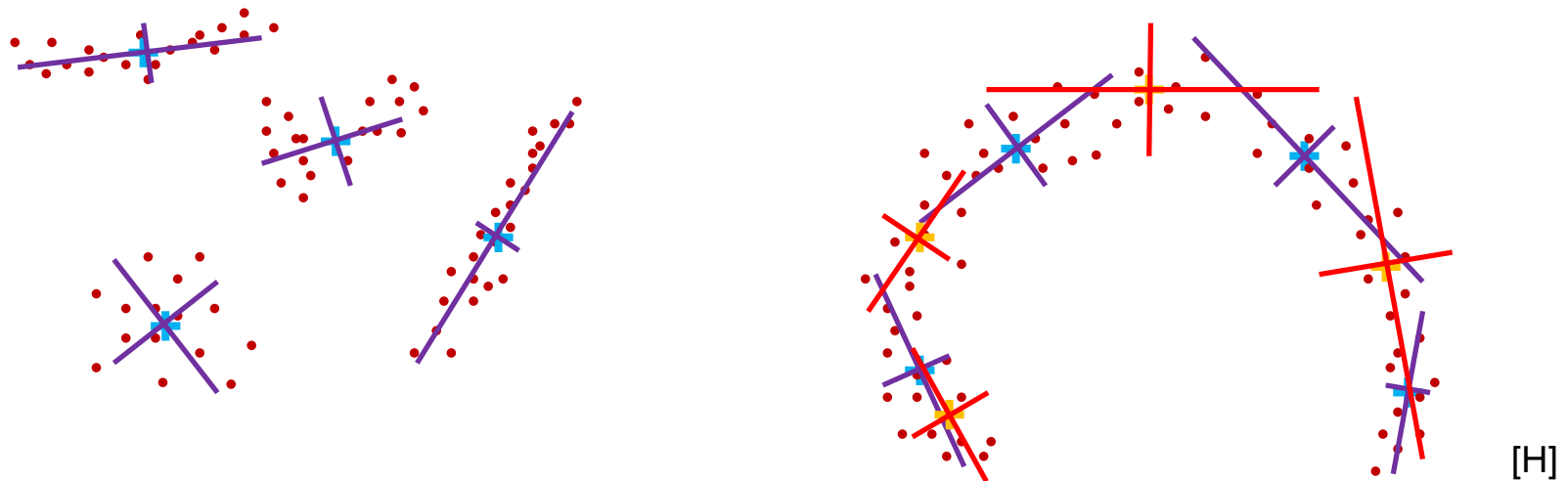
Both eigenvalues are large, but clearly the data does exhibit structure and does not “fill” the entire data space.



[H]

- Clustered distributions can be handled by **local PCA**.
- Nonlinear distributions can be handled by **principal curves**.

Find cluster centers, then compute PCA individually for each cluster.
Better: Iteratively improve position of cluster centers and local PCs.



- For continuous, nonlinear distributions (not clustered), local PCA does not yield a continuous description of the manifold.
- Further, **different clusterings** are possible on a continuous distribution, leading to entirely **different local projection** systems.

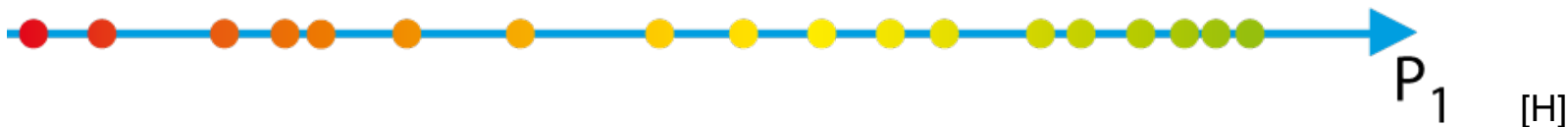
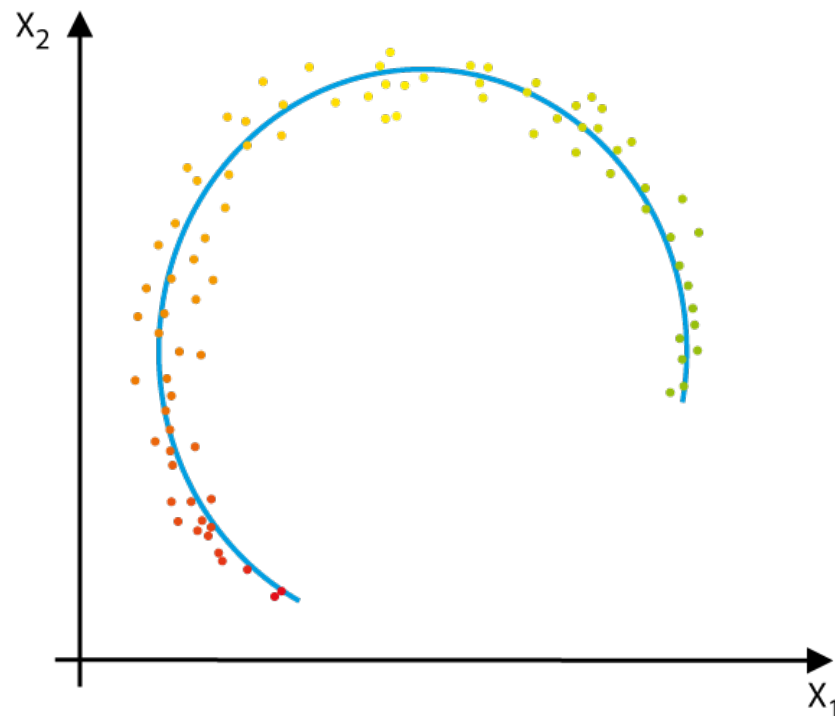
Principal curves / surfaces are a nonlinear extension of PCA.

Example in 2d (“banana”):

- Principal curve provides an approximation of the shape of the data distribution.
- Data are projected onto the principal curve to obtain a linear order.

Issues:

- Find appropriate dimensionality.
- Find appropriate “flexibility”, i.e., parameterization, of the fit function.



PCA:

Mean square error function of for a data set given by a probability density $P(\vec{x})$:

$$E = \frac{1}{2} \int (\vec{x} - \sum_{i=1 \dots m} a_i(\vec{x}) \cdot \vec{p}_i)^2 \cdot P(\vec{x}) d\vec{x}.$$

Approximate manifold by m vectors \vec{p}_i , find coefficients $a_i(\vec{x})$ for each \vec{x} .

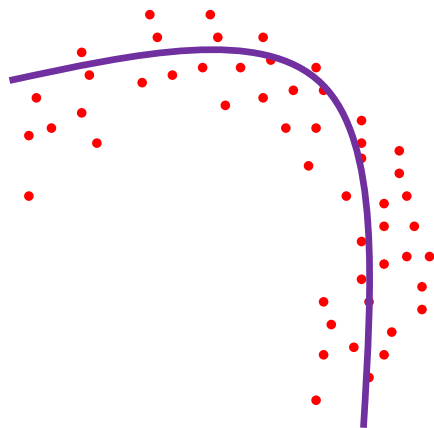
Principal curves:

Generalize approximation to a parameterized surface $\vec{X}(a_1 \dots a_m; \vec{w})$ of m dimensions. The vector $\vec{w} \in \mathbb{R}^n$ summarizes the n parameters which determine the shape of the surface and need to be found by minimizing

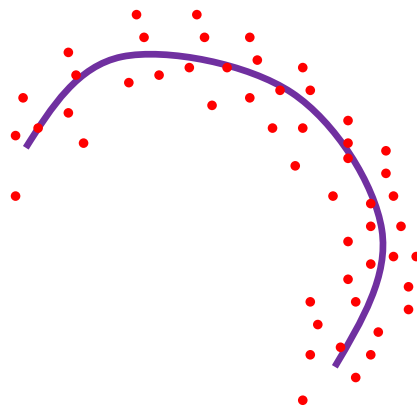
$$E = \frac{1}{2} \int (\vec{x} - \vec{X}(a_1(\vec{x}) \dots a_m(\vec{x}); \vec{w}))^2 \cdot P(\vec{x}) d\vec{x}.$$

The m parameters $a_i(\vec{x})$ determine the point on the surface of \vec{X} that best matches \vec{x} and have to be computed for each \vec{x} individually.

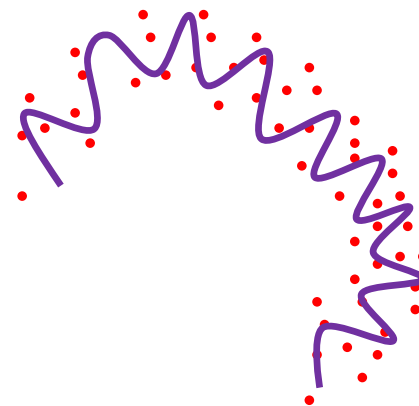
The number n of parameters \vec{w} is responsible for the ability of \vec{x} to fit a manifold.



Small n :
Underfit



Reasonable n



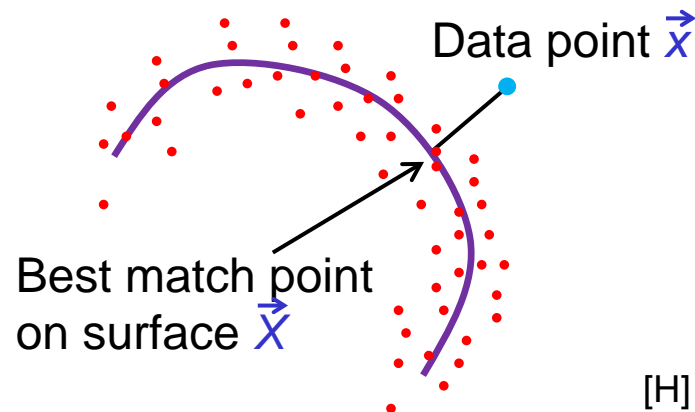
Too large n :
Overfit

[H]

For example, n might be the degree of a polynomial.

To ensure a good fit, additional **smoothness constraints** should be used.

The m parameters $a_i(\vec{x})$ determine the point on the surface of \vec{X} that best matches \vec{x} .



In the 2d example, there is only one parameter a_1 ($m = 1$) as \vec{X} is a curve.

$a_1(\vec{x})$ indicates, e.g., the way from the end of the curve to the best match point for \vec{x} .

The parameters \vec{w} can be found for $P(\vec{x})$, e.g., by gradient descent:

$$\partial E / \partial \vec{w} = - \int (\vec{x} - \vec{X}(a_1(\vec{x}) \dots a_m(\vec{x}); \vec{w})) \cdot \partial / \partial \vec{w} \vec{X}(a_1(\vec{x}) \dots a_m(\vec{x}); \vec{w}) \cdot P(\vec{x}) d\vec{x}.$$

A downhill step is

$$\Delta \vec{w} = -\varepsilon \partial E / \partial \vec{w}$$

with stepsize ε .

Problem: Each step requires integration over *all* data.

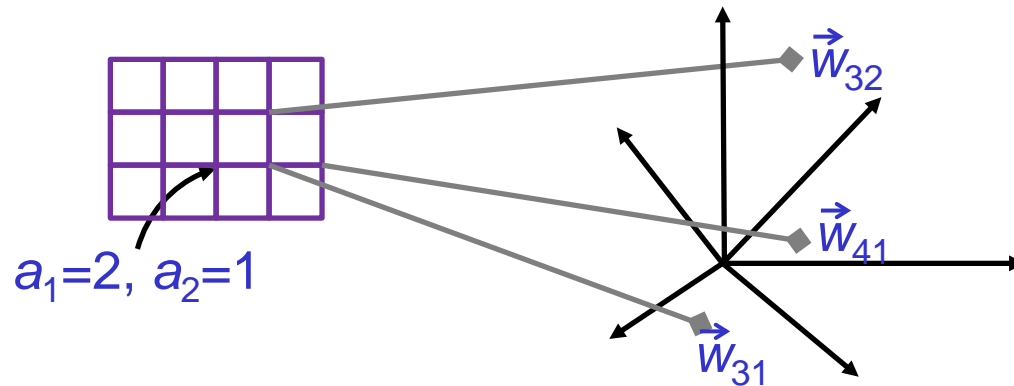
Solution: **Stochastic approximation**

Make downhill step only with respect to a single sample:

$$\Delta \vec{w} = \varepsilon (\vec{x} - \vec{X}(a_1(\vec{x}) \dots a_m(\vec{x}); \vec{w})) \cdot \partial / \partial \vec{w} \vec{X}(a_1(\vec{x}) \dots a_m(\vec{x}); \vec{w})$$

hoping that minimization for a succession of many samples with small stepsize results in minimization of E .

Now we make a special choice for the surface \vec{X} : A discrete **grid**.
Parameters a_i become a discrete grid index, e.g., for $m=2$:



[H]

The parameters \vec{w}_j determine the shape of the surface in the embedding space of high dimensionality. We define \vec{w}_{ij} to be the location of grid point (i, j) .

Mean square error for grid:

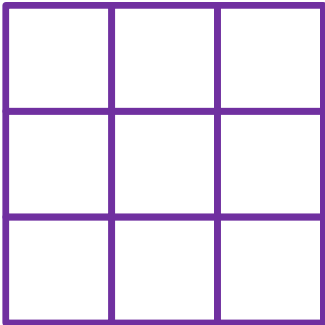
$$\begin{aligned} E &= \frac{1}{2} \int (\vec{x} - \vec{X}(a_1(\vec{x}) \dots a_m(\vec{x}); \vec{w}))^2 \cdot P(\vec{x}) d\vec{x} \\ &= \frac{1}{2} \sum_{i,j} \int_{F_{ij}} (\vec{x} - \vec{w}_{ij})^2 \cdot P(\vec{x}) d\vec{x} \end{aligned}$$

where F_{ij} is the Voronoi tessellation cell in data space belonging to grid point (i, j) :

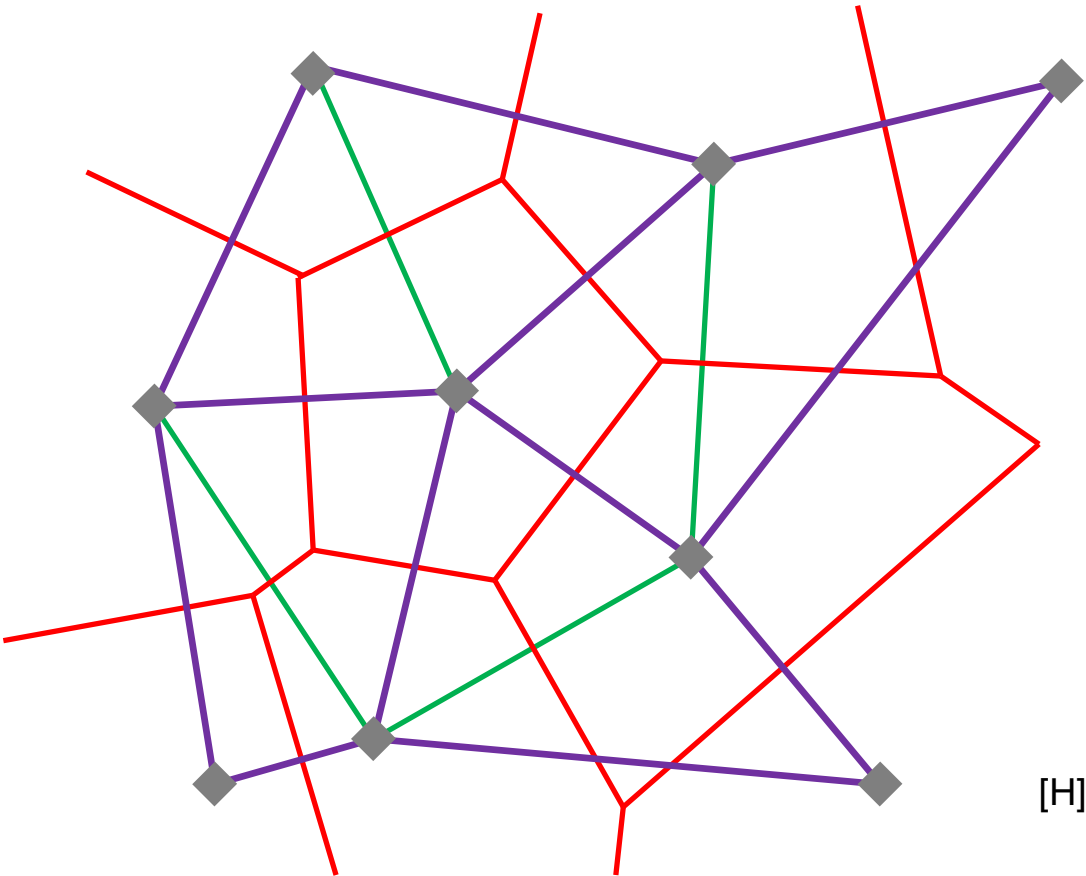
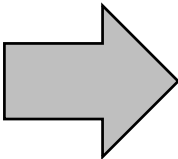
$$F_{ij} = \{ \vec{x} \mid \forall i', j' : |\vec{x} - \vec{w}_{ij}| \leq |\vec{x} - \vec{w}_{i'j'}| \},$$

i.e., all points \vec{x} in data space which are closer to \vec{w}_{ij} than to any other $\vec{w}_{i'j'}$.




So while for a given \vec{x} the best match point on \vec{X} had to be found, on a grid simply the closest grid point \vec{w}_{ij} needs to be found.



2d grid



2d grid in high dimensional space

-  \vec{w}_{ij}
-  Helper lines (Delaunay triangulation)
-  Boundaries of Voronoi tessellation cells

Minimization:

$$\begin{aligned}
 E &= \frac{1}{2} \sum_{i,j} \int_{F_{ij}} (\vec{x} - \vec{w}_{ij})^2 \cdot P(\vec{x}) \, d\vec{x} \\
 \partial E / \partial \vec{w}_{kl} &= - \int_{F_{kl}} (\vec{x} - \vec{w}_{kl}) \cdot P(\vec{x}) \, d\vec{x} \\
 \Delta \vec{w}_{kl} &= -\varepsilon \partial E / \partial \vec{w} = \varepsilon \int_{F_{kl}} (\vec{x} - \vec{w}_{kl}) \cdot P(\vec{x}) \, d\vec{x} \\
 &= \varepsilon \langle \vec{x} - \vec{w}_{kl} \rangle_{F_{kl}}
 \end{aligned}$$

where $\langle . \rangle_{F_{kl}}$ denotes averaging over the Voronoi cell F_{kl} .

Stochastic approximation:

$$\Delta \vec{w}_{kl} = \varepsilon (\vec{x} - \vec{w}_{kl}),$$

where kl is the index of the \vec{w} closest to \vec{x} .

So instead of averaging over the density $P(\vec{x})$, gradient descent steps are performed only for a single \vec{x} drawn from $P(\vec{x})$.

The adaptation

$$\Delta \vec{w}_{kl} = \varepsilon (\vec{x} - \vec{w}_{kl})$$

is a “winner takes all” rule since only the best match \vec{w} is adapted, the rest remains unchanged.

Instead of adapting only the best match \vec{w} , also other, neighboring \vec{w} can be adapted according to their distance to the location of the best match \vec{w} on the grid (not the distance in the embedding space):

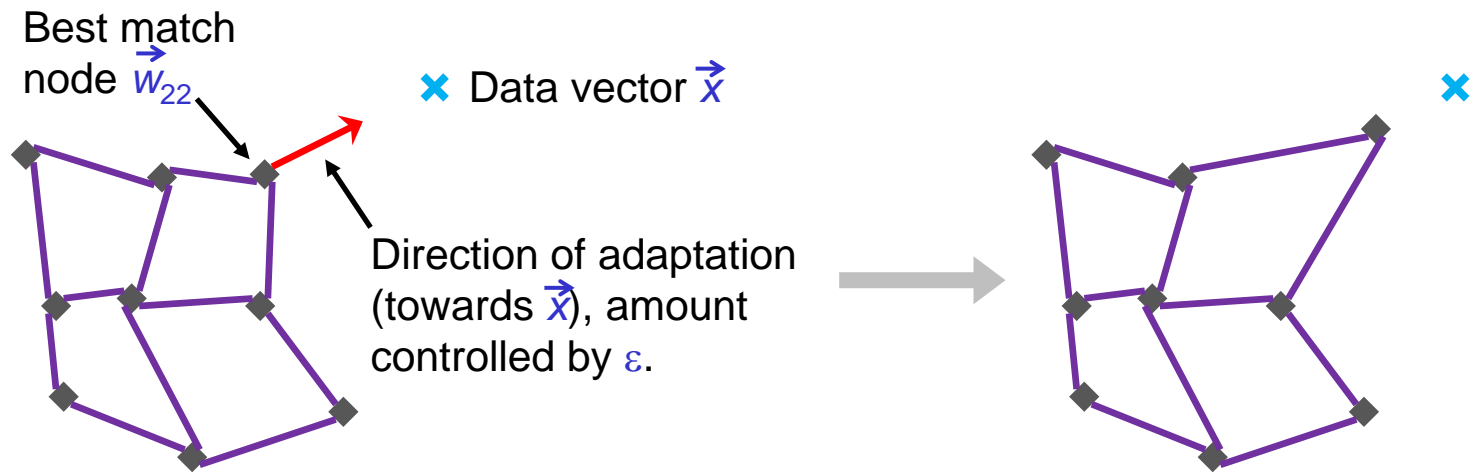
$$\Delta \vec{w}_{ij} = \varepsilon \exp\left(-((i-k)^2 + (j-l)^2)^{1/2} / 2\sigma^2\right) \cdot (\vec{x} - \vec{w}_{ij}).$$

This enforces smoothness and helps to avoid “meandering” overfits.

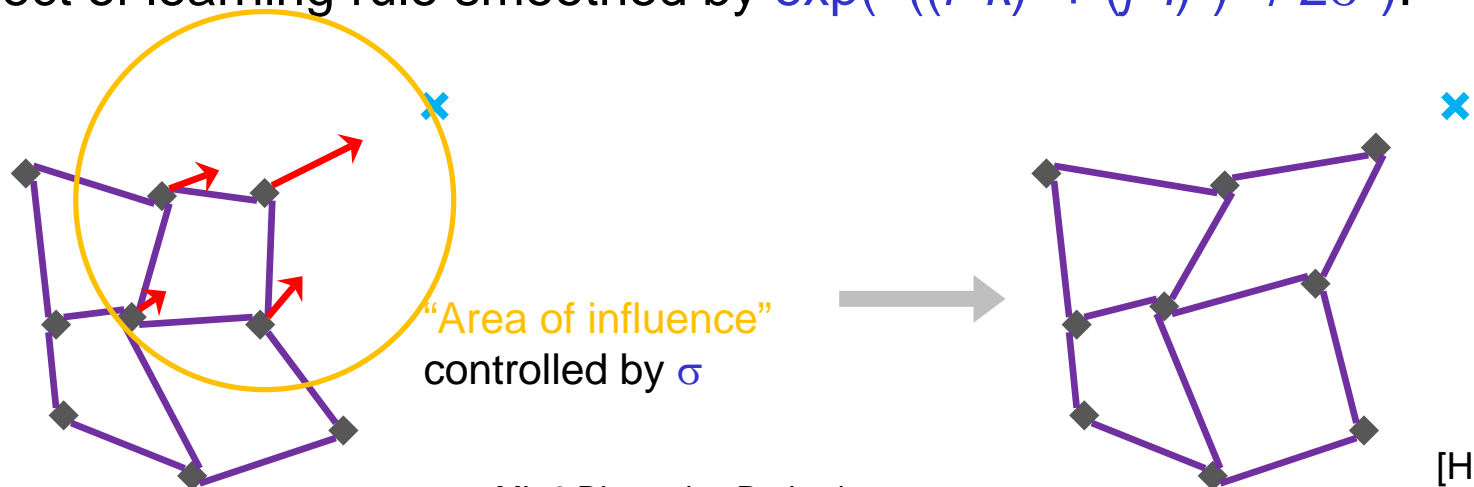
The parameter σ controls the range within which the “pull” on the best match \vec{w}_{kl} affects its grid neighbors.

The smoothed version is the **Kohonen adaptation rule** used for the training of **Self-Organizing Maps (SOMs)**.

Effect of learning rule $\Delta \vec{w}_{kl} = \varepsilon (\vec{x} - \vec{w}_{kl})$:

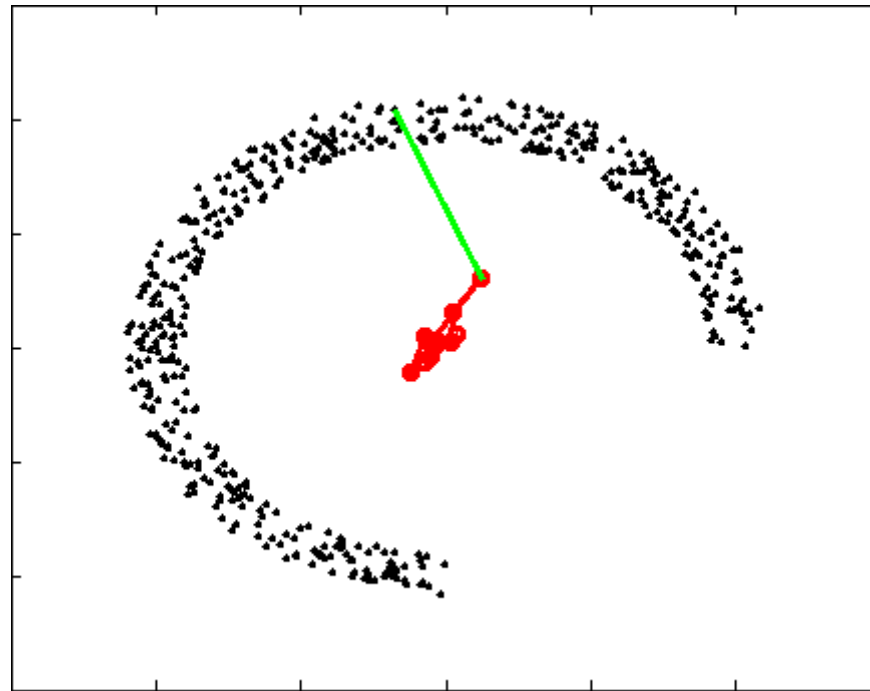


Effect of learning rule smoothed by $\exp(-((i-k)^2 + (j-l)^2)^{1/2} / 2\sigma^2)$:



To obtain fast and smooth adaptation,

- reduce step size ε over time, and
- start with large σ and reduce gradually.



[P]

Note the smoothing factor $\exp(-((i-k)^2 + (j-l)^2)^{1/2} / 2\sigma^2)$ was introduced on the level of the adaptation rule.

This is a pragmatic decision – the “principled” way would have been to add a smoothness constraint to the error function:

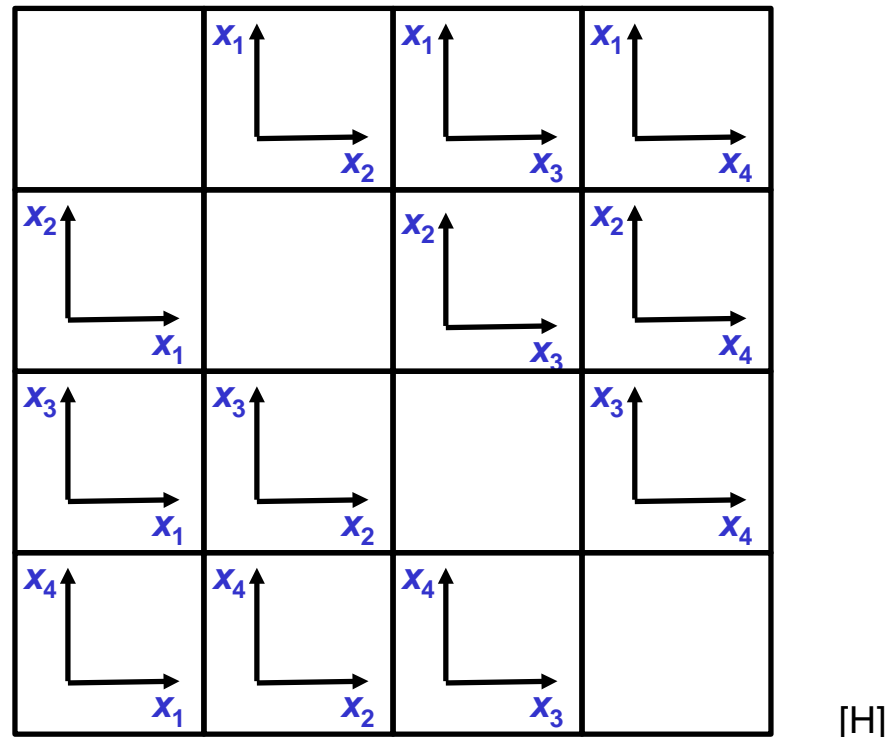
$$E = \frac{1}{2} \int (\vec{x} - \vec{X}(a_1(x) \dots a_m(x); \vec{w}))^2 \cdot P(\vec{x}) d\vec{x} + \text{“punishment for high curvature of } \vec{X}\text{”} .$$

- *Curse of dimensionality* leads to tremendous increase of the volume of a data space with increasing dimension.
- As a consequence, any real data set can not “fill” the data space.
- Usually, real data are not randomly distributed in the data space but form a manifold of (at least locally) smaller intrinsic dimensionality but complex shape.
- PCA is the standard method for dimensionality reduction:
 - Directions of maximum variance are eigenvectors of the covariance matrix, project onto these directions for dimension reduction.
 - Linear method, can not capture the structure of clustered or nonlinear data.
- Local PCA: Find clusters, analyse locally by PCA.
- Principal curves: Nonlinear extension of PCA. Eigenvectors are replaced by an arbitrary surface in the embedding data space.

Dimension reduction for visualization

- Fully automated analysis of high dimensional data is often not possible.
- Humans have highly developed pattern recognition abilities.
- Visualization is a huge field – only a brief overview of basic techniques for the problem of high dimensionality is given.
- Two major directions:
 - For not too high dimensionality (typically < 10), some methods allow display of the *complete* information (alternatively, some selected dimensions) in 2 or 3 dimensions.
 - For high dimensionality, reduction techniques such as PCA can be used to display a *projection* to 2 or 3 dimensions.
- PCA is the most well known projection technique but other methods are better suited to show structure.

Project on two of the dimensions and display all combinations as a matrix of 2d plots:



If a display of all axes is infeasible, PCA may yield suitable directions.

Classification of Iris sub-species

- Iris data set (Fisher, 1936)
- Three sub-species: *I. setosa*, *I. versicolor*, *I. virginica*

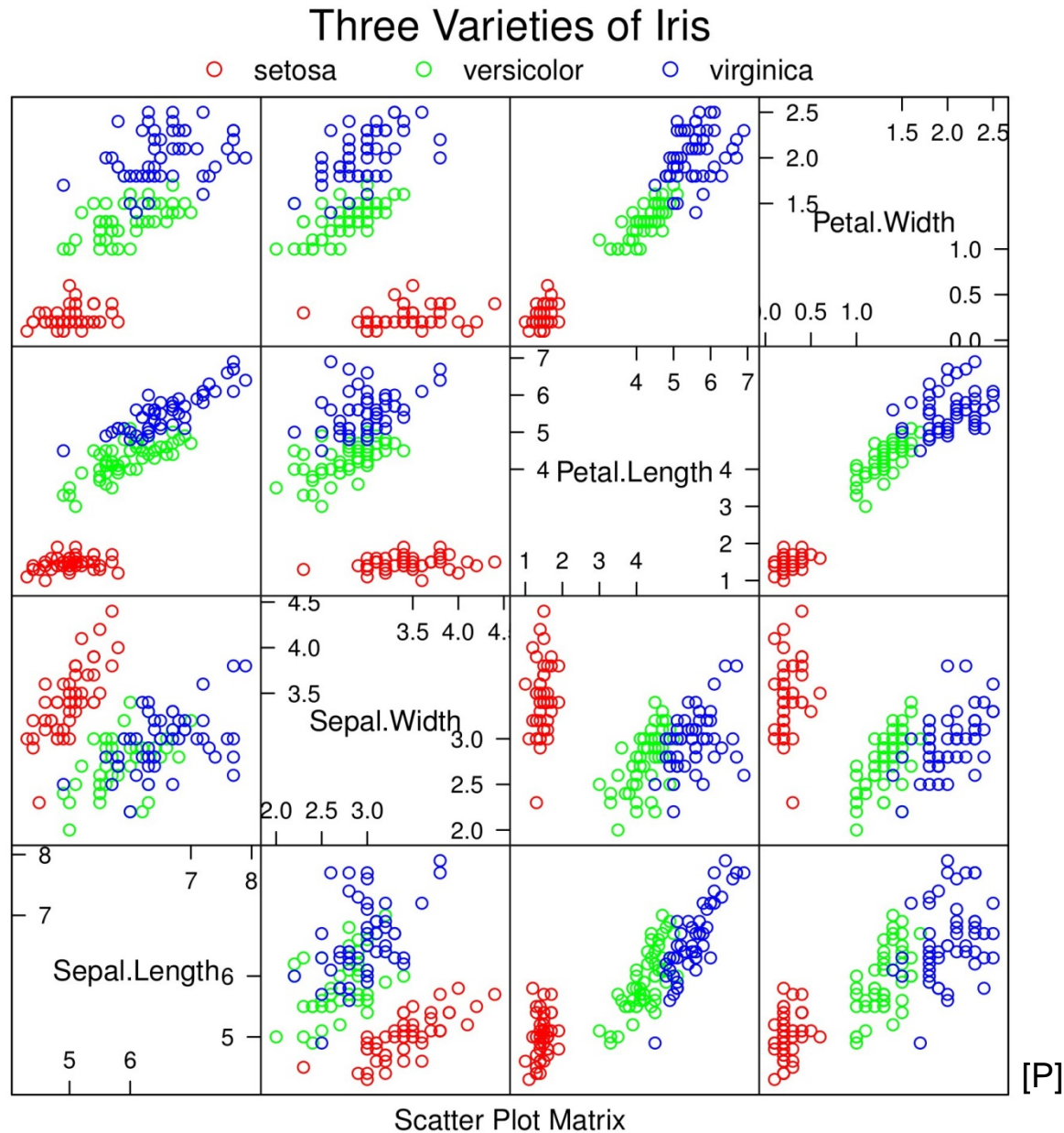


[P]

- Features: Length & Width of Petal & Sepal Leaves

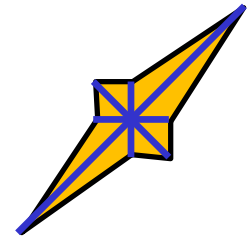
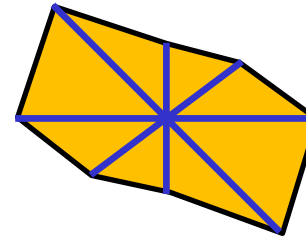
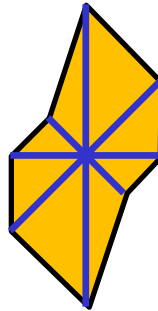
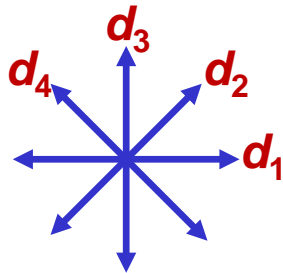
■ [All Images: wikipedia.org]

Scatterplot matrix for Iris Dataset



Map each dimension (or some selected dimensions) onto the parameters of a geometrical figure.

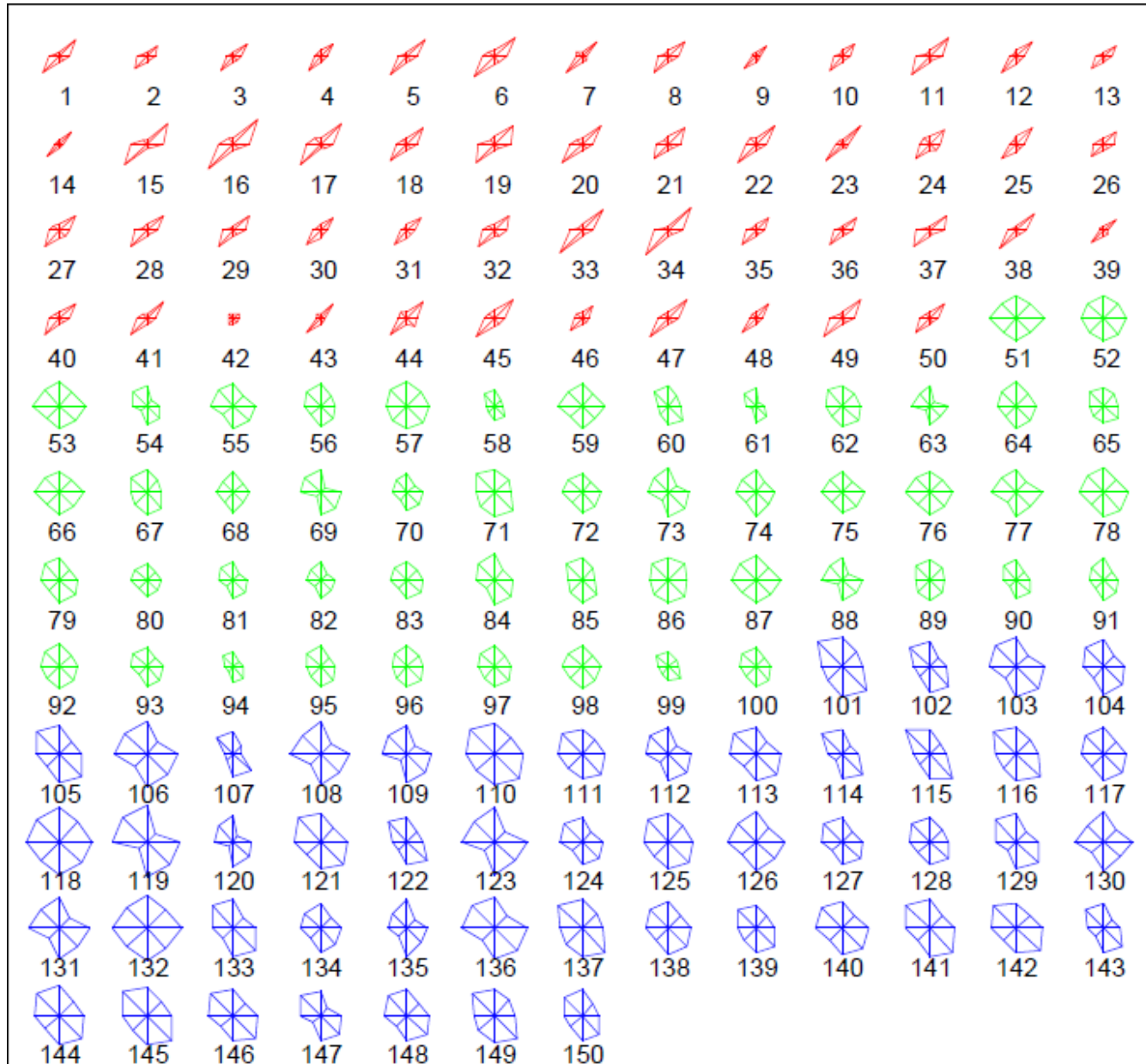
Example: Star glyphs



[H]

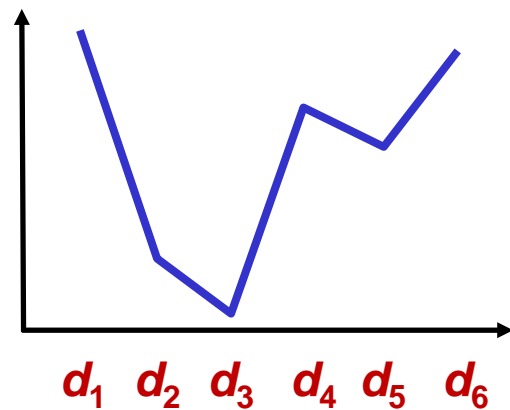
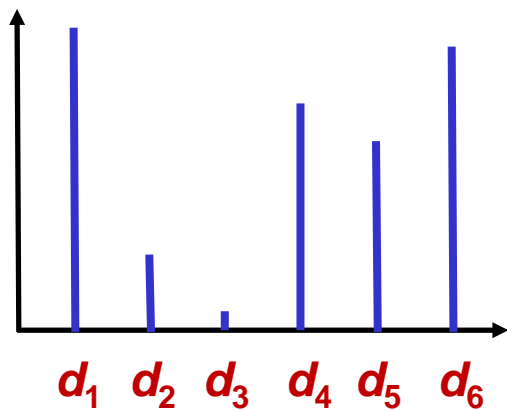
- Glyphs are usually perceived as a whole, not component wise.
- Glyphs require training.
- Re-numbering of the vector components leads to perceptually totally different glyphs.

Glyphs

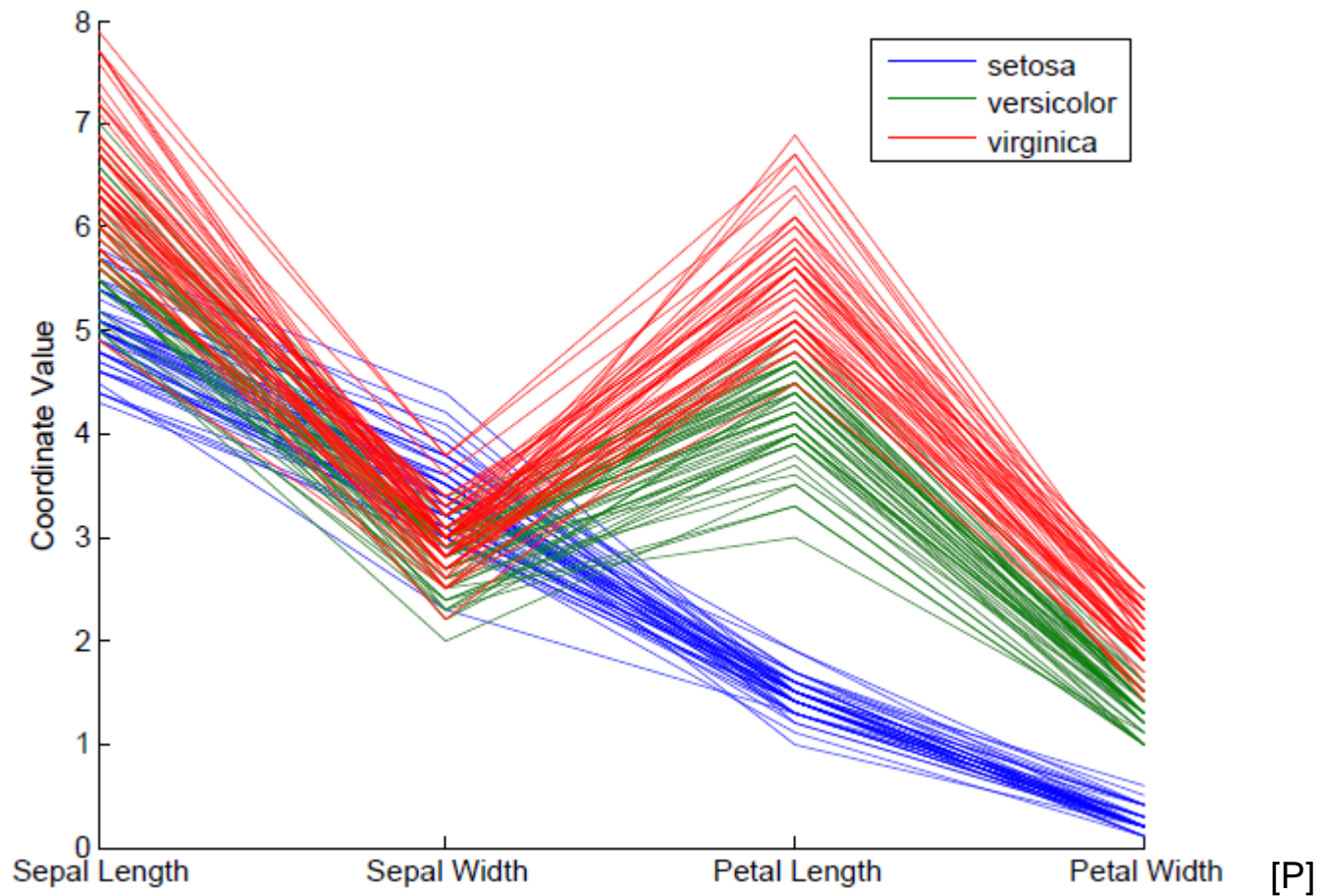


[P]

Parallel coordinates:

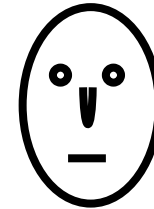
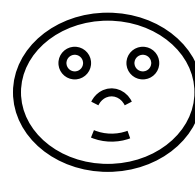
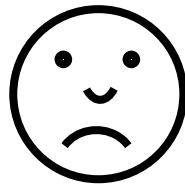
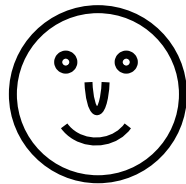


[H]



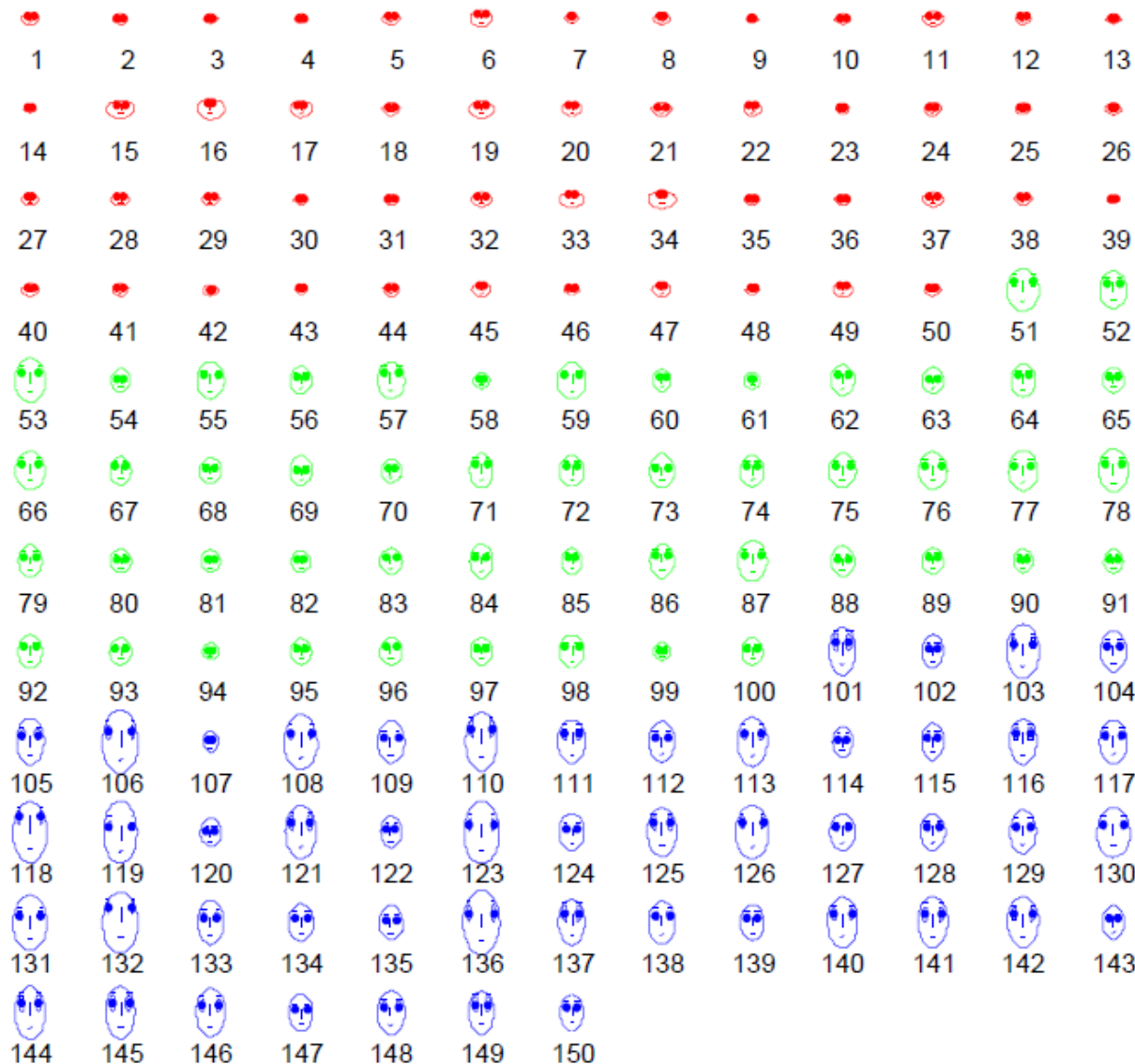
Chernoff faces:

Parameters are mapped to facial features.



[H]

Chernoff Faces



Mapping:

S.Length -> Face size,

S.Width -> 'Forehead/jaw relative length',

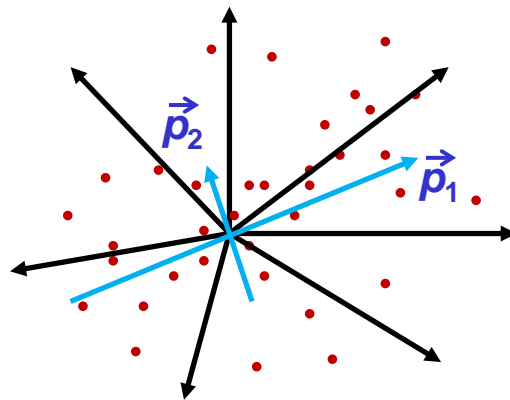
P. Length -> Forehead Shape,

P. Width -> Jaw shape

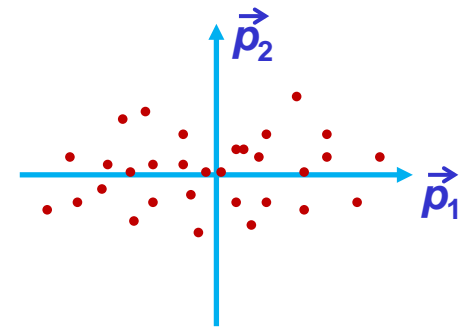
[P]

Other than scatterplots or glyphs, projection techniques do not represent the complete information in a data point but project onto selected directions.

Using PCA for visualization:



Data in many dimensions



Visualization:

Projection onto 2 or 3 PCs [H]

Problem:

PCA extracts maximum variance directions, structure is disregarded. Visualization may be an unstructured point cloud.

Idea (Friedman & Tukey, 1974):

Project onto 2-3 selected directions like PCA, but choose directions where the data exhibit interesting **structure**.

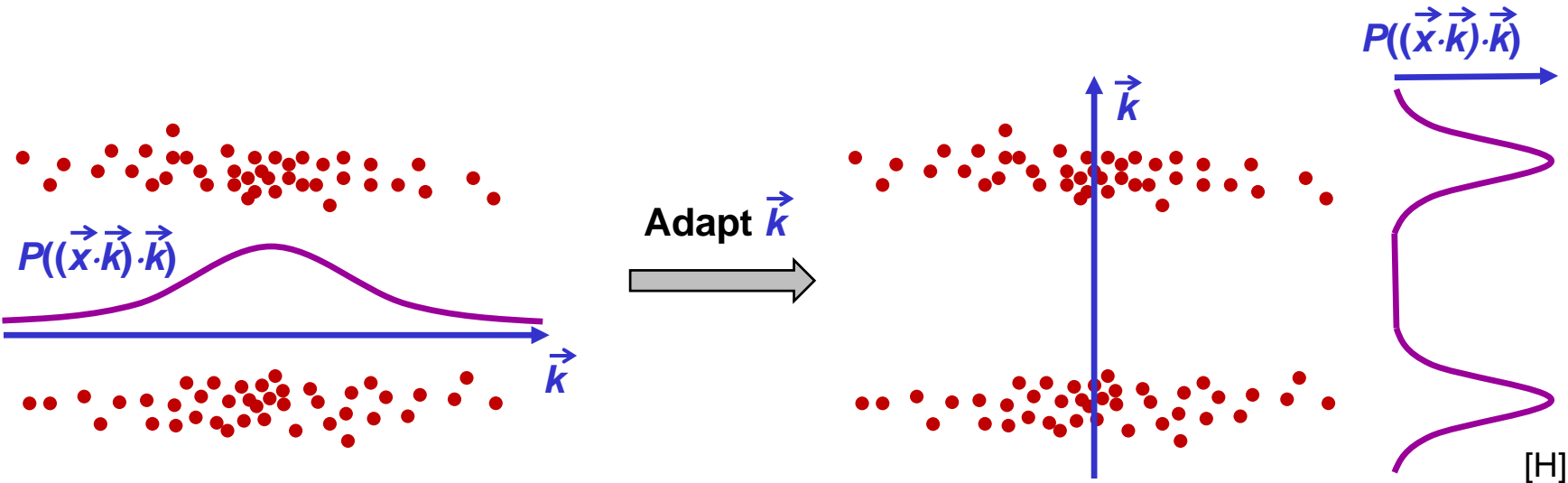
What is *interesting*?

- **Variance** (otherwise, data are on the same spot),
- distribution that is **not Gaussian**,
- **clusters**.

Procedure:

- Select 1-3 directions (e.g., by PCA or simply original dimensions).
- Project onto these directions, get density $P(\vec{x})$ of the projected data.
- Compute an **index** function (how interesting is the density).
- **Maximize** index by searching for better directions.

A Gaussian distribution is most uninformative, so we search for directions \vec{k} where the projection exhibits more structure:



Indices to measure the deviation of a distribution $P(\vec{x})$ (which has been normalized to zero mean and variance 1) from the standardized normal distribution $N_1(\vec{x})$ (with variance 1):

$$I_{\text{FT}} = \int P^2(\vec{x}) d\vec{x} \quad (\text{Friedman-Tukey index}),$$

minimized when P is a parabolic function similar to normal distribution.

$$I_{\text{H}} = \int (P(\vec{x}) - N_1(\vec{x}))^2 d\vec{x} \quad (\text{Hermite or Hall index}),$$

minimized when P is a standardized normal distribution.

$$I_{\text{NH}} = \int (P(\vec{x}) - N_1(\vec{x}))^2 N_1(\vec{x}) d\vec{x} \quad (\text{Natural Hermite index}),$$

like *Hermite index*, but higher weighting to the center.

$$I_{\text{E}} = \int P(\vec{x}) \log P(\vec{x}) d\vec{x} \quad (\text{Entropy index}),$$

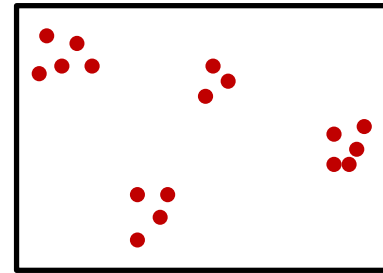
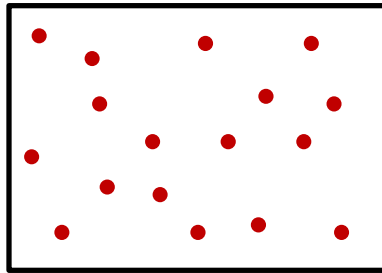
minimized by standardized normal distribution (note its not $-P \log P$!).

Problem:

Maximization requires estimation of the density $P(\vec{x})$ of the projected data. Methods are

- Kernel density estimation (Parzen windows),
- orthonormal function expansion.

Clustered distributions exhibit more short distances between pairs of points than unstructured distributions (for identical variance):



[H]

A projection index aimed at finding clusters was also proposed by Friedman and Tukey (1974):

$$I(\vec{k}) = s(\vec{k}) \cdot p(\vec{k}).$$

\vec{k} is the projection vector.

$s(\vec{k})$ is the standard deviation along \vec{k} (we still want big variance).

$p(\vec{k})$ is the average distance of points along \vec{k} .

(“Along” means: Project data onto direction \vec{k} , then measure s and p .)

Given: Data set $D = \{\vec{x}_1, \vec{x}_2, \dots\}$, $\vec{x}_i \in \mathbb{R}^d$ (with outliers removed).

Index

$$I(\vec{k}) = s(\vec{k}) \cdot p(\vec{k}).$$

Standard deviation:

$$s(\vec{k}) = \left(1/|D| \sum_{i=1 \dots |D|} (\vec{x}_i \cdot \vec{k} - \langle \vec{x} \rangle_{\vec{k}})^2 \right)^{1/2}$$

with mean

$$\langle \vec{x} \rangle_{\vec{k}} = 1/|D| \sum_{i=1 \dots |D|} \vec{x}_i \cdot \vec{k}.$$

Average pair distance:

$$p(\vec{k}) = \sum_{i,j=1 \dots |D|} f(x_{ij}) \cdot \Theta(R - x_{ij})$$

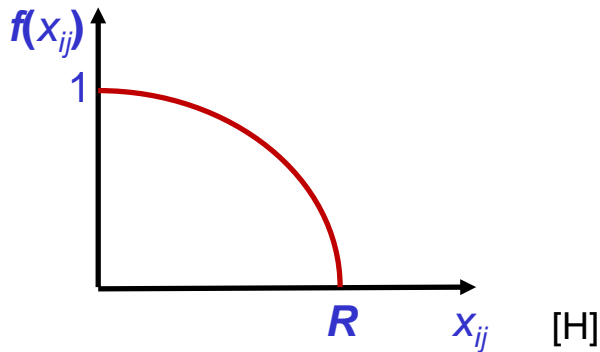
with pair distances $x_{ij} = |\vec{x}_i \cdot \vec{k} - \vec{x}_j \cdot \vec{k}|.$

Θ is the step function which implements a soft cut off together with $f(.)$:

The average pair distance

$$p(\vec{k}) = \sum_{i,j=1 \dots |D|} f(x_{ij}) \cdot \Theta(R - x_{ij})$$

is computed only up to a certain scale defined by $f(\cdot)$ and the cut off value R :



- Selection of a scale possible (remember clustering!)
- Reduces computational effort

Maximization of $l(\vec{k}) = s(\vec{k}) \cdot p(\vec{k})$:

- On the surface of the $d-1$ dimensional unit sphere (\vec{k} has unit length).
- PCA may yield suitable starting direction for \vec{k} .

To obtain a plot we need *two* projection directions. Two methods:

1. Repeated application of the 1d-method:

- Compute \vec{k}_1 by 1d-method as before.
- Apply 1d-method again to find \vec{k}_2 for the space orthogonal to \vec{k}_1 .
- To do this, compute the part orthogonal to \vec{k}_1 for each data point:

$$\vec{x}_i' = \vec{x}_i - (\vec{x}_i \cdot \vec{k}_1) \vec{k}_1.$$

(“ \vec{x} without its projection onto \vec{k}_1 .”)

2. Simultaneous maximization for \vec{k}_1 and \vec{k}_2 using 2d-index:

$$s(\vec{k}_1, \vec{k}_2) = \left(1/|D| \sum_{i=1 \dots |D|} (\vec{x}_i \cdot \vec{k}_1 - \langle \vec{x} \rangle_{\vec{k}_1})^2 + (\vec{x}_i \cdot \vec{k}_2 - \langle \vec{x} \rangle_{\vec{k}_2})^2 \right)^{1/2},$$

$$p(\vec{k}_1, \vec{k}_2) = \sum_{i,j=1 \dots |D|} f(x_{ij}) \cdot \Theta(R - x_{ij}) \quad \text{with}$$

$$x_{ij} = \left((\vec{x}_i \cdot \vec{k}_1 - \vec{x}_j \cdot \vec{k}_1)^2 + (\vec{x}_i \cdot \vec{k}_2 - \vec{x}_j \cdot \vec{k}_2)^2 \right)^{1/2}.$$

Aim:

Given data in a space of high dimension, find a manifold of low dimension such that projecting the data onto this manifold **preserves the structure** of the data as good as possible.

→ We must define what “structure” means !

An important aspect of structure is the **distances** between the data points.

Given:

- Data space \mathbb{R}^D of high dimension D .
- Projection space \mathbb{R}^d of small dimension d (for visualization, usually $d = 2$ or 3).

Aim: Find a mapping $\vec{\Phi}: \mathbb{R}^D \rightarrow \mathbb{R}^d$ such that the distances

$$\Delta_{ij} = \|\vec{x}_i - \vec{x}_j\|_D$$

between the data points \vec{x}_i and \vec{x}_j in \mathbb{R}^D are well approximated by the distances

$$\delta_{ij} = \|\vec{\Phi}(\vec{x}_i) - \vec{\Phi}(\vec{x}_j)\|_d$$

of the projections $\vec{\Phi}(\vec{x}_i)$ and $\vec{\Phi}(\vec{x}_j)$ in \mathbb{R}^d :

$$\forall i, j: \quad \Delta_{ij} \approx \delta_{ij}.$$

Sammon's stress measure:

$$E[\Phi] = (1 / \sum_{i < j} \Delta_{ij}) \sum_{i < j} (\Delta_{ij} - \delta_{ij})^2 / \Delta_{ij}$$

Minimization, e.g., by gradient descent with respect to the parameters of the mapping Φ (all the usual remarks on gradient descent go here).

Suitable initial parameters may be obtained by projection of the data onto the subspace spanned by the eigenvectors of largest eigenvalues.

The most difficult problem, however, is not minimization but finding a suitable ansatz for the mapping function Φ .

- Visualization of high dimensional data can capture the complete information of a single data point as long as the dimension is not too high.
- A scatterplot matrix shows all 2d combinations of dimensions.
- Glyphs represent several dimensions but require training to the specific glyph design.
- For higher dimension, projection onto directions / surfaces is necessary.
- Common dimension reduction techniques aiming at high variance may lead to projections showing an unstructured distribution.
- Projection pursuit:
 - Finds suitable projection direction(s) by maximizing a projection index.
 - The index aims at non-Gaussian and/or clustered distributions.

- [M] Online material available at www.cs.cmu.edu/~tom/mlbook.html for the textbook: Tom M. Mitchell: *Machine Learning*, McGraw-Hill
- [A] *Artexplosion Explosion*® Photo Gallery, Nova Development Corporation, 23801 Calabasas Road, Suite 2005 Calabasas, California 91302-1547, USA.
- [TP] M. Turk, A. Pentland: Eigenfaces for Recognition, *J. Cognitive Neuroscience*, Vol. 3, p. 71-86, 1991.
- [H] Gunther Heidemann, 2012.
- [P] Michael Pardowitz, 2014.