# Lecture 11

## Dimensionality Reduction

University of Amsterdam

# About dimensionality

University of Amsterdam

In order to perform our task, we want to have as much information as possible...

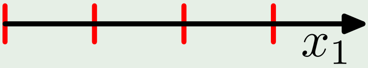
- In general we want to have as many features as possible
- Correlated features do not, together, convey as much information as the sum of both features separately
- Working in high dimensions causes difficulties:
  - We need exponential amounts of data to characterise the density as the dimensionality goes up
  - Intuitions we have from low-dimensional spaces do not always hold in higher dimensions
  - Often the data lies in a low dimensional manifold, embedded in a high-dimensional space

  This is often called the *curse of dimensionality*
- It is hard to visualise high-dimensional data

# Curse of dimensionality



## Example: Estimating densities with histograms

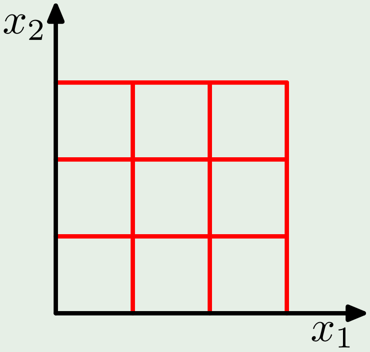



$$D = 1$$

# Curse of dimensionality

UNIVERSITY OF AMSTERDAM

## Example: Estimating densities with histograms

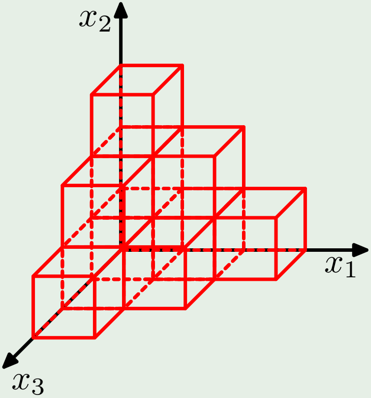



$$D = 2$$

# Curse of dimensionality

University of Amsterdam

## Example: Estimating densities with histograms




$$D = 3$$

# Embeddings in high-dimensional spaces



**Example: Low-dimensional manifold in High-dim data**

To illustrate, consider a $16 \times 16$ image of a digit. A machine sees this as a 256-dimensional vector. If we consider only rotations of this digit, all images will lie on a one-dimensional manifold in a 256-dimensional space. If we consider translations and rotations, the data is intrinsically 3D, in a 256D space.

# Dimensionality reduction

UNIVERSITY OF AMSTERDAM

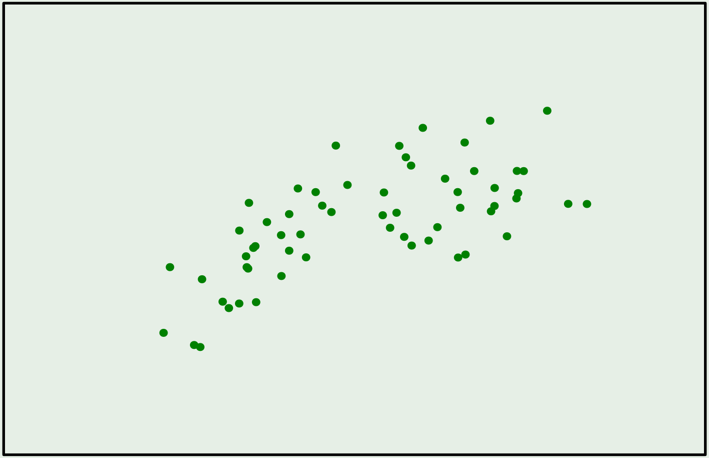
The purpose of dimensionality reduction is to project the data to a low-dimensional space, while retaining the information present in the data.

- This is a form of lossy data compression
- It can be useful to visualize the data
- Learning machines working in the lower-dimensional space may obtain better results with less training data, as we can obtain better density estimates from the data in the low-dimensional space.

# Dimensionality reduction by linear projection



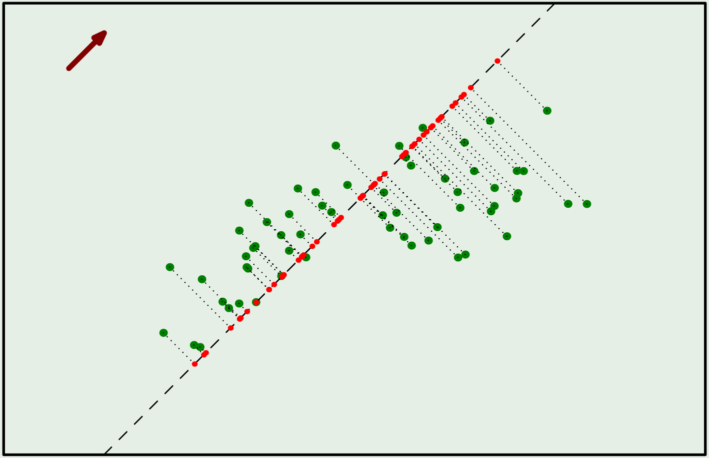
## Example



# Dimensionality reduction by linear projection



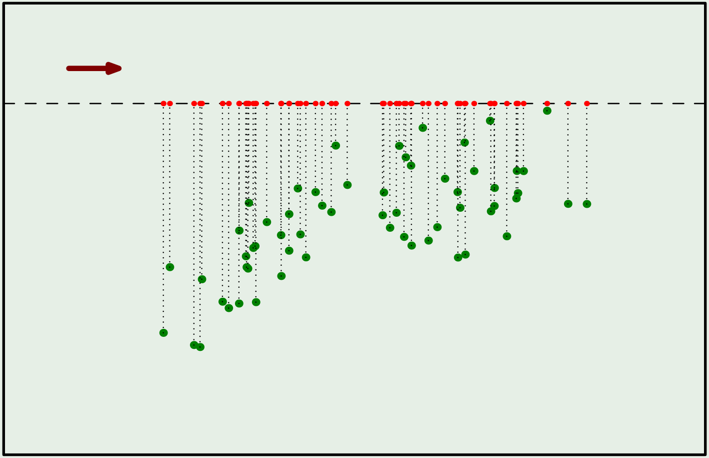
## Example



# Dimensionality reduction by linear projection



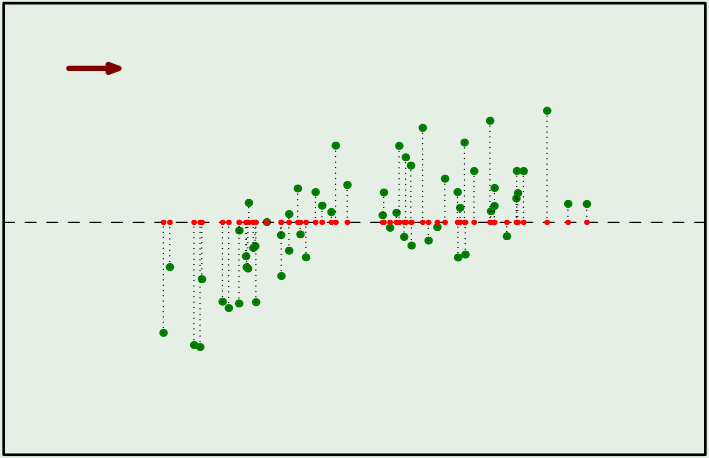
## Example



# Dimensionality reduction by linear projection



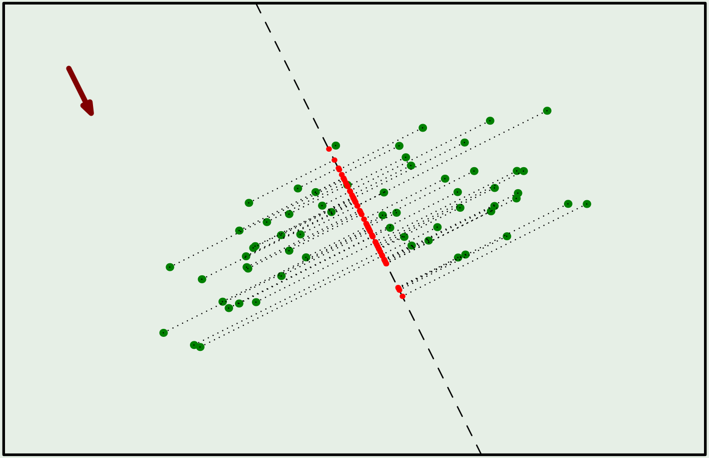
## Example



# Dimensionality reduction by linear projection

UNIVERSITY OF AMSTERDAM

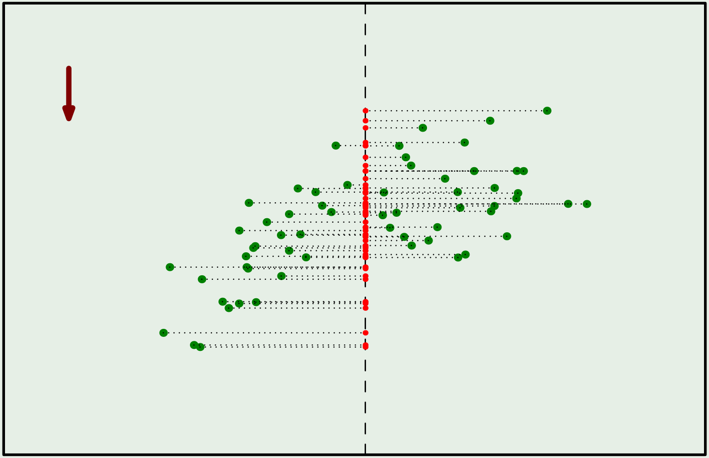
## Example



# Dimensionality reduction by linear projection

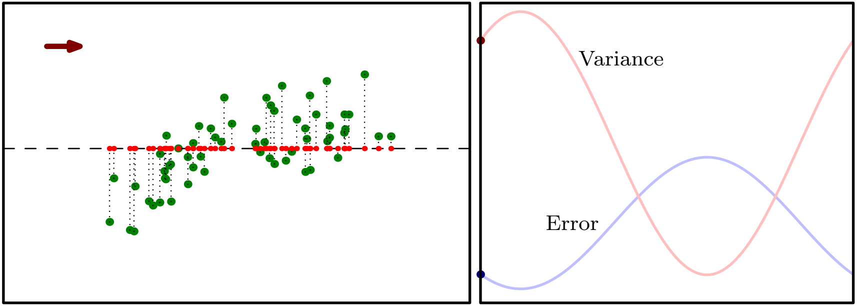


## Example



# Dimensionality reduction by linear projection

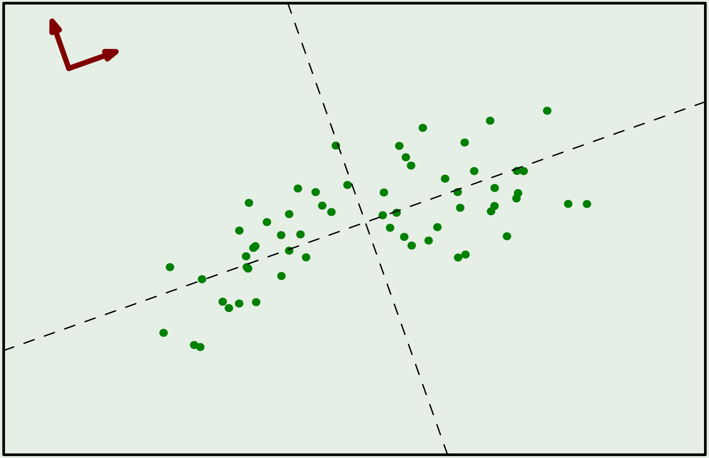




# Dimensionality reduction by linear projection

UNIVERSITY OF AMSTERDAM

## Example



# Principal Component Analysis



PCA is probably the most well-known method for dimensionality reduction.

- Also known as the *Karhunen-Loève transform*
- Orthogonal projection of the data into a lower-dimensional subspace, so that the variance of the projected data is maximised
- Equivalently: linear projection that minimises the mean-squared distance between data points and their projection

# Maximising the projected variance



Consider projecting on $\mathbf{u}_1$, with unit length for convenience.

- Each vector $\mathbf{x}_n$ is then projected into $\mathbf{u}_1^\top \mathbf{x}_n$
- Mean of the projected data equals the projected mean $\mathbf{u}_1^\top \bar{\mathbf{x}}$
- To maximise the variance of the projected data, we maximise

$$\frac{1}{N}\sum_{n=1}^{N}(\mathbf{u}_1^\top \mathbf{x}_n - \mathbf{u}_1^\top \bar{\mathbf{x}})^2 = \mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1 \tag{1}$$

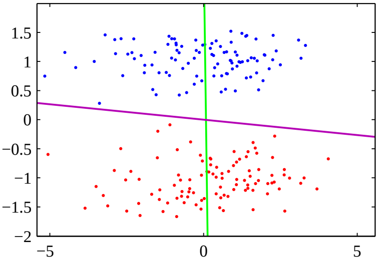
  where $\mathbf{S}$ is the data covariance
- Using a Lagrange multiplier to constrain $\mathbf{u}_1^\top \mathbf{u}_1 = 1$, we get

$$L(\mathbf{x}, \lambda) = \mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^\top \mathbf{u}_1) \tag{2}$$

  resulting in $\mathbf{S}\mathbf{u}_1 = \lambda \mathbf{u}_1$. That is, $\mathbf{u}_1$ is an eigenvector of $\mathbf{S}$ and the maximum is obtained for the largest eigenvalue.

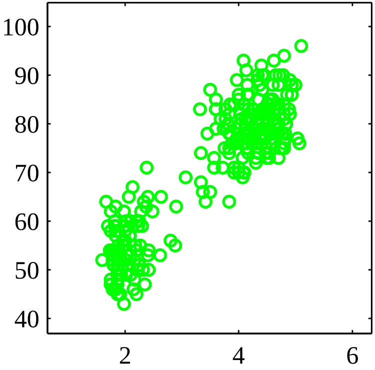
# About PCA

- The maximal variance does not always correspond to optimal class separation. Fisher's linear discriminant includes class label information in finding the projection
- PCA can be used to normalise the dataset so as to make different dimensions decorrelated



# About PCA

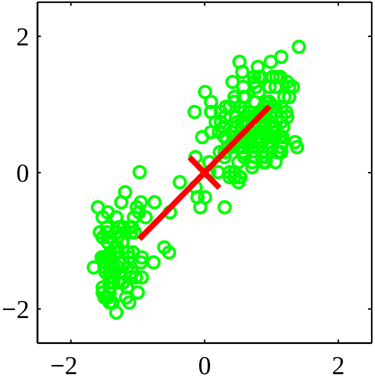
- The maximal variance does not always correspond to optimal class separation. Fisher's linear discriminant includes class label information in finding the projection
- PCA can be used to normalise the dataset so as to make different dimensions decorrelated



# About PCA



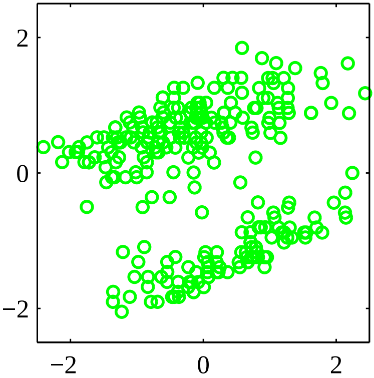
- The maximal variance does not always correspond to optimal class separation. Fisher's linear discriminant includes class label information in finding the projection
- PCA can be used to normalise the dataset so as to make different dimensions decorrelated



# About PCA



- The maximal variance does not always correspond to optimal class separation. Fisher's linear discriminant includes class label information in finding the projection
- PCA can be used to normalise the dataset so as to make different dimensions decorrelated



# Principal Component Analysis

University of Amsterdam

The additional principal components can be found by incrementally selecting the eigenvectors with the largest eigenvalues.

## To Summarise: Principal Component Analysis

1. Find the empirical mean of the data
2. Compute the covariance matrix
3. Perform eigenvector decomposition, and select sufficient eigenvectors to preserve the chosen amount of variation in the data

One problem with this formulation of PCA is that the covariance matrix becomes huge as the dimensionality of the data increases.

- PCA is often used to represent pictures in a compact fashion. A $256 \times 256$ pixels image resolution results in a $65536 \times 65536$ element covariance matrix (which requires 48GB to store)
- However the number of non-zero eigenvectors is limited by the number of data elements. We can reduce the computational complexity.
- The resulting computation of PCA has revolutionised the field of face recognition in the 90's.

# PCA on high-dimensional data

Define $\mathbf{X}$ where row $n$ contains $\mathbf{x}_n - \bar{\mathbf{x}}$, then $\mathbf{S} = 1/N\,\mathbf{X}^\top\mathbf{X}$ so that PCA is given by

$$\frac{1}{N}\mathbf{X}^\top\mathbf{X}\mathbf{u}_i = \lambda_i\mathbf{u}_i \tag{3}$$

Now consider instead

$$\frac{1}{N}\mathbf{X}\mathbf{X}^\top\mathbf{v}_i = \lambda_i\mathbf{v}_i \tag{4}$$

Premultiplying both sides by $\mathbf{X}^\top$ gives

$$\frac{1}{N}\mathbf{X}^\top\mathbf{X}\mathbf{X}^\top\mathbf{v}_i = \lambda_i\mathbf{X}^\top\mathbf{v}_i \tag{5}$$

so that if $\mathbf{v}_i$ is an eigenvector of the $N \times N$ matrix $\mathbf{X}\mathbf{X}^\top$ with eigenvalue $\lambda_i$, $\mathbf{u}_i = \mathbf{X}^\top\mathbf{v}_i$ is an eigenvector of the $D \times D$ matrix $\mathbf{S}$ with eigenvalue $\lambda_i$

# Probabilistic PCA

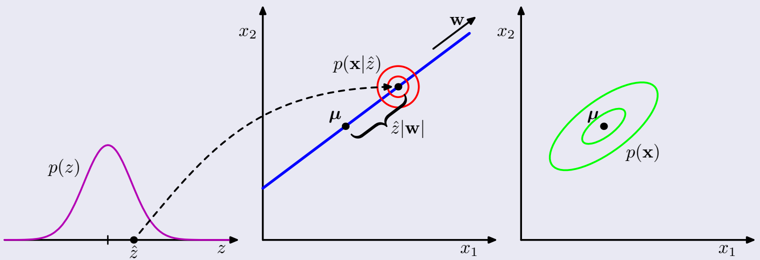


Alternatively, PCA can be seen as a probabilistic model:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \tag{6}$$

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I}) \tag{7}$$

## Generative view of PCA



# Advantages of PPCA



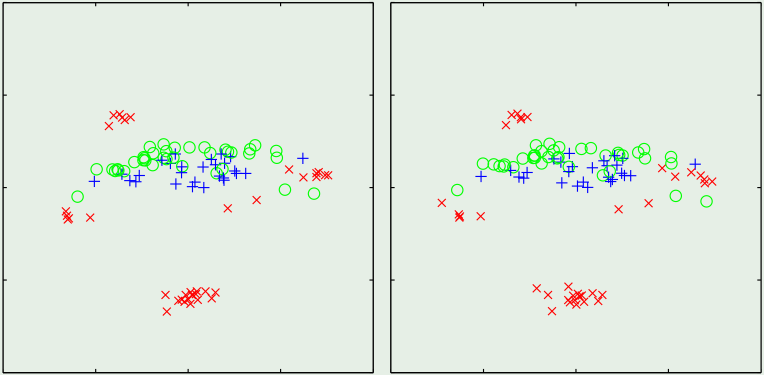
The probabilistic formulation of PCA is equivalent with PCA, but has several advantages:

- We can use EM to optimise the model without ever explicitly computing the covariance matrix
- It is easy to combine multiple PCA models into a mixture of PCA
- The existence of the likelihood function allows direct comparison of different models
- The model can be sampled from to generate new data

UNIVERSITY OF AMSTERDAM

## Example: Missing values

To illustrate, PPCA was applied twice to the same data, however in the second case 30% of the values were removed and EM was used to deal with the missing values.



# Factor Analysis



FA is very similar to PCA: the only difference is in the conditional probability of the data:

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \mu, \mathbf{\Psi}) \tag{8}$$

where $\mathbf{\Psi}$ is a diagonal matrix.

## PCA vs. FA

- PCA is sensitive to the scale of each feature, but is insensitive to rotation of the dataset
- FA is insensitive to the relative scale of the features, but is sensitive to rotation

# Factor Analysis



FA is very similar to PCA: the only difference is in the conditional probability of the data:

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}) \tag{8}$$

where $\boldsymbol{\Psi}$ is a diagonal matrix.

## PCA vs. FA

- PCA is sensitive to the scale of each feature, but is insensitive to rotation of the dataset
- FA is insensitive to the relative scale of the features, but is sensitive to rotation

# Kernel PCA



We can define PCA in function of a kernel function, and solve the eigenvalue problem

$$\frac{1}{N}\mathbf{K}\mathbf{a}_i = \lambda_i \mathbf{a}_i \tag{9}$$

- This allows us to find much richer mappings than the direct linear projection of PCA.
- However Kernel PCA has the problem that we often want to find projection directions using a training set, and find the low-dimensional projections for new data. This is not possible with Kernel PCA.
- There are however techniques to find approximate projections

# Multidimensional Scaling



Instead of minimising the reconstruction error after projection, we can do a linear projection of the data such as to preserve the distances between data points as best as possible

- If Euclidean distance is used, this is equivalent with PCA
- However MDS is much more general and can be defined on non-numerical data (for example: strings) using a similarity measure
- This is called *non-metric MDS*

# Isomaps



Isometric Feature Maps perform MDS on the data, using geodesic distances between points.

- For example, the distance between two points on a circle is measured along the circumference, not the straight Euclidean distance.
- The geodesic distance between two points is approximated by finding the closest points to each data point and computing the sum of the distances between the points on the shortest path connecting them.

# Generative Topographic Mappings

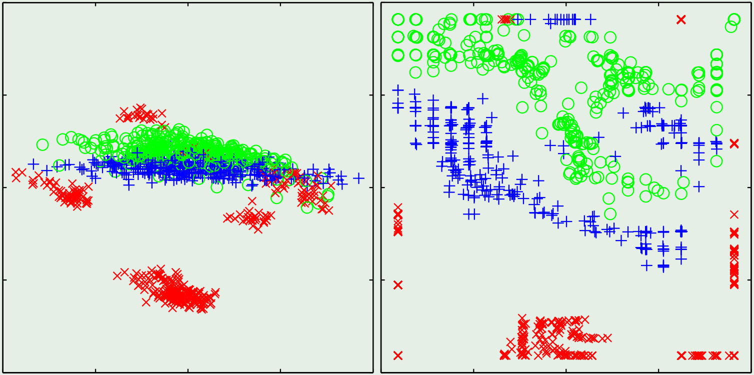


The GTM is a final model for dimensionality reduction:

$$p(\mathbf{z}) = \frac{1}{K} \sum_{i=1}^{K} \delta(\mathbf{z} - \mathbf{z}_i)$$
$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{y}(\mathbf{z}, \mathbf{w}), \sigma \mathbf{I})$$

- It defines a discrete, typ. 2D, latent space of binary variables and a continuous observed space
- A non-linear mapping is found by linear regression
- Using a discrete latent space allows us to compute the marginal by summing over the latent variables
- The model can then be optimised using EM

## Example: GTM vs PCA



# Piecewise Linear Modelling



In order to model a nonlinear manifold, we can approximate the structure through a combination of linear models.

- Cluster the data using K-Means and apply PCA to each group
- Better: Use the reconstruction error for cluster assignment
- Better still: Use mixture of PPCA. This provides a full probabilistic treatment and Bayesian treatment allows us to find the number of components automatically
- This can be extended to mixtures of factor analysers

# Locally Linear Embedding



As an extreme case, we can represent each data element as a linear function of its $K$ neighbours

$$\mathbf{x}_i = \sum_{j=1}^{K} w_{ij}\mathbf{x}_j$$

- Optimise $\sum_i |\mathbf{x}_i - \sum_j w_{ij}\mathbf{x}_j|$ to find $w_{ij}$.
- Create a representation $\mathbf{z}$ in any dimensionality, by optimising

$$E(\mathbf{Z}) = \sum_i |\mathbf{z}_i - \sum_j w_{ij}\mathbf{z}_j|$$

# Locally Linear Embedding



Pros

- Efficient for large datasets
- Single parameter to tune ($K$)
- Scale-invariant, rotation-invariant, translation-invariant

Cons

- Not useful for representing future data, really
- Can be unstable in sparse areas of dataset

# Independent component analysis



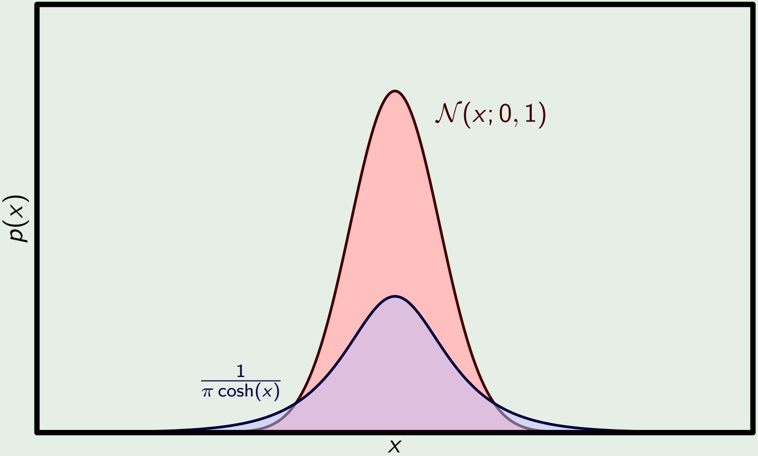
ICA views the $D$-dimensional observation as a linear combination of $N$ independent sources.

- To illustrate this, imagine $D$ microphones recording $N$ people speaking simultaneously (ignoring delays and considering only the attenuation due to the distance between speaker and microphone)
- This is a form of *blind source separation*
- This is only possible if we assume that the sources have non-Gaussian distributions. Often, we assume

$$p(z_j) = \frac{1}{\pi cosh(z_j)} \tag{10}$$

University of Amsterdam

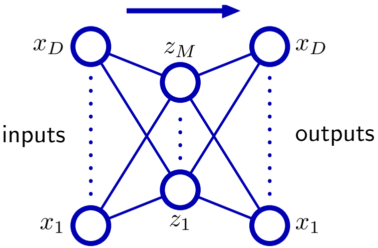
## Example: Typical ICA density function



# Autoassociative Neural Networks



Consider a 2-layer neural network, where we set the output values to be equal to the input values. This is called an auto-encoder or auto-associative Neural Network.



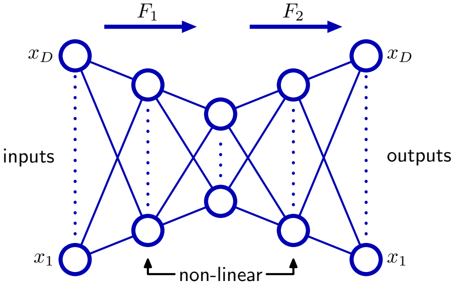

By setting the number of hidden nodes to be less than the number of inputs/outputs, we force the network to compress the data.

# Autoassociative Neural Networks



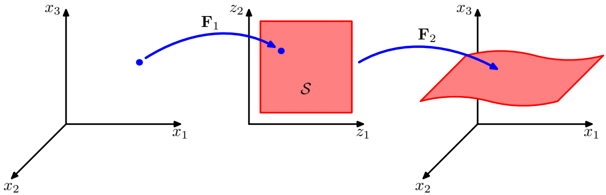
In the case of a 2-layer network with linear activation functions and minimizing the sum-squared-error, the network converges to PCA.

- However even if we use non-linear activation functions, we still obtain PCA with a 2-layer network
- Increasing the number of layers allows the network to do non-linear dimensionality reduction



# Dimensionality reduction with auto-encoders



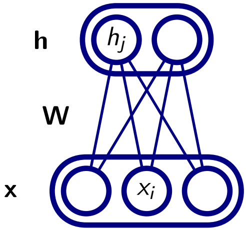



The mapping done by the auto-encoder can be very general

- Depending on the number of nodes in the intermediate layers, extremely complex functional mappings are possible
- However the optimisation problem is now non-convex — we can easily end up with non-optimal solutions
- Moreover, the dimensionality of the subspace must be defined beforehand

# Restricted Boltzmann Machines






- with three sets of parameters
  - The weights $W$, with $W_{ij}$ connecting node $x_i$ to $h_j$
  - The biases $a$ for the visible nodes
  - the biases $b$ for the hidden nodes

# The Restricted Boltzmann Machine

We use undirected connections between each node of one layer and all the nodes of the next layer. Therefore:

$$p(h|x) = \prod_j p(h_j|x) \qquad p(x|h) = \prod_i p(x_i|h) \tag{11}$$

- The log-likelihood is given by

$$-\log P(x,h) = -b^\top x - a^\top h - h^\top Wx + const \tag{12}$$

- The log-likelihood cannot be optimised in closed form, but nodes can be sampled from as:

$$p(x_i|h) \sim \sigma(b_i + \sum_j W_{ij} h_j) \qquad \text{for binomial nodes} \tag{13}$$

$$p(x_i|h) \sim \mathcal{N}(b_i + \sum_j W_{ij} h_j, 1) \qquad \text{for continuous nodes} \tag{14}$$

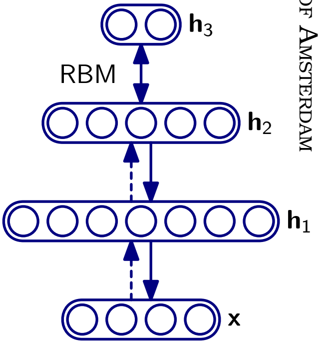
$$p(x_i|h) \sim \mathcal{S}_{\{x_i\}}[b_i + \sum_j W_{ij} h_j] \qquad \text{for multinomial nodes} \tag{15}$$

# Multiple Layers



We can use the RBM for non-linear dimensionality reduction:

- Similar to the autoencoder, we try to recover the original data
- The inverse mapping is done by the same nodes as the mapping, thus halving the number of free parameters
- This is a probabilistic model, and can therefore easily be combined with other probabilistic models
- We can use a sampling algorithm called contrastive divergence to optimise the parameters, using each layer as input to the next



# Summary

UNIVERSITY OF AMSTERDAM

We've seen dimensionality reduction:

- Principal Component Analysis (Bishop, p. 561-563, 569-570)
- PPCA (Bishop, p. 570-573)
- Factor analysis (Bishop, p. 583-586)
- Kernel PCA (Bishop, p. 586-590)
- Nonlinear models (Bishop, p. 591-598)