Neural Networks and Learning Systems
TBMI26 / 732A55
2023

Lecture 8
Unsupervised Learning –
Dimensionality Reduction, Clustering and Auto Encoders

Magnus Borga
magnus.borga@liu.se

Three main categories of machine learning methods

- **Supervised learning (predictive)**

  Learn to generalize and classify new data based on labeled training data.

  – Pattern recognition

  – Classification

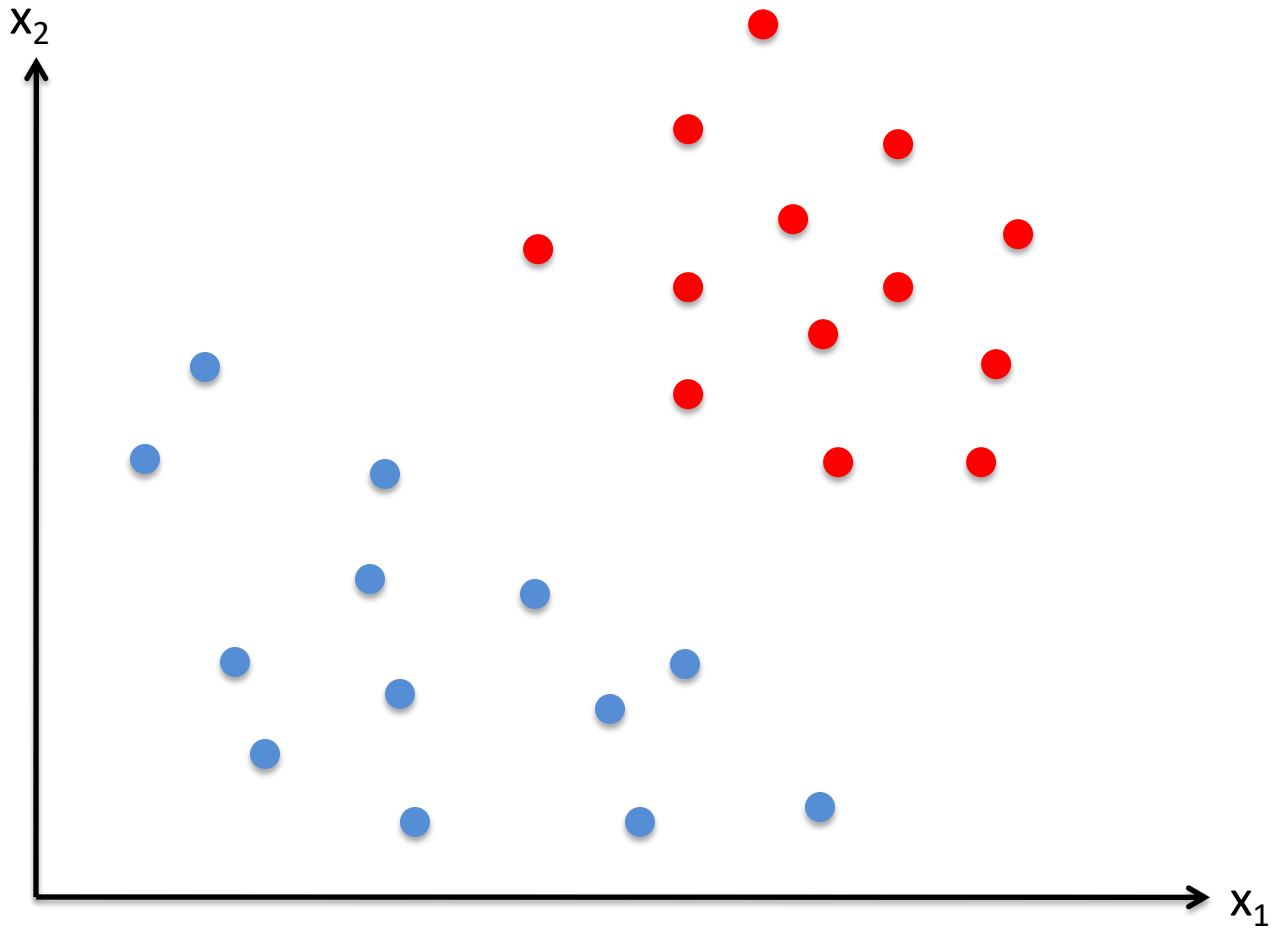
- **Reinforcement learning (active)**

  Generate policies/strategies that lead to a (possibly delayed) reward. Learning by doing.

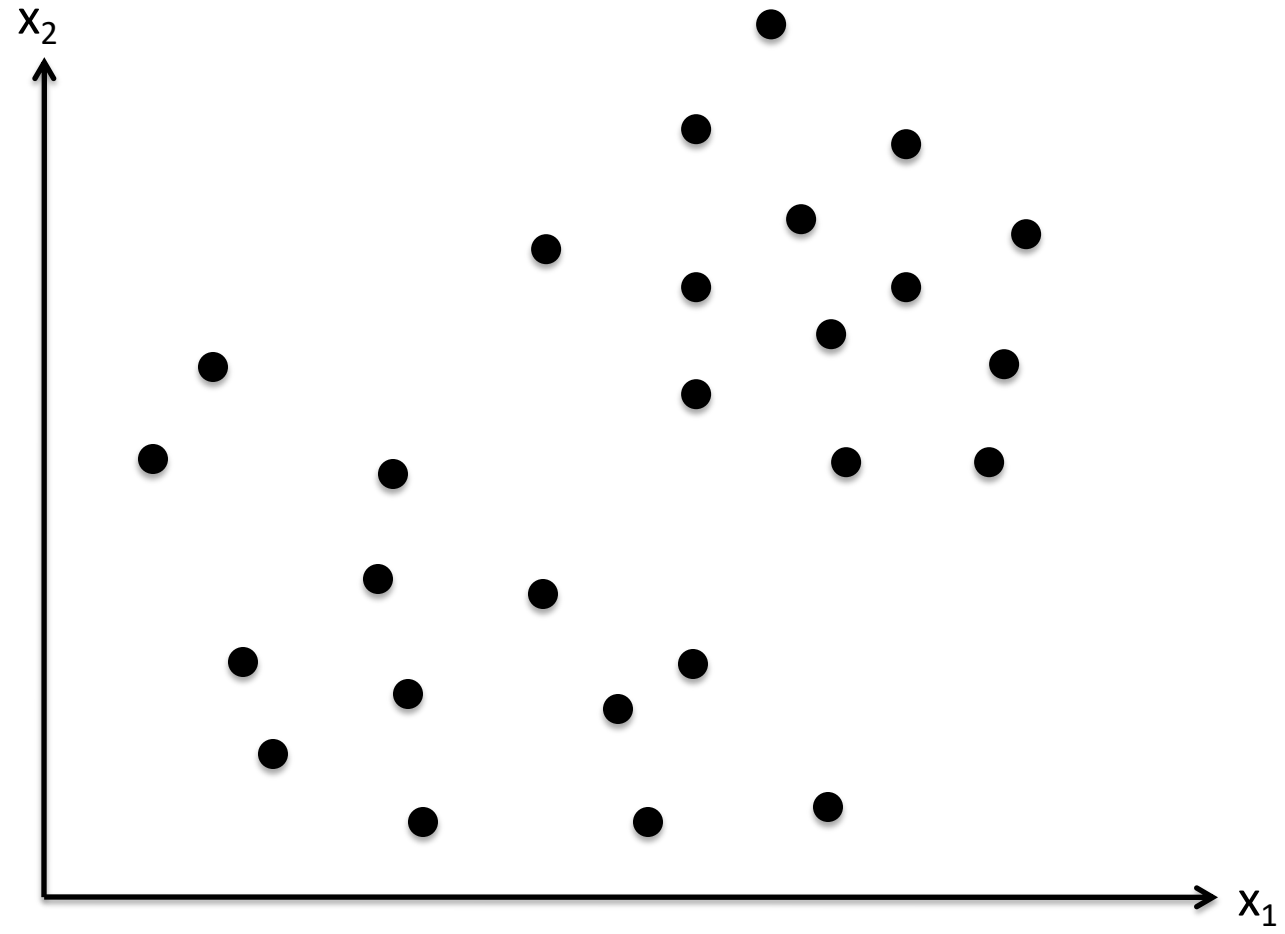
- **Unsupervised learning (descriptive)**

  Discover structure and relationships in complex high-dimensional data.

Supervised learning – labelled samples



Unsupervised learning – unlabelled samples



Unsupervised learning

- **Task:** Find underlying structure in data.
- **Input:** Training data examples $\{\mathbf{x}_i\}$ i=1...N.
- **Output:** Description of the data in a simpler form, e.g., with fewer dimensions or parameters.

Unsupervised learning

- Optimizes an internal loss function, e.g.
  - max variance (PCA)
  - min distance to cluster centre (k-means)

- Finds a new representation of the data

Applications

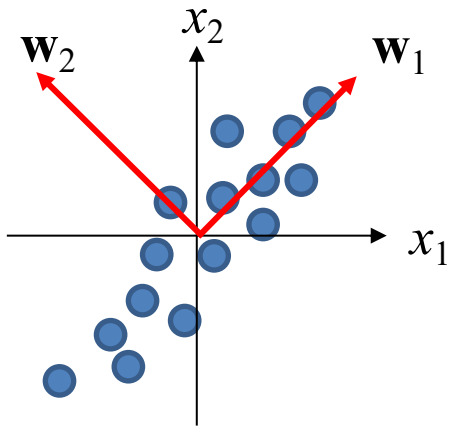
- Clustering
  - find order or structure in data
- Dimensionality reduction / Feature extraction
  - keep the most “important” parts of the signal
    - Image compression
    - image denoising,
    - image restoration,
    - outlier detection

Too many dimensions/features

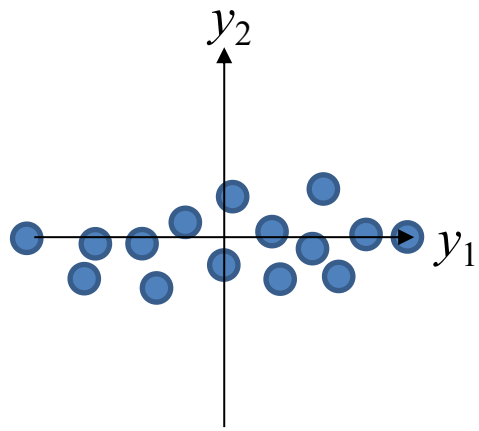
Correlated features or features that do not carry any information:

- Introduce noise in the analysis/classification
- Introduce more parameters in the learning model
  - More local optima in the optimization
  - Poorer generalization
  - Higher computational effort
- Difficult to visualize high-dimensional data

Projection / linear transformation



$$\mathbf{y} = \begin{pmatrix} \mathbf{w}_1^T \mathbf{x} \\ \mathbf{w}_2^T \mathbf{x} \end{pmatrix} = \mathbf{W}^T \mathbf{x}$$



Can ”uncorrelate” data through a linear transformation!!

$$\mathbf{C}_x = \begin{bmatrix} \mathrm{Var}(x_1) & \mathrm{Cov}(x_1, x_2) \\ \mathrm{Cov}(x_2, x_1) & \mathrm{Var}(x_2) \end{bmatrix}$$

$$\mathbf{C}_y = \begin{bmatrix} \mathrm{Var}(y_1) & 0 \\ 0 & \mathrm{Var}(y_2) \end{bmatrix}$$

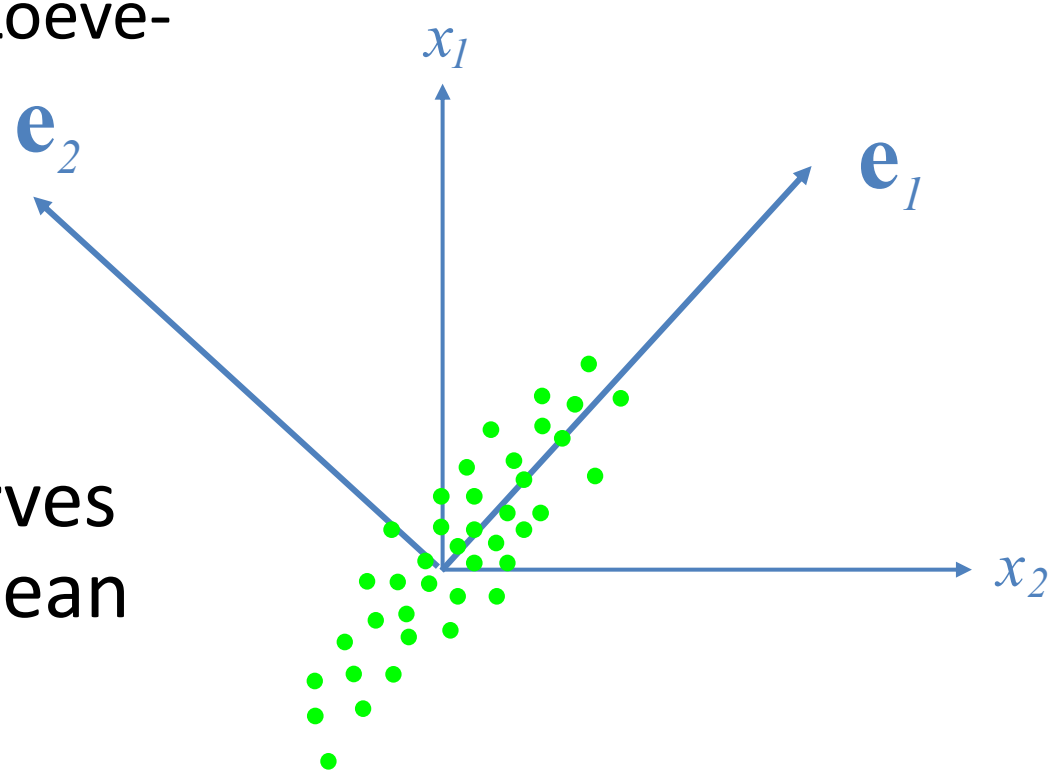
PCA

Principal Component Analysis

- Pearson 1901

  (A.k.a. Hotelling-transform or Karhunen-Loéve-transform)

- Coordinate transformation to an orthogonal basis where the data is uncorrelated.
- Dimensionality reduction that preserves maximum variance (minimizes the mean square error).



Maximize the variance

Normalized vector

The variance in direction $\hat{\mathbf{w}}$: (Suppose $\mathbf{x}$ has mean 0.)

$$\sigma^2_{\hat{\mathbf{w}}} = E[(\mathbf{x}^T\hat{\mathbf{w}})^2] = E[(\hat{\mathbf{w}}^T\mathbf{x})(\mathbf{x}^T\hat{\mathbf{w}})]$$

$$= \hat{\mathbf{w}}^T E[\mathbf{x}\mathbf{x}^T]\hat{\mathbf{w}} = \hat{\mathbf{w}}^T\mathbf{C}\hat{\mathbf{w}} = \frac{\mathbf{w}^T\mathbf{C}\mathbf{w}}{\mathbf{w}^T\mathbf{w}}$$

The covariance matrix of $\mathbf{x}$.

Maximize the variance

$$\sigma^2_{\hat{\mathbf{w}}} = \frac{\mathbf{w}^T \mathbf{C} \mathbf{w}}{\mathbf{w}^T \mathbf{w}}$$

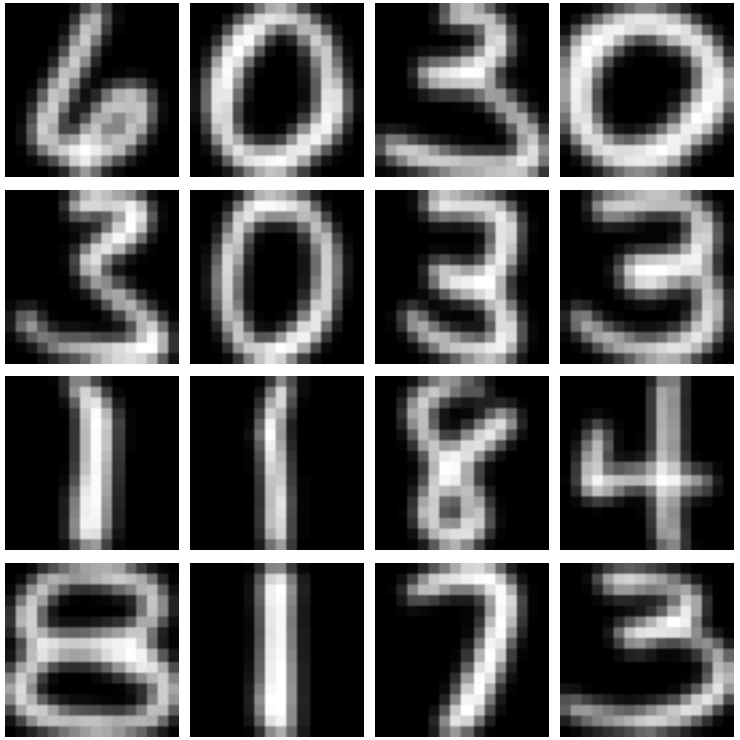
$$\frac{\partial \sigma^2_{\hat{\mathbf{w}}}}{\partial \mathbf{w}} = \frac{2}{\mathbf{w}^T \mathbf{w}} (\mathbf{C}\mathbf{w} - \sigma^2_{\hat{\mathbf{w}}} \mathbf{w}) = 0 \quad \Rightarrow$$

$$\mathbf{C}\mathbf{w} = \sigma^2_{\hat{\mathbf{w}}} \mathbf{w}$$

PCA is the Eigen-value decomposition of the data covariance matrix.

PCA - Example

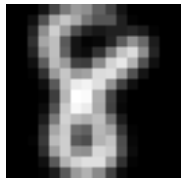
About 9000 training examples



US Postal Service Digit Data
http://www.gaussianprocess.org/gpml/data/

Feature vectors

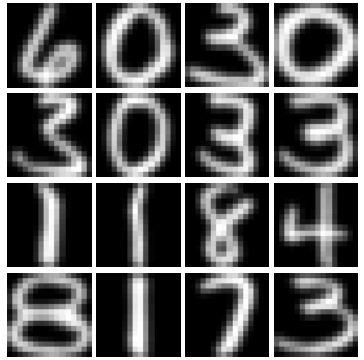
16



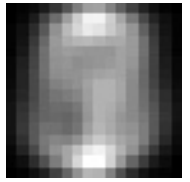
16

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_{256} \end{pmatrix}$$

PCA – Example, cont.



Mean digit

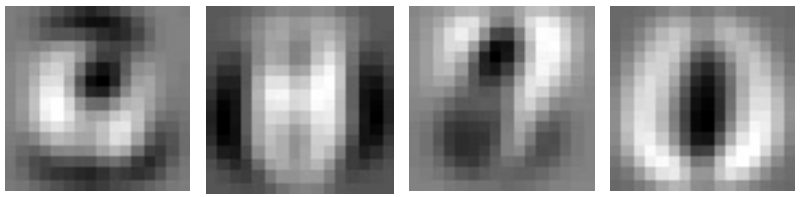
$\bar{\mathbf{x}} =$ 

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

256x256 matrix

Eigendecomposition of **C**:

$\mathbf{e}_1$ $\mathbf{e}_2$ $\mathbf{e}_3$ $\mathbf{e}_4$



…

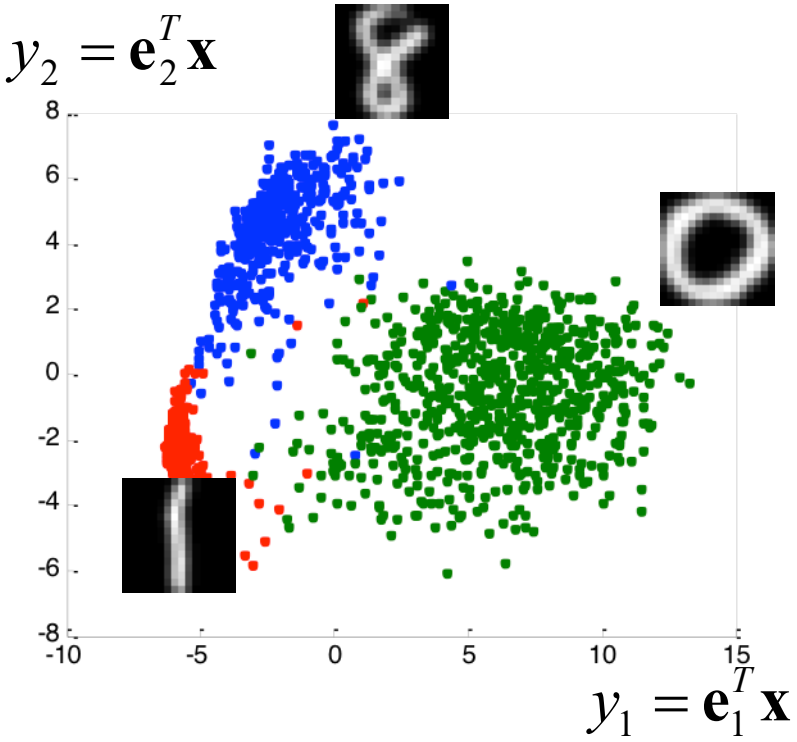
256 eigenvectors and eigenvalues

50 100 150 200 250

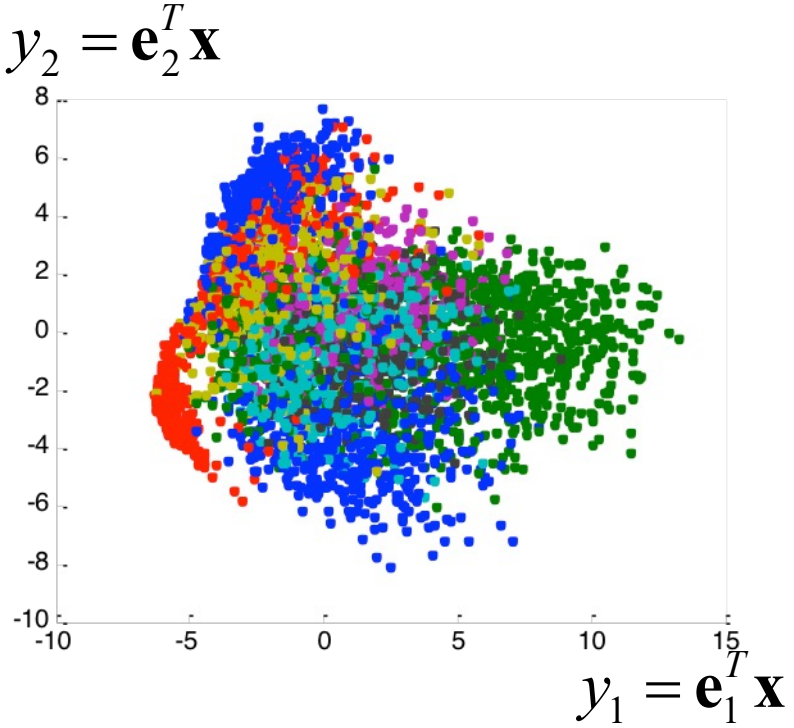
PCA – Example, cont.

From 256 to 2 feature dimensions!

3 classes

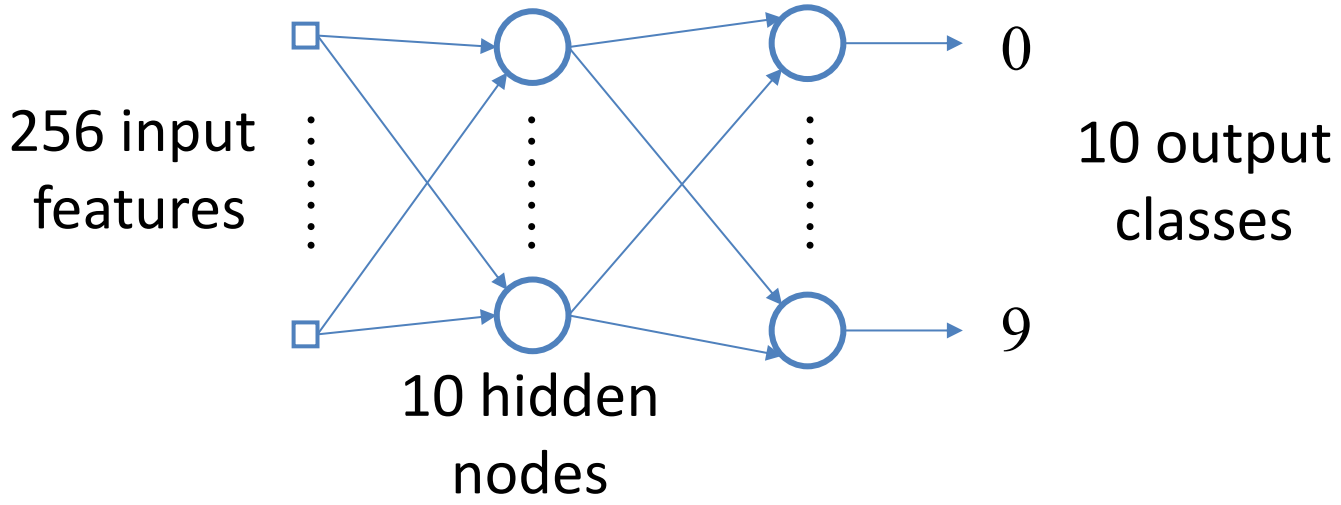


All 10 classes



PCA – Example, cont.

Say we try to classify with a
neural network with 10 hidden nodes

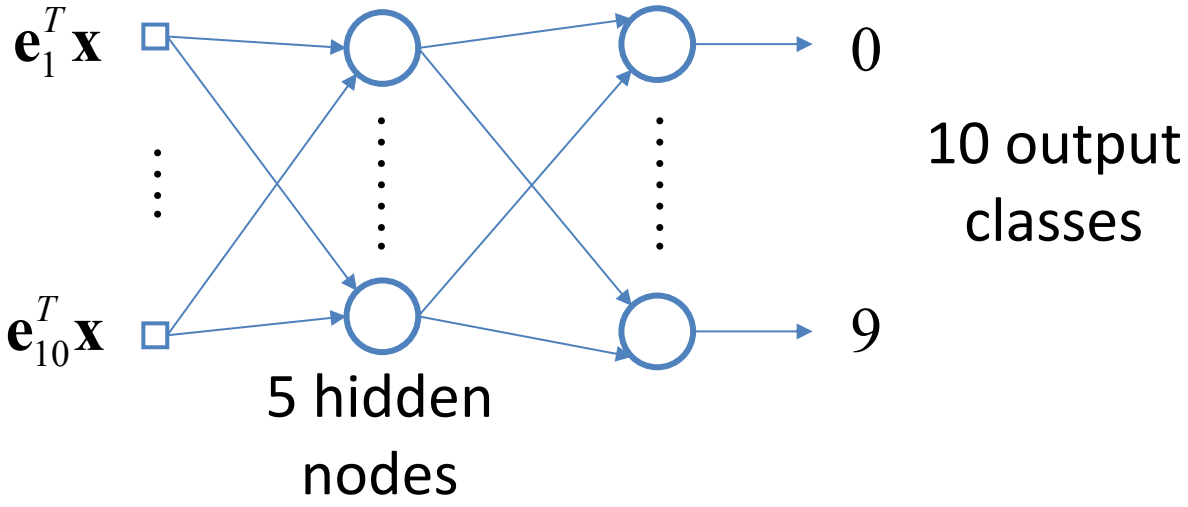


parameters in network: 257x10 + 11x10 = 2680

(including bias weights)

PCA – Example, cont.

If we reduce the input dimensionality first, we may be able to do the classification with a smaller network, e.g., 10 principal components as input and 5 hidden nodes.

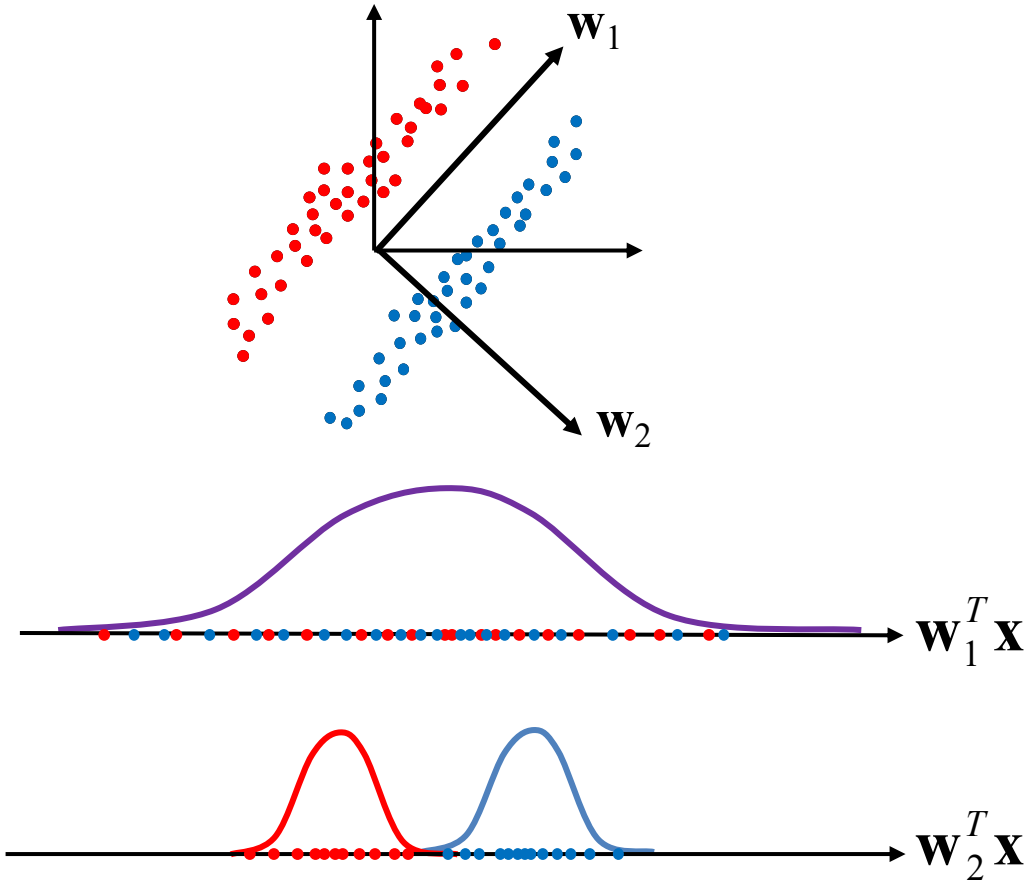


parameters in network: 11x5 + 6x10 = 115

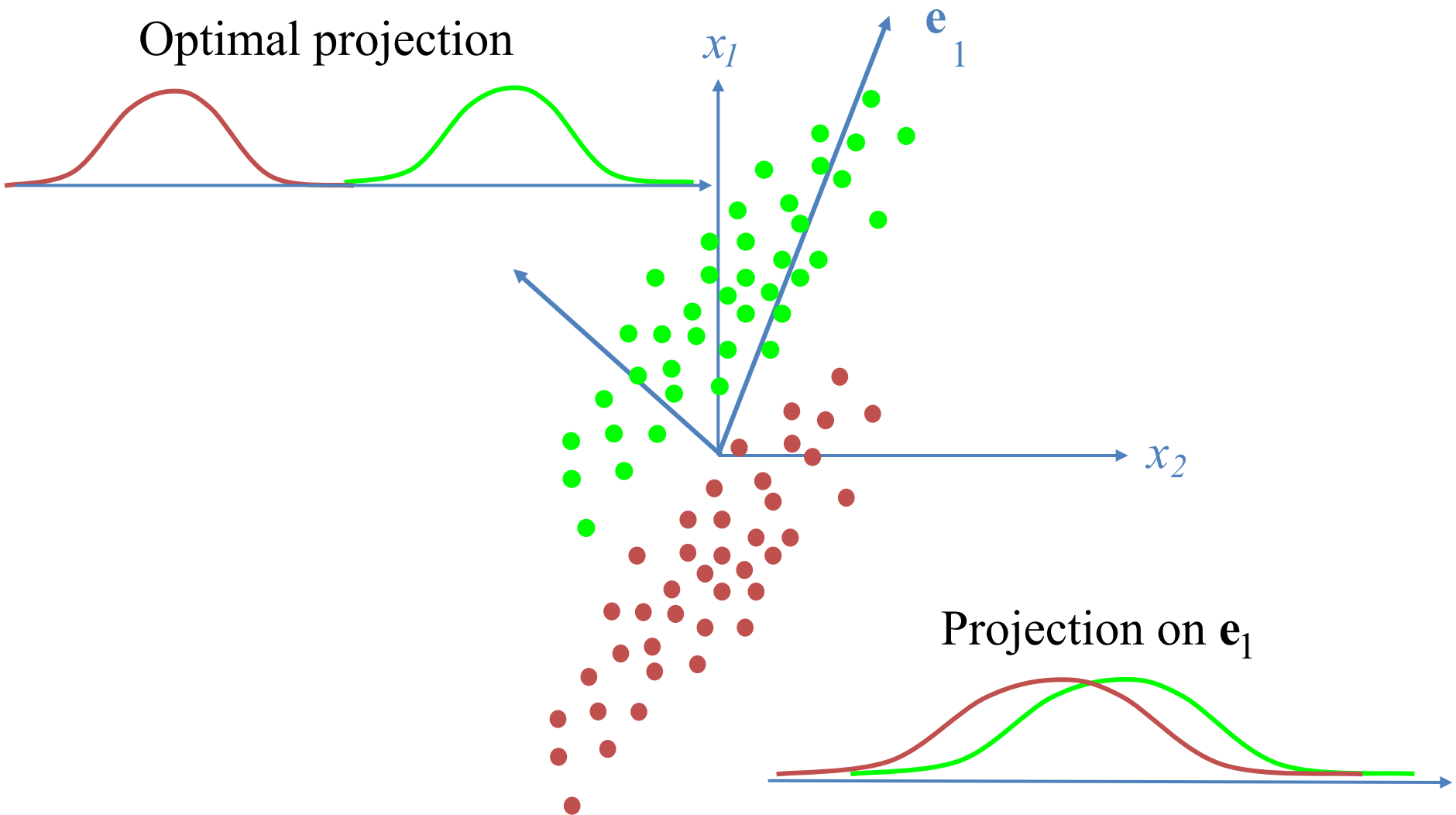
(including bias weights)

Limitations with PCA

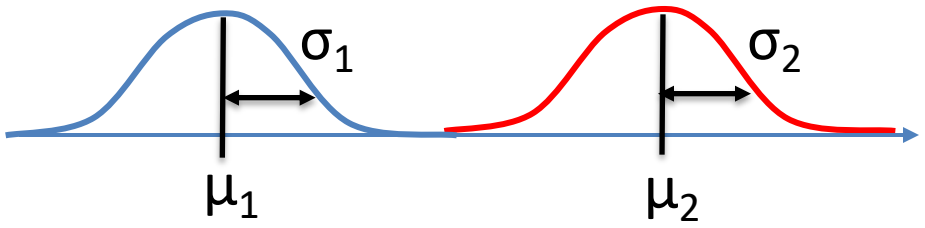
Variance is not always the most important goal!



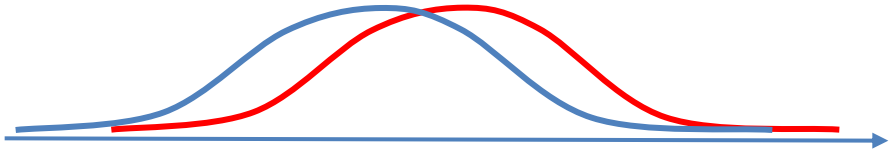
Optimal projection for separation of two clusters



Class separability



- Small variance
- Large distance



- Large variance
- Small distance

Goal: minimize variance and maximize distance.

Linear Discriminant Analysis (LDA)

a.k.a. Fishers Linear Discriminant (FLD)

- Minimize variance
- Maximize distance

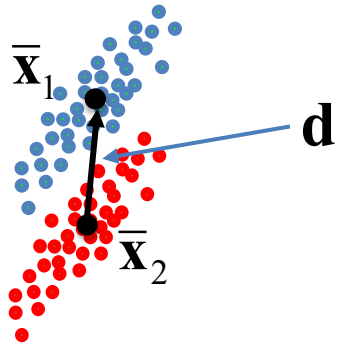
Maximize: $\varepsilon(\mathbf{w}) = \dfrac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$

LDA – Loss function

$$\varepsilon = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$

Distance:

$$\mu(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{w}^T \mathbf{x}_i = \mathbf{w}^T \left(\frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \right) = \mathbf{w}^T \bar{\mathbf{x}}$$



$$\left(\mu_1(\mathbf{w}) - \mu_2(\mathbf{w})\right)^2 = (\mathbf{w}^T\bar{\mathbf{x}}_1 - \mathbf{w}^T\bar{\mathbf{x}}_2)^2 = (\mathbf{w}^T\mathbf{d})^2 = \mathbf{w}^T \underbrace{\mathbf{d}\,\mathbf{d}^T}_{\mathbf{M}} \mathbf{w} = \mathbf{w}^T\mathbf{M}\mathbf{w}$$

Variance:

$$\sigma^2(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{w}^T(\mathbf{x}_i - \bar{\mathbf{x}})\right)^2 = \ldots = \mathbf{w}^T \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \right)}_{\mathbf{C}} \mathbf{w} = \mathbf{w}^T\mathbf{C}\mathbf{w}$$

$$\sigma_1^2(\mathbf{w}) + \sigma_2^2(\mathbf{w}) = \mathbf{w}^T\mathbf{C}_1\mathbf{w} + \mathbf{w}^T\mathbf{C}_2\mathbf{w} = \mathbf{w}^T\mathbf{C}_{tot}\mathbf{w}$$

Note the assumption $\mathbf{C}_1 = \mathbf{C}_2$!

Exercise:
Complete all the steps!

LDA – Solution

Compare with PCA!

$$\varepsilon(\mathbf{w}) = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} = \frac{\mathbf{w}^T \mathbf{M} \mathbf{w}}{\mathbf{w}^T \mathbf{C}_{tot} \mathbf{w}}$$

This form is called a Rayleigh quotient,
which is maximized by the largest eigenvector to the
generalized eigenvalue problem $\mathbf{C}_{\text{tot}}\mathbf{w} = \lambda\, \mathbf{M}\mathbf{w}$!

Simplification: $\mathbf{M}\mathbf{w} = \mathbf{d}\underbrace{\mathbf{d}^T\mathbf{w}}_{\text{Some scalar } k} = k\mathbf{d}$

$$\mathbf{w} \sim \mathbf{C}_{\text{tot}}^{-1}\mathbf{d}$$

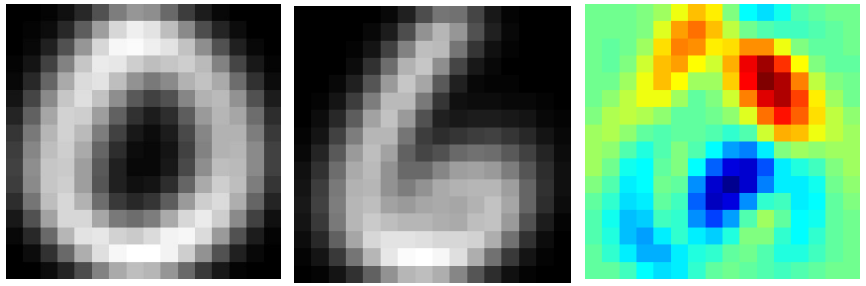
Scaling of **w** not important!

LDA - Example

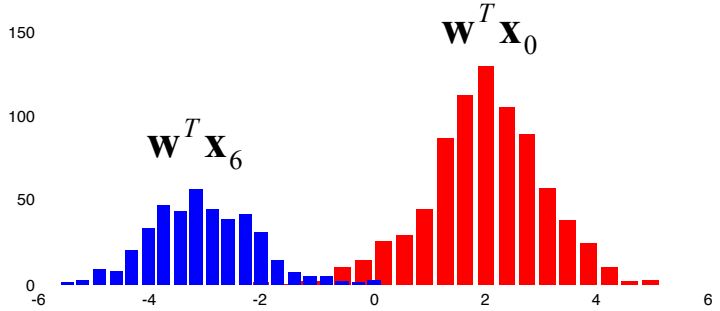
Feature vectors

$$\mathbf{x}_0 = \begin{pmatrix} x_1 \\ \vdots \\ x_{256} \end{pmatrix}_{\text{0-digits}} \qquad \mathbf{x}_6 = \begin{pmatrix} x_1 \\ \vdots \\ x_{256} \end{pmatrix}_{\text{6-digits}}$$

$\bar{\mathbf{x}}_0$ $\bar{\mathbf{x}}_6$ $\mathbf{w} \sim \mathbf{C}_{tot}^{-1}\left(\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_6\right)$

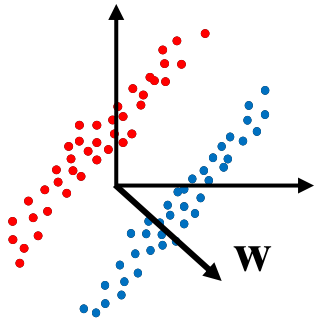


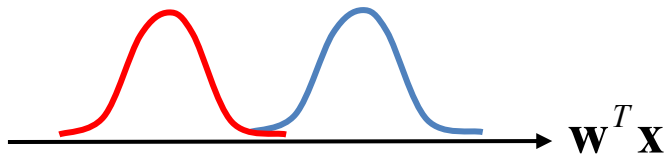
Reduction from 256 to 1 dimension



LDA - Summary

- Projection direction **w** on which two classes are maximally separated

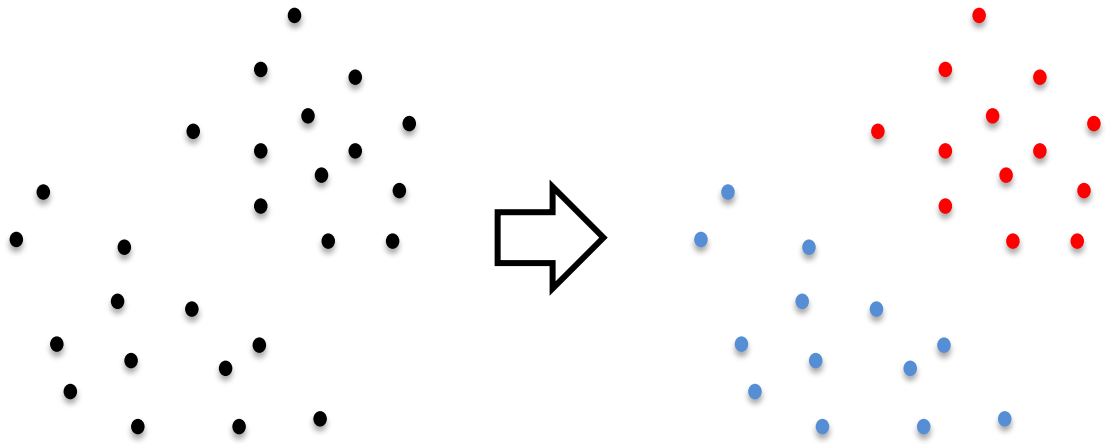




- Closed form solution, no parameters to set, no local optima.
- Components of **w** give the importance of each feature in **x**.
- Assumes identical distributions of the two classes!
- Note that this is actually supervised learning since labelled data is required!

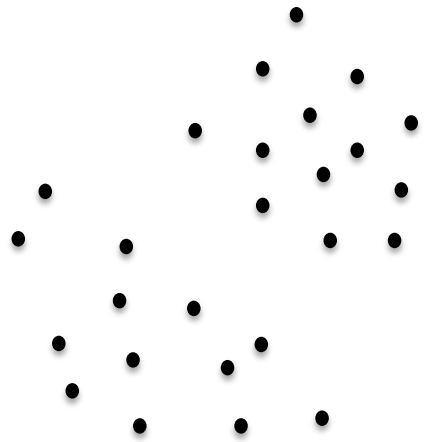
Categorization

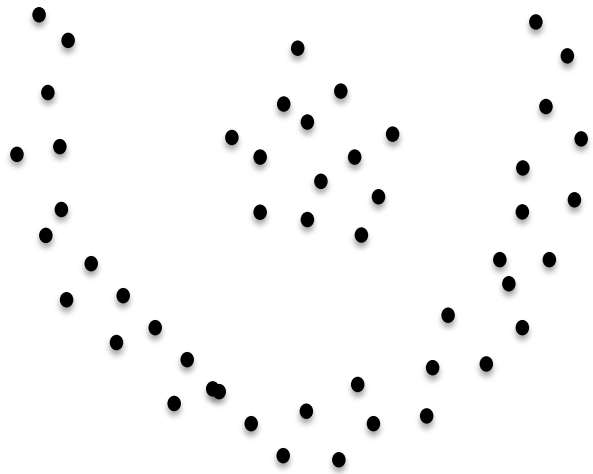
- Categorization and grouping of objects based on similar properties is an important functionality in learning and knowledge representation.
- In the machine learning area, this is usually referred to as ***clustering***.



What describes a cluster?

- Distances to other points?
- Connectivity?
- Different definitions lead to different algorithms.





k-Means algorithm

- Assume *k* clusters (user input).
- Represent each cluster with a mean prototype vector $\mathbf{p}_j$ at the cluster center.
- A data point belongs to the cluster with the closest prototype vector (Euclidian distance).

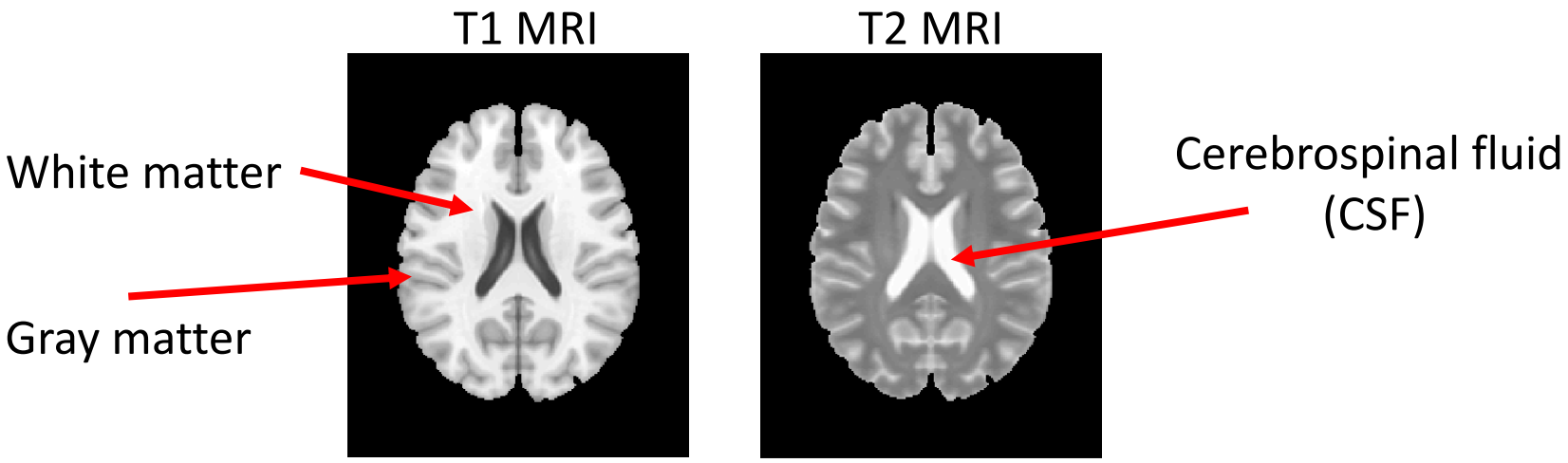
k-Means algorithm, cont.

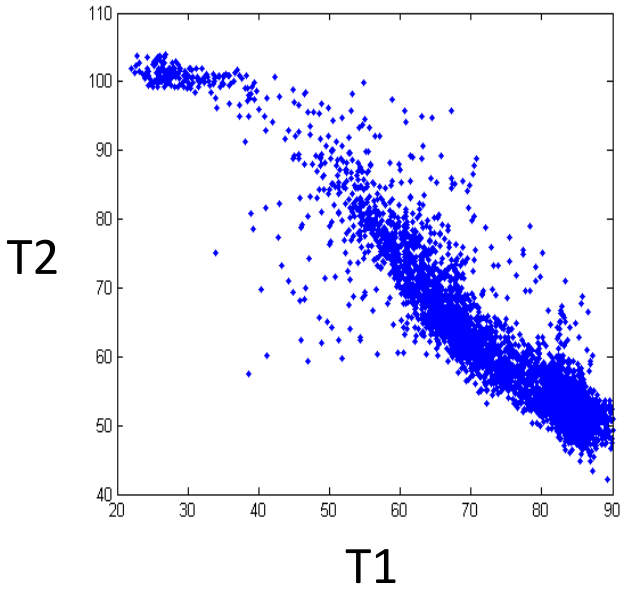
1. Start with *k* random prototype vectors $\mathbf{p}_j$
2. Iterate:
   1. Assignment: Assign each data vector $\mathbf{x}_i$ to the closest prototype vector $\mathbf{p}_j$. Denote the set of data vectors assigned to cluster $\mathbf{p}_j$ by S_j.
   2. Update all prototype vectors to the mean of the clusters:

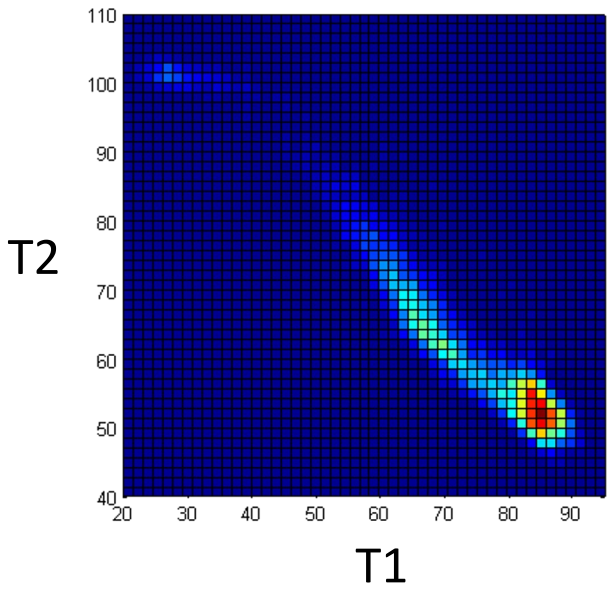
$$\mathbf{p}_j = \frac{1}{|S_j|} \sum_{k \in S_j} \mathbf{x}_k$$

Number of elements in S

k-Means - Example

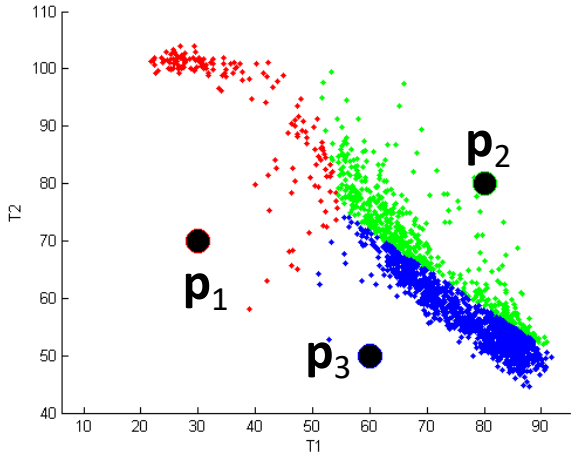




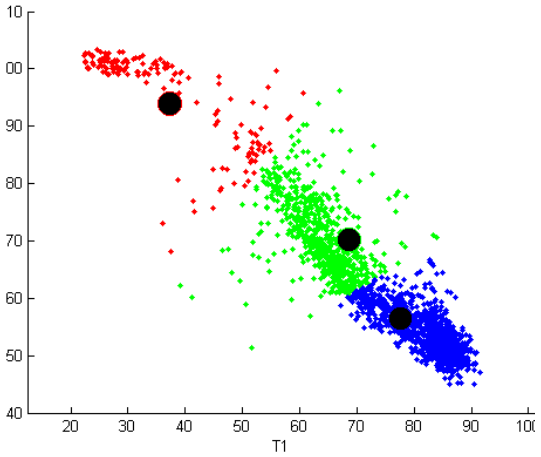


k-Means - Example

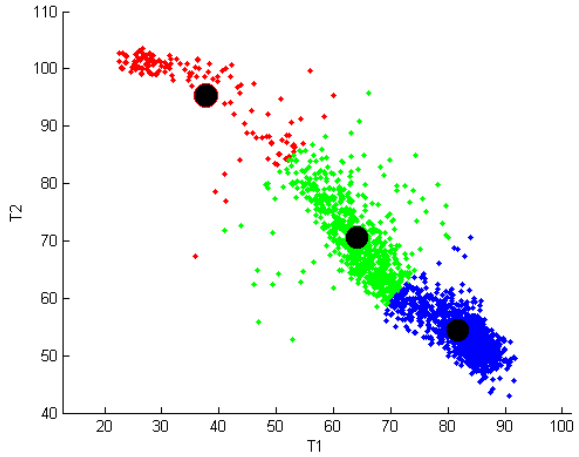
Random initialization



1st iteration

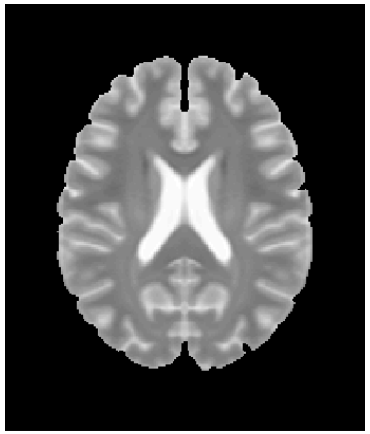


2nd iteration

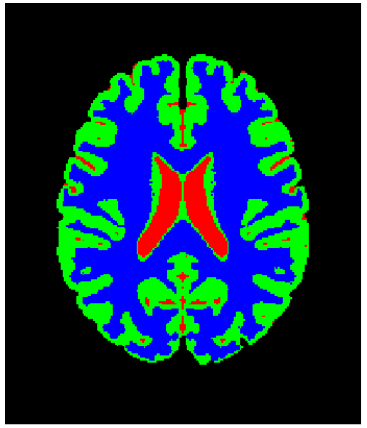


T1 MRI

T2 MRI



k-means result



k-Means - Discussion

- Must specify *k*
- Tries to minimize the loss function

$$\varepsilon(\mathbf{p}_1,\ldots\mathbf{p}_k,S_1,\ldots,S_k)=\sum_{i=1}^{k}\sum_{\mathbf{x}_j\in S_i}\left\|\mathbf{x}_j-\mathbf{p}_i\right\|^2$$

- Note that this function is not differentiable as the S_i:s are discrete sets, i.e., we cannot do gradient descent.

Expectation Maximization (EM) – Intro

$$\varepsilon(\mathbf{p}_1,\ldots\mathbf{p}_k, S_1,\ldots,S_k) = \sum_{i=1}^{k}\sum_{\mathbf{x}_j\in S_i}\left\|\mathbf{x}_j-\mathbf{p}_i\right\|^2$$

Unknown class labels

Assume we know $S_1,..,S_k$!

$$\frac{\partial\varepsilon}{\partial\mathbf{p}_i} = \frac{\partial}{\partial\mathbf{p}_i}\left(\sum_{\mathbf{x}_j\in S_i}\left\|\mathbf{x}_j-\mathbf{p}_i\right\|^2\right) = -2\sum_{\mathbf{x}_j\in S_i}\left(\mathbf{x}_j-\mathbf{p}_i\right) = 2\left|S_i\right|\mathbf{p}_i - 2\sum_{\mathbf{x}_j\in S_i}\mathbf{x}_j$$

$$\frac{\partial\varepsilon}{\partial\mathbf{p}_i} = 0 \quad\Rightarrow\quad \mathbf{p}_i = \frac{1}{\left|S_i\right|}\sum_{\mathbf{x}_j\in S_i}\mathbf{x}_j$$

Mean vector!

Expectation Maximization (EM) – Intro

$$\varepsilon\left(\underbrace{\mathbf{p}_1,\ldots\mathbf{p}_k}_{\text{Continuous parameters}},S_1,\ldots,S_k\right)=\sum_{i=1}^{k}\sum_{\mathbf{x}_j\in S_i}\left\|\mathbf{x}_j-\mathbf{p}_i\right\|^2$$

Now, assume we know $\mathbf{p}_1,..,\mathbf{p}_k$!

Each $\mathbf{x}_j$ independently contributes a distance $\|\mathbf{x}_j - \mathbf{p}_i\|$ to ε

Obvious that ε is minimized if each $\mathbf{x}_j$ is assigned to the set S associated with the closest $\mathbf{p}$!

k-Means algorithm!!

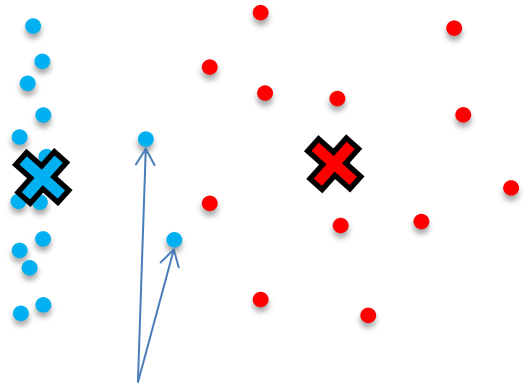
Expectation Maximization

- The *k*-means algorithm is a special case of a general optimization approach called *Expectation Maximization* (EM).
- EM can be used when we want to estimate model parameters (the prototypes $\mathbf{p}_i$), but for each data sample $\mathbf{x}_j$ there is a hidden/missing discrete parameter (the class labels S_j).
- EM iterates between optimizing the hidden parameters and the model parameters.

Mixture of Gaussians (MoG) clustering

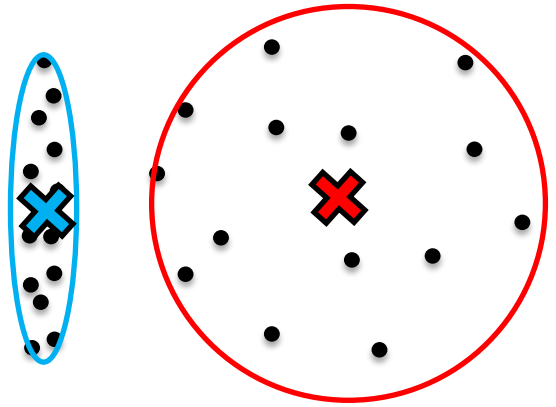
Another application of EM!

k-means



May not be correctly clustered!

Idea: Represent each cluster using a Gaussian distribution



Problem: Each data sample $\mathbf{x}_j$ belongs to one of k Gaussian distributions $N(\mathbf{p}_i,\mathbf{C}_i)$, i=1..k.

Find the sets S_i of samples that belong to distribution $N(\mathbf{p}_i,\mathbf{C}_i)$ and the mean $\mathbf{p}_i$ and covariance matrix $\mathbf{C}_i$ of that distribution.

Let's use EM!

Assume we know the hidden parameters $S_1,..,S_k$!

That is, we know the samples for each set, e.g., $S_1 = \{\mathbf{x}_1, \mathbf{x}_7, \mathbf{x}_{12}, \mathbf{x}_{13}\}$

We can then estimate the mean $\mathbf{p}_i$ and covariance $\mathbf{C}_i$ using standard estimation for the Gaussian:

$$\mathbf{p}_i = \frac{1}{|S_i|} \sum_{k \in S_i} \mathbf{x}_k$$

$$\mathbf{C}_i = \frac{1}{|S_i|} \sum_{k \in S_i} (\mathbf{x}_k - \mathbf{p}_i)(\mathbf{x}_k - \mathbf{p}_i)^T$$

Let's use EM, cont!

Assume now that we know the Gaussian distribution parameters, i.e., we know all distributions

$$f\left(\mathbf{x};\mathbf{p}_i,\mathbf{C}_i\right)=\frac{1}{\left(2\pi\right)^{d/2}\left|\mathbf{C}_i\right|^{1/2}}e^{-\frac{1}{2}\left(\mathbf{x}-\mathbf{p}_i\right)^T\mathbf{C}_i^{-1}\left(\mathbf{x}-\mathbf{p}_i\right)}$$

Let the hidden parameter S_i be the set of all data samples for which $f(\mathbf{x};\mathbf{p}_i,\mathbf{C}_i)$ is larger than for all other distributions. That is, each data sample is assigned to the Gaussian distribution **to which it most likely belongs**!

Mixture of Gaussians – Using EM

1. Start with *k* random Gaussians ($\mathbf{p}_j, \mathbf{C}_j$)
2. Iterate:
   1. Assign each sample $\mathbf{x}_i$ to the most likely Gaussian

$$\max f\left(\mathbf{x}_i; \mathbf{p}_j, \mathbf{C}_j\right) = \frac{1}{\left(2\pi\right)^{d/2}\left|\mathbf{C}_j\right|^{1/2}} e^{-\frac{1}{2}\left(\mathbf{x}_i-\mathbf{p}_j\right)^T \mathbf{C}_j^{-1}\left(\mathbf{x}_i-\mathbf{p}_j\right)}$$

   Denote the set of samples assigned to Gaussian j by S_j.

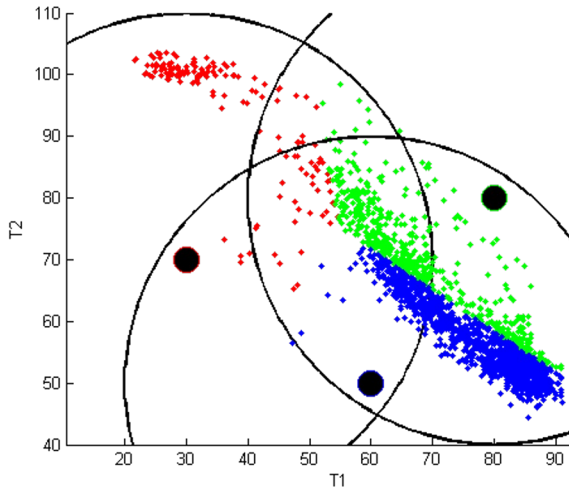
   2. Update all Gaussian's means and covariances:

$$\mathbf{p}_j = \frac{1}{\left|S_j\right|}\sum_{k\in S_j}\mathbf{x}_k$$

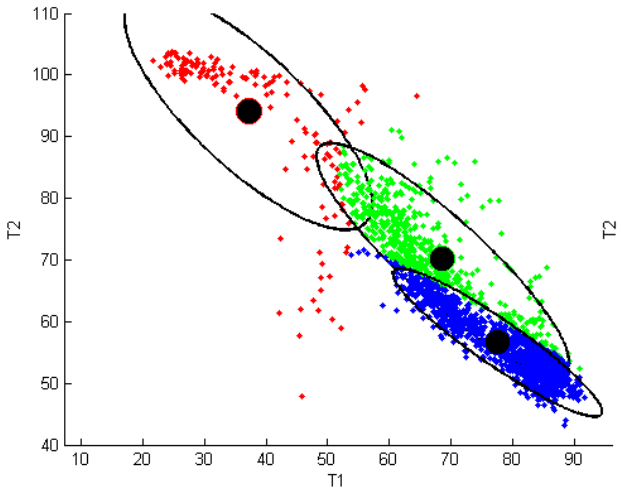
$$\mathbf{C}_j = \frac{1}{\left|S_j\right|}\sum_{k\in S_j}\left(\mathbf{x}_k-\mathbf{p}_i\right)\left(\mathbf{x}_k-\mathbf{p}_i\right)^T$$

Mixture of Gaussians - Example

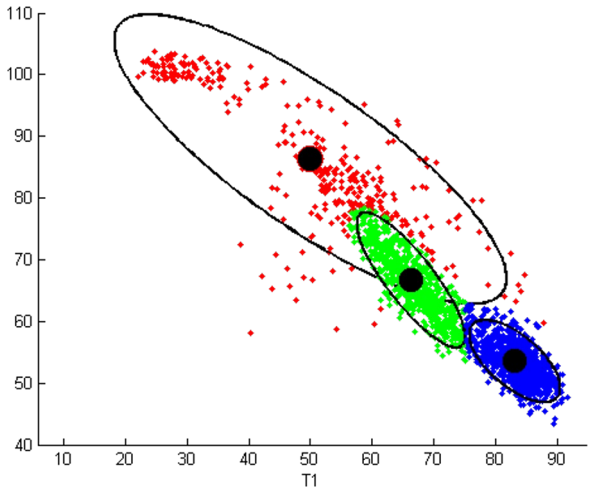
Random initialization



1st iteration

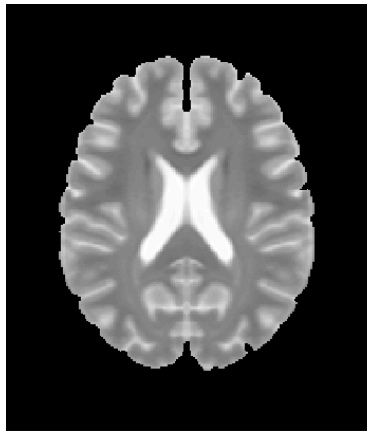


7th iteration

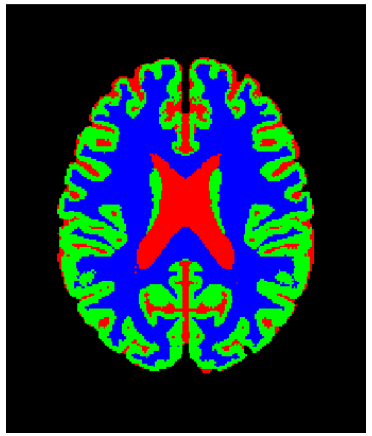


T1 MRI

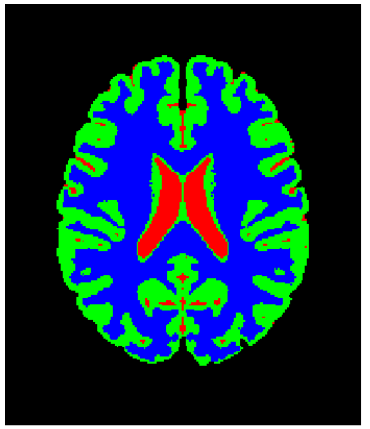
T2 MRI



MoG result



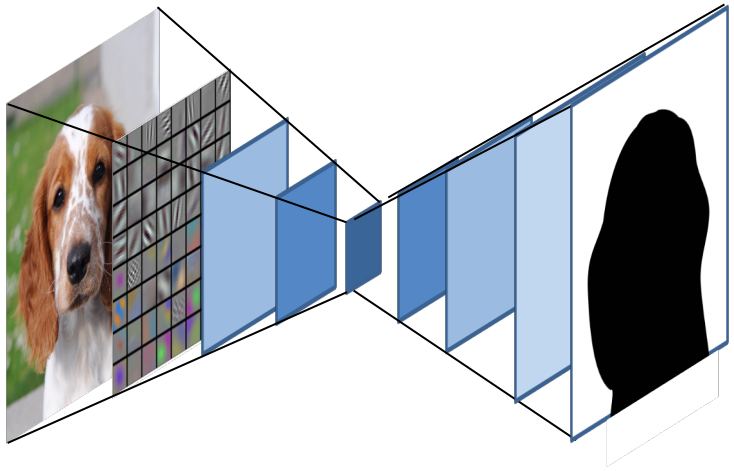
k-means result



Summary of *k*-Means and MoG clustering

- Must choose the number of clusters *k*.
- Different initializations may give different results.
- May converge to degenerate solutions, e.g., empty clusters.
- MoG allows for elliptic cluster shapes with different sizes.

Auto Encoders



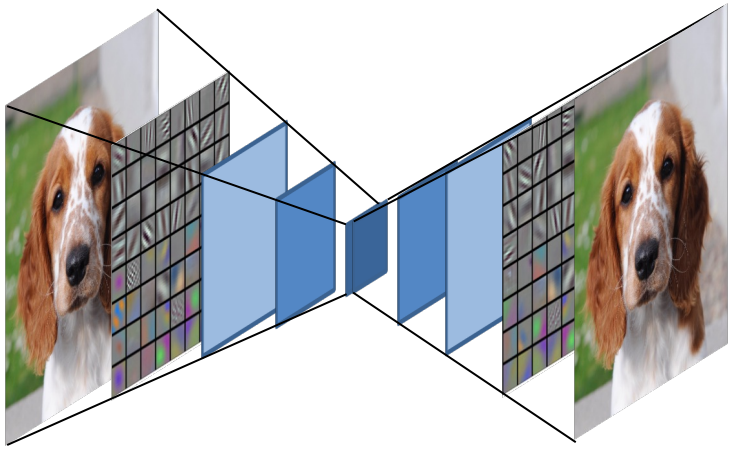


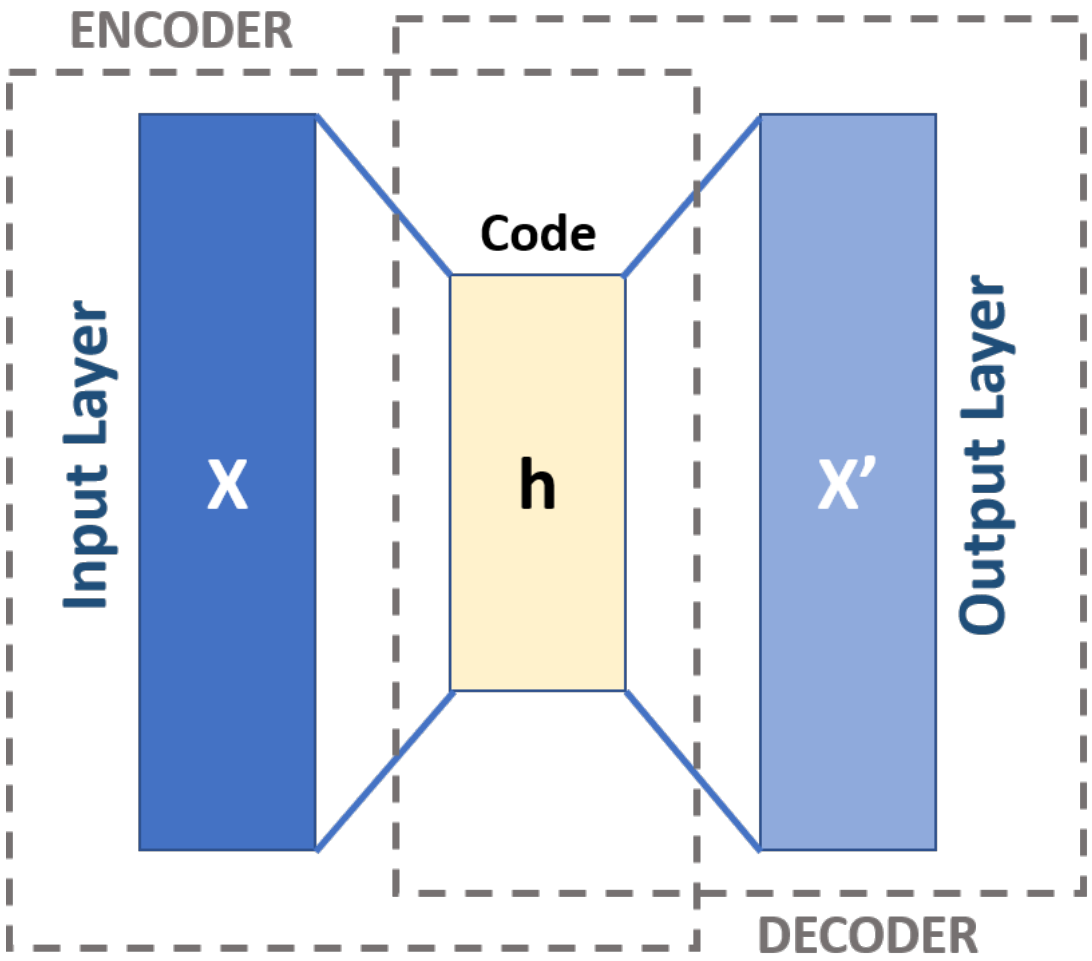
Image segmentation

- Training data $\{\mathbf{x}_i , \mathbf{y}_i\}$
- Binary output (foreground / background)
- Classification of each pixel

Auto encoder

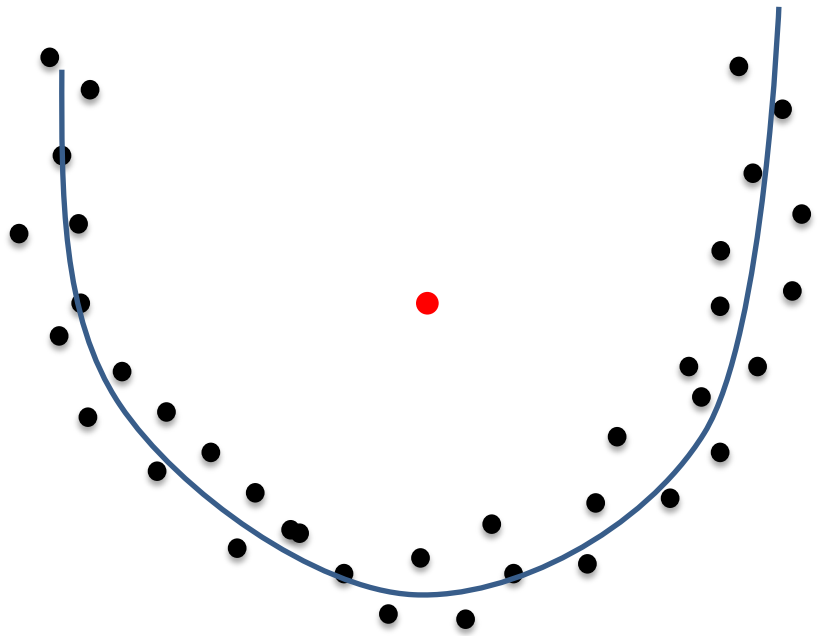
- Training data $\{\mathbf{x}_i , \mathbf{x}_i\}$
- Does not require labelled training data – Unsupervised learning!
- The smallest layer must be able to represent all training data
- The bottle neck forces an efficient coding of the original data (images)

Auto Encoders



- The hidden representation **h** is a compact encoding of the distribution of training data **x**
- If the dimensionality of **h** is chosen properly, **h** can represent all training data well, but not data points that deviate from the distribution of **x**

Data manifold



- The data lies in a low-dimensional manifold in a high-dimensional feature space.
- An auto encoder learns to represent the manifold
- Data points that are outside the manifold cannot be accurately represented

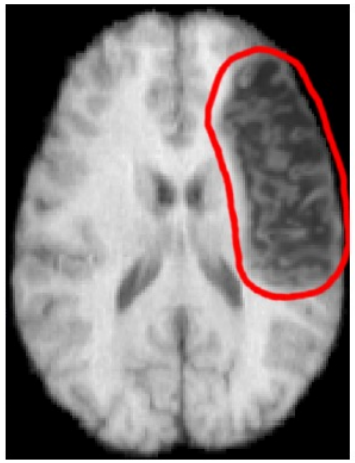
Outlier detection

- Data points that deviate from the distribution of normal samples (the learned manifold)
- These data points can indicate anomalies such as production errors or pathologies

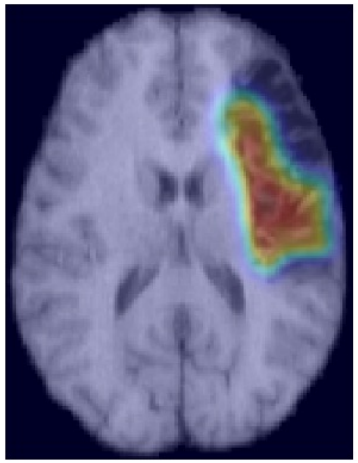
Application in Medical Image Analysis

- Train on “normal images”
- Compare input and output
- Use the residual as indication of abnormality

Normal brain



Brain with lesion



Residual overlay

Uzunova, H. *et al.* *Int J CARS* **14,** 451–461 (2019).