Neural Networks and Learning Systems
TBMI 26, 2017

**Lecture 6**

**Unsupervised Learning**

*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
- **Unsupervised learning (descriptive)**
  Discover structure and relationships in complex high-dimensional data.
- **Reinforcement learning (active)**
  Generate policies/strategies that lead to a (possibly delayed) reward. Learning by doing.

## 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 cost function, e.g.
  - max variance (PCA)
  - max class separability (LDA)
- Finds a new <u>representation</u> of the data

## Applications

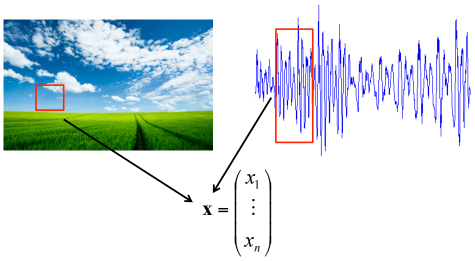
- Feature extraction
  - find order or structure in data
- Dimensionality reduction
  - keep the most "important" parts of the signal

## 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

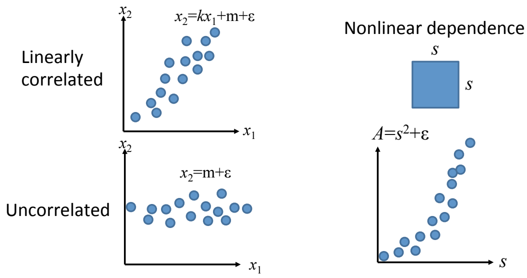
## Structure in signals




Lots of structure (autocorrelation) in signals/images!
Generally **not** a good idea to use data samples directly as features.

## Data/feature dependence

Want features that carry *independent* information.
Simpler but weaker condition is to use *uncorrelated* variables.



## Independent vs. uncorrelated

- Statistically independent means that there is no relationship (linear or nonlinear) between variables. **Independent -> uncorrelated**.
- Uncorrelated means that there is no <u>linear</u> relationship between variables. **Uncorrelated does not imply independent**.
- Special case: For Gaussian distributions, uncorrelated also means independent.

## Describing linear data dependence

Mean value operator

$$\mathrm{Var}(x) = \sigma^2 = \mathrm{E}\left[(x-\bar{x})^2\right] = \frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})^2$$

$$\mathrm{Cov}(x,y) = \mathrm{E}\left[(x-\bar{x})(y-\bar{y})\right] = \frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y})$$

$$\rho(x,y) = \mathrm{Corr}(x,y) = \frac{\mathrm{Cov}(x,y)}{\sqrt{\mathrm{Var}(x)\mathrm{Var}(y)}} \qquad -1 \le \rho \le 1$$

**Scaling:** $\mathrm{Var}(a\cdot x) = a^2\mathrm{Var}(x)$ Influenced by scaling

$\rho(a\cdot x, b\cdot y) = \rho(x,y)$ Invariant to scaling

## Multidimensional linear dependence

$$\mathbf{C} = \mathrm{Cov}(\mathbf{x}) = \mathrm{E}\left[(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^T\right] = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i-\bar{\mathbf{x}})(\mathbf{x}_i-\bar{\mathbf{x}})^T$$

Example:

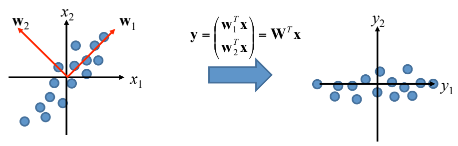
Symmetric covariance matrix

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \qquad \mathbf{C} = \begin{bmatrix} \mathrm{Var}(x_1) & \mathrm{Cov}(x_1,x_2) & \mathrm{Cov}(x_1,x_3) \\ \mathrm{Cov}(x_2,x_1) & \mathrm{Var}(x_2) & \mathrm{Cov}(x_2,x_3) \\ \mathrm{Cov}(x_3,x_1) & \mathrm{Cov}(x_3,x_2) & \mathrm{Var}(x_3) \end{bmatrix}$$

Sometimes we also use the correlation matrix

$$\mathbf{R} = \begin{bmatrix} 1 & \mathrm{Corr}(x_1,x_2) & \mathrm{Corr}(x_1,x_3) \\ \mathrm{Corr}(x_2,x_1) & 1 & \mathrm{Corr}(x_2,x_3) \\ \mathrm{Corr}(x_3,x_1) & \mathrm{Corr}(x_3,x_2) & 1 \end{bmatrix}$$

## Projection/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!!

$$\mathbf{C}_x = \begin{bmatrix} \mathrm{Var}(x_1) & \mathrm{Cov}(x_1,x_2) \\ \mathrm{Cov}(x_2,x_1) & \mathrm{Var}(x_2) \end{bmatrix} \qquad \mathbf{C}_y = \begin{bmatrix} \mathrm{Var}(y_1) & 0 \\ 0 & \mathrm{Var}(y_2) \end{bmatrix}$$

## 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).

## Dimensionality reduction with least mean square error

- Chose a new basis whose dimensionality is smaller than the original
- The basis vectors that preserve maximum variance gives the least mean square error.

## Dimensionality reduction with least mean square error

Suppose **y** is a representation of **x** where some dimensions are removed, i.e. some $y_i = 0$.

(Assume mean = 0)

$$\varepsilon = E\left[\sum_i (x_i - y_i)^2\right] = E\left[\sum_k x_k^2\right] = \sum_k E[x_k^2] = \sum_k \sigma_k^2$$

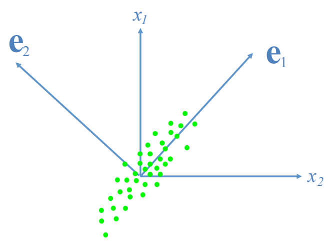
$k$ is the indices of the removed components, i.e. $k : y_k = 0$

## Dimensionality reduction with least mean square error

- Expected mean square error = sum of the variance in the removed dimensions.
- The mean square error is minimized by keeping the components with highest variance.

## PCA

Principal Component Analysis




## Maximize the variance

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

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

$$= \hat{\mathbf{w}}^T \underbrace{E[\mathbf{x}\mathbf{x}^T]}_{\text{The covariance matrix of } \mathbf{x}.}\hat{\mathbf{w}} = \hat{\mathbf{w}}^T\mathbf{C}\hat{\mathbf{w}} = \frac{\mathbf{w}^T\mathbf{C}\mathbf{w}}{\mathbf{w}^T\mathbf{w}}$$

## Maximize the variance

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

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

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

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

## The Eigen-value decomposition

$$\mathbf{C}\mathbf{e} = \lambda\mathbf{e}$$

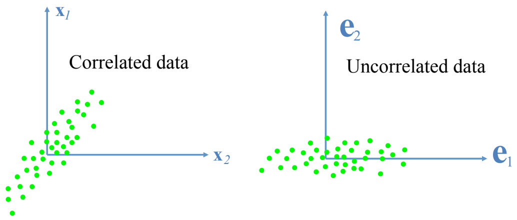
$$\mathbf{C} = \sum_i \lambda_i \mathbf{e}_i \mathbf{e}_i^T$$

If $\mathbf{C}$ is a covariance matrix:

$$\mathbf{C} = E[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T] \qquad \lambda_i = \sigma^2_{\mathbf{e}_i}$$

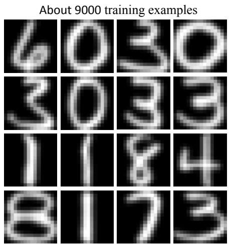
## Uncorrelated components

$$E[\mathbf{e}_i^T \mathbf{x}\mathbf{x}^T \mathbf{e}_j] = \mathbf{e}_i^T \mathbf{C}\mathbf{e}_j = \lambda_j \mathbf{e}_i^T \mathbf{e}_j = 0$$



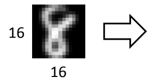

## PCA - Example

About 9000 training examples



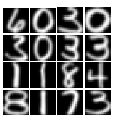
US Postal Service Digit Data
http://www.gaussianprocess.org/gpml/data/

Feature vectors




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

## PCA – Example, cont.



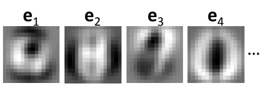
Mean digit

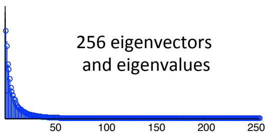
$\bar{\mathbf{x}} =$ 

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

256x256 matrix

Eigendecomposition of **C**:



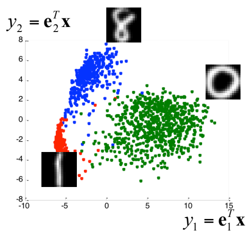




256 eigenvectors and eigenvalues

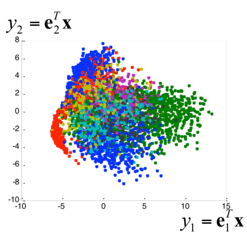
## PCA – Example, cont.

From 256 to 2 feature dimensions!

3 classes

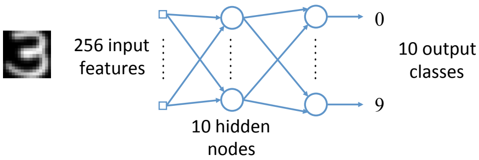



All 10 classes



## PCA – Example, cont.

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

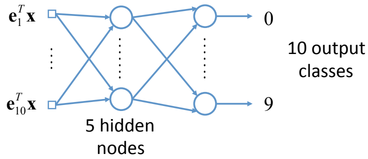



# parameters in network: 256x10 + 10x10 = 2700

(ignoring the 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.

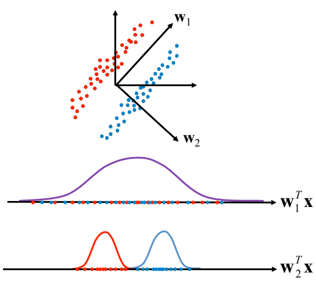



# parameters in network: 10x5 + 5x10 = 100

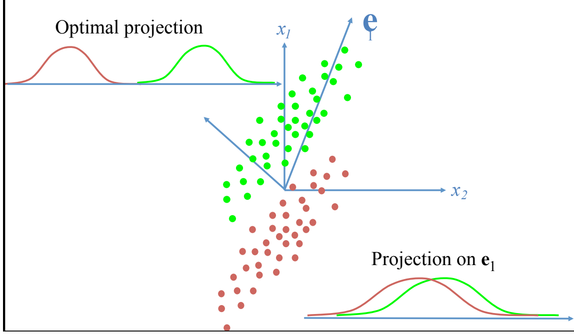
(ignoring the bias weights)

## Limitations with PCA

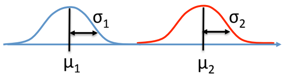
Variance is not always the most important goal!



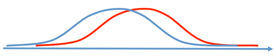

## Optimal projection for separation of two clusters



## Class separability




- Small variance
- Large distance



- Large variance
- Small distance

Goal: <u>minimize variance</u> and <u>maximize distance</u>.

## Linear Discriminant Analysis (LDA)

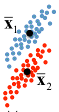
a.k.a. Fishers Linear Discriminant (FLD)

- Minimize variance
- Maximize distance

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

## LDA – Cost function

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




Distance:

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

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

Variance:

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

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

Exercise:
Complete all the steps!

## LDA – Solution

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

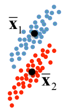
Compare with PCA!

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

Simplification: $\mathbf{M}\mathbf{w} = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)\underbrace{(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T \mathbf{w}}_{\text{Some scalar } K} = K(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$

$$\boxed{\mathbf{w} \sim \mathbf{C}_{tot}^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)}$$

Scaling of **w** not important!

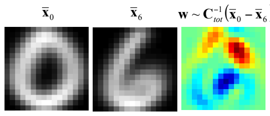


## LDA - Example

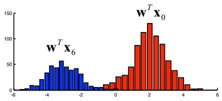
Feature vectors ⇒

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

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

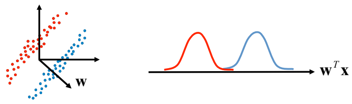


Reduction from 256 to 1 dimension




## LDA - Summary

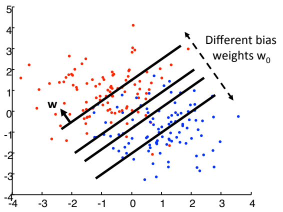
- Projection direction **w** on which two classes are maximally separated



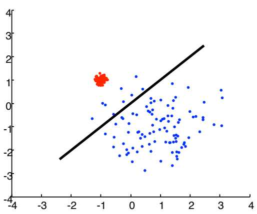

- Closed form solution, no parameters to set – easy to calculate.
- Components of **w** give the importance of each feature in **x**.
- Can find a series of directions (as in PCA) by applying the algorithm several times – *orthogonal LDA/FLD*
- Can be used as a classifier!

## LDA as a classifier

Handles overlapping classes. Must decide the bias weight to define exact separation line.




Approximates the training data with a distribution. Loose some precision when setting the exact separation line.



## LDA as a classifier

- Easy to use!
  - Closed form solution
    - Fast to calculate
    - No dependency on initialization
    - No step length to choose
    - No local optima
  - No parameters to set

Very useful as a first classifier to try and as a benchmark!