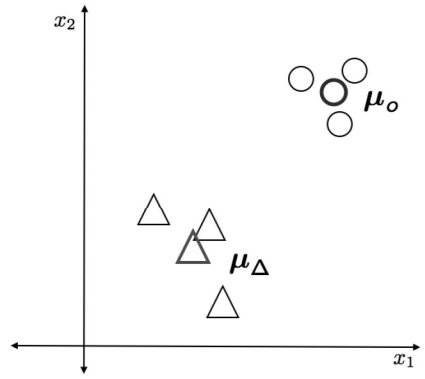
Machine Learning

Lecture 7
Perceptrons

Felix Bießmann

Beuth University & Einstein Center for Digital Future

Recap: Nearest Centroid Classification



Prototypes $\boldsymbol{\mu}_\Delta$ and $\boldsymbol{\mu}_o$ can be the class means

$$\boldsymbol{\mu}_\Delta = 1/N_\Delta \sum_n^{N_\Delta} \mathbf{x}_{\Delta,n}$$

$$\boldsymbol{\mu}_o = 1/N_o \sum_n^{N_o} \mathbf{x}_{o,n}$$

Distance from w_Δ to new data x

$$\|\boldsymbol{\mu}_\Delta - \mathbf{x}\|_2$$

From Prototypes to Linear Classification

$$\text{distance}(\mathbf{x}, \boldsymbol{\mu}_\Delta) > \text{distance}(\mathbf{x}, \boldsymbol{\mu}_o) \quad (1)$$
$$\|\mathbf{x} - \boldsymbol{\mu}_\Delta\| > \|\mathbf{x} - \boldsymbol{\mu}_o\|$$

From Prototypes to Linear Classification

$$\text{distance}(\mathbf{x}, \boldsymbol{\mu}_\Delta) > \text{distance}(\mathbf{x}, \boldsymbol{\mu}_o) \quad (1)$$
$$\|\mathbf{x} - \boldsymbol{\mu}_\Delta\| > \|\mathbf{x} - \boldsymbol{\mu}_o\|$$
$$\Leftrightarrow \|\mathbf{x} - \boldsymbol{\mu}_\Delta\|^2 > \|\mathbf{x} - \boldsymbol{\mu}_o\|^2$$
$$\Leftrightarrow \mathbf{x}^\top\mathbf{x} - 2\boldsymbol{\mu}_\Delta^\top\mathbf{x} + \boldsymbol{\mu}_\Delta^\top\boldsymbol{\mu}_\Delta > \mathbf{x}^\top\mathbf{x} - 2\boldsymbol{\mu}_o^\top\mathbf{x} + \boldsymbol{\mu}_o^\top\boldsymbol{\mu}_o$$
$$\Leftrightarrow \boldsymbol{\mu}_\Delta^\top\mathbf{x} - \boldsymbol{\mu}_\Delta^2/2 < \boldsymbol{\mu}_o^\top\mathbf{x} - \boldsymbol{\mu}_o^2/2$$
$$\Leftrightarrow 0 < \underbrace{(\boldsymbol{\mu}_o - \boldsymbol{\mu}_\Delta)}_{\mathbf{w}}^\top\mathbf{x} - 1/2\underbrace{(\boldsymbol{\mu}_o^\top\boldsymbol{\mu}_o - \boldsymbol{\mu}_\Delta^\top\boldsymbol{\mu}_\Delta)}_{\beta}$$

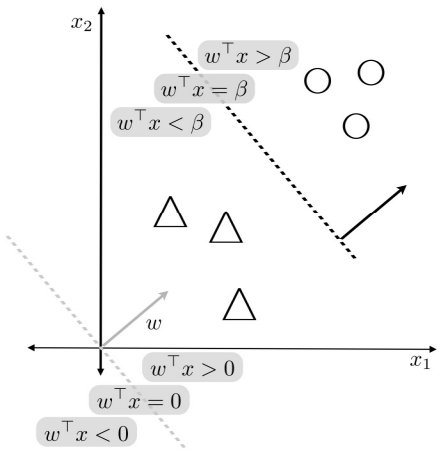
From Prototypes to Linear Classification

$$
\begin{aligned}
\text{distance}(\mathbf{x}, \boldsymbol{\mu}_\Delta) &> \text{distance}(\mathbf{x}, \boldsymbol{\mu}_o) \qquad (1)\\
\|\mathbf{x} - \boldsymbol{\mu}_\Delta\| &> \|\mathbf{x} - \boldsymbol{\mu}_o\| \\
\Leftrightarrow \|\mathbf{x} - \boldsymbol{\mu}_\Delta\|^2 &> \|\mathbf{x} - \boldsymbol{\mu}_o\|^2 \\
\Leftrightarrow \mathbf{x}^\top\mathbf{x} - 2\boldsymbol{\mu}_\Delta^\top\mathbf{x} + \boldsymbol{\mu}_\Delta^\top\boldsymbol{\mu}_\Delta &> \mathbf{x}^\top\mathbf{x} - 2\boldsymbol{\mu}_o^\top\mathbf{x} + \boldsymbol{\mu}_o^\top\boldsymbol{\mu}_o \\
\Leftrightarrow \boldsymbol{\mu}_\Delta^\top\mathbf{x} - \boldsymbol{\mu}_\Delta^2/2 &< \boldsymbol{\mu}_o^\top\mathbf{x} - \boldsymbol{\mu}_o^2/2 \\
\Leftrightarrow 0 &< \underbrace{(\boldsymbol{\mu}_o - \boldsymbol{\mu}_\Delta)}_{\mathbf{w}}{}^\top\mathbf{x} - 1/2\underbrace{(\boldsymbol{\mu}_o^\top\boldsymbol{\mu}_o - \boldsymbol{\mu}_\Delta^\top\boldsymbol{\mu}_\Delta)}_{\beta}
\end{aligned}
$$

Linear Classification

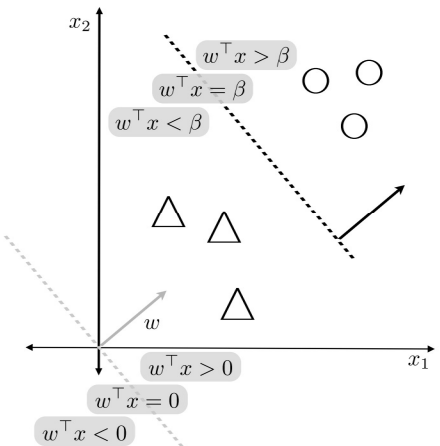
$$
\mathbf{w}^\top\mathbf{x} - \beta = \begin{cases} > 0 & \text{if } \mathbf{x} \text{ belongs to class } \circ \\ < 0 & \text{if } \mathbf{x} \text{ belongs to class } \Delta \end{cases} \qquad (2)
$$

Linear Classification



$$
\mathbf{w}^\top\mathbf{x} - \beta = \begin{cases} > 0 & \text{if } \mathbf{x} \text{ belongs to } o \\ < 0 & \text{if } \mathbf{x} \text{ belongs to } \Delta \end{cases}
$$

Linear Classification



What is a good $\mathbf{w}$?

$\rightarrow$ We need an **error function** that tells us how good $\mathbf{w}$ is.

Two classical Error Functions

Given data $\mathbf{x} \in \mathbb{R}^D$ and corresponding labels $y \in \{-1, +1\}$, two classical error functions $\mathcal{E}(\mathbf{x}, y, \mathbf{w})$ to find the optimal $\mathbf{w} \in \mathbb{R}^D$ are:

Error Function	Used in
$\frac{1}{2}(y - \mathbf{w}^\top\mathbf{x})^2$	Adaline [Widrow and Hoff, 1960]
$\max(0, -y\mathbf{w}^\top\mathbf{x})$	Perceptron [Rosenblatt, 1958]

Rosenblatt's Perceptron

Frank Rosenblatt
(1928-1969)

Rosenblatt proposed the **perceptron**, an artificial neural network for pattern recognition [Rosenblatt, 1958]

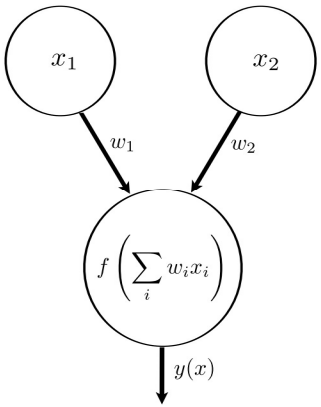
Perceptrons gave rise to the field of artificial neural networks

Rosenblatt's Perceptron

Rosenblatt studied Perceptrons, so that

" [. . .] the fundamental laws of organization which are common to all information handling systems, machines and men included, may eventually be understood."

Artificial Neural Networks



Input nodes x_i receive information

Inputs are multiplied with a weighting factor w_i and summed up

Integrated input is mapped through some (non-linear) function $f(.)$

$$f(x) = \begin{cases} +1 & \text{if } \mathbf{x} \text{ is preferred stimulus} \\ -1 & \text{if } \mathbf{x} \text{ is any other stimulus} \end{cases}$$

Rosenblatt's Perceptron

Perceptron Hardware (pictures from [Bishop, 2007])

The Perceptron Learning Algorithm

Goal Binary classification of multivariate data $\mathbf{x} \in \mathbb{R}^D$

The Perceptron Learning Algorithm

Goal Binary classification of multivariate data $\mathbf{x} \in \mathbb{R}^D$

Input Learning rate η and N tupels $(\mathbf{x}_n, y_n)$ where
$\mathbf{x}_n \in \mathbb{R}^D$ is the D-dimensional data
$y_n \in \{-1, +1\}$ is the corresponding label, such that

$$y_n = \begin{cases} +1 & \text{if } \mathbf{x}_n \text{ is a correct example} \\ -1 & \text{if } \mathbf{x}_n \text{ is } \textbf{not} \text{ a correct example} \end{cases}$$

The Perceptron Learning Algorithm

Goal Binary classification of multivariate data $\mathbf{x} \in \mathbb{R}^D$

Input Learning rate η and N tupels $(\mathbf{x}_n, y_n)$ where
$\mathbf{x}_n \in \mathbb{R}^D$ is the D-dimensional data
$y_n \in \{-1, +1\}$ is the corresponding label, such that

$$y_n = \begin{cases} +1 & \text{if } \mathbf{x}_n \text{ is a correct example} \\ -1 & \text{if } \mathbf{x}_n \text{ is } \textbf{not} \text{ a correct example} \end{cases}$$

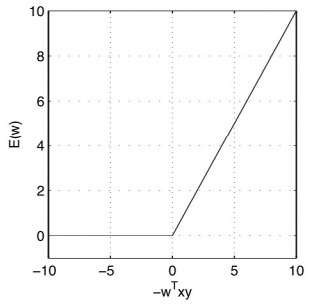
Output Weight vector $w \in \mathbb{R}^D$ such that

$$\mathbf{w}^\top \mathbf{x}_n = \begin{cases} \geq 0 & \text{if } y_n = +1 \\ < 0 & \text{if } y_n = -1 \end{cases}$$

The Perceptron Error Function

$$\mathcal{E}_{\mathcal{P}}(\mathbf{w}) = -\sum_{m \in \mathcal{M}} \mathbf{w}^\top \mathbf{x}_m y_m \tag{3}$$

Classification Error as Function of Weights



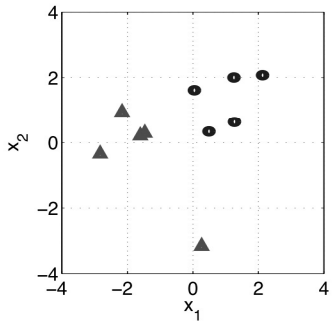
Given data $\mathbf{x} \in \mathbb{R}^D$ and corresponding labels $y \in \{-1, +1\}$ the classification error $\mathcal{E}$ is a function of the weights $\mathbf{w}$ (and the data $\mathbf{x}$, y)

$$\mathcal{E}(\mathbf{w}, \mathbf{x}_m, y_m) = -\sum_{m \in \mathcal{M}} \mathbf{w}^\top \mathbf{x}_m y_m \quad (4)$$

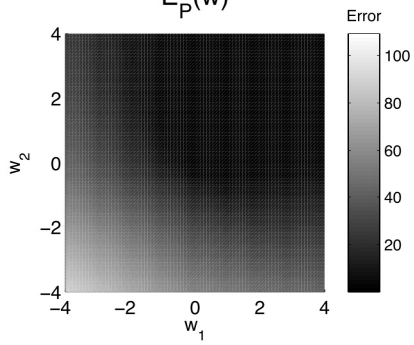
where $\mathcal{M}$ denotes the index set of all *misclassified* data $\mathbf{x}_m$

Classification Error as Function of Weights

Data $\mathbf{x} \in \mathbb{R}^2$

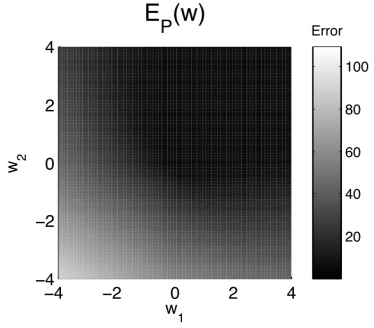


$E_P(w)$



Gradient Descent

$E_P(w)$



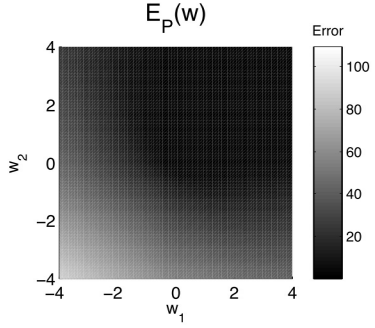
How to minimize the error function?

$$\mathcal{E}(\mathbf{w}, \mathbf{x}_m, y_m) = -\sum_{m \in \mathcal{M}} \mathbf{w}^\top \mathbf{x}_m y_m$$

$\rightarrow$ **Gradient Descent**

Gradient Descent

$E_P(w)$

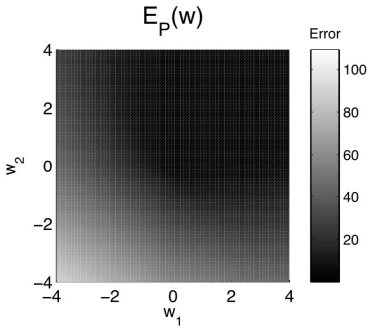


We minimize $\mathcal{E}(\mathbf{w}, \mathbf{x}_m, y_m)$ by walking in the opposite direction of the gradient.

$$\mathbf{w}^{\text{new}} \leftarrow \mathbf{w}^{\text{old}} - \eta \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \nabla \mathcal{E}(\mathbf{w}, \mathbf{x}_i, y_i)$$

where $\mathcal{X}$ is the set of data points and η is called a **learning rate**.

Stochastic Gradient Descent



A noisy estimate of

$$\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \nabla \mathcal{E}(\mathbf{w}, \mathbf{x}, y)$$

is obtained by [Robbins and Monro, 1951]

$$\mathbf{w}^{\text{new}} \leftarrow \mathbf{w}^{\text{old}} - \eta \nabla \mathcal{E}(\mathbf{w}, \mathbf{x}_i, y_i)$$

Note that only $\mathbf{w}$ is stored and only one data point $\mathbf{x}_i$ and label y_i are considered at a time!

$\rightarrow$ Scales to large data sets [Bottou, 2010]

Perceptron Training

Training a Perceptron means finding weights $\mathbf{w}$ that minimize $\mathcal{E}_{\mathcal{P}}$:

$$\underset{w}{\text{argmin}} \quad \left(- \sum_{m \in \mathcal{M}} \mathbf{w}^{\top} \mathbf{x}_m y_m \right) \tag{5}$$

where $\mathcal{M}$ denotes the index set of all *misclassified* data $\mathbf{x}_m$

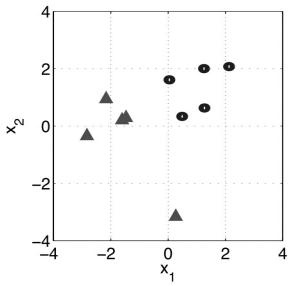
Perceptron Training

Training a Perceptron means finding weights $\mathbf{w}$ that minimize $\mathcal{E}_{\mathcal{P}}$:

$$\underset{w}{\text{argmin}} \quad \left(- \sum_{m \in \mathcal{M}} \mathbf{w}^{\top} \mathbf{x}_m y_m \right) \tag{5}$$

where $\mathcal{M}$ denotes the index set of all *misclassified* data $\mathbf{x}_m$

Data $\mathbf{x} \in \mathbb{R}^2$



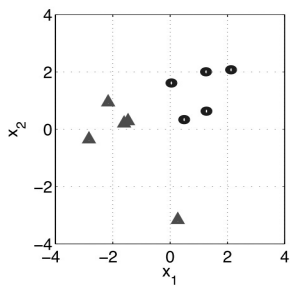
Perceptron Training

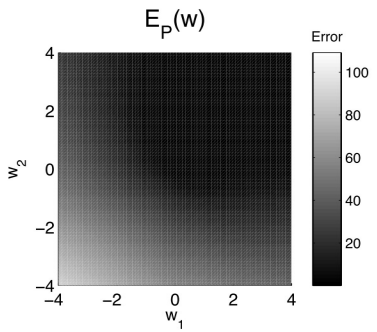
Training a Perceptron means finding weights $\mathbf{w}$ that minimize $\mathcal{E}_{\mathcal{P}}$:

$$\underset{w}{\text{argmin}} \quad \left(- \sum_{m \in \mathcal{M}} \mathbf{w}^{\top} \mathbf{x}_m y_m \right) \tag{5}$$

where $\mathcal{M}$ denotes the index set of all *misclassified* data $\mathbf{x}_m$

Data $\mathbf{x} \in \mathbb{R}^2$





The Perceptron Learning Algorithm

Eq. 5 can be minimized *iteratively*
using **stochastic gradient descent**
[Bottou, 2010; Robbins and Monro, 1951]

1. Initialize $\mathbf{w}^{\text{old}}$ (randomly, $1/n$, ...)
2. While there are misclassified data points $\mathbf{x}_m$
   Pick a random misclassified data point $\mathbf{x}_m$

$$\mathbf{w}^{\text{new}} \leftarrow \mathbf{w}^{\text{old}} - \eta \nabla \mathcal{E}_{\mathcal{P}}(\mathbf{w}) = \mathbf{w}^{\text{old}} + \eta \mathbf{x}_m y_m \qquad (6)$$

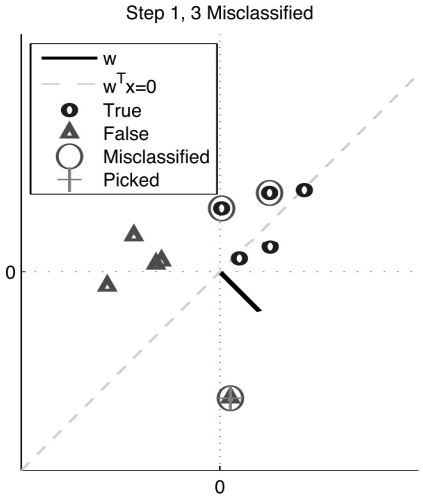
The Perceptron Learning Algorithm

Algorithm 1 Perceptron learning

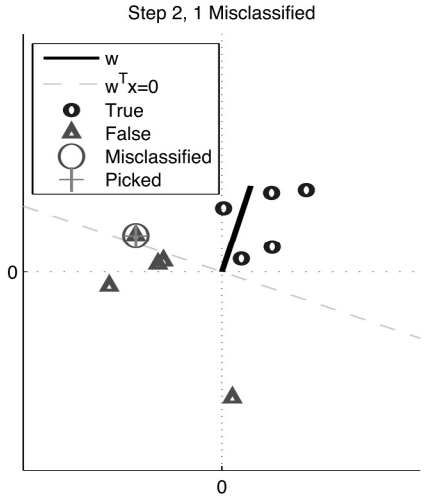
Require: Data $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N], x_i \in \mathbb{R}^D$, Labels $y_1, \ldots, y_N \in \{-1, +1\}$, iterations *its*, learning rate η
Ensure: $\mathbf{w}$

```
w = 1_D/(D)
for it = 1, ..., its do
    # Pick a random example
    i = ceil(rand*N)
    # If this example is not correctly classified
    if sign(w^T x_i) != y_i then
        # Update weight vector
        w ← w + η/it x_i y_i
    end if
end for
```

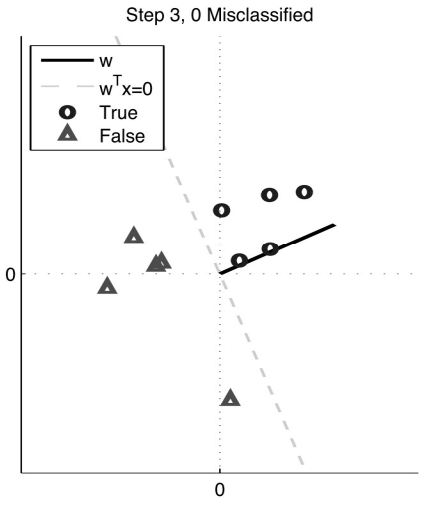
The Perceptron Learning Algorithm in Action



The Perceptron Learning Algorithm in Action



The Perceptron Learning Algorithm in Action



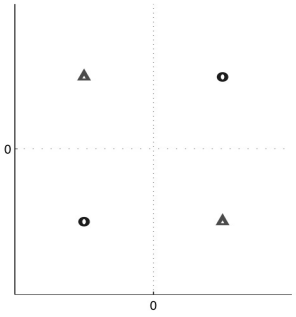
Problems with Perceptrons

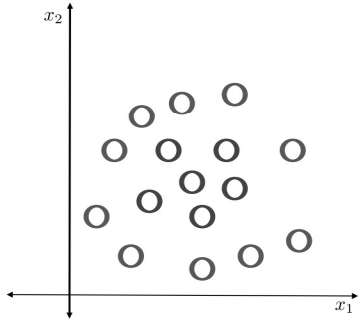
- Each update might lead to new misclassifications
- Many solutions might exist; which one is correct?
- Convergence might be slow
- Only solves linearly separable problems
- If there is no solution, the algorithm will not converge

Some solutions have been proposed by Freund and Schapire [1998]

Problems with Perceptrons

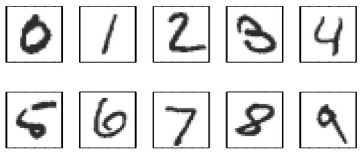
Perceptrons can only learn linearly separable problems.





Application example: Handwritten Digit Recognition

Handwritten digits from UPS data set



Each digit represented as 16×16 pixel image

$\rightarrow$ $x \in \mathbb{R}^{256}$ input nodes

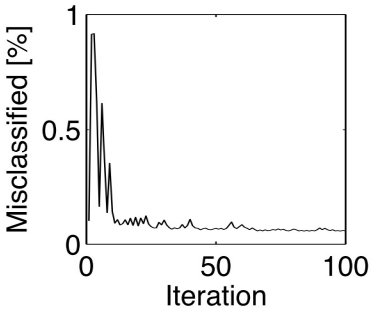
Each image is associated with a label $y \in \{0, 1, \ldots, 9\}$

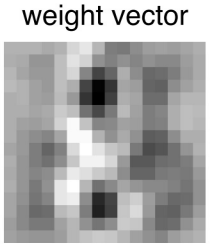
Goal Artificial neural network that recognizes the digit 8

$\Rightarrow$ We need a function $f(.)$ such that

$$f(x) = \begin{cases} -1 & \text{if } y \in \{0, 1, \ldots, 7, 9\} \\ +1 & \text{if } y = 8 \end{cases}$$

Application example: Handwritten Digit Recognition





Application example: Handwritten Digit Recognition

- Handwritten Digit Recognition is a **Multiclass Problem**
- But the Perceptron is a **two-class** (binary) classifier
- How can we **binarize** the multi class problem?

→ Train a perceptron for each digit against all others

→ Concatenate all weight vectors $W = [\mathbf{w_0}, \mathbf{w}_1, \ldots, \mathbf{w}_9]$

→ For new data point $\mathbf{x}$ compute the label as $\operatorname{argmax}\left(W^\top \mathbf{x}\right)$

Minimizing Functions

Remember:

- ML algorithms are functions f that need to be fitted to data
- ML is finding minima/maxima of functions
- Function minimization is done by **Gradient Descent**

Taking Derivatives of Univariate Functions

Function	Derivative
$f(x) = x^n$	
$cf(x)$	
$f(x) + g(x)$	
$f(x)g(x)$	
$\frac{f(x)}{g(x)}$	
$f(g(x))$	

References

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer US, 2007.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In Y. Lechevallier and G. Saporta, editors, *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187, Paris, France, 2010. Springer.

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, pages 277–296, 1998.

H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400—407, 1951.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, Nov. 1958.

B. Widrow and M. E. Hoff. Adaptive switching circuits. In *1960 IRE WESCON Convention Record, Part 4*, pages 96–104, New York, 1960. IRE.

Machine Learning

Lecture 8
Regression

Felix Bießmann

Beuth University & Einstein Center for Digital Future

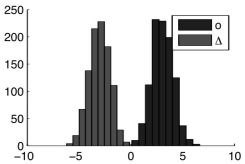
June 11, 2019

Signal-to-Noise Ratio in Classification Settings

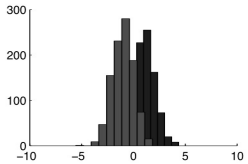
A useful definition of Signal-to-Noise Ratio for classification is

$$\frac{\text{Between Class Variance}}{\text{Within Class Variance}} \tag{1}$$

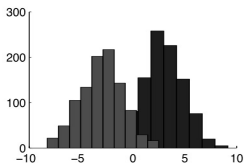
Good SNR



Bad SNR: Close means



Bad SNR: Large Variance

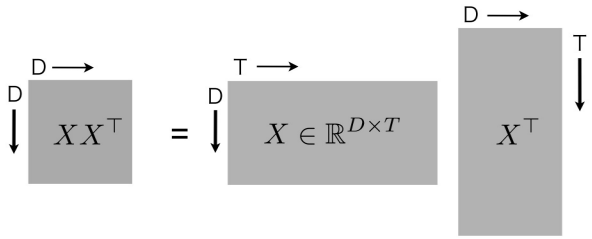


Covariance Matrices

Given T data points $\mathbf{x} \in \mathbb{R}^D$ in a data matrix $\mathbf{X} \in \mathbb{R}^{D \times T}$ the empirical estimate of the **covariance matrix** is defined as

$$1/T \;\; \mathbf{X}\mathbf{X}^\top \tag{2}$$

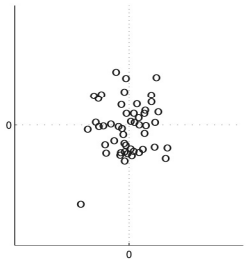
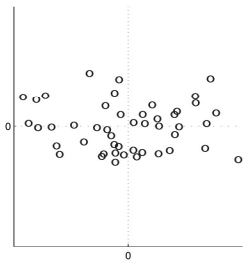
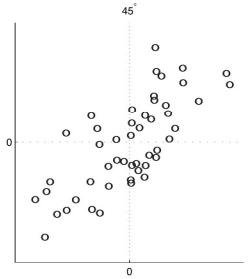
where we assume centered data, i.e. $\sum_{t=1}^{T} \mathbf{x}_t = 0$.



Correlated Data and Linear Mappings

Simulating correlated data can help understanding it

We generate uncorrelated data $\mathbf{x} \in \mathbb{R}^2$ drawn from a normal distribution $\mathbf{x} \sim \mathcal{N}(0, 1)$
We induce *correlations* by a diagonal scaling matrix D and a rotation matrix R

Uncorrelated	Uncorrelated, scaled	Scaled, rotated by 45°
		
$\mathbf{x} \sim \mathcal{N}(0, 1)$	$\begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x}$	$\begin{bmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x}$
$\mathbf{x}\mathbf{x}^\top = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\mathbf{x}\mathbf{x}^\top = \begin{bmatrix} 9 & 0 \\ 0 & 1 \end{bmatrix}$	$\mathbf{x}\mathbf{x}^\top = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$

Fisher's Linear Discriminant Analysis

$$\underset{w}{\operatorname{argmax}} \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}} \tag{3}$$

$$\mathbf{S}_B = \underbrace{(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)}_{\text{Distance Class Means}} (\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)^\top \tag{4}$$

$$\mathbf{S}_W = \sum_{i \in \mathcal{Y}_{+1}} (\mathbf{x}_i - \boldsymbol{\mu}_+)(\mathbf{x}_i - \boldsymbol{\mu}_+)^\top + \sum_{j \in \mathcal{Y}_{-1}} (\mathbf{x}_j - \boldsymbol{\mu}_-) \underbrace{(\mathbf{x}_j - \boldsymbol{\mu}_-)^\top}_{\text{Distance from Class Mean}} \tag{5}$$

Fisher's Linear Discriminant Analysis

Setting the first derivative of eq. 3 to zero, we obtain the generalized eigenvalue equation

$$\mathbf{S}_B w = \mathbf{S}_W w \lambda \tag{6}$$

Left multiplying with $\mathbf{S}_W^{-1}$ yields

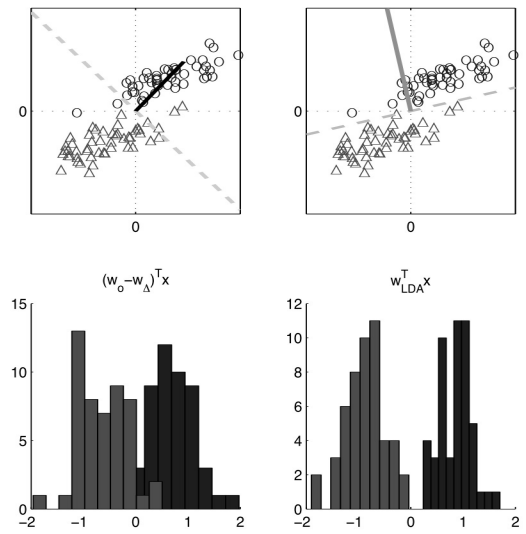
$$\mathbf{S}_W^{-1} \mathbf{S}_B w = \mathbf{S}_W^{-1} \mathbf{S}_W w \lambda$$

$$\mathbf{S}_W^{-1} (\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-) \underbrace{(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)^\top w}_{\beta} = w \tag{7}$$

$$w \propto \mathbf{S}_W^{-1} (\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)$$

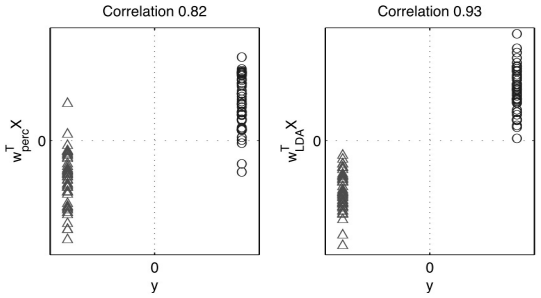
$\rightarrow$ Fisher's LDA first *decorrelates* the data followed by nearest centroid classification

Fisher Linear Discriminant Analysis



Fisher Linear Discriminant Analysis

Another view on LDA:
Maximizing the correlation between class labels $\mathbf{y}$ and $\mathbf{w}^\top \mathbf{X}$:



What if our labels are not $\in \{-1, +1\}$ but $\in \mathbb{R}$?

From Classification to Regression

$y \in \{-1, +1\}$	$y \in \mathbb{R}$
Classification	**Regression**

The most popular and best understood type of regression is **linear regression** using a *least-squares cost function*.

Linear Regression

Target variable $y \in \mathbb{R}$ is modeled as a **linear combination** $w \in \mathbb{R}^N$ of N regressors $\phi(\mathbf{x}) \in \mathbb{R}^N$

$$y = \mathbf{w}^\top \phi(\mathbf{x}) \tag{8}$$

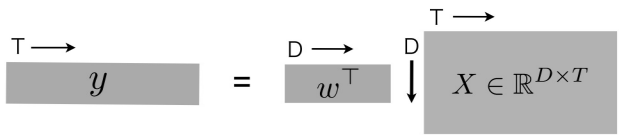
where $\phi(.)$ is one or more (potentially non-linear) function on $\mathbf{x}$.

For the sake of simplicity we assume $\phi(\mathbf{x}) = \mathbf{x}$.

Linear Regression

Let T be the number of samples, so $\mathbf{y} \in \mathbb{R}^{1 \times T}$ and $\mathbf{X} \in \mathbb{R}^{N \times T}$.
The Linear Regression model in matrix notation then becomes

$$\mathbf{y} = \mathbf{w}^\top \mathbf{X}. \tag{9}$$



Linear Regression

The most popular loss function to optimize w is the **least-square error** [Gauß, 1809; Legendre, 1805]

$$\mathcal{E}_{lsq}(w) = \sum_{t=1}^{T}(y_t - \mathbf{w}^\top \mathbf{X}_t)^2 \tag{10}$$

C.F. Gauß (1777-1855)

A.M. Legendre (1752-1833)

Linear Regression

To minimize the least-squares loss function in eq. 10

$$\mathcal{E}_{lsq}(w) = (\mathbf{y} - \mathbf{w}^\top \mathbf{X})^2 = \mathbf{y}\mathbf{y}^\top - 2\mathbf{w}^\top \mathbf{X}\mathbf{y}^\top + \mathbf{w}^\top \mathbf{X}\mathbf{X}^\top \mathbf{w}$$

we compute derivative w.r.t. $\mathbf{w}$

Linear Regression

To minimize the least-squares loss function in eq. 10

$$\mathcal{E}_{lsq}(w) = (\mathbf{y} - \mathbf{w}^\top \mathbf{X})^2 = \mathbf{y}\mathbf{y}^\top - 2\mathbf{w}^\top \mathbf{X}\mathbf{y}^\top + \mathbf{w}^\top \mathbf{X}\mathbf{X}^\top \mathbf{w}$$

we compute derivative w.r.t. $\mathbf{w}$

$$\frac{\partial \mathcal{E}_{lsq}(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{X}\mathbf{y}^\top + 2\mathbf{X}\mathbf{X}^\top \mathbf{w} \tag{11}$$

Linear Regression

To minimize the least-squares loss function in eq. 10

$$\mathcal{E}_{lsq}(w) = (\mathbf{y} - \mathbf{w}^\top \mathbf{X})^2 = \mathbf{y}\mathbf{y}^\top - 2\mathbf{w}^\top \mathbf{X}\mathbf{y}^\top + \mathbf{w}^\top \mathbf{X}\mathbf{X}^\top \mathbf{w}$$

we compute derivative w.r.t. $\mathbf{w}$

$$\frac{\partial \mathcal{E}_{lsq}(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{X}\mathbf{y}^\top + 2\mathbf{X}\mathbf{X}^\top \mathbf{w} \tag{11}$$

set it to zero and solve for w

$$\begin{aligned} -2\mathbf{X}\mathbf{y}^\top + 2\mathbf{X}\mathbf{X}^\top \mathbf{w} &= 0 \\ \mathbf{X}\mathbf{X}^\top \mathbf{w} &= \mathbf{X}\mathbf{y}^\top \\ \mathbf{w} &= (\underbrace{\mathbf{X}\mathbf{X}^\top}_{\text{Cov. Mat.}})^{-1}\mathbf{X}\mathbf{y}^\top \end{aligned} \tag{12}$$

Linear Regression for Vector Labels

Prediction of vector-valued labels $\mathbf{y} \in \mathbb{R}^M$
is called **Multiple Linear Regression**:

For a measurement $\mathbf{X} \in \mathbb{R}^{N \times T}$, $\mathbf{Y} \in \mathbb{R}^{M \times T}$ the MLR model is

$$\mathbf{Y} = \mathbf{W}^\top \mathbf{X} \tag{13}$$

where $\mathbf{W}^\top \in \mathbb{R}^{M \times N}$ is a **linear mapping** from data to labels.

Linear Regression for Vector Labels

Given Data $\mathbf{X} \in \mathbb{R}^{N \times T}$ and labels $\mathbf{Y} \in \mathbb{R}^{M \times T}$
the error function for multiple linear regression is

$$\mathcal{E}_{MLR}(\mathbf{W}) = \sum_{m=1}^{M} (\mathbf{Y}_m - \mathbf{W}_m^\top \mathbf{X})^2 \tag{14}$$

where $\mathbf{Y}_m$ denotes the m-th output dimension
and $\mathbf{W}_m$ the corresponding weight vector

Linear Regression for Vector Labels

Given Data $\mathbf{X} \in \mathbb{R}^{N \times T}$ and labels $\mathbf{Y} \in \mathbb{R}^{M \times T}$
the error function for multiple linear regression is

$$\mathcal{E}_{MLR}(\mathbf{W}) = \sum_{m=1}^{M} (\mathbf{Y}_m - \mathbf{W}_m^\top \mathbf{X})^2 \tag{14}$$

where $\mathbf{Y}_m$ denotes the m-th output dimension
and $\mathbf{W}_m$ the corresponding weight vector

Eq. 14 is minimized by

$$W = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{Y}^\top \tag{15}$$

Linear Discriminant Analysis and Linear Regression

Remember the solution to linear discriminant analysis

$$\mathbf{w}_{LDA} \propto \mathbf{S}^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)$$

Linear Discriminant Analysis and Linear Regression

Remember the solution to linear discriminant analysis

$$\begin{aligned}\mathbf{w}_{LDA} &\propto \mathbf{S}^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-) \\ &\propto \mathbf{S}^{-1}(\underbrace{(+1)}_{y}\ \underbrace{(1/N_{+1})}_{\gamma_1} \sum_{i \in \mathcal{Y}_{+1}} \mathbf{x}_i + \underbrace{(-1)}_{y}\ \underbrace{(1/N_{-1})}_{\gamma_2} \sum_{j \in \mathcal{Y}_{-1}} \mathbf{x}_j)\end{aligned}$$

Linear Discriminant Analysis and Linear Regression

Remember the solution to linear discriminant analysis

$$\begin{aligned}\mathbf{w}_{LDA} &\propto \mathbf{S}^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-) \\ &\propto \mathbf{S}^{-1}(\underbrace{(+1)}_{y}\ \underbrace{(1/N_{+1})}_{\gamma_1} \sum_{i \in \mathcal{Y}_{+1}} \mathbf{x}_i + \underbrace{(-1)}_{y}\ \underbrace{(1/N_{-1})}_{\gamma_2} \sum_{j \in \mathcal{Y}_{-1}} \mathbf{x}_j) \\ &\propto \mathbf{S}^{-1}\mathbf{X}\mathbf{y}^\top \text{ assuming } N_{+1} = N_{-1}\end{aligned}$$

Linear Discriminant Analysis and Linear Regression

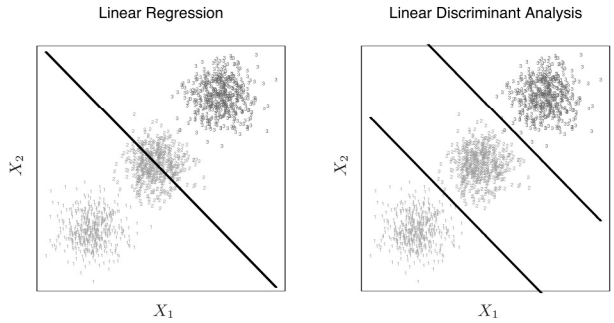
Remember the solution to linear discriminant analysis

$$\begin{aligned}\mathbf{w}_{LDA} &\propto \mathbf{S}^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-) \\ &\propto \mathbf{S}^{-1}(\underbrace{(+1)}_{y}\ \underbrace{(1/N_{+1})}_{\gamma_1} \sum_{i \in \mathcal{Y}_{+1}} \mathbf{x}_i + \underbrace{(-1)}_{y}\ \underbrace{(1/N_{-1})}_{\gamma_2} \sum_{j \in \mathcal{Y}_{-1}} \mathbf{x}_j) \\ &\propto \mathbf{S}^{-1}\mathbf{X}\mathbf{y}^\top \text{ assuming } N_{+1} = N_{-1}\end{aligned}$$

LDA	Linear Regression
$\mathbf{w} \propto \mathbf{S}^{-1}\mathbf{X}\mathbf{y}^\top$	$\mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y}^\top$

Classification by Linear Regression?

So when coding the class as a boolean multivariate label vector, can we do classification with linear regression?



No, for more than 2 classes, this can lead to poor classification

References

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer US, 2007.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In Y. Lechevallier and G. Saporta, editors, *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187, Paris, France, 2010. Springer.

C. F. Gauß. Theoria motus corporum coelestium in sectionibus conicis solem ambientium. *Göttingen*, 1809.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. 2003.

A. E. Hoerl and R. W. Kennar. Ridge regression: Applications to nonorthogonal problems. *Technometrics*, 12(1):69–82, 1970.

A.-M. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*, chapter Sur la methode des moindres quarres. Firmin Didot, http://imgbase-scd-ulp.u-strasbg.fr/displayimage.php?pos=-141297, 1805.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning. The MIT Press, 1 edition, 2012. ISBN 0262018020,9780262018029.

A. Rahimi and B. Recht. Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning. In D. Koller, D. Schuurmans, Y. Bengio, L. Bottou, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *NIPS*, pages 1313–1320. MIT Press, 2008.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

A. N. Tychonoff. On the stability of inverse problems. *Doklady Akademii Nauk SSSR*, 39(5):195–198, 1943.

Machine Learning

Lecture 9
Principal Component Analysis and Extensions

Felix Bießmann

Beuth University & Einstein Center for Digital Future

June 10, 2019

Supervised vs Unsupervised Algorithms

Often there is no label information available

- Ongoing neural activity
- Mixtures of different speakers in a audio recording
- Complex artefacts in experimental recordings

Unsupervised algorithms

- Find structure in data sets
- Allow partitioning of data in *meaningful* parts
- Allow to remove unwanted aspects (e.g. noise)

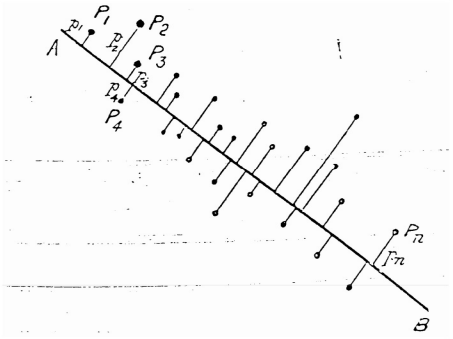
Principal Component Analysis

Principal Component Analysis (PCA):

Popular dimensionality reduction technique

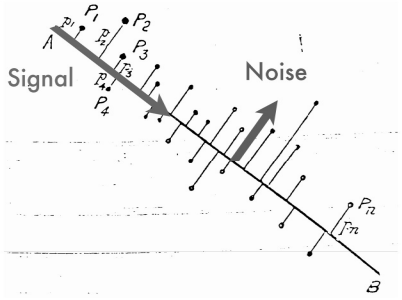
Easy to implement

Principal Component Analysis



Which line fits data best?

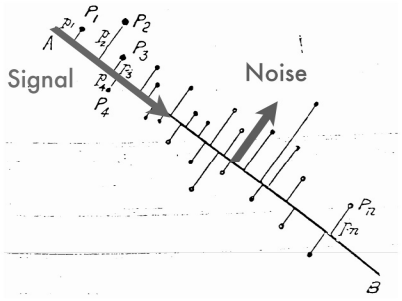
Principal Component Analysis



Which line fits data best?

The line w that minimizes the noise and maximizes the signal [Pearson, 1901]

Principal Component Analysis



Which line fits data best?

The line w that minimizes the noise and maximizes the signal [Pearson, 1901]

Or equivalently:

The line w that maximizes the variance within the data set

Maximizing variance in a data set

We obtained some data $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$

PCA finds a direction $\mathbf{w}^* \in \mathbb{R}^D$ such that

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}}\, \mathbf{w}^\top \mathbf{X}\mathbf{X}^\top \mathbf{w} \tag{1}$$

Maximizing variance in a data set

We obtained some data $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$

PCA finds a direction $\mathbf{w}^* \in \mathbb{R}^D$ such that

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}}\, \mathbf{w}^\top \mathbf{X}\mathbf{X}^\top \mathbf{w} \tag{1}$$

When optimizing eq. 1 we have to constrain $\mathbf{w}$

$$\|\mathbf{w}\|^2 = \mathbf{w}^\top \mathbf{w} = 1 \tag{2}$$

yielding the Lagrangian

$$\mathcal{L} = \mathbf{w}^\top \mathbf{X}\mathbf{X}^\top \mathbf{w} + \lambda(1 - \mathbf{w}^\top \mathbf{w}) \tag{3}$$

Short excursion: Lagrangians and Optimization

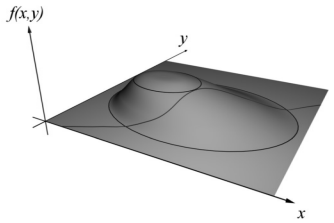
Optimizing a function subject to some constraint

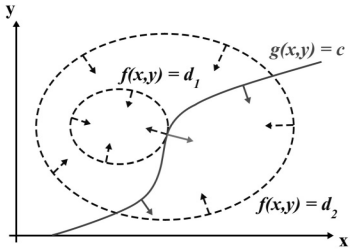
maximize $f(x, y)$

subject to the constraint $g(x, y) = c$

Lagrangian: $\mathcal{L}(x, y, \lambda) = f(x, y) + \lambda(g(x, y) - c)$

where λ is called a *Lagrangian Multiplier*





Source: `http://en.wikipedia.org/wiki/Lagrange_multipliers`

Maximizing variance in a data set

$$\mathcal{L} = \mathbf{w}^\top \mathbf{X}\mathbf{X}^\top \mathbf{w} + \lambda(1 - \mathbf{w}^\top \mathbf{w})$$

Setting the derivative w.r.t. w to zero yields

$$\frac{\partial \mathcal{L}}{\partial w} = 2\mathbf{X}\mathbf{X}^\top \mathbf{w} - 2\lambda \mathbf{w} = 0$$

$$\Rightarrow \mathbf{X}\mathbf{X}^\top \mathbf{w} = \lambda \mathbf{w} \tag{4}$$

This is a standard eigenvalue problem.

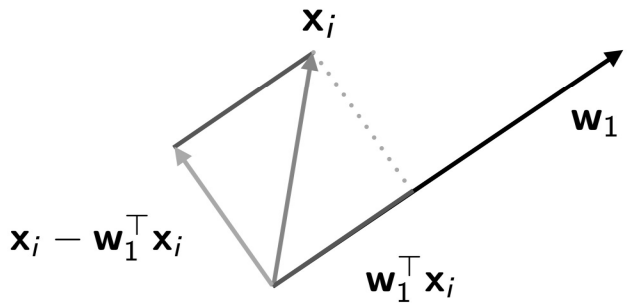
$\mathbf{w}$ is the eigenvector of $\mathbf{X}\mathbf{X}^\top$ corresponding to the largest eigenvalue

Finding k Principal Components

- We found the strongest variance direction
- Now we want to find the second strongest variance direction
- The second direction should not be correlated with the first
- We *project out* the variance explained by the first direction
- And again find the strongest variance direction

Finding k Principal Components

How can we *project out* the variance along the first principal direction $\mathbf{w}_1$?



Finding k Principal Components

How can we *project out* the variance along the first principal direction $\mathbf{w}_1$?

Project data on $\mathbf{w}_1$ and back through $\mathbf{w}_1$

$$\mathbf{X}_{\mathbf{w}_1} = \mathbf{w}_1 \underbrace{\mathbf{w}_1^\top \mathbf{X}}_{\text{First Principal Component}} \tag{5}$$

Finding k Principal Components

How can we *project out* the variance along the first principal direction $\mathbf{w}_1$?

Project data on $\mathbf{w}_1$ and back through $\mathbf{w}_1$

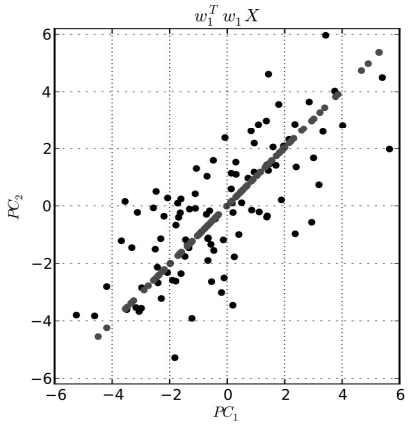
$$\mathbf{X}_{\mathbf{w}_1} = \mathbf{w}_1 \underbrace{\mathbf{w}_1^\top \mathbf{X}}_{\text{First Principal Component}} \tag{5}$$

Now we can subtract $\mathbf{X}_{\mathbf{w}_1}$ from the original data $\mathbf{X}$.

$$\mathbf{X} - \mathbf{X}_{\mathbf{w}_1} = \mathbf{X} - \mathbf{w}_1\mathbf{w}_1^\top \mathbf{X} = (\mathbf{I} - \mathbf{w}_1\mathbf{w}_1^\top)\mathbf{X} \tag{6}$$

Finding k Principal Components

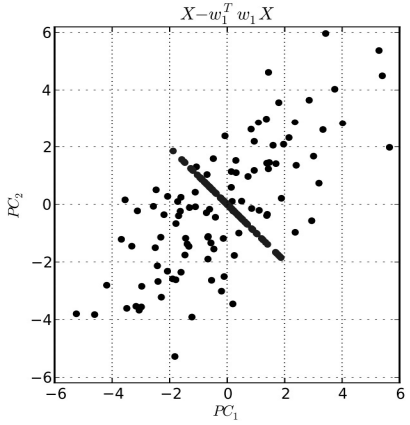
$$\mathbf{X}_{\mathbf{w}_1} = \mathbf{w}_1\mathbf{w}_1^\top \mathbf{X}$$



Data projected on $\mathbf{w}_1$ and back through $\mathbf{w}_1$

Finding k Principal Components

$$\mathbf{X} - \mathbf{w}_1\mathbf{w}_1^\top\mathbf{X} = (\mathbf{I} - \mathbf{w}_1\mathbf{w}_1^\top)\mathbf{X}$$



Data projected on $\mathbf{w}_1$ and back through $\mathbf{w}_1$ subtracted from $\mathbf{X}$

Finding k Principal Components

Finding the largest eigenvector using gradient descent, projecting it out and finding the next eigenvector is called the **Power Method** for solving eigenvalue equations

Principal Directions are Eigenvectors of Covariance Matrix

The k first PCA basis vectors are the eigenvectors corresponding to the largest k eigenvalues

$$\mathbf{X}\mathbf{X}^\top\mathbf{W} = \mathbf{W}\Lambda \qquad (7)$$

where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k]$ contains the eigenvectors sorted according to their eigenvalues and Λ is a diagonal matrix containing all eigenvalues.

Principal Directions are Eigenvectors of Covariance Matrix

The k first PCA basis vectors are the eigenvectors corresponding to the largest k eigenvalues

$$\mathbf{X}\mathbf{X}^\top\mathbf{W} = \mathbf{W}\Lambda \qquad (7)$$

where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k]$ contains the eigenvectors sorted according to their eigenvalues and Λ is a diagonal matrix containing all eigenvalues.

A useful property of the new PCA basis:
Eigenvectors $\mathbf{w}_i$, $i \in \{1, 2, \ldots, k\}$ are orthogonal to each other:

$$\mathbf{w}_i^\top\mathbf{w}_j = 0, \forall i \neq j$$

PCA Algorithm

Algorithm 2 Principal Component Analysis

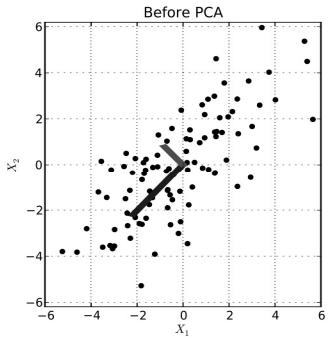
Require: data $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$, number of principal components k

Ensure: $\mathbf{W}$

1: # Center Data
2: $\mathbf{X} = \mathbf{X} - 1/N \sum_i \mathbf{x}_i$
3: # Compute Covariance Matrix
4: $\mathbf{C} = 1/N\ \mathbf{X}\mathbf{X}^\top$
5: # Compute largest k eigenvectors
6: $\mathbf{W} = \text{eig}(\mathbf{C})$

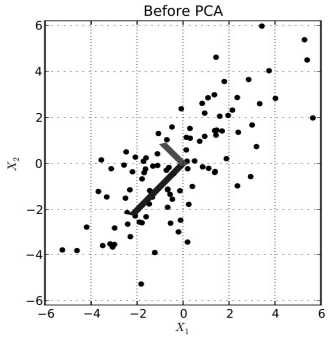
Principal Component Analysis

$\mathbf{X}$

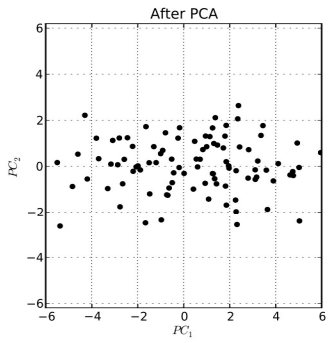


Principal Component Analysis

$\mathbf{X}$



$\mathbf{W}^\top\mathbf{X}$



PCA aligns maximum variance directions with standard basis
→ Variance along each dimension is **uncorrelated**
→ Now we can remove each dimension separately

Dimensionality Reduction by PCA

We can reduce the dimensionality of $\mathbf{X}$ from d to k

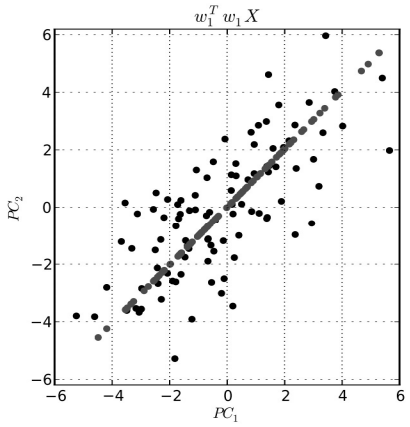
$$\mathbf{X}_{PCA} = \mathbf{W}^\top\mathbf{X} \tag{8}$$

If we want only a set $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ of principal components

$$\mathbf{X}_{\mathcal{I}} = \sum_{i \in \mathcal{I}} \mathbf{w}_i \mathbf{w}_i^\top \mathbf{X}$$

Note that $\mathcal{I}$ does not need to contain the *strongest* components

Dimensionality Reduction by PCA



Here we assume the relevant signal is along the high variance direction

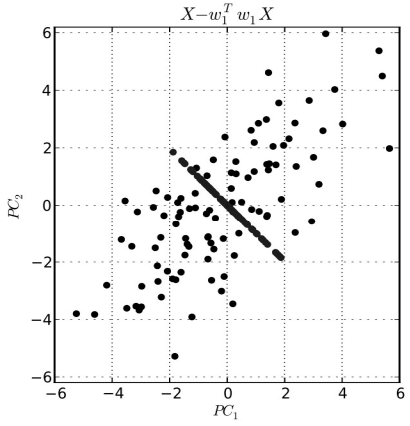
Denoising by PCA

Assuming that noise has high (or low) variance we can remove those components

If we want to project out a set $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ of principal components but we want the data to be in the input space

$$\mathbf{X}_{PCA} = \mathbf{X} - \mathbf{W}_{\mathcal{I}}\mathbf{W}_{\mathcal{I}}^{\top}\mathbf{X} = (\mathbf{I} - \mathbf{W}_{\mathcal{I}}\mathbf{W}_{\mathcal{I}}^{\top})\mathbf{X} \tag{8}$$

Denoising by PCA



Here we assume noise is along the high variance direction

Summary

Unsupervised Data Analysis

- Finds structure in data in explorative fashion
- Can be used for
  - Dimensionality reduction
  - Visualization
  - Denoising

Principal Component Analysis (PCA)

- Finds directions of maximal variance
- Is solved by eigendecomposition of Covariance/Kernel Matrix

Linear PCA

- Finds *linear* subspaces
- If there are more dimensions than data points

$\rightarrow$ Do eigendecomposition on kernel matrix

References

K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(6):1299–1319, 1998.

Machine Learning

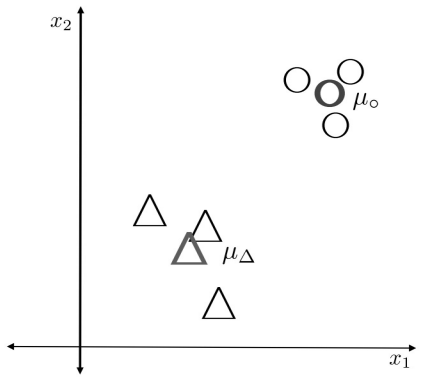
Lecture 10
Clustering

Felix Bießmann

Beuth University & Einstein Center for Digital Future

Clustering

Psychological Models of Categorization: Prototypes



Prototypes $\boldsymbol{\mu}_\Delta$ and $\boldsymbol{\mu}_o$:

$$\boldsymbol{\mu}_\Delta = 1/N_\Delta \sum_n^{N_\Delta} \mathbf{x}_{\Delta,n}$$

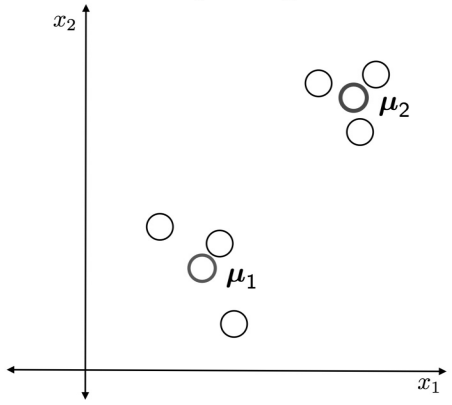
$$\boldsymbol{\mu}_o = 1/N_o \sum_n^{N_o} \mathbf{x}_{o,n}$$

New data points x are assigned to their closest cluster center μ^*

$$\boldsymbol{\mu}^* = \underset{i}{\operatorname{argmin}}(\|\boldsymbol{\mu}_i - \mathbf{x}\|_2) \qquad (1)$$

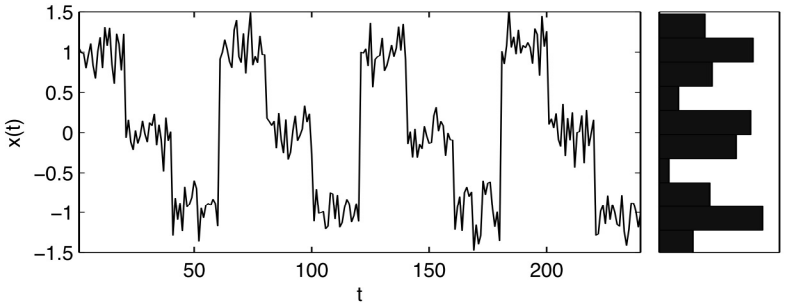
Clustering

Psychological Models of Categorization: Prototypes



The only difference for clustering is:
We do not have labels.

Clustering For Quantization of Analog Signals



Quantization transforms an analog signal into discretized states
This is important for Audio Processing, Compression, ...
The most popular Clustering Algorithm was proposed for Quantization [Lloyd, 1982]

K-means Clustering

K-Means Algorithm

Re-iterating two steps:

1. Assign each data point $\mathbf{x}_i$ to their closest cluster $\boldsymbol{\mu}_k$
2. Update $\boldsymbol{\mu}_k$ to the mean of the members in that cluster

K-means Clustering

Goal: Given data $\mathbf{x}_1, \ldots, \mathbf{x}_N$ find cluster centers $\boldsymbol{\mu}_1, \ldots \boldsymbol{\mu}_K$ such that the distances of data points to their respective cluster centre are minimized

$$\mathcal{J} = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbf{c}_{n,k} \|\mathbf{x}_n - \boldsymbol{\mu}_{\mathbf{c}_k}\| \tag{2}$$

$$\text{where } \mathbf{c}_{n,k} \begin{cases} 1 & \text{if } \mathbf{x}_n \text{ belongs to cluster } k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

K-means Clustering Algorithm

Algorithm 1 K-means clustering

Require: data $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^D$, number of clusters k, iterations m.

Choose random data points as initial cluster centres $\boldsymbol{\mu}_1 \leftarrow \mathbf{x}_{i_1}, \ldots, \boldsymbol{\mu}_k \leftarrow \mathbf{x}_{i_k}$ where $i_j \neq i_l$ for all $j \neq l$.
$\mathbf{c} \leftarrow \mathbf{0}_N$
$\mathbf{c}^{\text{old}} \leftarrow \mathbf{0}_N$
$i \leftarrow 0$
while $i < m$ **do**
  for $j = 1$ to N **do**
    Find nearest cluster centre $\mathbf{c}_j \leftarrow \text{argmin}_{1 \leq l \leq k} \|\mathbf{x}_j - \boldsymbol{\mu}_l\|_2$
  end for
  for $j \leftarrow 1$ to k **do**
    Compute new cluster centre $\boldsymbol{\mu}_j \leftarrow \frac{1}{|\{l : \mathbf{c}_l = j\}|} \sum_{l : \mathbf{c}_l = j} \mathbf{x}_l$
  end for
  if $\mathbf{c}^{\text{old}} = \mathbf{c}$ **then**
    break
  end if
  $\mathbf{c}^{\text{old}} \leftarrow \mathbf{c}$
  $i \leftarrow i + 1$
end while
return cluster centres $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \in \mathbb{R}^D$, assignment vector $\mathbf{c}^{\text{old}} \in \mathbb{R}^n$

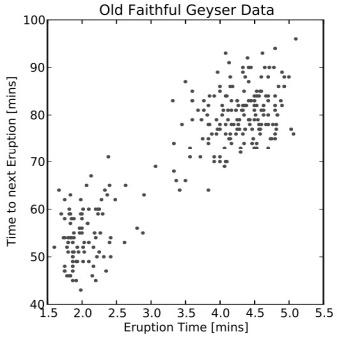
Application Example: Geyser Eruptions

Old Faithful Geyser
Yellowstone National Park, USA

Famous data set for clustering

- Old Faithful Eruptions
- Two dimensions
  1. Time of Eruption [mins]
  2. Time until next Eruption [mins]

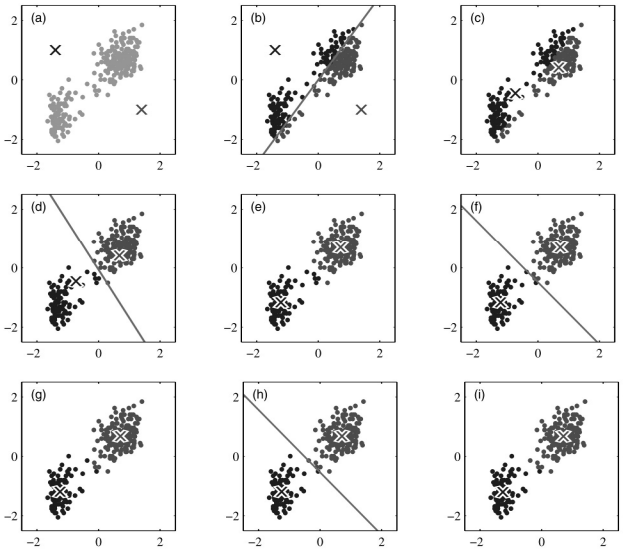
Application Example: Geyser Eruptions



Famous data set for clustering

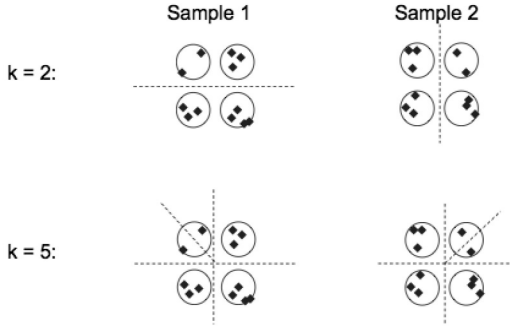
- Old Faithful Eruptions
- Two dimensions
  1. Time of Eruption [mins]
  2. Time until next Eruption [mins]

K-means Clustering Step-by-Step



Clustering Instability

Number of Clusters is a critical parameter



Clusterings are instable if number of clusters is too small or too large

Clustering Instability

- Number of clusters is critical hyper parameter
- In supervised settings we use cross-validation to optimize hyper-parameters for accuracy on test data
- How can we optimize the number of clusters?

→ Choose that k that results in most **stable** clusterings
For a review see e.g. [von Luxburg, 2009]

Clustering Instability Algorithm

Algorithm 2 Clustering Instability

Require: data points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, clustering algorithm $\mathcal{A}$, maximal number of clusters K, iterations i.
Ensure: optimal number of clusters k^*

```
for k = 2 to K do
    Resample data set (e.g. random draws with replacement)
    for it = 1 to i do
        Cluster data using algorithm A into k clusters
    end for
    Compute minimal (across all label permutations) distance between clusterings
end for
Chose that k that has the minimal instability over resamplings
```

Distance Measures for Real-Valued Data $\mathbf{x} \in \mathbb{R}^D$

Clustering Algorithms need a **distance function** $d(\mathbf{x}_i, \mathbf{x}_j)$

- For real valued data $\mathbf{x} \in \mathbb{R}^D$ we can use the Euclidean distance

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \tag{5}$$

- More robust (less sensitive to outliers) is the **city block distance** or $\mathcal{L}_1$ norm

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_1 \tag{6}$$

- Another alternative is the correlation coefficient (also called cosine similarity)

$$d(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_1^\top \mathbf{x}_2}{\|\mathbf{x}_1\|_2 \|\mathbf{x}_2\|_2} \tag{7}$$

For standardized data $\sum_i \mathbf{x}_i = 0$, $\sum_i \mathbf{x}_i^2 = 1$ maximizing correlation is the same as minimizing euclidean distance.

Distance Measures for Non-Real-Valued Data

- For ordinal variables $\mathbf{x} \in \{\text{low}, \text{medium}, \text{high}\}^D$ we can transform the values into real-valued numbers (for three possible values e.g. $1/3, 2/3, 3/3$) and then apply distance functions for real-valued data
- For categorial variables $\mathbf{x} \in \{\text{red}, \text{green}, \text{blue}\}^D$ we can use a binary coding for the differences

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_d^D \mathbf{x}_{id} \neq \mathbf{x}_{jd} \tag{8}$$

This metric is called **Hamming Distance**

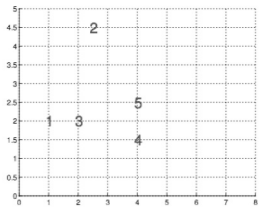
For the sake of simplicity we only consider the euclidean distance

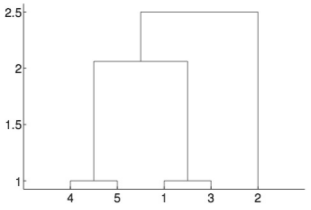
Hierarchical Clustering

- K-Means produces **flat** clusterings
- Often we are interested in a **hierarchy** of clusterings
- Examples:
  - Biological Species
  - Topics in Text Documents

Hierarchical Clustering

- K-Means produces **flat** clusterings
- Often we are interested in a **hierarchy** of clusterings
- Examples:
  - Biological Species
  - Topics in Text Documents

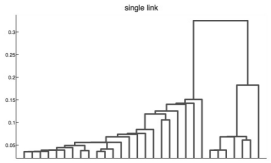




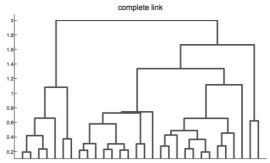
Examples Hierarchical Clustering

Dendrograms (binary clustering trees) of yeast gene expression data

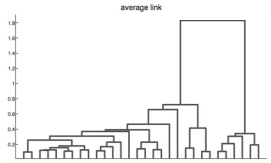
Single Linkage



Complete Linkage



Average Linkage



Taken from [Murphy, 2012]

Summary

- Clustering Algorithms find clusters in data
- Clustering Performance depends on distance function used
- K-Means is one of the most popular clustering algorithms
- For large data sets use Online K-Means
- Wrong number of clusters leads to unstable results
- Hierarchical clustering

References

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer US, 2007.

S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, Mar. 1982. ISSN 0018-9448.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning. The MIT Press, 1 edition, 2012. ISBN 0262018020,9780262018029.

U. von Luxburg. Clustering stability: An overview. *Foundations and Trends in Machine Learning*, 2(3):235–274, 2009.

Machine Learning

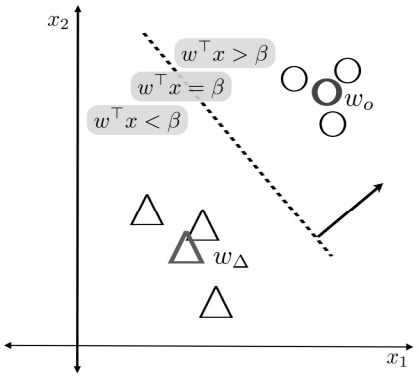
Neural Networks

Felix Bießmann

Beuth University & Einstein Center for Digital Future

June 24, 2019

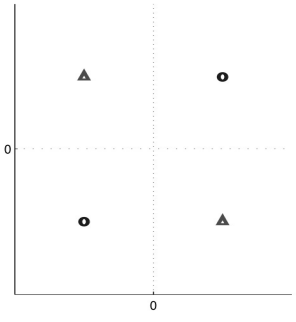
Linear Classification

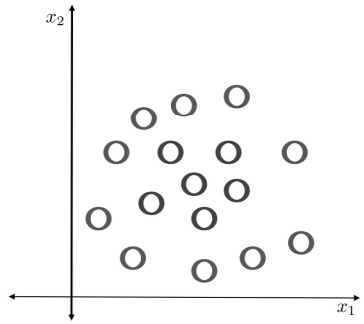


$$\phi(\mathbf{w}^\top \mathbf{x} - \beta) = \begin{cases} > 0 & \text{if } \mathbf{x} \text{ is from class } o \\ < 0 & \text{if } \mathbf{x} \text{ is from class } \Delta \end{cases}$$

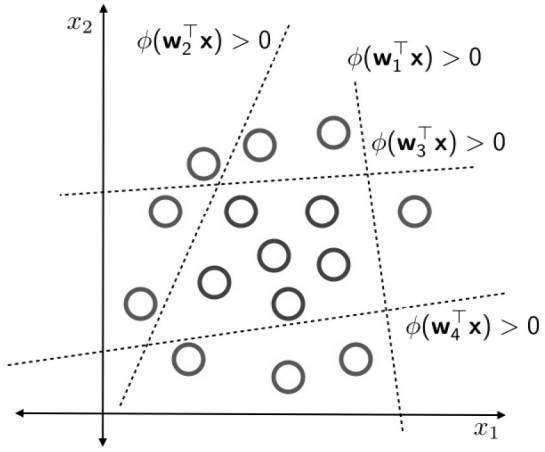
Problems with Perceptrons

Perceptrons can only learn linearly separable problems.



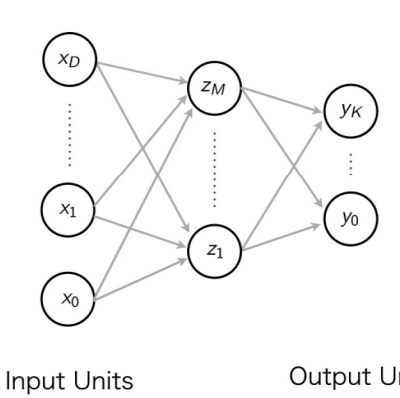


Deep Neural Networks



Deep Neural Networks

Combinations of Perceptrons (Multi Layer Perceptrons):



Neurons (Units), that are neither output nor input are called **Hidden Units**.

A Short History of Deep Learning

1943 First mathematical Neuron Model (Mcculloch and Pitts, 1943)

1957 Perceptron Algorithm (Rosenblatt, 1958)

1969 Perceptrons cannot solve non-linearen Problems (Minsky and Papert, 1969)

1970 Backpropagation: Efficient gradient computations (Linnainmaa, 1970)

1980 Computer Hardware $\approx$ 10,000 faster compared to 1960/1970 – Automatic Differentiation (Speelpenning, 1980)

1986 Backpropagation learns meaningful representations (Rumelhart et al., 1986), NETtalk (Sejnowski and Rosenberg, 1986)

1992 Support-Vector Machines (SVMs) (Boser et al., 1992)

2000 Computer Hardware (GPUs) $\approx$ 10,000 faster compared to 1980/1990, Bigger datasets render kernel SVMs computationally infeasible

2012 Deep Convolutional Networks wins ImageNet (Krizhevsky et al., 2012)

2014 Neural Machine Translation surpasses traditional methods

2017 Neural Networks for Reinforcement Learning excell at Go (AlphaGo Zero)

2018 ImageNet Moment for Neural Language Models (BERT / ELMO)

Sources: Juergen Schmidhuber's page and others

Universal Approximation Theorem

[Cybenko, 1989]

Multilayer Perceptrons with one hidden layer and a finite number of hidden units can approximate any function.

Training of Deep Neural Networks

- Training: Gradient Descent
- Problem: Gradient Computations
  - Mathematically challenging for complex models
  - Computationally challenging

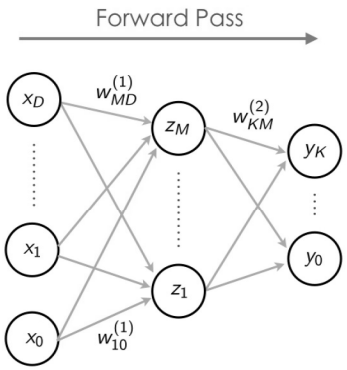
→ Solution: **Backpropagation**
  - Elegant formulation
  - Efficient implementation

Backpropagation Algorithm

Algorithm 1 Backpropagation Algorithm - Pseudocode

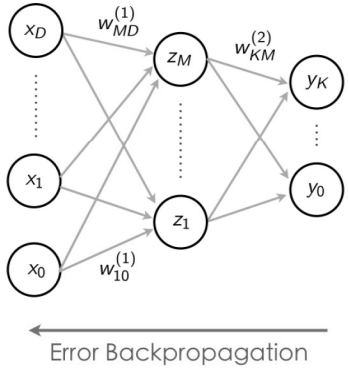
Require: Data $\mathbf{X} \in \mathbb{R}^{D \times N}$, labels $\mathbf{Y} \in \mathbb{R}^{K \times N}$, untrained network
Ensure: Network parameters $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(V)}$
1: **while** Not converged **do**
2: # Compute network predictions
3: # Evaluate error function
4: # Propagate error from output layer back to input layer
5: # Take gradient descent step
6: **end while**

Learning with Backpropagation in Neural Networks



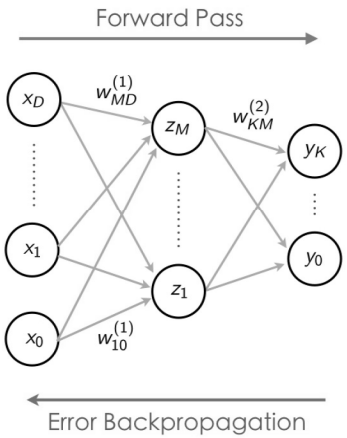
Computation of network predictions is called **Forward Propagation**.

Learning with Backpropagation in Neural Networks



Backpropagation refers to efficient computation of error function gradients for all connections.

Learning with Backpropagation in Neural Networks



After a forward and backward pass a gradient step is performed.

Summary

- Perceptrons cannot separate linearly non-separable problems
- Using combinations and stacking of standard Perceptrons, Multi Layer Perceptrons (MLPs) can approximate any function with one hidden layer
- Gradient descent for MLPs is challenging, mathematically and computationally
- Backpropagation: Efficient Gradient computation

References

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer US, 2007.

G. Cybenko. Approximations by superpositions of sigmoidal functions. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. 2003.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning. The MIT Press, 1 edition, 2012. ISBN 0262018020,9780262018029.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, Nov. 1958.

B. Widrow and M. E. Hoff. Adaptive switching circuits. In *1960 IRE WESCON Convention Record, Part 4*, pages 96–104, New York, 1960. IRE.