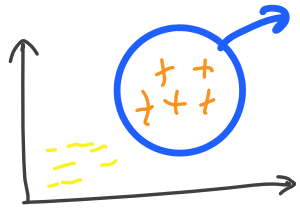
Intro

8 Şubat 2017 Çarşamba 13:45

Online Courses : Yaser Abou Moustafa - Caltech
Nando Freitas - Oxford

Book : Ffont.Intro to Machine Learning 3rd -Ethem Alpaydın
Machine Learning & Pattern Recognation- Christopher Bishop

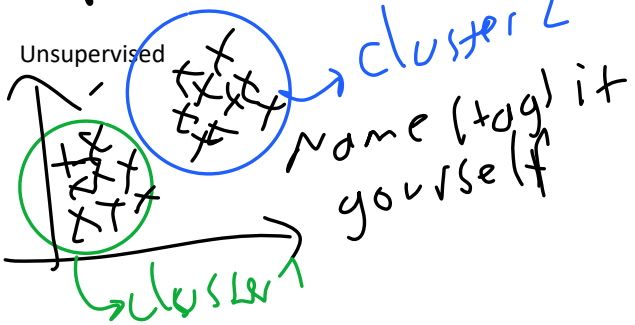


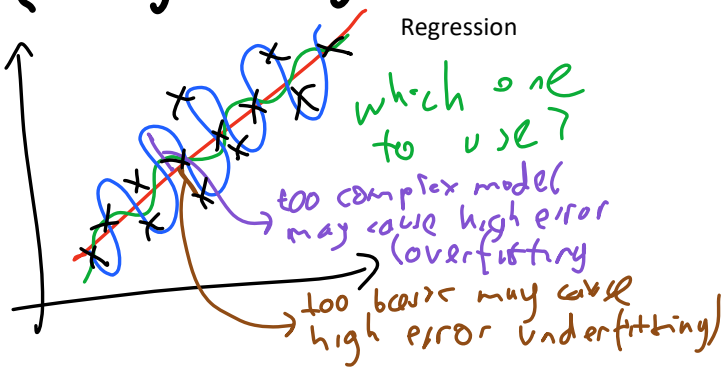
Cluster (Check video of Anil K. Join videoteachers.com
- Oranges
- lemons

Data is given : Supervised Learning -find regulars of data points
This graph two dimensinal but can be increased Like color and sugar Level

we need to turn color into math function so we test our features with training dataset. And we have to do it more than once to increase statistical accuracy, least clarification error and highest accuracy with biggest data set is the winner. We will use this diverting rule in the application

But also which features to use is important too. (Feature engineering)

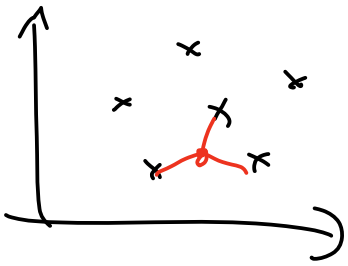




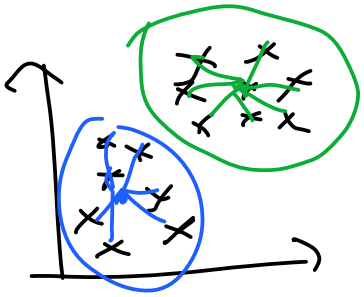
Classifiers

8 Şubat 2017 Çarşamba 14:54

First clastering K-NN



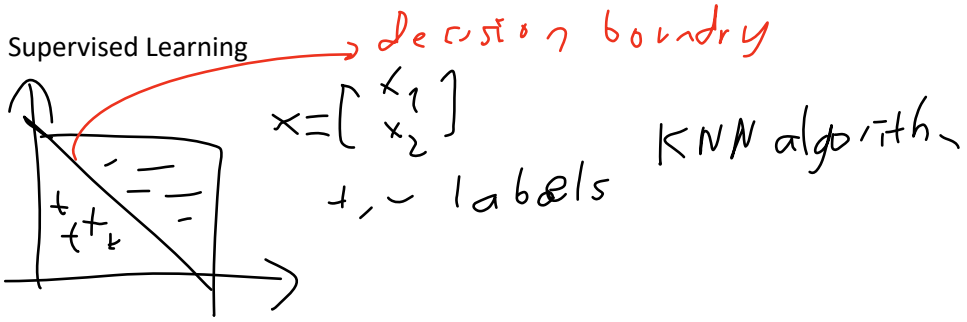
Find three closest points and assume new point is their type
Drawbacks:
KEEPING ALL TRANING DATA POINTS
ComPUTING ALL THE DISTENCES



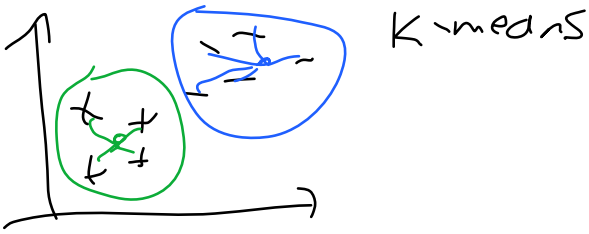
K-means clustering
K=2
Iterate to cluster centers
You may come up with Totally different clusters
Need to know numbers Of clusters
Group and calculate Mean untill distances does not change very much
Other statistics like Hubert's (not covered in class)
You can also calculate distance between cluster means and merge if its lower than threshold

15 Şubat 2017 Çarşamba 13:46

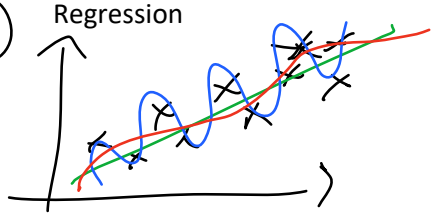
1. Supervised Learning



2. Unsupervised Learning (Clustering)



3. Regression



Keep the model simple as possible

You are given the data. You will select data randomly or Smartly (later in course) and create a model
If model has low error for both training and real data model is good
But it's impossible to get right data with random choosed trainning data (possibility might be used to select right data- check statistics notes

Statistics

15 Şubat 2017 Çarşamba 14:07

$$P(A/B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B/A)P(A)}{P(B)}$$

$$P(C/X) = \frac{P(X/C)\,P(C)}{P(X)}$$

Prior

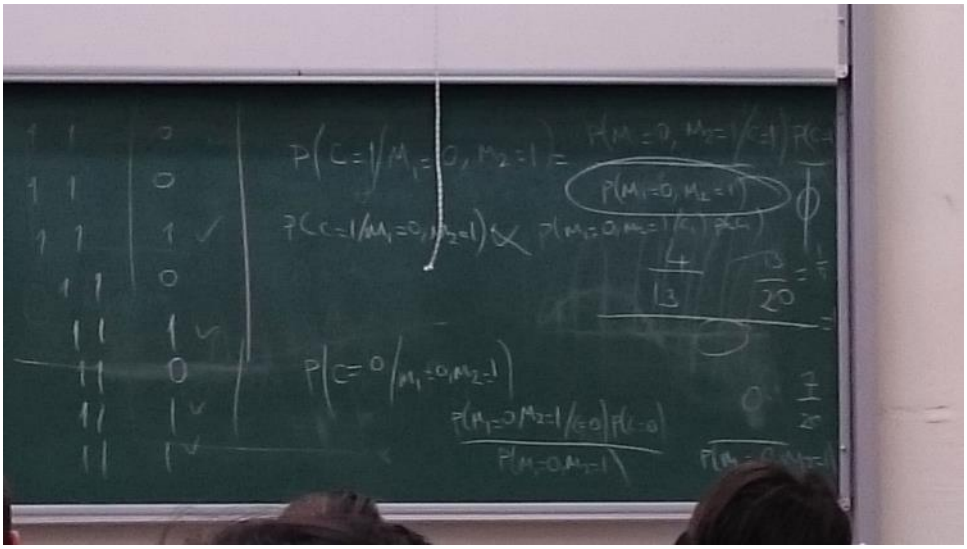
Likelihood

Class label

Feature vector

Evidence

Posterior prob.



Distribition & Expectations-Barneuli Random

A binary random variable X is a mapping from the sample space Ω to a discrete space $E = \{0, 1\}$

$X(\omega)\ \Omega \rightarrow E$

↳ set of Ω

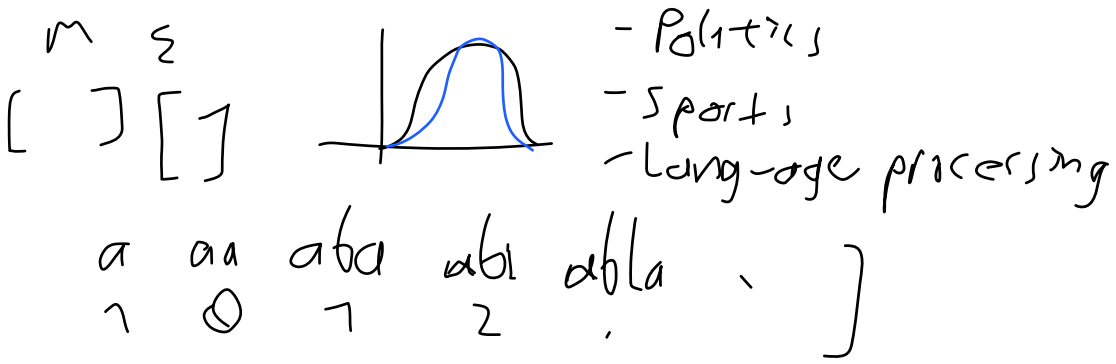
Probability Distribution (with random variable - check statistic notes

$$P(X=0) = \theta^{x}\,\theta^{x}(1-\theta)^{x}$$

Parameter

Naïve Bayes Classifiers

8 Mart 2017 Çarşamba 13:51



Naïve Bayes Classifiers : Features Conditionally Independent

$P(x_1, x_2, x_3 \dots x_n / C) \rightarrow$ Assume rindependent

$P(x_1/C)\ P(x_2/C)$

$P(C/X) = \frac{P(X/C)\ P(C)}{P(X)}$

$P(C/x_1, x_2 \dots x_n) = \frac{P(x_1, x_2 \dots x_n/C)\ P(C)}{P(X)}$ ⇒ Naive Bayes

$= \frac{\prod_{i=1}^{n} P(x_i/C)\ P(C)}{P(X)}$ Select C_i that maximizes $P(C_i/X)$

Naive Bayes works as

k classes d dimensional features

$P(C_i) = \frac{\#\text{ that belongs to } C_i}{|T|}$ ← Size of training set

Let features be X_i

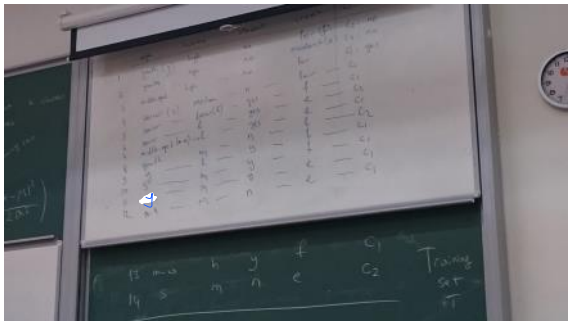
If the features are categorized and if the feature x_i has category θ (for binary features;0,1)

$P(X_i = \theta_i / C_i) = \frac{\#\theta_i}{\text{Total number of examples in class } C_i}$

x are continous

$P(X_i/C_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right)$

Method is same with bayes but assuming possibilities are independent, calculating mean and covargance matrix becomes unneccasari therefore, number of calculations will be reduced.



Classify $X = \{$age = youth, income = medium, student = yes, credit = fair$\}$

$P(C_2) = 5/14 \quad P(C_1) = \frac{9}{14}$

$P(\text{age} = \text{youth} / C = \text{buy}) = 2/9$ (C_1)

$P(\text{age} = \text{youth} / C = \text{no}) = 3/5$ (C_2)

$P(\text{student} = \text{yes} / C_1) = \frac{6}{9}$,

$P(\text{student} = \text{yes} / C_2) = 2/5$

$P(\text{income} = \text{m} / C_1) = \frac{4}{9}$

$P(\text{income} = \text{m} / C_2) = 2/5$

$P(\text{credit} = \text{f} / C_1)$

$P(\text{credit} = \text{f} / C_2) = 2/5$

$P(X / C_1) = \frac{2}{9} \cdot \frac{4}{9} \cdot \frac{6}{9} \cdot \frac{6}{9} = 0,044$

What if special set does not exist P(X) will be zero. This set may happen in real life, but it might included in training set, it may be caused by little training data. How can we solve this problem without expanding dataset available.

$P(X_i = 0) \Rightarrow \frac{0 + 1}{9 + 3}$ (3) → number of different features

0 feature için kaç farklı değer olabileceği → If there is a lot of features this will be negligible

Linear Prediction

8 Mart 2017 Çarşamba 15:03

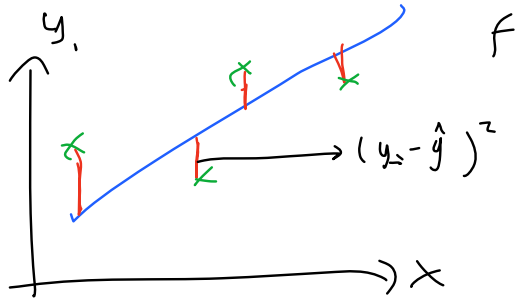
Linear Regression (Least Squares Model)
Given a training set n instances {X1in,Y1in}

Each input Xi e R^1xd a vector with d attributes (features)

Output yi referred as target yi e R n=4

Wind Speed	#people	Energy Req.
100	2	5
50	20	18
45	44	21
60	35	40

Learn a model that represents the effect of inputs to otputs



$f(x)=\theta_0+\theta_1 x_1+\theta_2 x_2 \Rightarrow$ Cost

$\sum_{i=1}(y_i-\hat{y})^2=J(\theta)$

$\sum_{i=1}(y_i-(\theta_0+\theta_1 x_i))^2$

$\arg\min_{\theta} \sum_{i=1}(y_i-\theta_0-\theta_1 x_i)^2$

$\frac{\partial J(\theta)}{\partial \theta_0}=0$

$\frac{\partial J(\theta)}{\partial \theta_1}=0$

$\vdots$

$\frac{\partial J(\theta)}{\partial \theta_d}=0$

$y_i=\theta_0+\theta_1 x_1+\ldots+\theta_d x_d \Rightarrow \hat{y}=\sum_{j=1}^{d} x_{ij}\theta_j+\theta_0=\sum_{j=1}^{d} x_{ij}\theta_j$

$X \rightarrow \begin{bmatrix} x_{10} & x_{11} & \ldots & x_{1d} \\ \vdots & & & \\ x_{n0} & & \ldots & x_{nd} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad Y: \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$

↳ artificial column added for θ_0

$X\theta=\hat{Y}$

$(Y-\hat{Y})^T(Y-\hat{Y})=J(\theta) \Rightarrow$ make square differences a matrix to make working with higher dimensions easier

$\begin{bmatrix} y_1-\hat{y}_1 & y_2-\hat{y}_2 \end{bmatrix} \begin{bmatrix} y_1-\hat{y}_1 \\ y_2-\hat{y}_2 \end{bmatrix} = (y_1-\hat{y}_1)^2+(y_2-\hat{y}_2)^2$

$$\begin{bmatrix} y_1 - \hat{y}_1 & y_2 - \hat{y}_2 \end{bmatrix} \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \end{bmatrix} = (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2$$

$$(Y - X\theta)^T (Y - X\theta) = J(\theta)$$

$$= Y^T Y - Y^T X\theta - (X\theta)^T + (X\theta)^T (X\theta)$$

$$= Y^T Y - Y^T X\theta - \theta^T X^T Y^T + \theta^T X^T X\theta$$

$$\frac{\partial}{\partial \theta} (Y^T Y - 2Y^T X\theta + \theta^T X^T X\theta) \qquad a = a^T$$

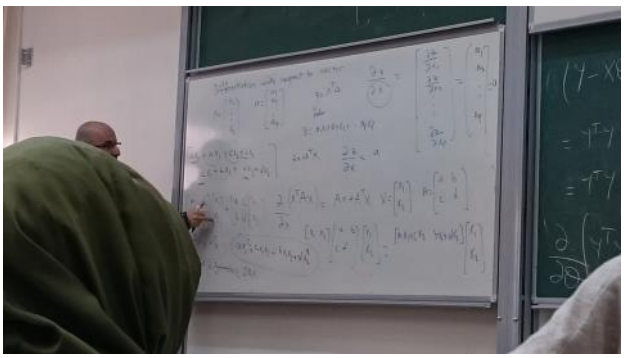
Differentiation with respect to vector

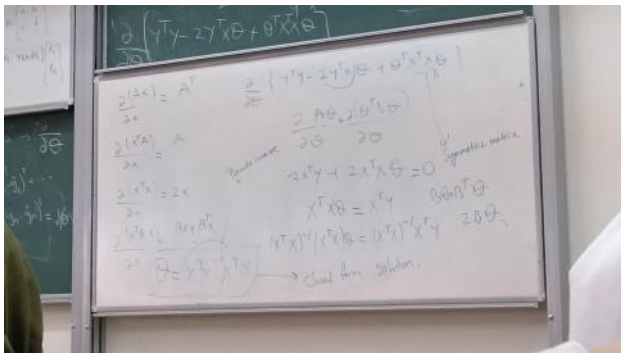
$$x = \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix} \quad a = \begin{bmatrix} a_1 \\ a_2 \\ \\ a_n \end{bmatrix} \quad z = x^T a$$

$$z = a_1 x_1 + a_2 x_2 + \dots a_p x_p$$

$$\frac{\partial z}{\partial x} = \begin{bmatrix} \frac{\partial z}{\partial x_1} \\ \frac{\partial z}{\partial x_2} \\ \\ \frac{\partial z}{\partial x_p} \end{bmatrix}$$

$$z = a^T x \quad \frac{\partial z}{\partial x} = a$$



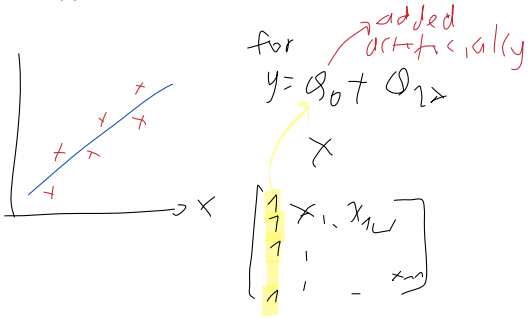


Complexity high because of large matrix multiplacations and matrix derivitavies

Q'nün boyutu (d+1)x1 - Tüm q ları bulduk

Week 5

15 Mart 2017 Çarşamba 13:41



Require the inversion of:

$X^T X$ -> If system eq. are poorly conditioned you may not be able to obtain the inverse
A solution to this problem is to add a small element (δ^2) to the diogonal:

$$\theta = (X^T X + \delta^2 I_{(d+1)(d+1)})^{-1} X^T y$$

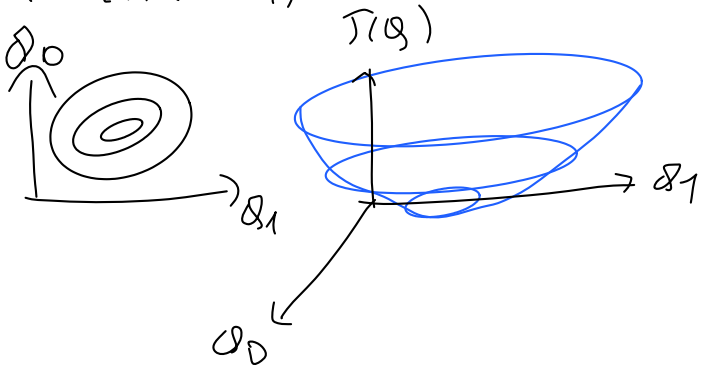
$$J(\theta) = [y_1 - \bar{y}_1 \quad y_2 - \bar{y}_2] \begin{bmatrix} y_1 - \bar{y}_1 \\ y_2 - \bar{y}_2 \end{bmatrix} = (y - \bar{y}_1)^2 + (y_2 - \bar{y}_2)^2$$

error function

$$J(\theta) = (\bar{y} - y)^T (\bar{y} - y) = \sum_{i=1}^{n} (y - \bar{y}_i)^2$$

$$= \sum_{i=1}^{n} (y_i - \theta_0 - \theta_1 x_1 - \theta_2 x_2 \dots \theta_d x_d)^2$$

$$J(\theta) = (Y - X\theta)^T (Y - X\theta)$$



To optimize (in machine learning and noral networks) we must minify J(Q). Now we will learn about iterative methotds to minimize J(Q)

Don't let big θ's to dominate others

$$J(\theta) = (Y - X\theta)^T (Y - X\theta) + \delta^2 \theta^T \theta$$

↳ Regulized cost function

δ^2 → Regularization
↳ Regulization parameter

Taking derivative of $J(\theta)$

Reminder

$$\frac{\partial \theta^T Y}{\partial X} = a$$

$$\frac{\partial X^T A X}{\partial X} = A^T y + AX$$

Taking derivative of $J(\theta)$

$$J(\theta) = Y^TY - Y^TX\theta - \theta^TX^TY + \theta^TX^TX\theta + \delta^2\theta^T\theta$$

$$\frac{\partial}{\partial\theta}\left[Y^TY - 2Y^TX\theta + \theta^TX^TX\theta + \delta^2\theta^T\theta\right]$$

$$-2X^TY + 2X^TX\theta + 2\delta^2\theta = 0$$

$$X^TX\theta + \delta^2 I\theta = X^TY$$

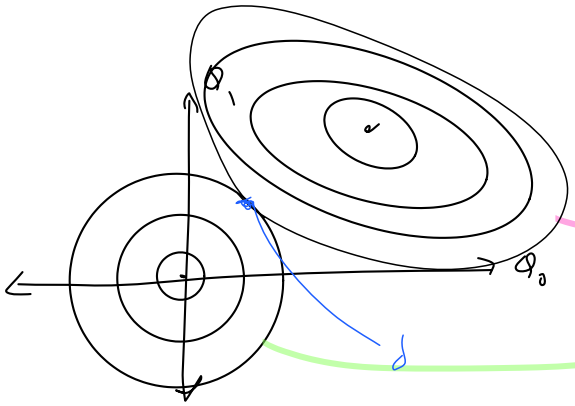
$$(XX^T + \delta^2 I)\theta = X^TY$$

$$\theta = (X^TX + \delta^2 I)^{-1}X^TY$$

this parameter must be fine tuned from the dataset

$$\frac{\partial x^TAx}{\partial x} = A^Ty + Ax$$

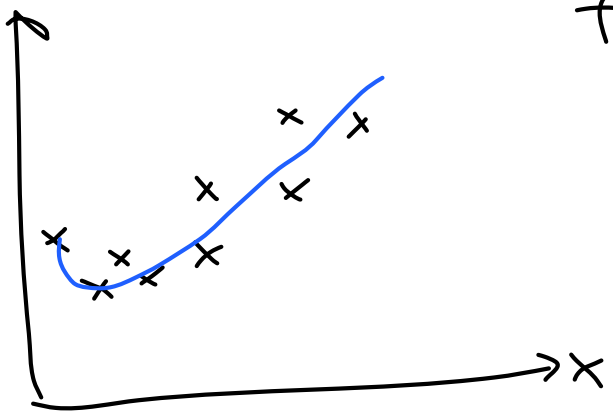
$$\frac{\partial x^TX}{\partial x} = 2x$$



$$J(\theta) = (Y - X\theta)^T(Y - X\theta) + \delta^2\theta^T\theta$$

NonLinear Prediction

15 Mart 2017 Çarşamba 14:42



$$f(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

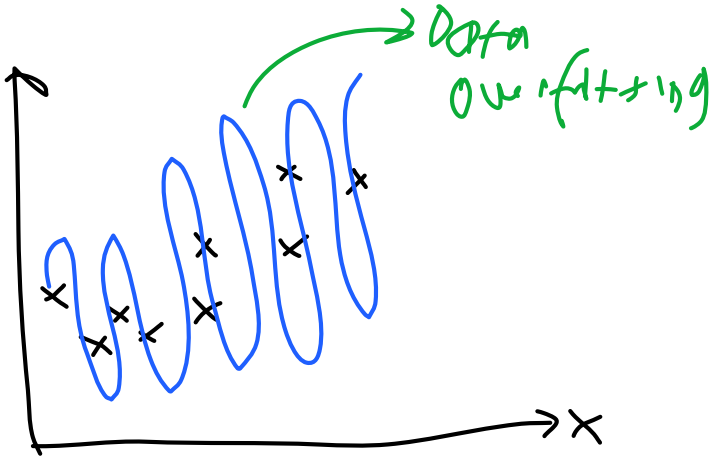
$$\underbrace{\begin{bmatrix} 1 & x & x^2 \end{bmatrix}}_{\phi(x)} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$$

Basis Vector

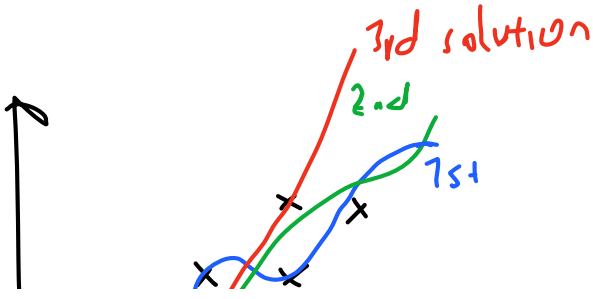
θ

Assume x^2 is another function and create basis and q matrix according to that.

$$\underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{19} \\ 1 & x_2 & x_2^2 & & x_2^{19} \\ \vdots & & & & \\ 1 & x_n & x_n^2 & & x_n^{19} \end{bmatrix}}_{X} \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \\ \theta_{19} \end{bmatrix}$$

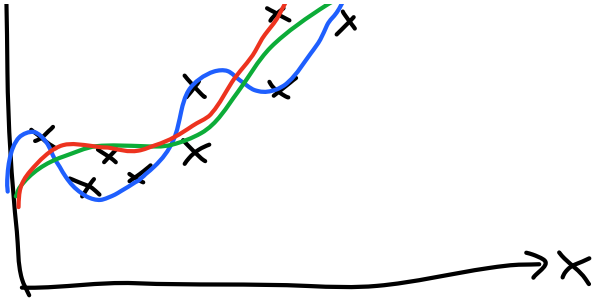


If you don't Use regulator yor. Data will be overfitted



$$f(x) = \theta_0 + \theta_1 x + \cdots \theta_{14} x^{14}$$

$$\delta_1^2 < \delta_2^2 < \delta_3^2$$



$\sigma_1 < \sigma_2 < \sigma_3$

cross validation

Kernel Regression and RBF (Radial Basis Functions)

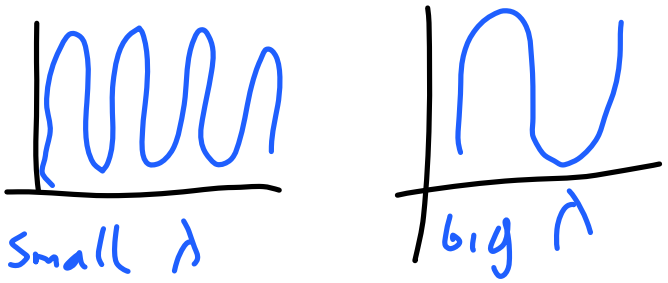
15 Mart 2017 Çarşamba 15:00

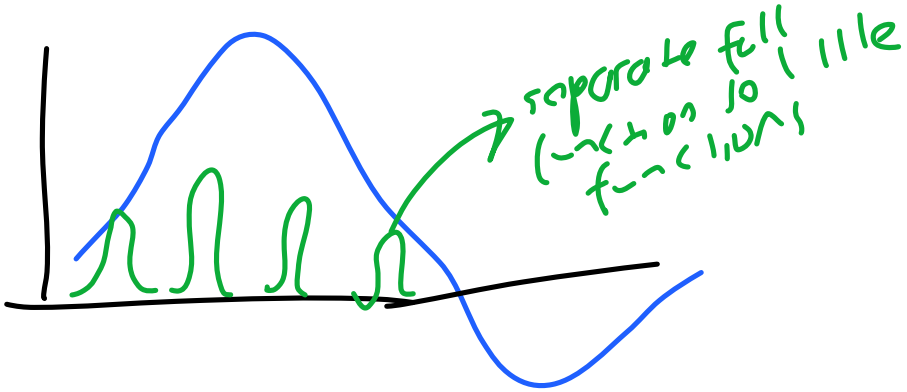
$$\phi(x) = [K(x, \mu_1, d), \quad K(x, \mu_2, \lambda)]$$

$$K(x_1, \mu_1, \lambda) = e^{-\frac{1}{\lambda}\|x - \mu_1\|^2}$$

Assume that $\mu_1 = 1$ $\mu_2 = 2$ $\mu_3 = 4$ $\lambda = 1$

If you have small lambda you can overfit data (little signals)





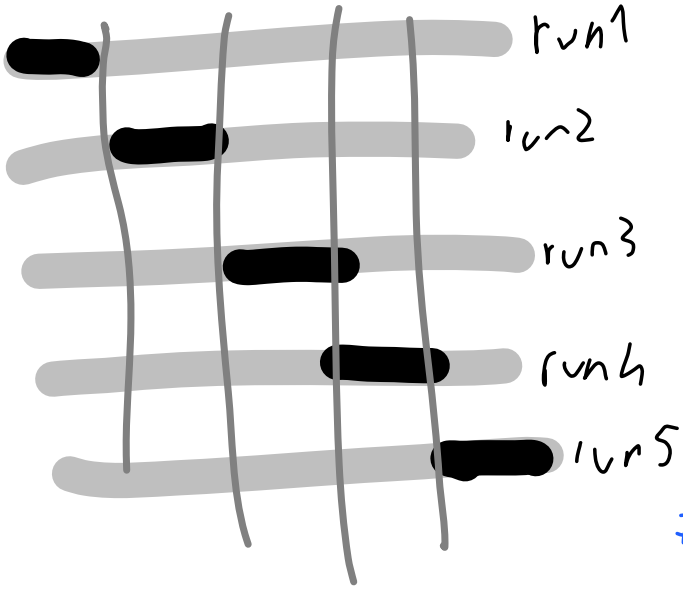
Cross validation

15 Mart 2017 Çarşamba 15:19

Finding best δ

k-fold Cross validation

k=5



N : the total number of data samples k=N

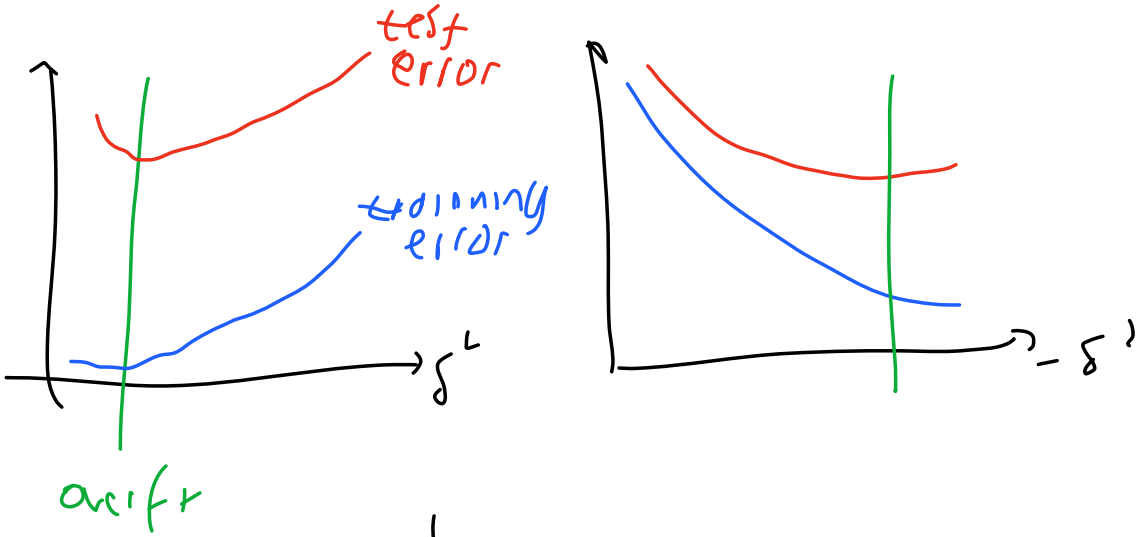
train	test	run
—	—	1
—	—	2
—	—	3
—	—	4
—	—	5
+ —	—	

switch each

Avg —

Find for each δ and pick minimum one

If you have too much classes, this method is hard and unnecessary to apply



An extreme example

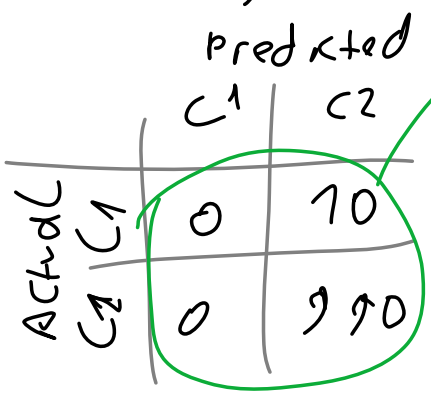
10 C1 990 C2

Unbalanced data set

If you classify everything as C2 you will have %99 accuracy but it's not a good model

→ confusion matrix

accuracy but ifs not



confussion matrix

Note Ask about dataset
If you want to get a
good idea about model

Bias-Variance Dilemma

22 Mart 2017 Çarşamba 13:42

Genarilaztion Error: Erron on the unseen examples. There is a mathematical formula:

Generilization Error: $Bias^2$+Variance

Difference between the actual function and predicted function is named as bias

For example: actual function is linear but you fit 16th order function. Bias will be high. If I will increase the complexity of the model bias will decrease. If we fit 19th order polynomial they wont vary too much from the 16th order polynomial. If we increase complexity of the model, variance will increase.

We don't have a chance to compute bias. Its only possible in simulations.

But we can estimate variance

Since we cant calculate generilization error we will use cross validation

Gradient Descent Optimization Method

22 Mart 2017 Çarşamba 14:09

Used in neural networks.

Closed Form solution: $Q=Q_0+Q_1x$ (Q calculated in one step with $(X^TX)X^{-1}Y$

Solution Method
Let Q be a d dimensional vector

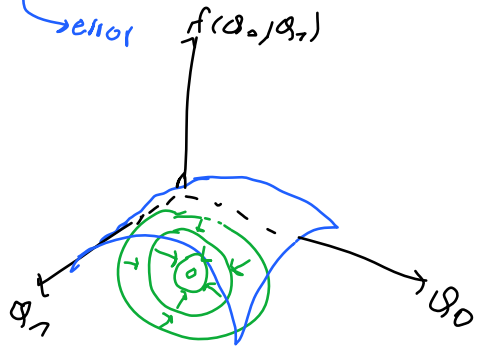
f(Q) is a scalar valued function.

Gradient vector of f(.) with respect to Q = $\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_n \end{bmatrix}$

$$\nabla_\theta f(\theta) = \begin{bmatrix} \frac{\partial f(\theta)}{\partial \theta_1} \\ \frac{\partial f(\theta)}{\partial \theta_2} \\ \vdots \\ \frac{\partial f(\theta)}{\partial \theta_n} \end{bmatrix}$$

$$J(\theta) = \sum_{i=1}^{n} (y_i - x\theta)^2 \quad d=2 \quad f(\theta_0, \theta_1)$$

error



Steepest Gradient Algorithm
-Start with a random Q guess vector
-Update Q $\nabla f(\theta_k)$ learning rate ex 0.1 or 2
$Q_{k+1} = Q_k -$

Perception Algorithm

12 Nisan 2017 Çarşamba 13:49

$x_1^{(1)}, x_2^{(1)}, x_3^{(1)} \dots x_d^{(1)}$

$w_1 x_1 + w_2 x_2 + \dots + w_d x_d > thr$

$< -w_0$

Always 1 feature

$w_0 x_0 + w_1 x_1 + \dots + w_d x_d < 0$

$w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \quad x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$

$r \in \{+1, -1\}$

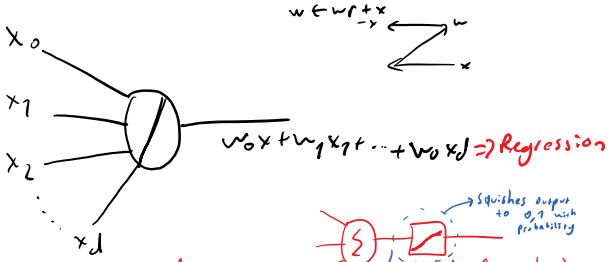
if $> thr$ then $+$

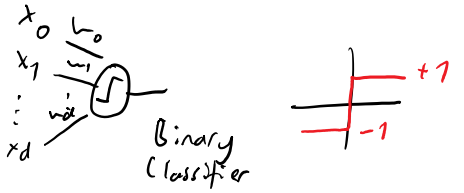
else $-$

Class label $= sign(w^T x)$

Starts with an arbitrary weight vector.
Classify all training examples using $w^T x$
Select one misclassified example

$w \leftarrow w r x$



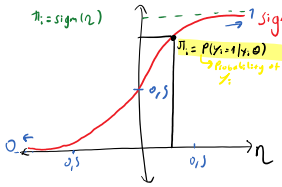


Sigmoid function

$g(s) = \frac{1}{1+e^{-s}} \Rightarrow$ Logistic (Logit) function

↳ apply this to Bernoulli random variable

$g(w^T x) = \frac{1}{1+e^{-w^T x}}$

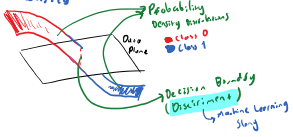


Done like this because we can implement distributions easily

$\eta = x\theta$ is a plane!

$\eta = x\theta$

$y \in \{0,1\}$



0,5 is threshold for classifying

$y = \frac{1}{exp(w^T x) + 1} \gtrless 0,5$

Using maximum likelihood parameter $\Rightarrow \prod_{i=1}^{N} (y^t)^{r^t} (1-y^t)^{(1-r^t)}$ ⇒ Maximize this for a good model

$\log \prod_{i=1}^{N} (y^t)^{r^t} (1-y^t)^{(1-r^t)} = \sum_{t=1}^{N} \log \left((y^t)^{r^t} (1-y^t)^{(1-r^t)} \right) = \sum_{t=1}^{N} \left(r^t \log y^t + (1-r^t) \log (1-y^t) \right)$

this minimize minus my last edf of this maximizing

$\sum (| |$

$-\sum p_i \log p_i \Rightarrow$ Entropy

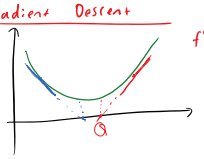
↳ information theory ⇒ Claude Shannon, John Von Neumen, Turing

Bayesian Classes ⇐ David Mackay

Bill gates's book: Seeing future or not?

$- \sum_{t=1} r^t \log y^t + (1-r^t) \log (1-y^t)$

Gradient Descent

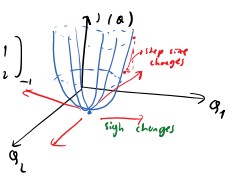


$f'(\theta)$

$\theta \leftarrow \theta - \eta f(\theta)$

$\theta \leftarrow \theta - \eta f'(\theta)$

$\nabla(\theta) = \begin{bmatrix} \frac{\partial}{\partial \theta_1} \\ \frac{\partial}{\partial \theta_2} \end{bmatrix}$



$\frac{1}{1+exp(-w^T x)} = \frac{1}{1+exp(-w^T x - w_0)}$

d+1 d

start with w vector

$w \leftarrow w - \eta \nabla J(w)$

$\frac{\partial}{\partial s} g(s) = g(s)(1-g(s))$

↳ need to add ds (additional multiplication)

$\frac{\partial}{\partial s} \frac{1}{(1+e^{-s})} = \frac{0+e^{-s}}{(1+e^{-s})^2} = \frac{1+e^{-s}-1}{(1+e^{-s})^2} = \frac{1+e^{-s}}{(1+e^{-s})^2} - \frac{1}{(1+e^{-s})^2} = \frac{1}{1+e^{-s}} - \left(\frac{1}{1+e^{-s}}\right)^2 = g(s) - g(s)^2 = g(s)(1-g(s))$

$$\frac{\partial}{\partial s}\frac{1}{(1+e^{-s})} = \frac{0+e^{-s}}{(1+e^{-s})^2} = \frac{1+e^{-s}-1}{(1+e^{-s})^2} = \frac{1+e^{-s}}{(1+e^{-s})^2} - \frac{1}{(1+e^{-s})^2} = \frac{1}{1+e^{-s}} - \frac{1}{(1+e^{-s})^2} = g(s)(1-g(s))$$

$$\Delta w_j = \sum_{t=1}^{N}\left[\frac{r^t}{y^t} + \frac{(1-r^t)}{1-y^t}\right]\frac{\partial y^t}{\partial w}$$

coming from $\frac{\partial}{\partial w}\log = \frac{w'}{w}$

derivative for each w

$$= \sum_{t=1}^{N}\left[\frac{r^t}{y^t}\frac{\partial y^t}{\partial w} + \frac{(1-r^t)}{(1-y^t)}\frac{\partial}{\partial w}(1-y^t)\right] = \sum_{t=1}^{N}\left[\frac{r^t}{y^t} - \frac{(1-r^t)}{1-y^t}\right]\frac{dy^t}{dw_j}$$

$$\frac{\partial}{\partial w_j} = \frac{1}{1+e^{-(w^T x + w_0)}} = \frac{0+x_j e^{-(w^T x+w_0)}}{1+(e^{-(w^T x+w_0)})^2} \Rightarrow \frac{\partial}{\partial s}g(s) = g(s)\,(1-g(s))\,x_j'$$

$g(s)$ → outcome of the model

$$w_j \leftarrow w_j - \left(-\eta \sum_{t=1}^{N}\left[\frac{r^t}{y^t} - \frac{(1-r^t)}{1-y^t}\right] y^t(1-y^t)x_j^t\right)$$

$$\Delta w_0 = -\sum_{t=1}^{N}(r^t - y^t)$$

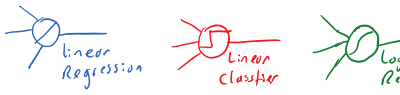
use this two to calculate α values (weights)

At the beginning of the algorithm use very small α's to stay away from overfitting. Using big random numbers can create phantom dominant or phantom weak features

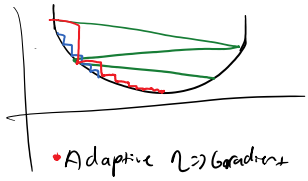
r^t 1 if y=1 0 if y=0 (In the training data)

- Find a good η (learning rate)
- Generalize algorithm; afterall its for binary.

An Example Neural Cell:



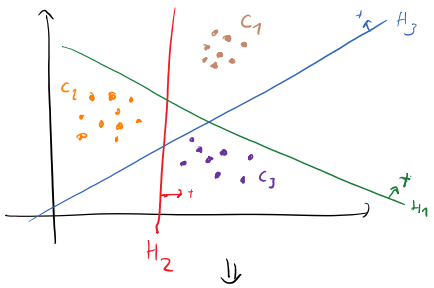
$\log\left(\prod_{t=1}^{N}(y^t)^{r^t}(1-y^t)^{(1-r^t)}\right)$ - Construct Method - Optimize it



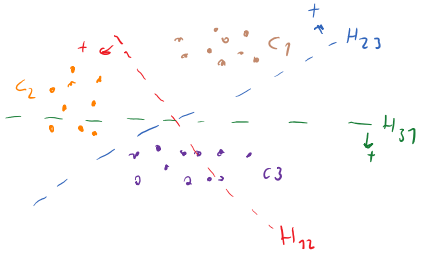
- small η means small steps => slow learning
- big η means big steps => fast learning but possibility to pass point => starting to do back and forth
- Adaptive η => Gradient ↳ Update η during iterations

Increasing K

$K > 2$?



$\text{class} = \max_i H_i(x)$, $i = 1,2,\dots K$



H_{23} >0 C_2 <0 C_3 | H_{31} >0 C_3 <0 C_1 | H_{12} >0 C_1 <0 C_2

Choose C_i if $\forall j \neq i$, $H_{ij}(x) > 0$

$H_{ij}(x) > 0$ if $x \in C_i$

≤ 0 if $x \in C_j$

Sum of Outputs of posterior models must be one since:

$$P(C|x) = y^t = \frac{1}{1+\exp(w^T x + w_0)} \qquad r^t \in \{0,1\}$$

$$P(C=1|x) = y^t = \frac{1}{1+\exp(w^T x + w_0)}$$

$$P(C=0|x) = 1 - \frac{1}{1+\exp(w^T x + w_0)}$$

for $K>2$:

modify function to handle sum of probabilities >1

$r^t \in \text{mult}(K)$

$$\frac{1}{1+e^{-s}} = \frac{e^s}{1+e^s}$$

$$p(C_i \mid x) = \frac{\exp(w_i^T x + w_0)}{\sum_{j=1}^{K} \exp(w_j^T x + w_0)} \qquad i = 1, \dots K$$

Softmax function

$$\begin{matrix} p(C=1 \mid x) \\ p(C=2 \mid x) \\ \vdots \\ + \quad p(C=K \mid x) \end{matrix} \qquad 1$$

$(y^t)^{r^t} (1-y^t)^{(1-r)^t}$ ⇒ $r_i^t = 1$ ⇒ if class belongs to C_i ; $r_i^t = 0$ ⇒ else its 0.

$r_i = [0 \;\; 1 \;\; 0]$ → C_2

y_i [0,1 0,8 0,1]

We must change those too

$$L = \prod_{t=1}^{N} \prod_{i=1}^{K} (y_i)^{r_i^t}$$

$$l = \log \prod_{t=1}^{N} \prod_{i=1}^{K} (y_i)^{r_i^t} \Rightarrow \Delta w_{ij} \Rightarrow \text{apply gradient descent}$$

HW: ↳ expand and take derivative for w_{ij}

Logistic regression: (Is a batch learning ⇒ use all data in one time)

$$p(y \mid X, \theta) = \prod_{i=1}^{n} \text{Ber}(y_i \mid \text{sigm}(x_i \theta))$$

↳ $p(y_i \mid x_i, \theta) \Rightarrow$ Bernoulli distribution

$$= \prod_{i=1}^{n} \left[\frac{1}{1+e^{-x_i \theta}}\right]^{y_i} \left[1 - \frac{1}{1+e^{-x_i \theta}}\right]^{1-y_i} \Rightarrow \text{where } x_i \theta = \theta_0 + \sum_{j=1}^{d} \theta_j x_{ij} = \bar{y}_i$$

y_i is either 1 or 0 so one of the probabilities will not affect results

$$c(\theta) = -\log p(Y \mid X; \theta)$$

$$= -\sum_{i=1}^{n} y_i \log \pi_i + (1-y_i) \log(1-\pi_i) \Rightarrow \text{cross entropy (cost - optimize - minimize)}$$

$$\theta_{k+1} = \theta_k - H^{-1} g_k$$

$g_k = X^T(\pi_k - y)$

$H_k = X^T S_k X$

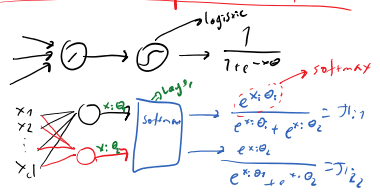
$S_k = \text{diag}(\pi_{1k}(1-\pi_{1k}), \dots \pi_{nk}(1-\pi_{nk}))$

$\pi_{ik} = \text{sigm}(x_i \theta_k)$

$$\theta_{k+1} = (X^T S_k X)^{-1} X^T [S_k X \theta_k + y - \pi_k]$$

(next iteration)

Neural Network representation of loss



$\pi_{i1} + \pi_{i2} = 1$

Indicator

$$\Pi_c(y_i) = \begin{cases} 1 & \text{if } y_i = c \\ 0 & \text{otherwise} \end{cases}$$

$$p(Y \mid X, \theta) = \prod_{i=1}^{n} p(y_i \mid x_i; \theta) = \prod_{i=1}^{n} \pi_{i1}^{\Pi_0(y_i)} \pi_{i2}^{\Pi_1(y_i)}$$

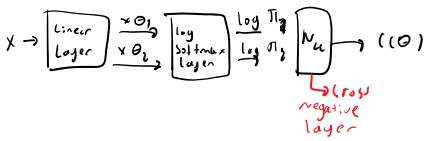
one of them will be ineffective since $\Pi_0 = 1 \Rightarrow \Pi_1 = 0$, $\Pi_1 = 1 \Rightarrow \Pi_0 = 0$

$$c(\theta) = -\log p(y \mid x, \theta) = -\sum_{i=1}^{n} \Pi_0(y_i) \log \pi_{i1} + \Pi_1(y_i) \log \pi_{i2}$$

Log will make things easier if you are trying to minimize.

$x \rightarrow$ [Linear Layer] $\xrightarrow{\times \theta_1 \; \times \theta_2}$ [log Softmax] $\xrightarrow{\log \pi_1 \; \log \pi_2}$ [N_{LL}] $\rightarrow c(\theta)$

to minimize.



Modifing Learning mode with new coming data (Online Learning)

Learning problem can vary in change. If you apply online learning, learning will be better. Applying online learning to gradient descent is called Stochastic Gradient Oesent

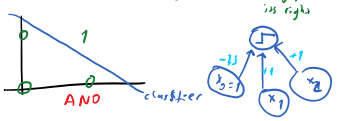
Assume applying to one data:

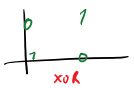
$$\Delta w = \eta \sum_{t=0} (r^t - y^t) x^t \quad = \eta (r^t - y^t) x^t$$

dissapears (one input) ($\eta = 1$)

→ input
→ observed
→ desired
minimize $(r^t - y^t) = 0$ its right

x_1	x_2	AND	XOR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



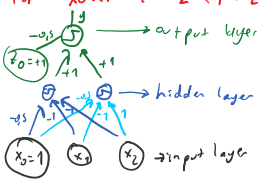


A linear classifier can not classifiy more than 3 points in two dimensions.

For linear classifier in d dimension d+1 data points can be correctly classified

Multi-Layer Neural Network:

For XOR: $x_1 \oplus x_2 = (x_1 \wedge x_2) \vee (x_1 \wedge x_2)$

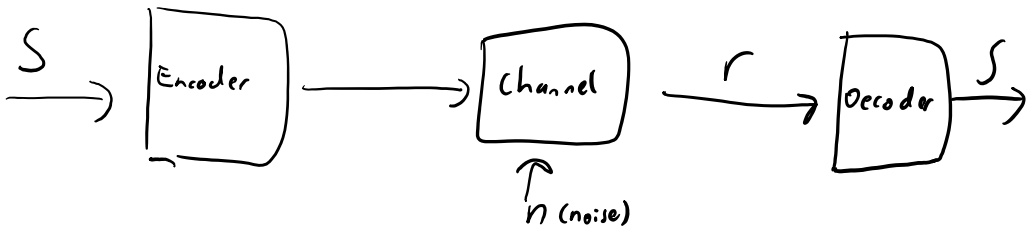


They can easily overfit the data
But they can give non-linear boundries

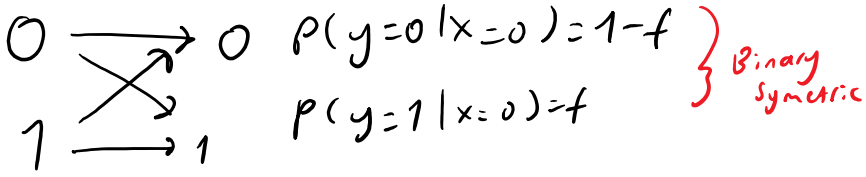
Information Theory

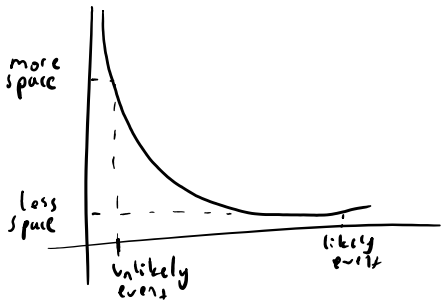
13 Nisan 2017 Perşembe 22:25

Making a reliable communications systems
↳ Change channel to less noisy one
↳ Accept noise and handle it



Binary Symetric Channel (Binary)



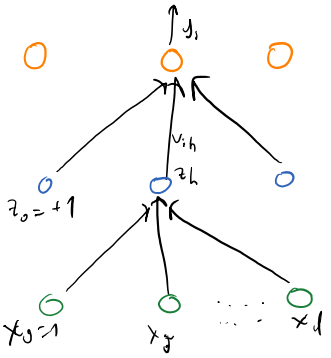


$$H(x) = -\sum_{x} p(x|\theta) \log p(x|\theta)$$

↳ uncertainty

Mutilayer Perceptrons - chapter 11

19 Nisan 2017 Çarşamba 14:48



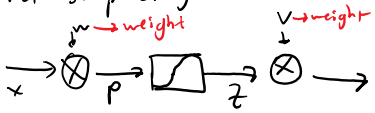
$$y_i = v^T z = \sum_{h=1}^{H} v_{ih} z_h + v_{i0}$$

$$z_h = \text{sigmoid}(w_h^T x) = \frac{1}{1+\exp\left[-\left(\sum_{j=1} w_{hj} x_j + w_{h0}\right)\right]}$$

An example model for parameter learning will be given and then we will apply backpropagation

Back propagation

For simplicity we will consider one input & neuron: (can be duplicated)



Going backwards

Using Regression

minimize error ($E = \frac{1}{2}(r-y)^2$) → r: actual value

Another regression method can be used If you want to use gradient descent just apply it on E

$$\frac{\partial E}{\partial v} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial v} = -1(r-y) \cdot z$$

$$\Delta v = -\eta \frac{\partial E}{\partial v} \quad \text{update for } v$$

$$\Delta v = \eta (r-y) z \qquad v_{i+1} = v_i + \Delta v$$

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial w} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial p} \cdot \frac{\partial p}{\partial w}$$

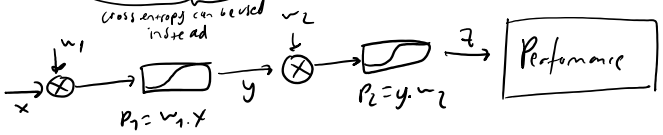
$$-(r-y) v \cdot z(1-z) \cdot x \qquad \Delta w = -\eta \frac{\partial E}{\partial w}$$

$$\Delta w = \eta (r-y) v \cdot z(1-z) \cdot x$$

For multiple inputs:

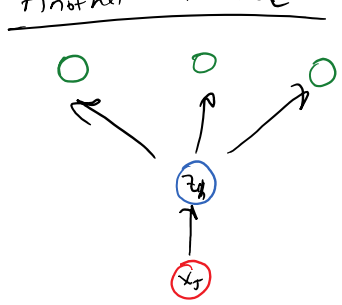
$$E = \frac{1}{2} \sum_t (r^t - y^t)^2 \qquad \Delta v = \eta \sum_t (r^t - y^t) \cdot z_h^t$$

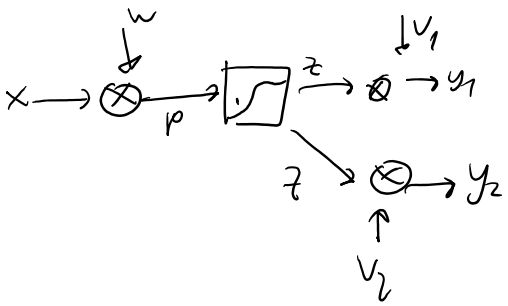
cross entropy can be used instead



$$\log\left(y^r (1-y)^{1-r}\right) \qquad \frac{\partial E}{\partial w_2} = -(r-z) \cdot z(1-z) w_2 y(1-y) x$$

Another Model





x → r_1, r_2 (z output)

input

Error function

$$E = \frac{1}{2}\left[(r_1 - y_1)^2 + (r_2 - y_2)^2\right]$$

$$\frac{\partial E}{\partial v_1} = \frac{\partial E}{\partial y_1} \cdot \frac{\partial y_1}{\partial v_1} = -(r_1 - y_1) \cdot z \qquad \frac{\partial E}{\partial v_2} = -(r_2 - y_2) \cdot z$$

$$= \frac{\partial}{\partial z}\left[\frac{1}{2}(r_1-y_1)^2\right] + \frac{\partial}{\partial z}\left[\frac{1}{2}(r_2-y_2)^2\right] \qquad \frac{\partial \xi}{\partial z} = \frac{\partial}{\partial z}\left[\frac{1}{2}(r_1-y_1)^2 + (r_2-y_2)^2\right]$$

$$\frac{\partial \xi}{\partial w} = \frac{\partial \xi}{\partial p}\cdot\frac{\partial p}{\partial w} = \frac{\partial \xi}{\partial z}\cdot\frac{\partial z}{\partial p}\cdot\frac{\partial p}{\partial w}$$

$\frac{\partial p}{\partial w}$ → $z(1-z)$

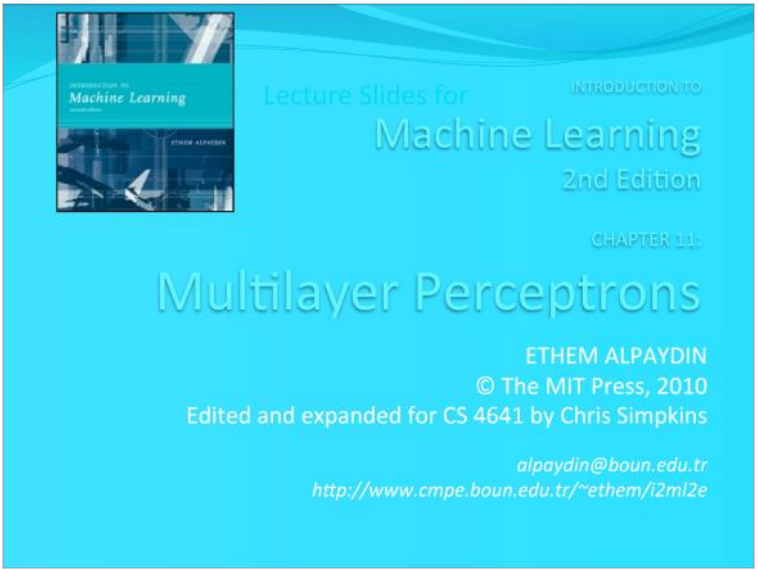
$\frac{\partial \xi}{\partial z}$ → $-(r_1-y_1)v_1-(r_2-y_2)v_2$

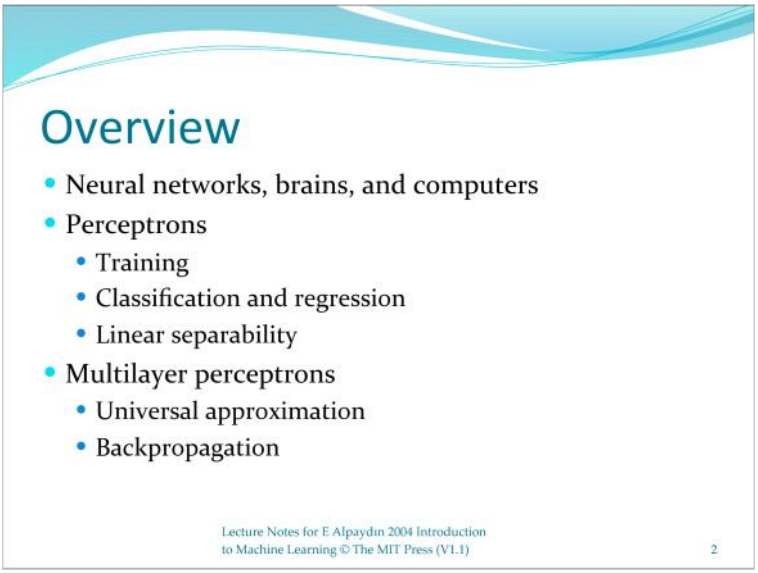
$$\frac{\partial \xi_1}{\partial z} + \frac{\partial \xi_2}{\partial z} = \frac{\partial \xi_1}{\partial y_1}\cdot\frac{\partial y_1}{\partial z} + \frac{\partial \xi_2}{\partial y_2}\cdot\frac{\partial y_2}{\partial z} = -(r_1-y_1)\cdot v_1 + -(r_2-y_2)v_2$$

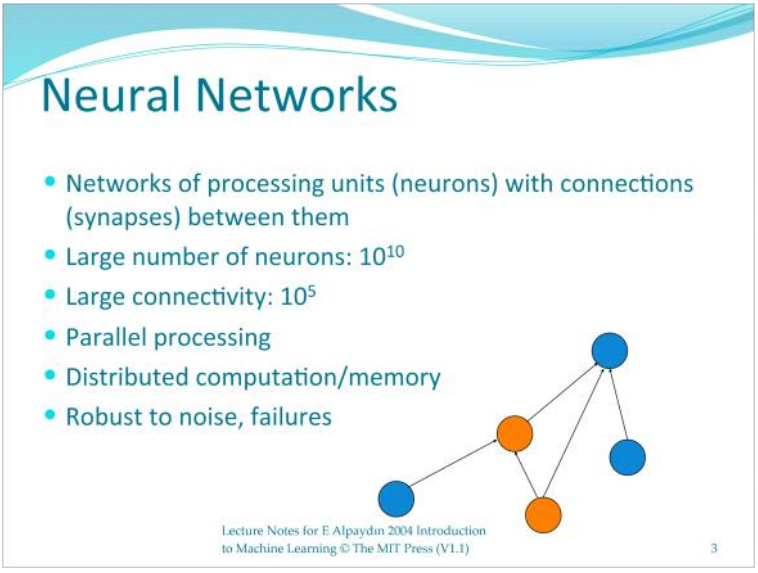
$$= [-(r_1-y_1)v_1 - (r_2-y_2)v_2]$$

$$= -\sum_k (r_k-y_k)v_k \quad \text{for } k \text{ output units}$$

multilayer-perceptro...







Understanding the Brain

- Levels of analysis (Marr, 1982)
  1. Computational theory
  2. Representation and algorithm
  3. Hardware implementation
- Reverse engineering: From hardware to theory
- Parallel processing: SIMD vs MIMD

  Neural net: SIMD with modifiable local memory

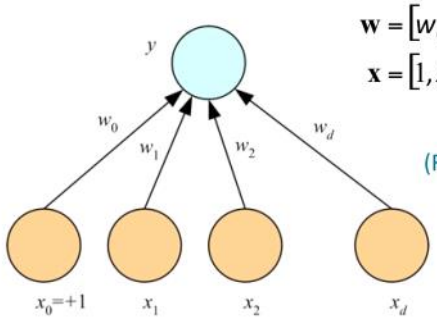
  Learning: Update by training/experience

Perceptron

$$y = \sum_{j=1}^{d} w_j x_j + w_0 = \mathbf{w}^T \mathbf{x}$$

$$\mathbf{w} = [w_0, w_1, \ldots, w_d]^T$$

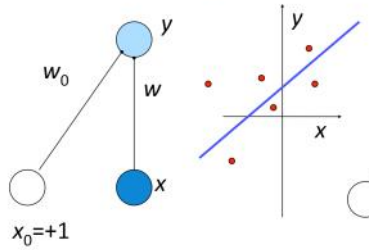
$$\mathbf{x} = [1, x_1, \ldots, x_d]^T$$



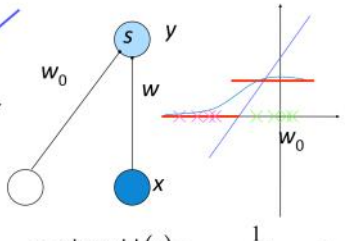
(Rosenblatt, 1962)

What a Perceptron Does

- Regression: $y=wx+w_0$
- Classification: $y=1(wx+w_0>0)$



Linear fit



$$y = \text{sigmoid}(o) = \frac{1}{1+\exp[-\mathbf{w}^T\mathbf{x}]}$$

Linear discrimination

K Outputs

Regression:

$$y_i = \sum_{j=1}^{d} w_{ij} x_j + w_{i0} = \mathbf{w}_i^T \mathbf{x}$$

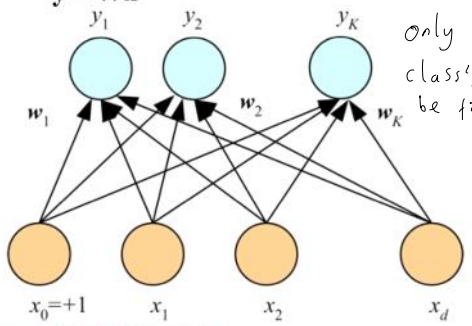
$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

Classification:

$$o_i = \mathbf{w}_i^T \mathbf{x}$$

$$y_i = \frac{\exp o_i}{\sum_k \exp o_k}$$

choose C_i if $y_i = \max_k y_k$



Only corresponding class's neuron will be fired ($y_i = 1$)

Training

- Online (instances seen one by one) vs batch (whole sample) learning:
  - No need to store the whole sample
  - Problem may change in time
  - Wear and degradation in system components
- Stochastic gradient-descent: Update after a single pattern
- Generic update rule (LMS rule):

$$\Delta w_{ij}^t = \eta \left(r_i^t - y_i^t \right) x_j^t$$

Update=LearningFactor*(DesiredOutput−ActualOutput)*Input

Training a Perceptron: Regression

- Regression (Linear output):

$$E^t\left(\mathbf{w} \mid \mathbf{x}^t, r^t\right) = \frac{1}{2}\left(r^t - y^t\right)^2 = \frac{1}{2}\left[r^t - \left(\mathbf{w}^T \mathbf{x}^t\right)\right]^2$$

$$\Delta w_j^t = \eta \left(r^t - y^t \right) x_j^t$$

Classification

- Single sigmoid output

$$y^t = \text{sigmoid}\left(\mathbf{w}^T\mathbf{x}^t\right)$$
$$E^t\left(\mathbf{w}\mid\mathbf{x}^t,\mathbf{r}^t\right) = -r^t \log y^t - \left(1-r^t\right)\log\left(1-y^t\right)$$
$$\Delta w_j^t = \eta\left(r^t - y^t\right)x_j^t$$

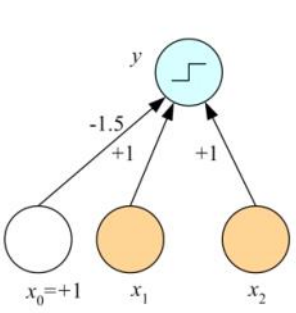
- $K>2$ softmax outputs

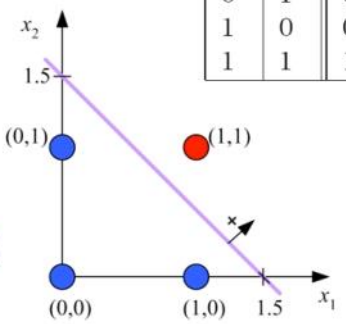
$$y_i^t = \frac{\exp \mathbf{w}_i^T\mathbf{x}^t}{\sum_k \exp \mathbf{w}_k^T\mathbf{x}^t} \qquad E^t\left(\{\mathbf{w}_i\}_i \mid \mathbf{x}^t,\mathbf{r}^t\right) = -\sum_i r_i^t \log y_i^t$$
$$\Delta w_{ij}^t = \eta\left(r_i^t - y_i^t\right)x_j^t$$

Same as for linear discriminants from chapter 10 except we update after each instance

Learning Boolean AND

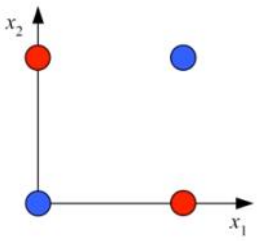
x_1	x_2	r
0	0	0
0	1	0
1	0	0
1	1	1





XOR

x_1	x_2	r
0	0	0
0	1	1
1	0	1
1	1	0



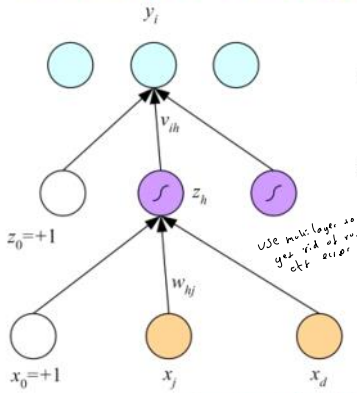
- No w_0, w_1, w_2 satisfy:

$$w_0 \le 0$$
$$w_2 + w_0 > 0$$
$$w_1 + w_0 > 0$$
$$w_1 + w_2 + w_0 \le 0$$

(Minsky and Papert, 1969)

important

Multilayer Perceptrons



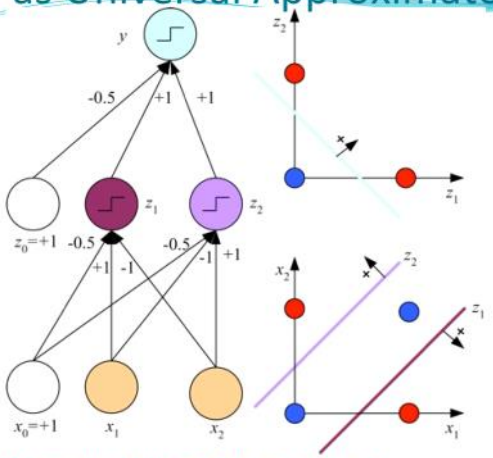
$$y_i = \mathbf{v}_i^T \mathbf{z} = \sum_{h=1}^{H} v_{ih} z_h + v_{i0}$$

$$z_h = \text{sigmoid}\left(\mathbf{w}_h^T \mathbf{x}\right) = \frac{1}{1+\exp\left[-\left(\sum_{j=1}^{d} w_{hj} x_j + w_{h0}\right)\right]}$$

use multilayer so you rid of round etc error

(Rumelhart et al., 1986)

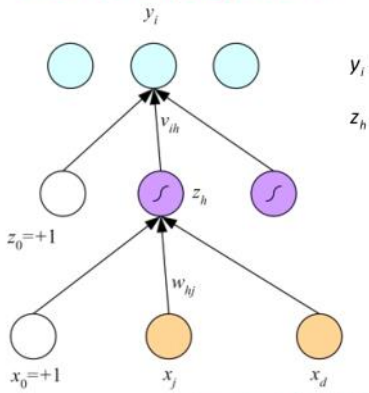
MLP as Universal Approximator



x_1 XOR x_2 = (x_1 AND ~x_2) OR (~x_1 AND x_2)

Backpropagation

Found de => weight b/c colors.



$$y_i = \mathbf{v}_i^T \mathbf{z} = \sum_{h=1}^{H} v_{ih} z_h + v_{i0}$$

$$z_h = \text{sigmoid}\left(\mathbf{w}_h^T \mathbf{x}\right) = \frac{1}{1+\exp\left[-\left(\sum_{j=1}^{d} w_{hj} x_j + w_{h0}\right)\right]}$$

$$\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial z_h} \frac{\partial z_h}{\partial w_{hj}}$$

Regression

$$E(\mathbf{W},\mathbf{v}\mid\mathcal{X})=\frac{1}{2}\sum_t\left(r^t-y^t\right)^2$$

$$y^t=\sum_{h=1}^{H}v_h z_h^t+v_0$$

$$\Delta v_h=\sum_t\left(r^t-y^t\right)z_h^t$$

Backward

Forward

$$\Delta w_{hj}=-\eta\frac{\partial E}{\partial w_{hj}}$$
$$=-\eta\sum_t\frac{\partial E}{\partial y^t}\frac{\partial y^t}{\partial z_h^t}\frac{\partial z_h^t}{\partial w_{hj}}$$
$$=-\eta\sum_t-\left(r^t-y^t\right)v_h z_h^t\left(1-z_h^t\right)x_j^t$$
$$=\eta\sum_t\left(r^t-y^t\right)v_h z_h^t\left(1-z_h^t\right)x_j^t$$

$$z_h=\text{sigmoid}\left(\mathbf{w}_h^T\mathbf{x}\right)$$

$\mathbf{x}$

If you have a classification problem use it and log

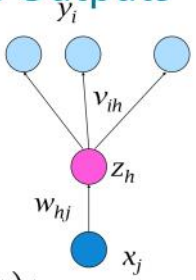
Regression with Multiple Outputs

$$E(\mathbf{W},\mathbf{V}\mid\mathcal{X})=\frac{1}{2}\sum_t\sum_i\left(r_i^t-y_i^t\right)^2$$

$$y_i^t=\sum_{h=1}^{H}v_{ih}z_h^t+v_{i0}$$

$$\Delta v_{ih}=\eta\sum_t\left(r_i^t-y_i^t\right)z_h^t$$

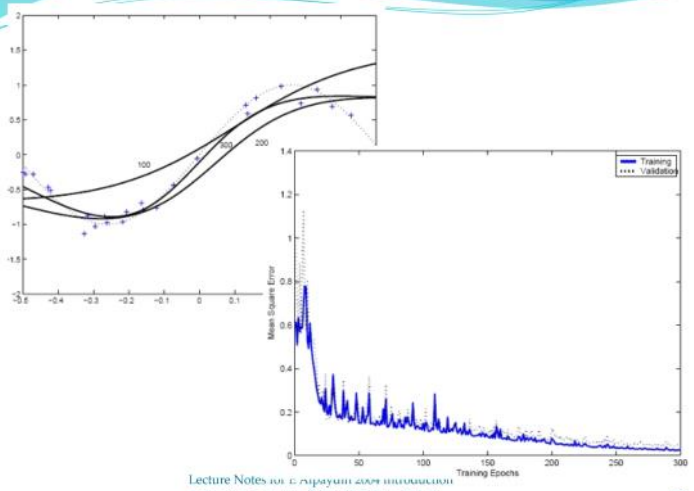
$$\Delta w_{hj}=\eta\sum_t\left[\sum_i\left(r_i^t-y_i^t\right)v_{ih}\right]z_h^t\left(1-z_h^t\right)x_j^t$$

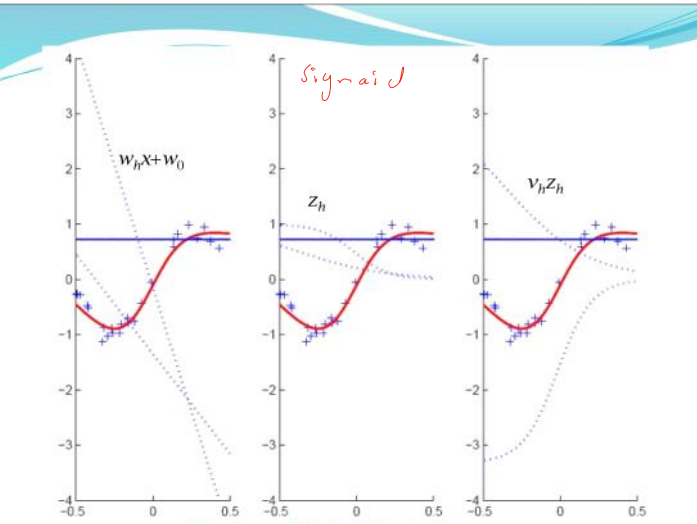


önemli

Initialize all v_{ih} and w_{hj} to rand$(-0.01, 0.01)$
Repeat
  For all $(\boldsymbol{x}^t, r^t) \in \mathcal{X}$ in random order
    For $h = 1, \ldots, H$
      $z_h \leftarrow \text{sigmoid}(\boldsymbol{w}_h^T \boldsymbol{x}^t)$
    For $i = 1, \ldots, K$
      $y_i = \boldsymbol{v}_i^T \boldsymbol{z}$
    For $i = 1, \ldots, K$
      $\Delta \boldsymbol{v}_i = \eta(r_i^t - y_i^t)\boldsymbol{z}$
    For $h = 1, \ldots, H$
      $\Delta \boldsymbol{w}_h = \eta(\sum_i (r_i^t - y_i^t) v_{ih}) z_h (1 - z_h) \boldsymbol{x}^t$
    For $i = 1, \ldots, K$
      $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \Delta \boldsymbol{v}_i$
    For $h = 1, \ldots, H$
      $\boldsymbol{w}_h \leftarrow \boldsymbol{w}_h + \Delta \boldsymbol{w}_h$
Until convergence => error rate

Apply this in this





Two-Class Discrimination

- One sigmoid output y^t for $P(C_1|\mathbf{x}^t)$ and $P(C_2|\mathbf{x}^t) \equiv 1-y^t$

$$y^t = \text{sigmoid}\left(\sum_{h=1}^{H} v_h z_h^t + v_0\right)$$

$$E(\mathbf{W}, \mathbf{v} \mid \mathcal{X}) = -\sum_t r^t \log y^t + (1 - r^t) \log (1 - y^t)$$

$$\Delta v_h = \eta \sum_t \left(r^t - y^t\right) z_h^t$$

$$\Delta w_{hj} = \eta \sum_t \left(r^t - y^t\right) v_h z_h^t \left(1 - z_h^t\right) x_j^t$$

K>2 Classes

$$o_i^t = \sum_{h=1}^{H} v_{ih} z_h^t + v_{i0} \qquad y_i^t = \frac{\exp o_i^t}{\sum_k \exp o_k^t} \equiv P(C_i \mid \mathbf{x}^t)$$

$$E(\mathbf{W}, \mathbf{v} \mid \mathcal{X}) = -\sum_t \sum_i r_i^t \log y_i^t$$

$$\Delta v_{ih} = \eta \sum_t \left(r_i^t - y_i^t\right) z_h^t$$

$$\Delta w_{hj} = \eta \sum_t \left[\sum_i \left(r_i^t - y_i^t\right) v_{ih}\right] z_h^t \left(1 - z_h^t\right) x_j^t$$

→ layer wise training

or

because of the numerical issues every layer learned individually and then combined

Multiple Hidden Layers

- MLP with one hidden layer is a universal approximator (Hornik et al., 1989), but using multiple layers may lead to simpler networks

$$z_{1h} = \text{sigmoid}\left(\mathbf{w}_{1h}^T \mathbf{x}\right) = \text{sigmoid}\left(\sum_{j=1}^{d} w_{1hj} x_j + w_{1h0}\right), h = 1, \ldots, H_1$$

$$z_{2l} = \text{sigmoid}\left(\mathbf{w}_{2l}^T \mathbf{z}_1\right) = \text{sigmoid}\left(\sum_{h=1}^{H_1} w_{2lh} z_{1h} + w_{2l0}\right), l = 1, \ldots, H_2$$

$$y = \mathbf{v}^T \mathbf{z}_2 = \sum_{l=1}^{H_2} v_l z_{2l} + v_0$$

Improving Convergence

- Momentum

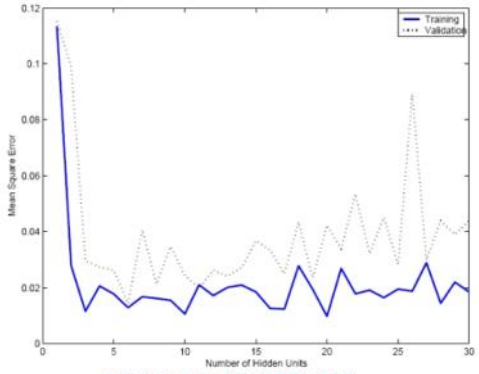
$$\Delta w_i^t = -\eta \frac{\partial E^t}{\partial w_i} + \alpha \Delta w_i^{t-1}$$

- Adaptive learning rate

$$\Delta \eta = \begin{cases} +a & \text{if } E^{t+\tau} < E^t \\ -b\eta & \text{otherwise} \end{cases}$$

Overfitting/Overtraining

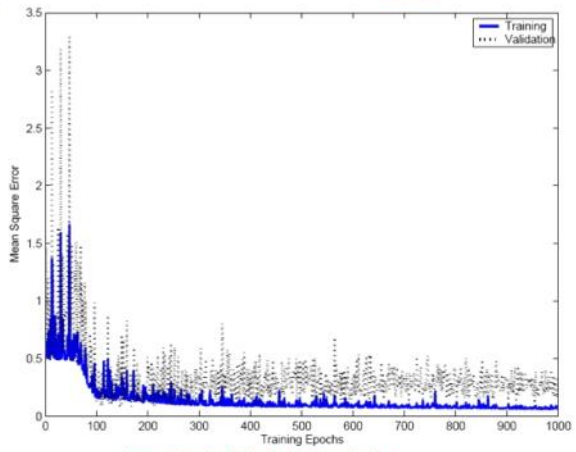
Number of weights: $H\ (d+1)+(H+1)K$



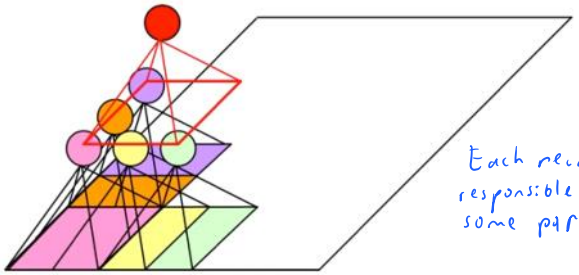
Stop early to prevent overfitting

Conclusion

- Perceptrons handle linearly separable problems
- Multilayer perceptrons handle any problem
- Logistic discrimination functions enable gradient descent-based packpropagation
  - Solves the structural credit assignment problem
  - Susceptible to local optima
  - Susceptible to overfitting



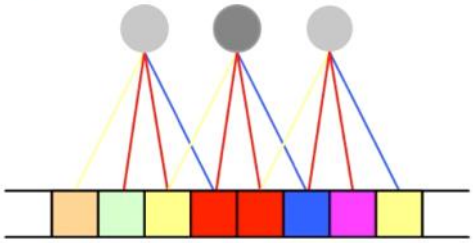
Structured MLP



Each neuron responsible for some part

(Le Cun et al, 1989)

Weight Sharing



Hints

(Abu-Mostafa, 1995)

- Invariance to translation, rotation, size

- Virtual examples
- Augmented error: $E'=E+\lambda_h E_h$

If $\boldsymbol{x}'$ and $\boldsymbol{x}$ are the "same": $E_h=[g(x|\theta)- g(x'|\theta)]^2$

Approximation hint:

$$E_h = \begin{cases} 0 & \text{if } g(x|\theta) \in [a_x, b_x] \\ (g(x|\theta)-a_x)^2 & \text{if } g(x|\theta) < a_x \\ (g(x|\theta)-b_x)^2 & \text{if } g(x|\theta) > b_x \end{cases}$$

Tuning the Network Size

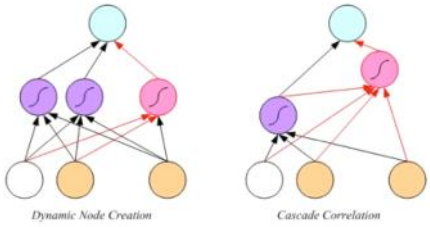
↑ If weights are small destroy it

- Destructive
- Weight decay:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} - \lambda w_i$$

$$E' = E + \frac{\lambda}{2}\sum_i w_i^2$$

- Constructive
- Growing networks



(Ash, 1989) (Fahlman and Lebiere, 1989)

Bayesian Learning

- Consider weights w_i as random vars, prior $p(w_i)$

$$p(\mathbf{w}\mid\mathcal{X}) = \frac{p(\mathcal{X}\mid\mathbf{w})p(\mathbf{w})}{p(\mathcal{X})} \quad \hat{\mathbf{w}}_{MAP} = \arg\max_{\mathbf{w}} \log p(\mathbf{w}\mid\mathcal{X})$$

$$\log p(\mathbf{w}\mid\mathcal{X}) = \log p(\mathcal{X}\mid\mathbf{w}) + \log p(\mathbf{w}) + C$$

$$p(\mathbf{w}) = \prod_i p(w_i) \text{ where } p(w_i) = c \times \exp\left[-\frac{w_i^2}{2(1/2\lambda)}\right]$$

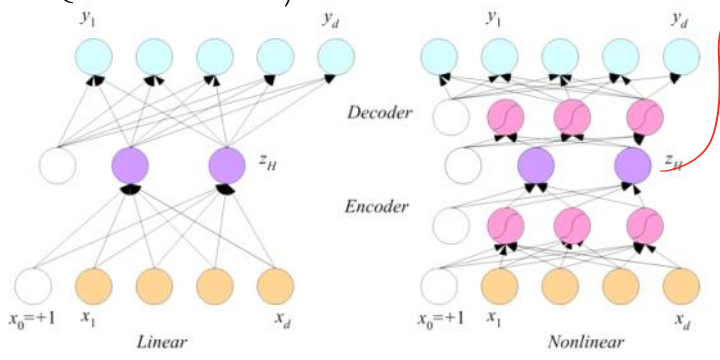
$$E' = E + \lambda\|\mathbf{w}\|^2$$

- Weight decay, ridge regression, regularization
  cost=data-misfit + λ complexity

More about Bayesian methods in chapter 14

Dimensionality Reduction

(Auto Encoder)



Learning from decoder and after, using middle layer you can feed classifiers

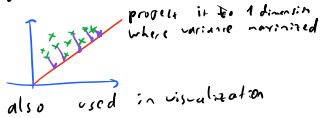
PCA — uses SVD

Principle Component Analysis

dimensional reduction

to avoid curse of dimensionality

Rule of thumb ($N \ll d$) used. This also might reduce noise.



also used in visualization

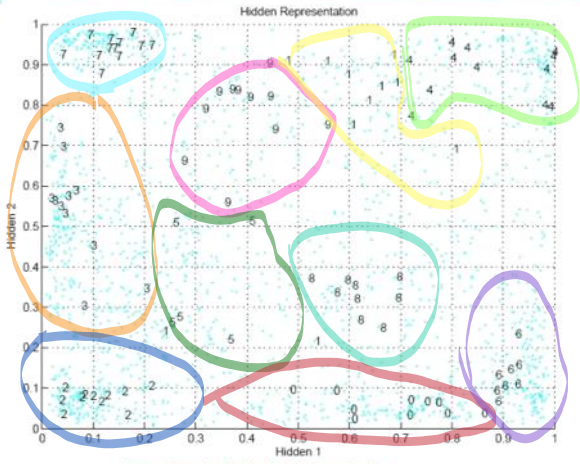
Maximize $(w^Tx - E[w^Tx])$

$X \rightarrow$ Training dataset

1. Substract mean from each instance
   ↳ creates a zero mean data
2. Then compute x^Tx (Covergence matrix (dxd))
3. Compute the eigen vectors of cov. matrix
   ↳ we have d eigen values and d eigen vectors
   ↳ Size of the eigen values will be dominant
4. Select k biggest o

eigen values / eigen vectors

↳ dimensions desired

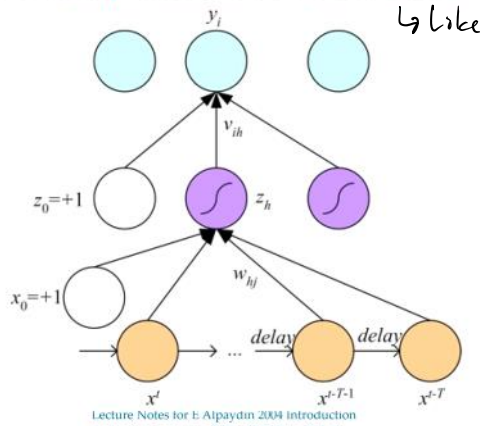


Learning Time

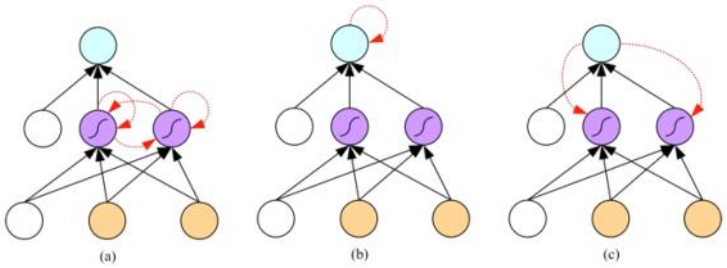
- Applications:
  - Sequence recognition: Speech recognition
  - Sequence reproduction: Time-series prediction
  - Sequence association
- Network architectures
  - Time-delay networks (Waibel et al., 1989)
  - Recurrent networks (Rumelhart et al., 1986)

Time-Delay Neural Networks

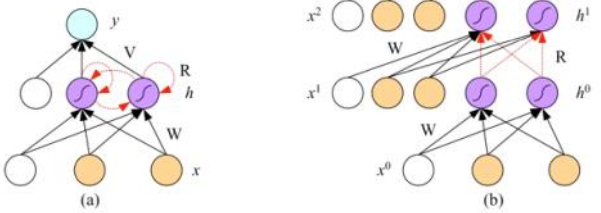
like borsa



Recurrent Networks



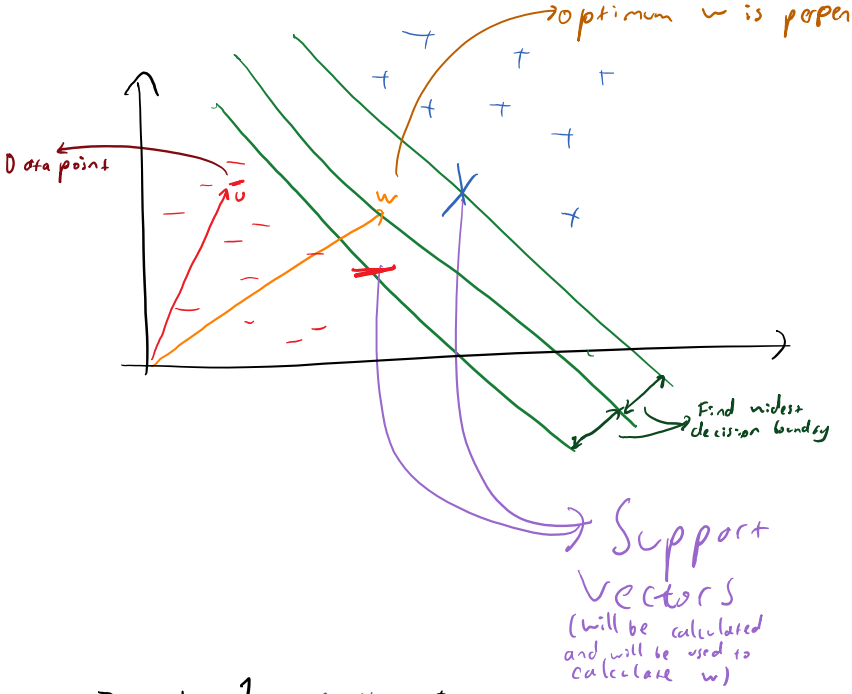
Unfolding in Time



Support Vector Machines

3 Mayıs 2017 Çarşamba 13:48

Linear model



$w^T x = 0$

$w^T u \geq c_i$ ⇒ Crosses classifier line. Its a plus

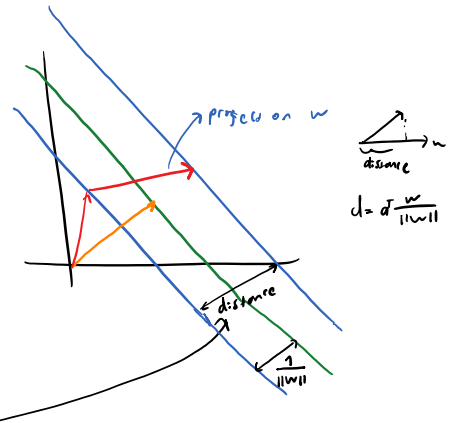
↳ $-b$

$w^T u + b \geq 0$ ⇒ optimize

$y=+1 \Rightarrow w^T x_+ + b \geq 1$

$y=-1 \Rightarrow w^T x_- + b \leq -1$

for the line ⇒ $w^T x_+ + b = 1$, $w^T x_- + b = 1$ (support vectors)



Combine two classifiers

$y_i (w^T x_i + b) \geq 1$

$y_i (w^T x_i + b) - 1 \geq 0$

$y_i (w^T x_i + b) - 1 = 0$

↳ Data points on the border (support vectors)

$width(d) = (x_+ - x_-)^T \cdot \frac{w}{\|w\|} = \frac{x_+^T w}{\|w\|} - \frac{x^T w}{\|w\|}$

$x_+^T w = \frac{(1-b)}{\|w\|} - \frac{(-b-1)}{\|w\|} = \frac{1-b+b+1}{\|w\|} = \frac{2}{\|w\|}$

minimize $\frac{1}{\|w\|}$ ⇒ maximize $\|w\| = \sqrt{w_1^2 + w_2^2 + \dots + w_n^2}$

minimize $\frac{1}{2} w^T w = (\|w\|_2)^2$

Since we have constrains (More than one function to satisfy) Lagrange Optimization will be used.

Lagrange Multipliers

Max $f(x)$ with respect to $g(x)=b$

$L(x, \lambda) = f(x) - \lambda (g(x) - b)$ $\lambda \geq 0$

$g(x) = y_i (w^T x_i + b) - 1$

$\mathcal{L} = \frac{1}{2} w^T w - \sum \alpha_i [y_i (w^T x_i + b) - 1]$ ⇒ primal form

↳ Lagrange multiplier for each data point (λ)

→ maximize

$\frac{\partial L}{\partial w} = w - \sum \alpha_i y_i x_i = 0$ $\quad w = \sum \alpha_i y_i x_i$

$L_d = \frac{1}{2} \left(\sum_i \alpha_i y_i x_i \right)^T \left(\sum_j \alpha_j y_j x_j \right) - \sum_i (\alpha_i y_i x_i)^T \left(\sum_j \alpha_j y_j x_j \right) - \sum \alpha_i y_i b + \sum \alpha_i$

($\sum \alpha_i y_i b$ → 0)

$\frac{\partial L}{\partial b} = \sum \alpha_i y_i = 0$ $\quad L_d = \sum \alpha_i - \frac{1}{2} \sum (\alpha_i y_i x_i)^T \sum \alpha_j y_j x_j$

$(x_1^T + x_2^T)(x_1 + x_2) = x_1^T x_1 + x_1^T y_2 + x_2^T y_1 + x_2^T x_1$

Quadratic

$(x_1^T + x_2^T)(x_1 + x_2) = x_1^T x_1 + x_1^T x_2 + x_2^T x_1 + x_2^T x_1$

Quadratic optimization tools

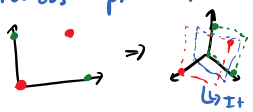
$\max_\alpha L_d = -\frac{1}{2}\sum_i \sum_\sigma \alpha_i \alpha_\sigma y_i y_\sigma \underbrace{x_i^T x_\sigma}_{z_i z_\sigma}$ => find maximum

$x_i^T x_\sigma = K(x_i, x_\sigma)$

Decision function: $w^T u + b \geq 0$ → vector to be classified

$\sum \alpha_i y_i \underbrace{x_i^T u}_{z_i^T} + b \geq 0$

Previous problem:



↳ It becomes linearly separable (3 dimension)

$z = \phi x$

$x_i^T u \Rightarrow \phi^T(x_i)\phi(u)$

$x_i^T x_\sigma \Rightarrow \phi^T(x_\sigma)\phi(x_i)$

$\phi(x_\sigma)\phi(x_i) = K(x_i, x_\sigma)$

↳ called "Kernel function"

An example ϕ function

$\phi(x) = [1, x_1 x_2, x_1^2, x_2^2]$

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ ϕ 2 to 4

ϕ defined according to mercer's conditions

$K(x,y) = (x^T y + 1)^2$ → polynomial kernel

$\phi(x) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2]$

$K(x,y) = \exp\left(-\frac{\|x-y\|}{2\sigma^2}\right)$ => Radial basis Function (RBF)

↳ parameter of the kernel

Finding α's

Coordinate Ascents

$\max w(\alpha_1, \alpha_2, \dots \alpha_m)$

Pseudo code:

Loop until convergence

for $i = 1, \dots, m$

$\alpha_i := \text{argmax}\, w(\alpha_1, \dots, \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, \dots \alpha_m)$

Sequential Minimal Optimization => tool box => libSVM

Repeat Until Converge

Select some pair α_i, α_σ to update next

Reoptimize $w(\alpha)$ with respect to α_i and α_σ

while holding all the other α_k $(k \neq i, \sigma)$

$\alpha_i + \alpha_\sigma = \zeta = \sum_{\substack{k=1 \\ k \neq i,\sigma}}^{N}$

If done right some of the α_i's are going to be 0

$\alpha_i \neq 0$ are going to be support vectors.

In the book: Kernel Machines

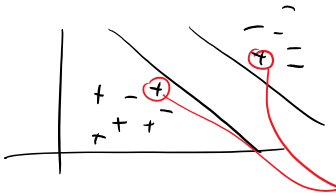
↳ Optimization kısmı çok

↳ Andrew Ing notlarında bak

↳ MIT-AT-SVM

$\sum \alpha_i y_i x_i u + b \geq 0$

If data is not linearly seperatable:



Violates boundry add constain

$L_p = \frac{1}{2} w^T w - \underbrace{\sum \alpha_i [y_i (w^T x_i + b) - 1 + \xi_i]}_{g(x)} + C\sum \xi_i - \sum \mu_i \xi_i$

$h(x)$

↳ punish violaters

$\xi_i \geq 0$ => constant

$g(x) = b$

$h(x) = c$

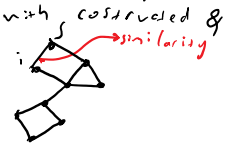
$$L(x,\lambda) = f(x) - \lambda(g(x)-b) - \mu h(x) - c$$

Hierarchical Clustering

10 Mayıs 2017 Çarşamba 13:46

K-means, Expectation maximization
↳ works well with: Mixture of gaussians
But to complicated optimization (?) - check

Spectral Clustering
Considers the similarities between data samples. It uses graph partitioning with constructed & Affinity matrix.



Hierarchical Clustering
Based on similarities and distances

L_p norm $L_2 \rightarrow$ Euclidian distance

Minkowski distance $d_n(x,y) = \left(\sum_{i=1}^{d} (x_i - y_i)^p \right)^{1/p}$

City Block (manhattan) distance $= d_{manhatten}(x,y) = \sum_{i=1}^{d} (x_i - y_i)$

Agglomarative clustering

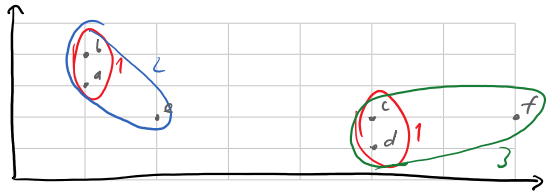
Each data sample assumed to be cluster.
Merge two closest groups at each iteration.
Iterate until there is one group.

Distance between groups:

1) Single linkage $d(G_i, G_j) = \min(x,y)$ $x \in G_i, y \in G_j$

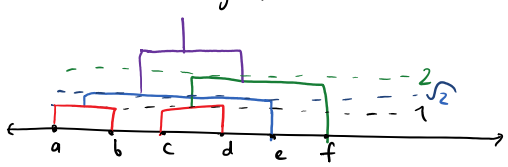
2) Complete link $d(G_i, G_j) = \max(x,y)$ $x \in G_i, y \in G_j$

3) Avaraye-link, centroid



Group Data
- Use euclodian distance
a-b: 1
c-d: 1
} combine

Dendogram



Single linkage
- Use min distance (Consider closgest data points to calculate)
a-c: $\sqrt{2}$

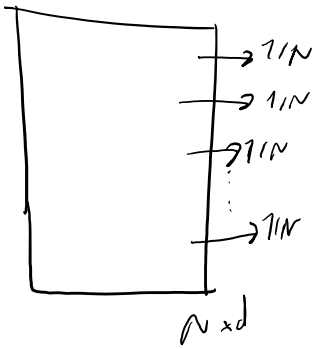
Complete linkage
- Use max distance (Consider most far data points.

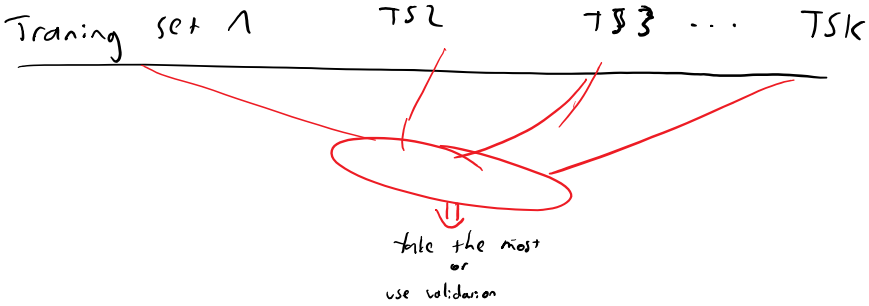
Avaraye Link, centroid
- Use Av. distance (Consider mean of cluster)

Classification Combination (Classfier Ensembles)

10 Mayıs 2017 Çarşamba 15:04

Bagging





Ella Boost

Diversity de incelenir

Unstable algoritmalar daha iyi çalışır.