Lecture 6: Bayesian learning

$P(h/D)$: probability of h given D (D is known)

Example:

D: weight = 121 Kg

h: orc or hobbit

Calculate $P(\text{hobbit} / 121\ Kg)$ vs. $P(\text{orc} / 121\ Kg)$ and select the highest

$P(D/h)$ read probability of 121 Kg given orc/hobbit

With μ and σ you can use in MatLab the function
normpdf(weight, μ, σ)
to get the probability value

$P(h)$ prior probability → knowledge without observation

how many orcs on average?

$P(D)$ probability of observation → probability of someone that gets 121 Kg

$\sum P(D/h_i) P(h_i)$ makes sure that $P(h_i/D)$ sum to 1

Bayesian probability = Posteriori probability

h_{MAP} : maximum A posteriori hypothesis → no other hypothesis is better

$$= \arg\max_{h \in H} P(h/D) = \arg\max_{h \in H} \frac{P(D/h) P(h)}{P(D)} = \arg\max_{h \in H} P(D/h) P(h)$$

↳ denominator is a constant

Bayesian, Maximum likelihood } doctors

- $h_{MAP} = \arg\max_{h \in H} P(h_i/D)$ → changes of tumor given headache
  - ↳ standard in medicine and gambling
- $h_{ML} = \arg\max_{h \in H} P(D/h_i)$ → chances of headache given tumor
  - ↳ standard in engineering

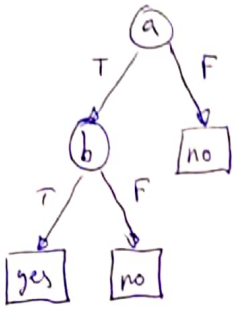
↳ types of decision

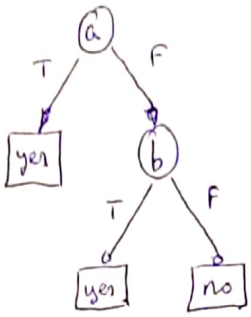
Homework 3

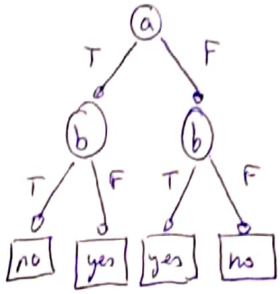
a) $A \wedge B$ (AND)

b) $A \vee B$ (OR)

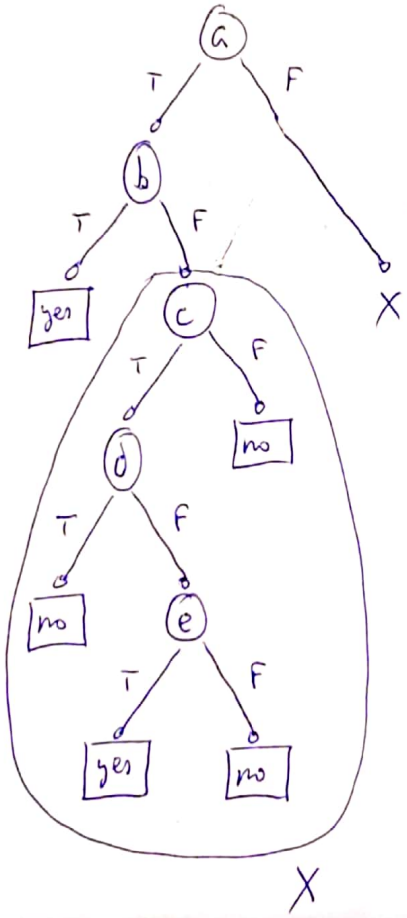
c) $A \oplus B$ (XOR)







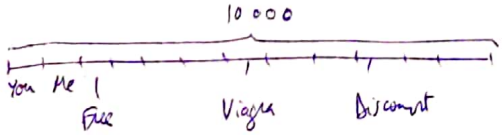
d) $(A \wedge B) \vee (C \wedge \neg D \wedge E)$



Spam classification

Representation of a text document

"bag of words"



P(Spam | <1, 0, 0, ..., 0, 0, 1>)

P(¬Spam | <1, 0, 0, ..., 0, 0, 1>)

If no examples of a specific combination both probabilities are zero

→ Naïve Bayes approach → Values closer to 1 or 0

The maximum likelihood approach gives for granted to have an enough quantity of samples

Lo in real world → sometimes it's not possible

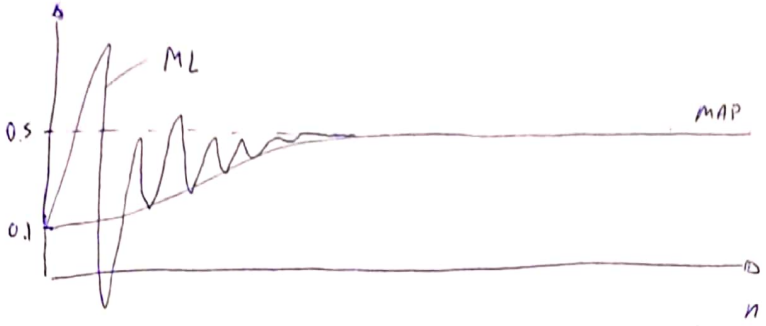
Tossing a coin as example

P(heads | n coin tosses)

$$heads\_prob_{ML} = \frac{\#\,heads}{n}$$ → with few samples, it's very inaccurate

Lo use prior information

$$heads\_prob_{MAP} = \frac{\#\,heads + m \cdot p}{n + m}$$

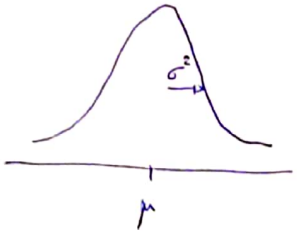


Lecture 7 : Bayesian Line Fitting (regression)

Whatever is my observation, don't take the decision only on the observation, keep on with the prior

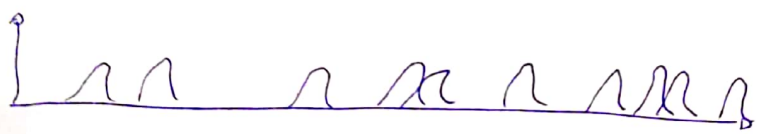
Gaussian :

$$\mu = \frac{1}{N} \sum_i x_i \qquad \sigma^2 = \frac{1}{N} \sum_i (x_i - \mu)^2$$



$$f(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi a^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

CIFAR - 10



Homework 4

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, .008 of the entire population have this cancer.

$P(\text{cancer}) = 0.008$ $\quad P(\neg\text{cancer}) = 0.992$

$P(+ \mid \text{cancer}) = 0.98$

$P(+ \mid \neg\text{cancer}) = 0.03$

h_{MAP} → $P(+ \mid \text{cancer}) \cdot P(\text{cancer}) = 0.98 \cdot 0.008 = 0.00784$ ← $P(\text{cancer} \mid +)$

→ $P(+ \mid \neg\text{cancer}) \cdot P(\neg\text{cancer}) = 0.03 \cdot 0.992 = 0.02976$ ← $P(\neg\text{cancer} \mid +)$

$h_{MAP} = \neg C$ → no cancer

Homework 5

Based on the table of positive and negative training examples for Play Tennis Compute posterior probabilities $P(\text{Play Tennis} = \text{yes} \mid \text{data})$ vs $P(\text{Play Tennis} = \text{no} \mid \text{data})$ where data is an observation vector

$S = \langle \text{Outlook} = \text{Sunny}, \text{Temp} = \text{Cool}, \text{Humidity} = \text{High}, \text{Wind} = \text{Strong} \rangle$

$P(\text{yes} \mid S) = P(\text{sunny} \mid \text{yes})\, P(\text{cool} \mid \text{yes})\, P(\text{high} \mid \text{yes})\, P(\text{strong} \mid \text{yes})\, P(\text{yes}) =$

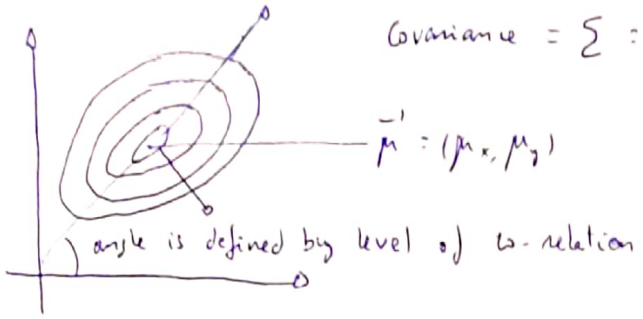
$= \frac{2}{9}\,\frac{3}{9}\,\frac{3}{9}\,\frac{3}{9}\,\frac{9}{14} = 5.3 \cdot 10^{-3}$

$P(\text{no} \mid S) = P(\text{sunny} \mid \text{no})\, P(\text{cool} \mid \text{no})\, P(\text{high} \mid \text{no})\, P(\text{strong} \mid \text{no})\, P(\text{no}) =$

$= \frac{3}{5}\,\frac{1}{5}\,\frac{4}{5}\,\frac{3}{5}\,\frac{5}{14} = 20.6 \cdot 10^{-3}$

$P(\text{no} \mid S) > P(\text{yes} \mid S) \rightarrow h_{map} = (\text{Play Tennis} = \text{no})$

2D Gaussian



$$\text{Covariance} = \Sigma = \begin{bmatrix} \text{var}(X_1, X_1) & \text{var}(X_1, X_2) \\ \text{var}(X_2, X_1) & \text{var}(X_2, X_2) \end{bmatrix}$$

$\text{var}(X_1, X_2) = \text{Var}(X_2, X_1)$

The largest eigen value of the eigen vector is the direction

↳ If one eigen value is zero

↳ one of the dimension in the data has completely disappeared

↳ you don't need to store that information at all

↳ data compression

Don't use distributions, let data speak by itself → Branch of maths

↳ Non-parametric methods

Lecture 8: Neural Networks (Black Box Methods)

Biology

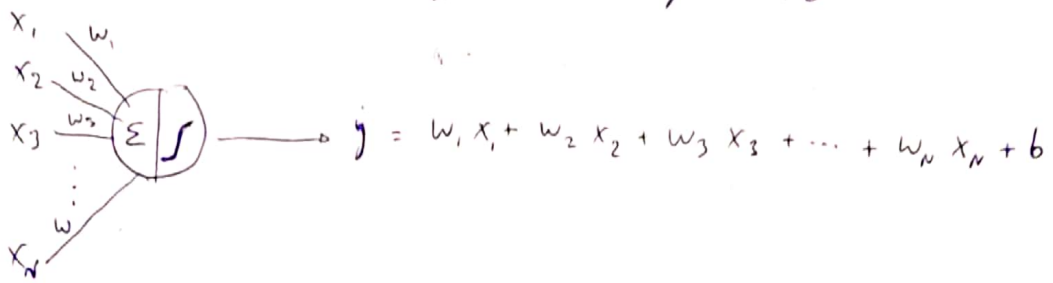
Dendrites → Inputs
Axon → Outputs
Neuron

Decision Trees → very appropiate for medical science because its transparency

Hypothesis: a neuron produces strong output when it is activated by a specific pattern (es. "I see a bicycle" → 1
"no bicycle" → 0)

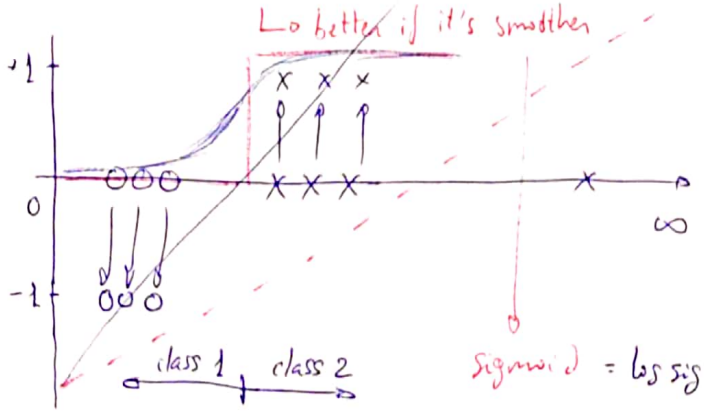
Pattern is specific to certain features ("bicycle features")

linear model: $y = ax + b$



Error: if correctly classified → error = 0
else → error = 1

Step function?
→ better if it's smoother

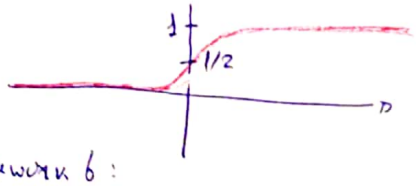


Problem with linear model for classification

→ error must NOT depend on how far on the correct/wrong side you are

Sigmoid = $\log sig(a \cdot x)$

→ ↑ a ↑ the quicker the jump is



$$\log sig(x) = \frac{1}{1+e^{-x}} \rightarrow y = \log sig\left(\sum_{i=1}^{n} w_i x_i + b_i\right)$$

Homework 6:
Define the formula for the number of free parameters in D-dimensional Gaussian distribution

$$\mu_{D\times 1} = D, \quad \Sigma_{D\times D} = \binom{D}{2} = \frac{D!}{(D-2)!\,2!} = \frac{D(D-1)(D-2)!}{(D-2)!\,2!} = \frac{D(D-1)}{2} \rightarrow \mu + \Sigma = D + \frac{D(D-1)}{2}$$

MSE $\frac{1}{N}\sum (y_i - \hat{y}_i)^2$ $\xrightarrow[\text{to be 0 or 1}]{\text{enforce}}$ $\frac{1}{N}\sum_i (y_i - \text{logsig}(\hat{y}_i))^2$

Error $\frac{1}{N}\sum_{i=1}^{n} (y_i - \text{logsig}(\sum_{j=1}^{D} w_i x_i - b))^2 = MSE$

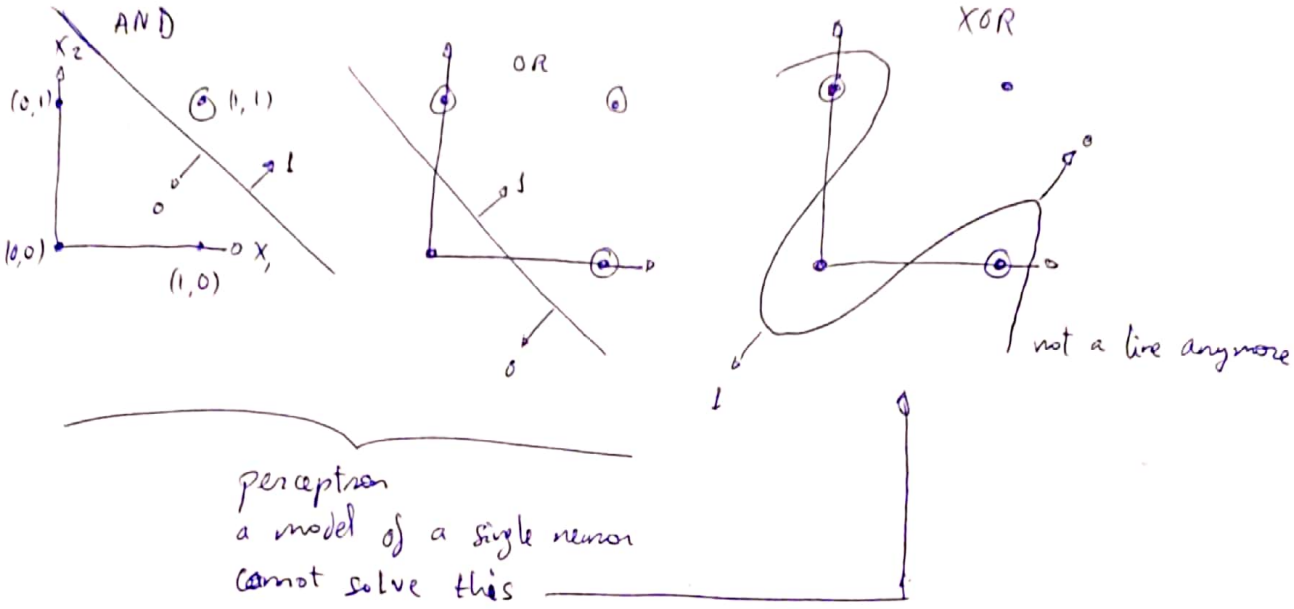
Initialize $w_i = \text{rand}(0,1)$, $b = \text{rand}(0,1)$

Then go towards the negative gradient

$$w_1^{k+1} = w_i^k - \nabla \frac{MSE}{\partial w_1} \qquad b^{k+1} = b^k - \alpha \frac{\partial MSE}{\partial b}$$

Gradient descent

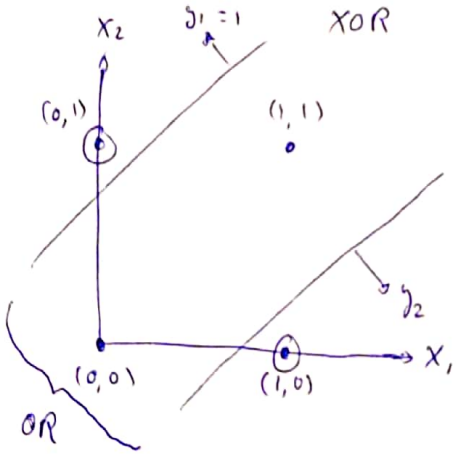
Example What can be classified ?



perceptron
a model of a single neuron
cannot solve this

↓

Multilayer perception model → MLP → invention of NN

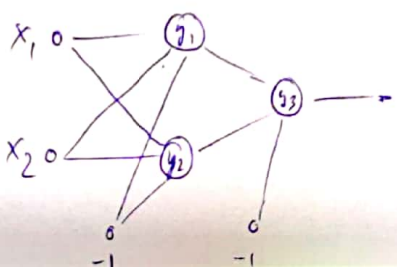


$y = \text{logsig}(w_1 x_1 + w_2 x_2 - b)$

$y_1 = \text{logsig}(-2x_1 + 2x_2 - 1)$

$y_2 = \text{logsig}(2x_1 - 2x_2 - 1)$

$y_3 = \text{logsig}(2y_1 + 2y_2 - 1)$



Lecture 9 - Reinforcement learning → how robot can learn

It started in the robotics field

learning for <u>embodied agents</u>

↳ somebody who is active in real world

Optimizing events in the future

Layers of behaviours in robots

↳ Lowest ones make sure it does not destroys itself nor others

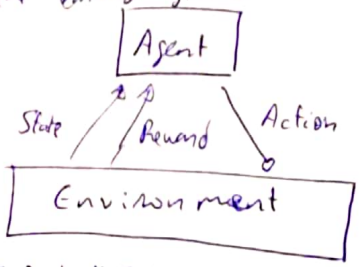
Intelligence → Multilayered perception - action patterns

Goal: as much reward as possible and ASAP

$$r_0 + \gamma r_1 + \gamma^2 r_2 \qquad 0 \le \gamma < 1$$

↳ changes the power of the learning system

$s_0 \xrightarrow[r_0]{a_0} s_1 \longrightarrow \dots$



State transition can be or not known

S	a_0	a_1	a_2	...
s_0	s_1			
s_1				
s_2				
⋮				

$s_{t+1} = \delta(s_t, a_t)$ ↳ random states

Always unknown

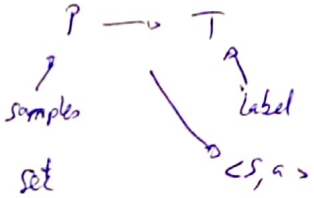
R	a_0	a_1	a_2	...
s_0	r_0			
s_1				
s_2				
⋮				

→ ~~take the maximum action~~ take the action whose reward is maximum

$r_t = r(s_t, a_t)$

Problem is to find the policy π

So far our training data ⟶

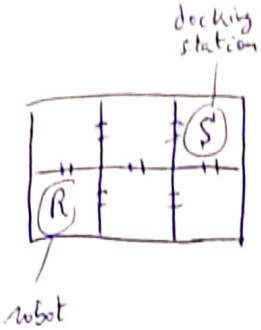


In robotics we don't have that training set

If you start a sequence of actions maybe at some point you will get the reward

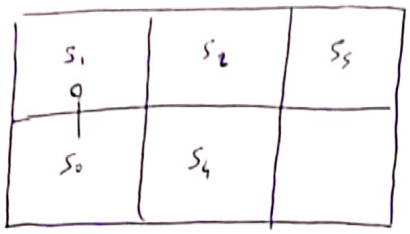
Evaluation function $V^{\pi}(s)$ for some policy π

$$\pi^*(s) = \arg\max_a \left[R(s,a) + \gamma V^* \left(\delta(s,a) \right) \right]$$



a = up, down, left, right

a_0 a_1 a_2 a_3



S	a_0	a_1	a_2	a_3	...
s_0	s_1	n/a	n/a	n/a	
s_1		s_0			
s_2				s_5	
s_3					
s_4					
s_5					

$\gamma = 0.9$

R	a_0	a_1	a_2	a_3	...
s_0	0	0	0	0	
s_1	0	0	0	0	
s_2	0	0	0	100	
s_3					
s_4	90				
s_5					

Prior knowledge makes quicker the learning progress

If there are huge amount of states and actions, finding the proper way can take ages in the context of robotics (non-simulated world)

Lecture 10 : Unsupervised learning

- So far we've been solving:

Find f so that

$f: X \to Y$

- Reinforcement learning

$f: S \to A$

Supervised learning

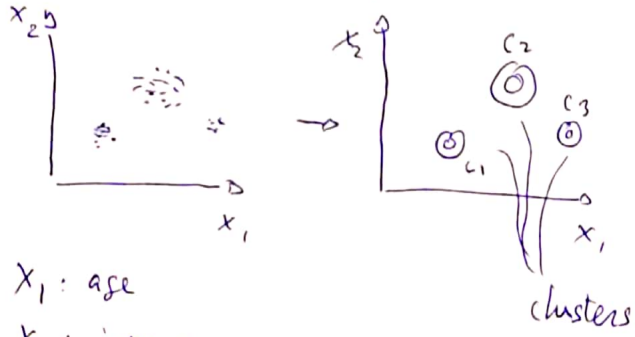
Given $\langle \bar{x}_1, y_1 \rangle, \langle \bar{x}_2, y_2 \rangle, \dots, \langle \bar{x}_N, y_N \rangle$ training set

$\bar{x}_i$ → input

y_i → ground truth → known labels

- How about if we only have $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_N$?

Example : Data $\begin{pmatrix} x_i^1 \\ x_i^2 \end{pmatrix}$ is points in 2D → Euclidean data



A phone company hires you and wants to know what are the features and types of clients who drop out their service

X_1 : age

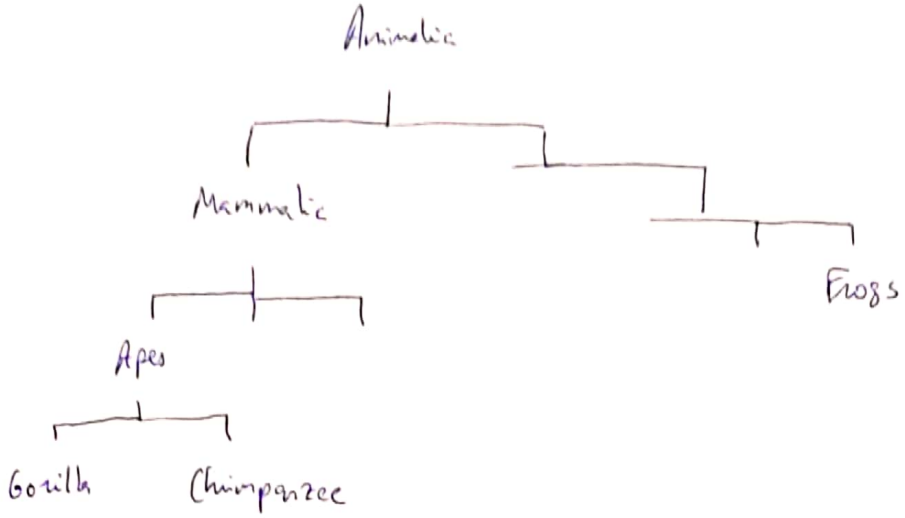
X_2 : income

C_1 : $\bar{\mu}_{C_1} = (15, 200)^T$ kids with smallish amount of money

C_3 : $\bar{\mu}_{C_3} = (78, 1500)^T$ elderly with OK income

C_2 : $\bar{\mu}_{C_2} = (30, 2000)^T$ taking variance (which is huge) into account, this is various types

Example → Hierarchical Data



Domain transfer → If I know how to classify objects, can I use this information for unknown data?

↳ Semi supervised learning

Clustering approaches → Hierarchical clustering

- Agglomerative (bottom-up)
- Divise (top-down)

↳ Every sample is its own cluster

We need:

1. Distance function
2. Lineage criteria

Distance function:
- L1: $\Sigma |a_i - b_i|$
- L2 (euclidean): $\sqrt{\Sigma (a_i - b_i)^2}$
- L_∞: $\max\{a_i - b_i\}$
- cosine: $\frac{a \cdot b}{\|a\| \|b\|}$ → best norms for nearest neighborhood classifier for tfidf data

Lineage criteria:
- Mean distance
- Distance to mean
- Min distance
- Max distance
- Median → Median Square error adequate for disperse data

Mixture model clustering — Clustering approach

Points are samples from a probability distribution

$$P(\bar{x}) = \sum_{i=1}^{k} \pi_i \, \text{prob}(\bar{x}; \theta_i)$$

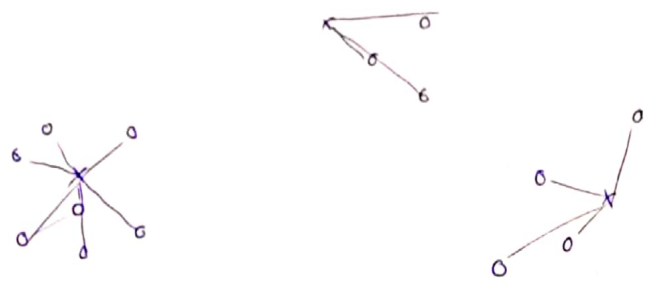
π_i: a priori

prob: basic distribution → Gaussian, $\theta_i = \bar{\mu}_i, \Sigma_i$

E.g. unsupervised Gaussian mixture model

→ needs a lot of data

Sum-of-squares method



Assum k clusters ($k=3$)
Find such $\mu_1, \ldots, \mu_k$ that distances from points in each cluster are minimized

This is brute force → every datum is compared to every cluster

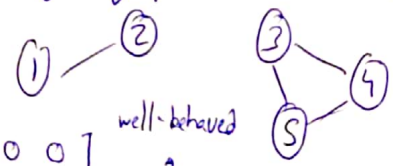
K-means (maximum likelihood solution)

1. randomly pick k points $\bar{\mu}_{1,\ldots,k}$
2. assign each point to its closest cluster center
3. calculate the mean of each cluster and update $\bar{\mu}_{1,\ldots,k}$
4. Repeat 2 and 3 until convergence
   * assignments do not change

Spectral clustering

Our data is of graph/network type

Example:



well-behaved

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Positive semidefinite matrix

$L = D - A \Rightarrow D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$ → N connections in the diagonal

$\rightarrow L = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & -1 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix}$ ← Graph laplacian

Eigen values
Eigen vectors

Many properties:

If any eigenvalues are zero, there is this amount of unconnected graphs and their eigenvectors reveal their nodes

$\lambda_4 = \lambda_5 = 0$ → the corresponding eigenvectors of those must be

① and ② and ③, ④ and ⑤

① II

Instead of finding clusters, can we somehow visualize high dimensional data?

Self-organizing map (SOM)

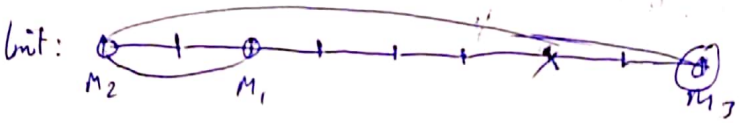
1. Repeat M times
2. Select a sample $\bar{x}$
3. Find the closest SOM cell $\bar{m}_j$ for $\bar{x}_i$ ($\bar{m}_{BMU}$)
4. Make $\bar{m}_{BMU}$ (best matching unit) more similar to $\bar{x}$
5. Make neighbours of $\bar{m}_{BMU}$ more similar
6. End repeat
7. Return $\bar{m}_j$

$$\bar{m}_{BMU} = \underset{\bar{m}_j}{\operatorname{argmin}} \|\bar{x}_i - \bar{m}_j\|^2$$

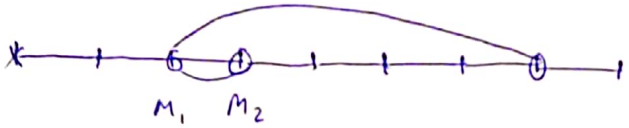
$$\bar{m}_j^{(l+1)} = \bar{m}_j^{(l)} + \alpha\,(\bar{x}_i - \bar{m}_j^{(l-1)})$$

example

$\alpha = 0.5$



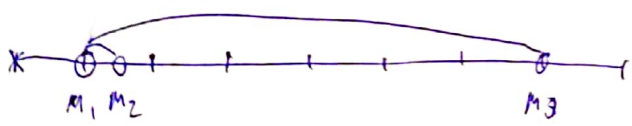
input $x_1 = 6$



input $x_2 = 0$

step $\Delta m_1 = \alpha(x - m_1) = 0.5(0 - 2) = -1$

step $m_2 = 0.5(0 - 3) = -1.5$



Lecture 13: Learning sets of rules

P	Q	P → Q
T	T	T
T	F	F → not ok
F	T	T
F	F	T

Think: "If rains then drive carefully"

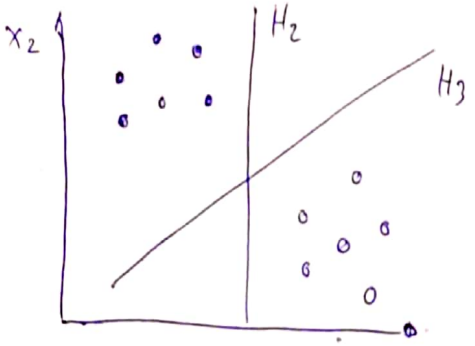
equivalent form: $P \rightarrow Q \quad \equiv \quad \neg P \vee Q$

learning first-order rules

search-based, but the search-space explodes so no feasible solutions

ProLog → mother (x, y)
father (x, y)

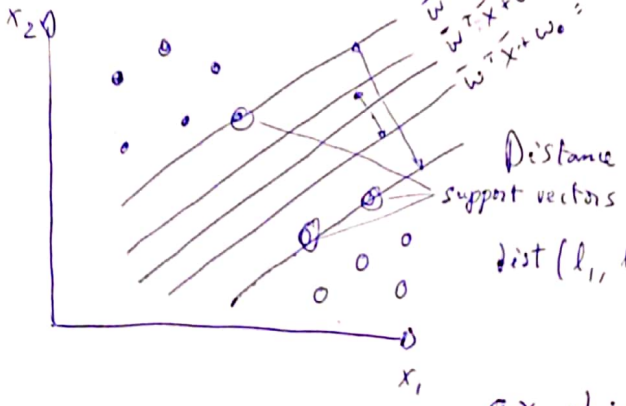
Lecture 9: Support Vector Machines



$MSE(H_2) = MSE(H_3) = 0$

For MLP they are equally good solutions

For all correctly classified points: $y_i(\bar{w}^T \bar{x} + w_0) > 0$



Goal: Maximize the distance of the two parallel lines (of the decision boundary $\bar{w}^T\bar{x} + w_0 = 0$)

↳ push c as far as possible

Distance of two parallel lines:

$$dist(l_1, l_2) = \frac{|c_2 - c_1|}{\sqrt{a^2 + b^2}} = \frac{2c}{\sqrt{\bar{w}^T\bar{w}}} = \frac{2}{\sqrt{\bar{w}^T\bar{w}}}$$

$$a x_1 + b x_2 + c = 0 \rightarrow \frac{a}{c} x_1 + \frac{b}{c} x_2 + 1 = 0 .$$

Problem: maximize $\frac{2}{\sqrt{\bar{w}^T\bar{w}}}$ $\quad \bar{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$

↳ minimize $\sqrt{\bar{w}^T\bar{w}} \rightarrow \min_{\bar{w}} \bar{w}^T\bar{w}$ → is 0

subj. to:

$y_1(\bar{w}, \bar{x}_1 + w_0) \geq 1 - \varepsilon_1$
$y_2(\bar{w}, \bar{x}_2 + w_0) \geq 1 - \varepsilon_2$
$\vdots$
$y_m(\bar{w}, \bar{x}_0 + w_0) \geq 1 - \varepsilon_m$

Homework 7: Give weights w_1, w_2 and w_b so that a perceptron model establishes: x_1 or x_2, x_1 and x_2, $\neg x_1$

x_1 and x_2 : $w_1 = w_2 = 0.5$, $w_b = -0.5$

x_1 or x_2 : $w_1 = w_2 = 0.5$, $w_b = 0.5$

not x_1 : $w_1 = -0.5$, $w_2 = 0$, $w_b = 0.3$

$$y = w_1 x_1 + w_2 x_2 + w_b$$

$$\text{ssn}(y) = \begin{cases} 1, & y > 0 \\ -1, & y < 0 \end{cases}$$

Optimization

1. linear optimization → e.g. simplex

max $w_1 x_1 + w_2 x_2 \left(- C \Sigma \varepsilon_i \right)$ ← C is restrictive is with points in the penalty for misclassifications wrong side

subj $x_1 + x_2 \leq B$ ← SVM addition

$x_1 \geq 0$, $x_2 \geq 0$

2. Unconstraint optimization

max/min $f(x)$

e.g. method gradient descent

3. Constrained problems

min $f(x)$

subject to $h_i(x) = 0$

$g_i(x) \leq 0$ ⇒ SVM

4. Discrete optimization

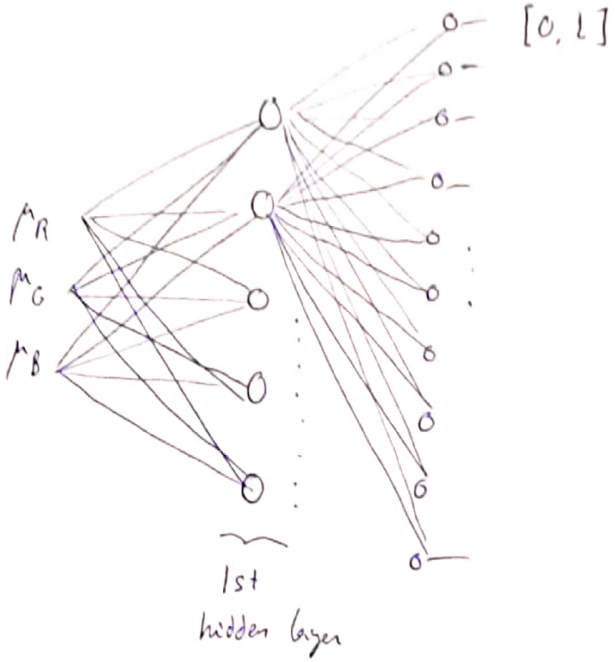
e.g. travelling salesman, ...

MLP

samples_train
labels_train
} learn a Multilayer perceptron network

samples_test
labels_test

↳ need to define # neurons of hidden layer



When the network is trained, labels_train need to be reshaped

GUI → see patternet in wiki matlab

```
>> nnstart
>> est_labels_test = sim(net, samples_test);
```

vec2ind([0 0 1 0 0]) → 3, see also ind2vec

patternet([10 10])

2 hidden layers
with 10 neurons each

take a subset of 10% of the training set while training
↳ when everything looks OK → the whole set

	1	2	3	4	(5)	6
1	0	4	13	24	12	8
2		0	10	22	11	10
(3)			0	7	(3)	9
4				0	6	18
5					0	8.5
6						0

→

	1	2	3-5	4	6
1	0	(4)	12	24	8
2		0	10	22	10
3-5			0	6	8.5
4				0	18
6					0

→

	1-2	3-5	4	6
1-2	0	10	22	8
3-5		0	(6)	8.5
4			0	18
6				0

→

	1-2	3-4-5	6
1-2	0	10	(8)
3-4-5		0	8.5
6			0

→

	1-2-6	3-4-5
1-2-6	0	(8)
3-4-5		0

→

	1-2-3-4-5-6
1-2-3-4-5-6	0

Decision Tree

- a, and b

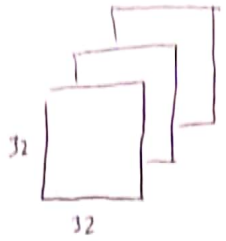


image x

too big → f : cifar_10_features

↳ mean of every channel

= 3072 dimension

f_1 = mean(x(:,:,1))

f_2 = ... ↳ 1 : end

f_b = ...

f = [f_1, f_2, f_b]

Bayesian classifier

c = cifar_10_bayes_classify (f, mu, sigma, P)

sample

[mu, sigmas, P] = cifar_10_bayes_learn (f_train, label_train)

mu: 10 × (3) = sigmas (squared); P: 10 × 1

f_train: 50000 × (3); label_train: 50000 × (1)

- P ↳ how many ones divided by total # labels
- sigmas ↳ " twos
- mu ↳ " threes
- var(x)
- function mean(x)

mu = mean(x)

$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(2)} & x_2^{(2)} \\ \vdots & \\ x_1^{(N)} & x_2^{(N)} \end{bmatrix}$$

- for class = 1 ... 10
  - for example = 1 ... length(f)
    - posteriori(class) = normpdf(f, mu(class, 1), sigma(class, 1)) * normpdf(...)

↳ Naive assumption → 3 single variable gaussian function

↳ Correlated assumption → 1 " " "

var → σ

sigma → Σ ↳ cov(X)

now instead of normpdf(...) → mvnpdf(), cov()

↳ matrix

Divide the image → 4 features → training images

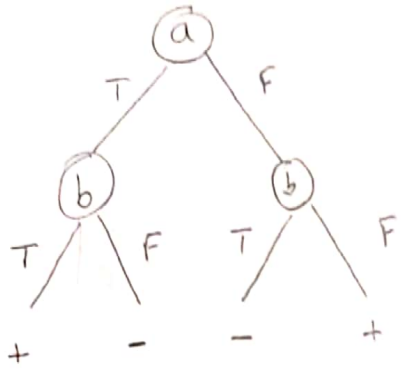
if it works → subdivide

if you keep subdividing → original image

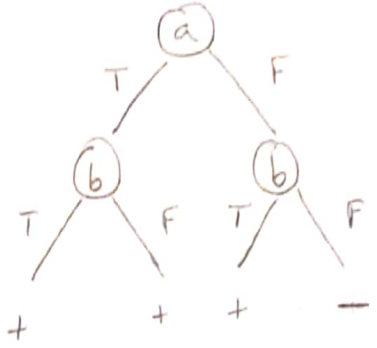
$$\Sigma_{3\times3} = \begin{bmatrix} \sigma_{RR}^2 & \sigma_{RG}^2 & \sigma_{RB}^2 \\ \sigma_{GR}^2 & \sigma_{GG}^2 & \sigma_{GB}^2 \\ \sigma_{BR}^2 & \sigma_{BG}^2 & \sigma_{BB}^2 \end{bmatrix}$$

Decision Tree

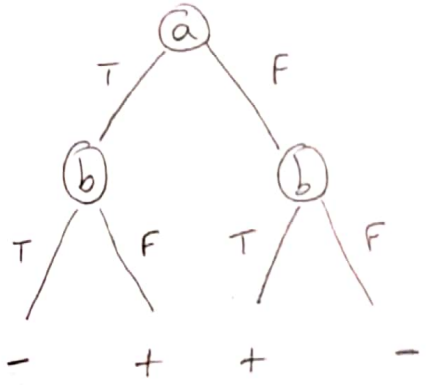
- a and b



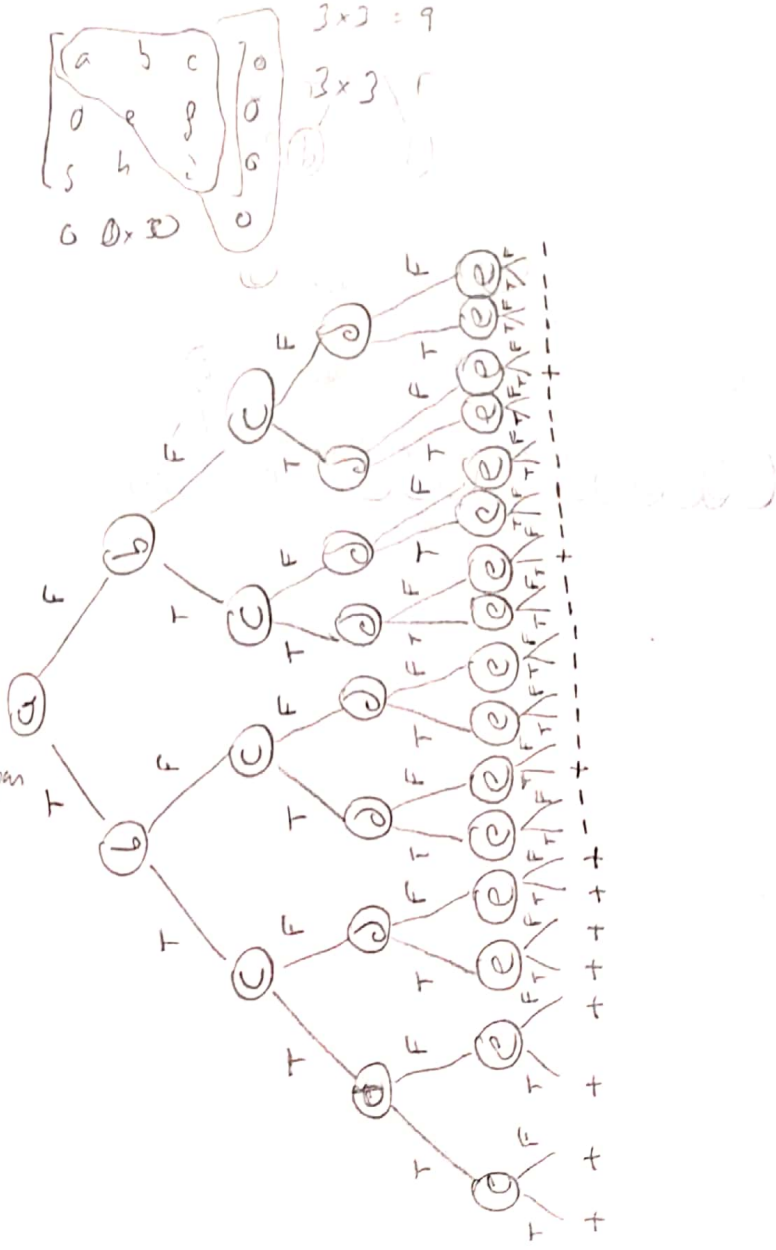
- a or b



- a xor b



- (a and b) or (c and not d and e)



formula # of free parameters in D-dimensional gaussian distro

$\vec{\mu}_{D\times 1} \rightarrow D$ par: $D + \frac{D^2+D}{2}$

$\Sigma_{D\times D} = \begin{bmatrix} \sigma & \sigma \\ \sigma & \sigma \end{bmatrix}$? $\left(D + \frac{D^2+1}{2}\right)$: irpar

$D + \frac{D^2}{2} \rightarrow D + \frac{1}{2}D$

$U = (D^2/2 - D)$

for every 3 parameters
↳ you need at least 3 examples