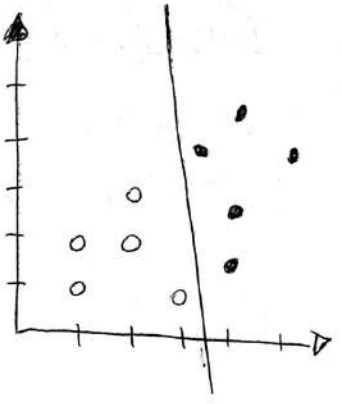
12/02/2018 (OK)

Ex 1

In Supervised learning we have dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$
x_n is input and t_n is value of target function associated to that particular input.
In supervised learn we have classification and regression the main difference between them is that for learning function $f: X \to Y$, in classification $Y = \{C_1, \dots, C_K\}$ and in regression $Y \subseteq R$

In unsupervised learning we have dataset $D = \{(x_n)\}$ contains only information about input, value of target function is not available.
the aim is to define or find a structure of our input using some application seech as clustering.

Ex 2



Perceptron method want to classify instance with this gradient to update weight that we use for change our decision boundary in each iteration when we find a point that is classified in a bad way

In particular we have funcke $o(x) = \begin{cases} 1 & \text{IF } w^T x > 0 \\ -1 & \text{otherwise} \end{cases}$

$\theta = w^T x$

Given a dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$

we want to minimize error funcke

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (t_n - \theta_n) = \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T x_n)^2$$

So we need to compute derivative respect to w

$$\frac{\partial E}{\partial w} = -\sum_{n=1}^{N} (t_n - w^T x_n) x_{i,n} = \nabla E$$

After this we need to update weights

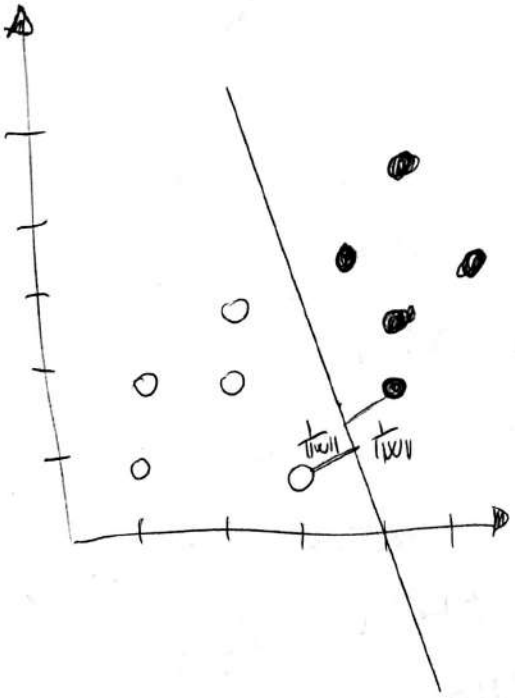
$$w \leftarrow w + \Delta w_i$$

$$\Delta w_i = -\eta \nabla E = \eta \sum_{n=1}^{N} (t_n - w^T x_n)$$

η is learning rate

Perceptron start with w choosing in a random way and then update in each iteration update weights UNTIL this method converge, so we don't need to update weight anymore.

Method converge IF Dataset is linearly separable and learning rate is small, otherwise if dataset is not linearly separable perceptron will not converge



SVM we wants to maximize the margin to provide a better accuracy.

Margin is smallest distance between closest points of decision boundary and the decision boundary itself We compute w^* and w_0^* using lagrange multipliers At the end this method converge if dataset is linearly seperable otherwise we need to introduce slack variable to avoid problem of overfitting and try to converge else if there are some points that are missclassified

I prefer SVM because if dataset is almost linearly seperable I can use slack variable to MAXIMIZE this margine While softly penalty points that are misclassified

Ex 3|

1)

IF $C = c_1 \wedge B = b_1$ THEN NO

IF $C = c_1 \wedge B = b_2$ THEN YES

IF $C = c_2 \wedge A = a_1$ THEN YES

IF $C = c_2 \wedge A = a_2 \wedge B = b_1$ THEN YES

IF $C = c_2 \wedge A = a_2 \wedge B = b_2$ THEN NO

IF $C = c_2 \wedge A = a_3$ THEN NO

IF $C = c_3$ THE NO

2)

$S = \{ S_1 = \langle a_1, b_1, c_1, NO \rangle , S_2 = \langle a_2, b_1, c_2, YES \rangle$

$S_3 = \langle a_1, b_2, c_3, NO \rangle , S_4 = \langle a_3, b_2, c_1, YES \rangle \}$

For S_1

- for the first rule we can say that T is consistent because $C = c_1 \wedge B = b_1$ THEN NO such as S_1

For S_2

IF $A_2 = a_2 \wedge B = b_1 \wedge C \in C_2 \Rightarrow$ YES, is true with four rule so also for this set T is consistent

For S_3

IF $C \in C_3 \neq$ THEN NO

Also for this set T is consistent

or S_4

T is constistent for this set of samples because IF $C = C_1 \wedge B = b_2$ We obtain YES such as the second rule

EX B4

1)

Boosting is an approch in we We consider to train differt learnerst instead of just one and then collect their result, but in particular with Boosting we train differt models in a sequential way this means that each classifier is trained on weighted data and that weights depends on performance of the previous classifier.
IF classifier give some error this means that we obtain weights with higher values.
At the end predictions are based on weighted majority of votes.

2)

AdamBoost can be explored else with sequential minimization of exponential error function

Consider error

$$E = \sum_{n=1}^{N} \exp\left(-t_n f_M(x_n)\right)$$

Theorem

$$f_M(x_n) = \frac{1}{2} \sum_{m=1}^{M} \alpha_m Y_m(x_n) \quad t_n \in \{-1, +1\}$$

We want to minimize

Error function E respect to $\alpha_m Y_m(x)$

$m = 1, \dots, M$

Instead of minimize error globally we assume

$Y_1(x), \dots, Y_{M-1}(x)$

$\alpha_1, \dots, \alpha_M$ fixed

Keep Y_M and α_M explicit we obtain

$$E = \sum_{n=1}^{N} \exp\left(-t_n f_{M-1}(x_n) - \frac{1}{2} t_n \alpha_M Y_M(x_n)\right)$$

$$= \sum_{n=1}^{N} w_n^{(M)} \exp\left[-\frac{1}{2} t_n \alpha_M Y_M(x_n)\right]$$

$$w_n^{(M)} = \exp\left(-t_n f_{M-1}(x_n)\right)$$

We know that from sequential optimisation of E

$$\alpha_m^{(m+1)} = \alpha_n^{(m)} \exp\left(\alpha_m I(Y_m(x) \neq t_n)\right)$$

$$\operatorname{sign}(f_M(x)) = \operatorname{sign}\left(\frac{1}{2} \sum_{m=1}^{M} \alpha_m Y_m(x)\right)$$

equivalent $Y_M(x) = \operatorname{sign} \sum_{m=1}^{M} \alpha_m Y_m(x)$

(5)

1) $f^{(2)}(f^{(1)}(x;\theta^{(1)}); \theta_2^{(2)})$

2) activation function For input could be relu because we need to predict a scalar $x \in R^3$ and For the activation function of the output we can have linear function because we don't work with binary classes or multiclass so we just need linear function

3) Since we consider linear function as an activation function of the output we consider mean squared error as a loss function for training the Network

$$J(\theta) = \frac{1}{2} \sum_{n=1}^{N} (t_n - f(x,\theta))^2$$

№6

1)

$$a = (K + \lambda I)^{-1} \cdot t$$

Where $K = XX^T = \begin{pmatrix} K(x_1, x_1) & \cdots & K(x_1, x_N) \\ \vdots & & \vdots \\ K(x_N, x_1) & \cdots & K(x_N, x_N) \end{pmatrix}$

this is Gram matrix

2) For Kernel trick we can say that if we find input vector x in form of inner product $x^T x$ we can replace this product with Kernal function

In case of ~~linear~~ regression we can say that

$$y(x) = \sum_{n=1}^{N} a_i K(x_i, x) \quad \text{with } a = (K + \lambda I)^{-1} t$$

19/01/2018

Ex 1

Ok

1) IMAGE 1242 x 378 x 3

- First Layer we have conv1, relu1 and pool1

After conv1 we obtain

$$W_{out_1} = \frac{W_{IN} - K + 2 \cdot P}{S} + 1 = \frac{1242 - 5 + 4}{1} + 1 = 1242$$

$$h_{out_1} = \frac{W_{IN} - K + 2 \cdot P}{S} + 1 = 378$$

the output after conv1 is 1242 x 378 x (64) features

- After RELU dimension of outputs doesn't change and so we pass through pooling 1

the out put after pooling should be $W'_{out} = \frac{W_{out_1} - Pol_{size}}{Stride} + 1$

$\Rightarrow W'_{out} = 621 \quad h'_{out} = 189$

So After First layer we obtain

621 x 189 x 64

- Second Layer

We repeate same thing but in this case conv2 has a kernel 3x3

$W_{out_2} = 310 \quad h_{out_2} = 94 \Rightarrow 310 \times 94 \times 128$

And after pool2 we have

78 x 24 x 128

2)

N° parameter in ~~[illegible]~~ conv layer would be

$((m * n) + 1) \cdot$ #features

dimension of kernel

So For conv 1 we have 1664 parameters

ReLU and ~~[illegible]~~ Pooling don't change number of parameter

N° of conv 2 are 1280

In table we obtain 2944 parameter for this CNN

EX 2

1) The dimensionality of data space is the dimension of images that in this case is 12x12 because we have this matrix in input and For ~~[illegible]~~ the intrinsic dimensionality we have 3 because it corrispond of 2D ~~[illegible]~~ translation and 1 rotation, so we have 3 degrees of Freedom

2) We ~~[illegible]~~ basis ~~[illegible]~~ are sorted by decreasing eigenvalues, ~~[illegible]~~ x_i can be transform in to the new space using projection but first we must ~~[illegible]~~ to compute mean of $x_1, \dots, x_6$ so $\bar{x} = \frac{1}{6} \sum_{n=1}^{6} x_n$

then $[\mu_1 \bar{x}_1, \dots, \mu_4 x_6]$, we maximize variance and obtain

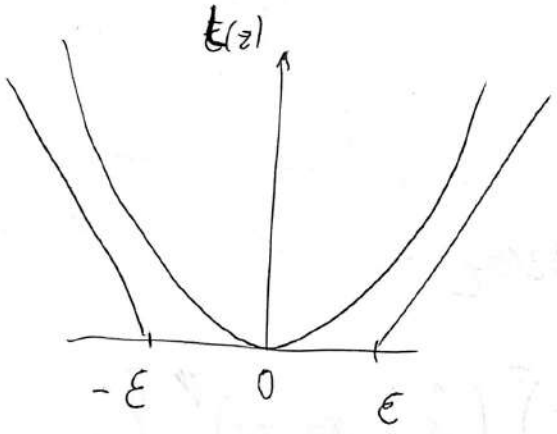
$\mu_i^T S \mu_i = \lambda_i$

When S is varianza and to have maximum varianza we consider to take μ_i eigen vector corrispondig to the largest eigen value λ_i
this called principal components

3) M, generally, diffent of 3 because principals components of PCA are NOT latent variables

Ex3

1)



the problem of minimizing T is that is not differenziable e it is difficult to solve

2) this slack variable ξ^+ and ξ^- have role to overcom of problem of not differenziable, in particular with them we can compute a quadratic program that can be easily to solve and we don't have any more the problem of not differenziable

Ex 4)

Least Squares is a linear classification method that aiths the aim to move prediction as close as possible to the target function

given a dataset $D = \{(x_n, t_n)\}$

$y = \tilde{w}^T \tilde{x}$ $\qquad \tilde{w}_K$

$\tilde{w} = (\tilde{w}_1, \tilde{w}_K, \tilde{w}_K)$

we want to compute $\tilde{w}$

we conde

$$\tilde{X} = \begin{pmatrix} \tilde{x}_1 \\ \vdots \\ \tilde{x}_N \end{pmatrix} \qquad t = \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix}$$

We want to MINIMIZE error

$$E(\tilde{w}) = \frac{1}{2} Tr \left\{ (\tilde{X}\tilde{W} - T)^T (\tilde{X}\tilde{W} - T) \right\}$$

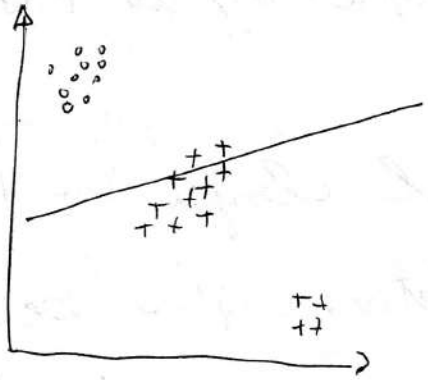
IN a closed form we obtain

$$\tilde{w} = (\tilde{X}\tilde{X}^T)^{-1} \tilde{X}^T \cdot T = \underbrace{(X^+)}_{\text{pseudo inverse}} \cdot T$$

$$y = \tilde{W}^T \tilde{x} = T^T (\tilde{X}^+)^T \cdot \tilde{x}$$

This method have problem with outliers because we minimize $E(w)$ that is a sum so each

[redacted] contribute to compute this error and in particular with outliers we can have a decision boundary that cannot divide perfectly [redacted] instances that have different classification value but we could have distance that [redacted] are [redacted] NOT in the right side



The decision boundary is NOT good because we have outliers that create problem so this method is not robust to outliers.

EX5

1) MAXIMUM A posteriori hypothesis we find the most probable hypothesis

$$h_{MAP} = \arg\max_{h \in H} \frac{P(D|h)P(h)}{P(D)} = \arg\max_{h \in H} P(D|h)P(h)$$

IF $P(h_i) = P(h_j)$

$$h_{ML} = \arg\max_{h \in H} P(D|h)$$

2) Bayes Optmal classfer $f : X \to V = \{v_1, \dots, v_K\}$

$$V_{BO} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | x, D) = \underset{v_j \in V}{\operatorname{argmax}} \sum_{h_i \in H} P(v_j | x, h_i) P(h_i | D)$$

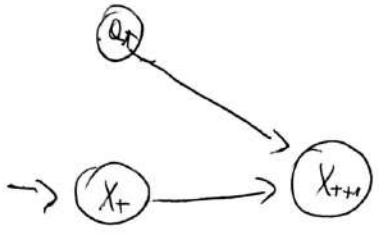
Bayes Optmal classfer is used to compute most probable classfication For new instance x, se what we use prior probabilty of h_i to compute this most probble classfication, we Find the optmal value

3) the problm of Bayes Optmal classfer is that is NOT pratcal for large Hypothesis spac because we cannot know all prior probabilties of all hypothesis, we selve this probls using Naive Bayes classfer that use conditonal independence to estimate most probble classfie, se it make approximations.

EX6

Markov property said that current state contain all information to know and it doesn't depend of precedent history.

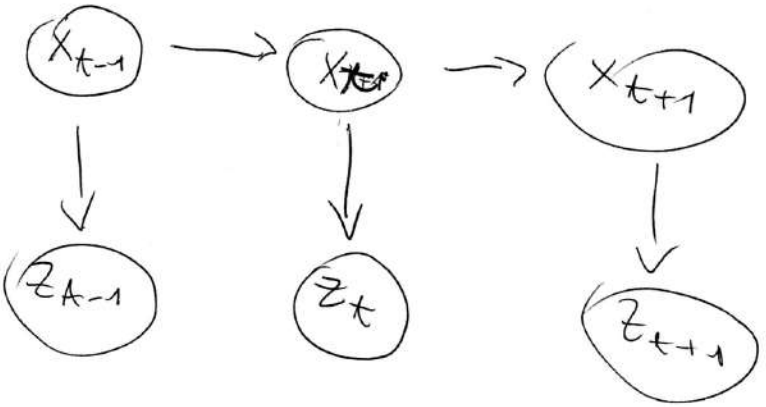
MDP composed by $\langle X, A, \delta, r \rangle$ where X is set of state A is set of actions, δ is transition function, r is reward function. States are fully observable so we don't need observation



Next state depend only of current state, are they are indipendent of past states and past observation given the current state

HMM is composed by $\langle X, Z, \Pi \rangle$ where X is set of states, Z is set of observation and Π_0 is initial distribution.

In this case state are not-observable so we need observation but also in this case we have always markow property



23/03/2018

Ex 1

1) Given an Hypostesis spee H, we can say that $h \in H$ overfits training that if choosen on $h' \in H$ we have that

$$error_S(h) < error_S(h)$$

se

$$error_D(h) > error_D(h')$$

2)

In the decision tree we can have problems of overfitting because when we have a dataset that contains noisy data, or when training dataset is too small to produce representative sample of target function but we can decide to have two decision Tree T and T' obtained by different configuration of ID3 with a data set that contains more data and we notice that

$$accuracy(T') > accuracy(T)$$

We cannot say that T' is better than T because certainly it has a problem with overfitting

En Decison tree to avoid overfitting we can stop to grow the tree, but it is very difficult to estimate so we can use method of post-pruning

Post pruning consist to prune the tree after growing full tree.

There are two different method: Reduced-Error-Pruning and Rule Post-Pruning

Reduce error pruning we split our dataset IN training and validation set

then until pruning is harmful we evaluate the impact on validation set of pruning each possible node so then we greedely remove the node that most improves the validation accuracy

This Pruning can give us only optimal subset of tree and for a small train set it can give us bad result

Rule Post pruning is a pruning method that infere the tree allowing for overfitting

Then we convert decison tree in set of rules

is because in this way we can distinguish different context, infact each path of Tree has different rule so in this way we can prune paths indipendently. At the end, after pruning each path indipendently of each other, we get final rules.

EX2

1) $f: X \to V = \{v_1, \dots, v_k\}$ $\quad X = \{a_1, \dots, a_m\} \notin D$, dataset

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \; P(v_j \mid D) \prod_i P(a_i \mid v_j, D)$$

Naive Bayes classifier is used for Hypothesis space that are very big so Bayes optimal classifier cannot know all prior probability of hypothesis h_i.

For this reason we use Naive Bayes classifier that use independence condition to do an approximation

For new instance x

x is described of values of attribute

$x = \{a_1, \dots, a_n\}$ and so there are no more hypothesis

What we want to find is the most probable classification and we do it using this approximation

In particular

$$P(v_j \mid x, D) = P(v_j \mid a_1, \dots, a_n, D)$$

$$= P(v_j \cdot D)\,(P(a_1, \dots, a_n \mid v_j D))$$

$$\text{arg max } P(v_j, x, D)$$

We make assumption that

$$P(a_1, \dots, a_n \mid v_j, D) = \prod_i^n P(a_i \mid v_j, D)$$

$$\Rightarrow v_{NB} = \underset{v_j \in V}{\text{arg max}}\; P(v_j \mid D) \prod_i P(a_i \mid v_j, D)$$

2)

3 categories of papers

We can use Naive Bayes with Multinomial distribution in which we can say that we count occurrence of w_i in documents

$f: Doc \rightarrow C = \{c_1, \dots, c_k\}$ Dataset $= \langle d_i, c_i \rangle$

New document d_i length(d_i)

$$P(d_i \mid c_j, D) = \prod_{i=1}^{} P(a_i = w_k \mid c_j, D)$$

We make this conditional independence assumption

$$P(a_i = w_k \mid c_j, D) = P(a_m = w_k \mid v_j, D) \rightarrow \forall i, m$$

$$\Rightarrow P(w_k \mid c_j, D)$$

or multinomial distribut

$P(\alpha_i | c_j, D) = \text{Multinomial}(\alpha, n, \theta)$

$\hat{P}(w_i | c_j, D) = \dfrac{TF_{ij} + 2}{TF_j + |V|}$ = approximate formula

where TF_{ij} is total number of times that word w_i occurs in subset of documents of D for which target value is c_j

TF_j is total number of word IN subset of documents of D for which target value is c_j

Ex 3

1) In Unsupervised learning problem we have a dataset $D = \{x_n\}$ where we have only information about inputs and the target value is not available We want to find the structures of our inputs using some applications, methods such as clustering

2) For example we have followers on twitter but we don't know about outputs so we can analyse salary, or N° of comment we don't know a prior we find with some methos such as clustering that through alot find natural cluster if they exist, Also compresion such as dimensionality reduct with PCA

3) We have K followers of twitter, $\{x_n, y_n\}_{n=1}^{N}$ value K can consider K-means se we want to compute K mean of data $\mu_1, \ldots, \mu_k$ generate from K gaussian mixture distribution.

① We consider that value K is N° of clusters

② We put any INITIAL partition that classifies the data INTO K clusters:
We take first K training samples as a single-element clusters. Then we assign each of remaining training samples N-K to a cluster with nearest centroid and we update this centroid after each assignment.

③ Take each sample IN sequence and we check if

is in the cluster with nearest centroid. IF sample is not in the cluster with nearest centroid, this sample change it's cluster with cluster with closest centroid. After this we recompute centroids of two clusters involved.

④ We repeat ③ until there is a convergence.

there is a convergence where there is no more sample to switch.

We obtain a circular shape for our input IF IT converge and it converge IF there are FINITE number of partition of K cluster and IF distance from sample to group of samples of centroid decreased.

EX 4

1) KNN is classification algorithm for non parametric model

We have Dataset D, target function $f: X \to C$ and K

the algorithm is compose by two steps for new instance $x \notin D$

① Choose K nearest neighbors of instance x

② Assign to x the most common value among K nearest neighbors

Likelihood Function

$$P(c|x, D, K) = \frac{1}{K} \sum_{i \in N_K(x,D)} I(x_i = c)$$

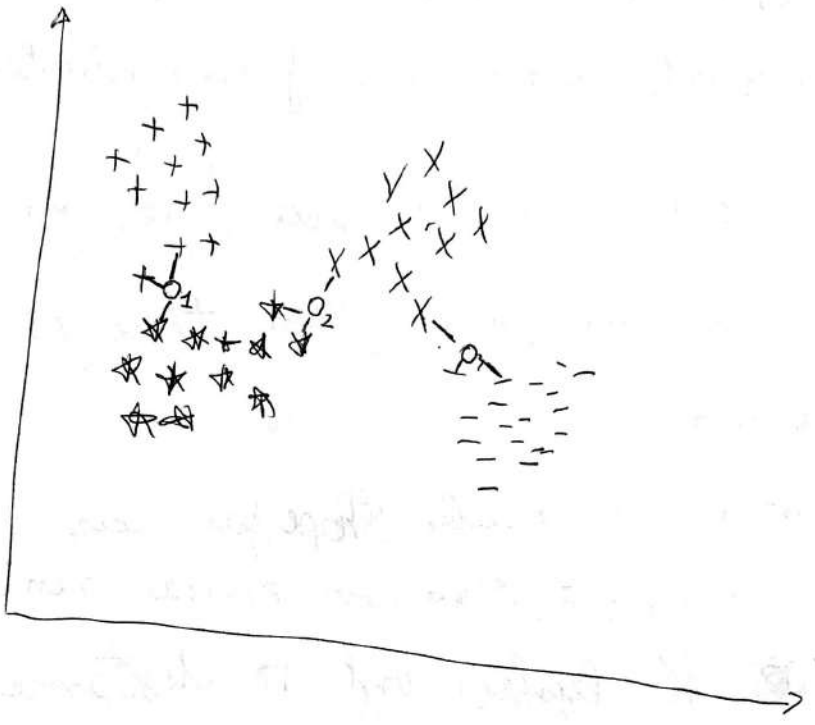
Where $N_k(x, D)$ are Nearest Neighbors and $I(e) = \{1$ if e is true, 0 otherwise$\}$

2)

K=3

4 classes

- O_1 has 2 + and 1 * so it is classified as "+"
- O_2 has 2 * and 1 X so it is classified as "*"
- O_3 has 2 - and 1 X so it is classified as "-"



EX 5

1) Backpropagation is algorithm used only to compute gradient ~~of cost~~ function $\nabla_\theta J(\theta)$

In particular the approach is bottom-up, we start from the output layer and we propagate at each level the gradient

Stochastic Gradient Descent is a learning algorithm that we use to improve the learning using the

gradient, in particular with it we update parameters at each step.

Forward and backward pass: Forward pass compute value from inputs to output and backward pass we perform the backpropagation that start from output and go to the inputs applies chain rule to compute the gradient

2)

Forward pass

Require: l depth of network
Require: x inputs
Require: t target value
Require: W^i matrices of weights $i = 1, \dots, l$
Require: $b^{(i)}$ bias $i = 1, \dots, l$

$h^0 = x$

for $k = 1, \dots, l$

$a^k = W^k \cdot h^{k-1} + b^k$

$h^k = f(a^k)$

end For

$y = h^{(l)}$

$J = L(t, y)$

Backward pass

$g \leftarrow \nabla_y J = \nabla_y L(t, y)$

for $k = l \dots 1$

Propagate gradients TO pre-non linearity activations

$g \leftarrow \nabla_{a^{(k)}} J = g \odot f'(a^k)$

$\nabla_{b^{(k)}} J = g$

$\nabla_{W^k} J = g (h^{(k-1)})^T$

$g \leftarrow \nabla_{h^{(k-1)}} J = W^{(k)} \cdot g$

end for

EX6

1) Linear regression we have learning function $f: X \to Y$ where $Y \subseteq R$, we want to find Relationship between inputs and output

In particular in linear regression we have

$y(x; w) = w_0 + w_1 x_1 + \dots w_d x_d = w^T x$

$w = \begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix}$ $x = \begin{pmatrix} 1 \\ \vdots \\ x_d \end{pmatrix}$ w are weights, x inputs

2)

We can consider linear regression with non function of input variable

$$y(x, w) = w^T \phi(x) \qquad w = \begin{pmatrix} w_0 \\ \vdots \\ w_{M-1} \end{pmatrix} \qquad \phi(x) = \begin{pmatrix} \phi_0(x) \\ \vdots \\ \phi_{M-1}(x) \end{pmatrix}$$

$$\phi_0(x) = 1$$

We want to maximize loss function

$$\min E_D(w) = \min \left(\frac{1}{2} \sum_{n=1}^{N} (t_n - w^T \phi(x))^2 \right)$$

this means

$$E_D(w) = \frac{1}{2} (T - W\phi)^T (T - W\phi)$$

$$t = \begin{pmatrix} t_1 \\ \vdots \\ t_N \end{pmatrix} \qquad \phi = \begin{pmatrix} \phi_0(x_0) & \cdots & \phi_{M-1}(x_1) \\ \vdots & & \\ \phi_0(x_N & & \phi_{M-1}(x_N) \end{pmatrix}$$

$$W_{ML} = (\phi^T \phi)^{-1} \phi^T t = \phi^+ t$$

For sequential learning we use it when the dataset is very huge so we use stochastic gradient to compute sequential learning

$$w \leftarrow w + \eta \nabla E$$

$$\hat{W} \leftarrow W + \eta \sum_{n=1}^{N} \left(t_n - W^T \phi(x_n)\right) \cdot \phi(x_n)$$

that converge for suitable value of learning rate η

13/12/2017

EX-1

1) F = Furniture
NR = Nr Rooms
NK = New Kitchen

$f: F \times NR \times NK \rightarrow \{Yes, No\}$

with $F = \{Yes, No\}$, $NR = \{3, 4\}$
$NK = \{Yes, No\}$

2)

Variable are chosen using highest value of information gain where information gain is

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

that is expected reduction of entropy

$$ENTropy(S) = -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

$p_\oplus$ portion of positive samples

$p_\ominus$ portion of negative samples

3) $S_A = [3+, 2-]$ means 3 positive 2 negative for acceptable

$S_F = \{Yes, No\}$ $\quad$ $Entropy(S_A) = 0.969$

$S_{FY} = [1+; 1-]$ $\quad$ $Entropy(S_{FY}) = 1$

$S_{FN} = [2+ \ 1-]$ $\quad$ $Entropy(S_{FN}) = 0.918$

$Gain(S, F) = 0.969 - \frac{2}{5} \cdot 1 - \frac{3}{5} 0.918 = 0.0182$

$S_{NR3} = [1+, 2-]$ $\quad$ $Entropy(S_{NR3}) = 0.918$

$S_{NR4} = [2+; 0]$ $\quad$ $Entropy(S_{NR4}) = 0$

$Gain(S, NR) = 0.969 - \frac{3}{5} 0.918 = 0.4182$

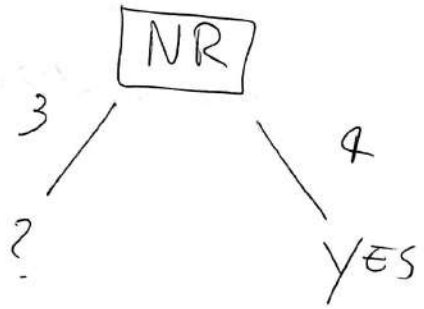
$S_{NKY} = [1; 0]$ $\quad$ $Entropy(S_{NKY}) = 0$

$S_{NKN} = [2+; 2-]$ $\quad$ $Entropy(S_{NKN}) = 1$

$Gain(S, NK) = 0.969 - \frac{4}{5} = 0.169$

the highest gain we have with Attribute NR

So the root is NR



Compute Gain of other in that condition So when

$NR = 3 \quad S_3 = [1+ \; 2-] \quad ENTropy(S_3) = 0.918$

$S_{FY} = [0; \; 1-] \quad ENTropy(S_{FY}) = 0$

$S_{FN} = [1+, \; 1-] \quad ENTropy(S_{FN}) = 1$

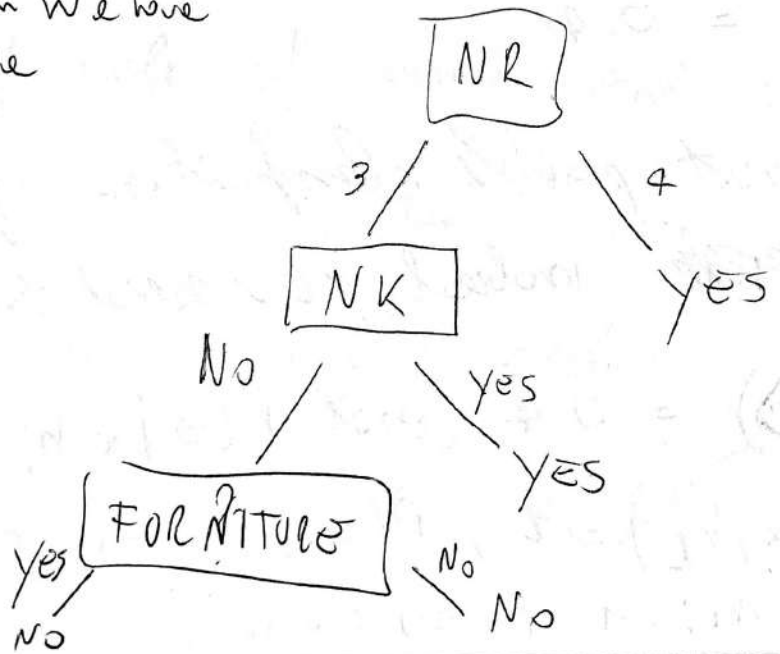
$Gain(S, F) = 0.918 - \frac{2}{3} = 0.251$

$S_{NKY} = [1+; \; 0] \quad ENTropy(S_{NKY}) = 0$

$S_{NKN} = [0; \; 2-] \quad ENTropy(S_{NKN}) = 0$

$Gain(S, NK) = 0.918$

We choose NK
so then We have
Furniture



Ex 2|

1) Consider hypothesis space H with $h \in H$ given a data D

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

We compute the most probable hypothesis IF $P(h_i) = P(h_j)$ with MAXIMUM likelihood

$$h_{ML} = \underset{h \in H}{\arg\max}\ P(h|D) = \underset{h \in H}{\arg\max}\ P(D|h)$$

We drop $P(D)$ and $P(h)$ because are constant

2) No, because IF we consider 3 hypothesis

$h_1 = \oplus$ $h_3 = \ominus$

$h_2 = \ominus$

$P(h_1|D) = 0.4$ $P(h_2|D) = 0.3$ $P(h_3|D) = 0.3$

with MAXIMUM likelihood IF we consider x a new instance we assign to it $\oplus$ because

$$h_{ML} = \underset{h \in H}{\arg\max}\ P(h|D) = 0.4$$

but this is NOT most probable classification of new instance Indeed you need to consider $P(\oplus|x, h_i) = 0.4$ and $P(\ominus|x, h_i) = 0.6$

IF we consider $P(\oplus|x, h_1) = 1$, $P(\oplus|x, h_2) = 0$, $P(\oplus|x, h_3) = 0$

$P(\ominus|x, h_1) = 0$, $P(\ominus|x, h_2) = 1$, $P(\ominus|x, h_3) = 1$

we can say a instance x should be classified as
and NOT as ⊕ as we can see with MAXIMUM CLEARANCE

Ex 3

1) We suppose to use a polynomial form

$$K(x', x) = (\beta x^T x + \gamma)^d \quad \text{where } d \Rightarrow \{?, \ldots\}$$

In this case we can use a polynomial of degree 3 so $d = 3$

2)

We can use slack variable $\xi_n \geq 0$ when dataset IS NOT perfectly separable and IN particular for dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$

We should be under soft margin constraint

In which $t_n y(x_n) \geq 1 - \xi_n$

IF $\xi_n \geq 1$ means that this point IS IN a wrong side of border and IN the margin

IF $0 \leq \xi_n < 1$ point IS INside the margine but in the right side.

EX4

1) $t = f(x, \theta)$

We consider to use as activation function

For output a sigmoid function because we consider binary classification problem with which we have that it assume value 1 or 0 or I have in output a result of sigmoid function ~~Function~~ ~~we~~ ~~with binary or multiclass in this problem~~

2) For this case we choose ~~binary~~ sigmoid outputs so We ~~don't~~ have problem of saturation ~~case~~ only when we give the correct ~~those outputs don't saturate~~ answer

the cost function that we can use is ~~Squared error loss function~~ binary - cross entropy

EX5

1) $W_{out} = \frac{W_{IN} - 3 + 2P}{S} + 1$

$H_{out} = \frac{H_{IN} - 3 + 2P}{S} + 1$

$r = W_{IN}$

$$W_{IN} = \frac{W_{IN} - 3 + 2P}{S} + 1 \qquad h_{IN} = \frac{h_{IN} - 3 + 2P}{S} + 1$$

$$S W_{IN} = W_{IN} - 3 + 2P + S \qquad S h_{IN} = h_{IN} - 3 + 2P + S$$

$$\boxed{S = \frac{W_{IN} - 3 + 2P}{W_{IN} - 1}}$$

$$2P = S(h_{IN} - 1) + 3 - h_{IN}$$

$$P = \frac{S(h_{IN} - 1) + 3 - h_{IN}}{2}$$

$$P = \frac{\left(\frac{W_{IN} - 3}{W_{IN} - 1}\right)(h_{IN} - 1) + 3 - h_{IN} + \frac{2P(h_{IN} - 1)}{W_{IN} - 1}}{2} = \frac{\left(\frac{W_{IN} - 3 + 2P}{(W_{IN} - 1)}\right)(h_{IN} - 1) + 3 - h_{IN}}{2}$$

$$P = \frac{\frac{(W_{IN} - 3)}{(W_{IN} - 1)}(h_{IN} - 1) + 3 - h_{IN}}{2 \cdot \left(1 - \frac{2(W_{IN} - 1)}{2(W_{IN} - 1)}\right)} = \frac{\frac{W_{IN} h_{IN} - 3h_{IN} + 3 - h_{IN}}{W_{IN} - 1}}{\frac{2(W_{IN} - 1) - 4(h_{IN} - 1)}{(W_{IN} - 1)}}$$

$$= \frac{W_{IN} h_{IN} - 3h_{IN} + 3 - 3W_{IN} + 3 + W_{IN} h_{IN}}{2(W_{IN} - 1) - 4(h_{IN} - 1)} = \frac{-3(W_{IN} + h_{IN})}{2W_{IN} - 2 - 4h_{IN} + 4}$$

$$= \boxed{\frac{-3(W_{IN} + h_{IN})}{+2(W_{IN} - 2h_{IN} + 1)}}$$

$$\boxed{S = \frac{W_{IN} - 3}{W_{IN} - 1} + \frac{-3(W_{IN} + h_{IN})}{(W_{IN} - 1)(W_{IN} - 2h_{IN} + 1)}}$$

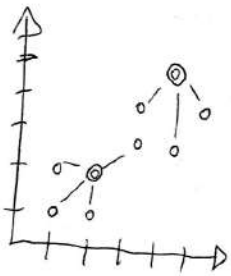
2)

1) Hyperbolic tangent
2) RELU
3) Sigmoid

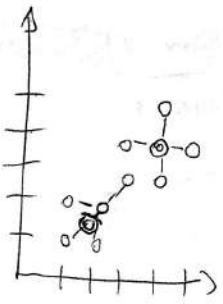
Ex 6

With K-Mean we compute each sample in sequence and we check if that sample is the cluster with nearest centroid

- K=2

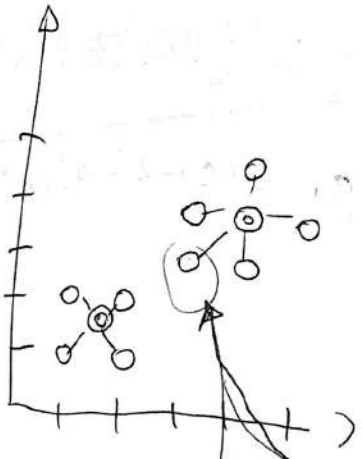


We show that ◎ are centroid In the next step, we need to update our centroid



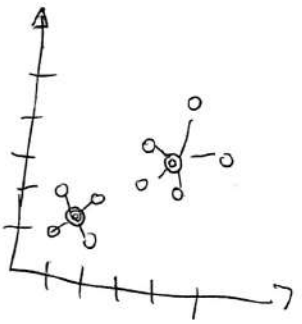
We update centroid

So now we need to check if there is a sample that should be more from one to the other cluster



Now we assign this sample to the other cluster on top because it is cluster with nearest centroid

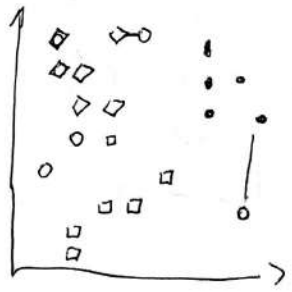
Code to update centroid of two clusters



We can say that there are no more points to switch between two cluster, so this K-means converge

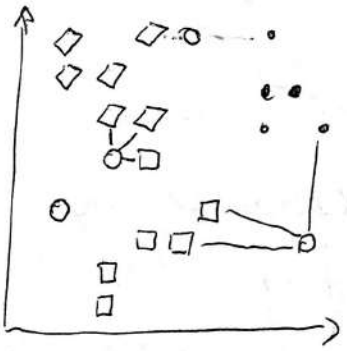
EXAM TEST

EX1



■ K=1

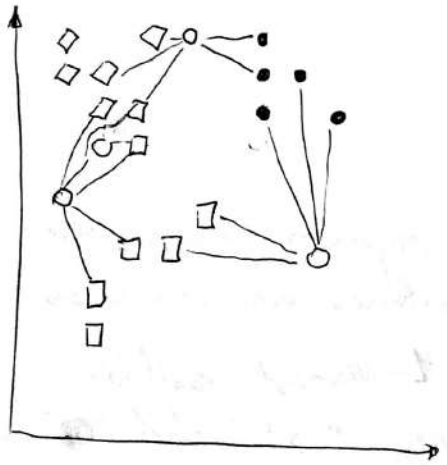
We choose 1-Nearest neighbors to our new instances two of this 4 New instances we can classify because We take 1 1-Nearest neighbor, in particular we assign "◇" and "●" for two instances that we collected in the picture, The other two instances we cannot assign 1-nearest neighbors because we have more than 1 neighbor that are in the same distance respect to new instance, we cannot assign a classified value



■ K = 3

In this case We can classify two of four new instances, one with value "□" because it is the most common value among 3-nearest neighbors and also for the other instance in picture we assign value "□"

For other two instance We have some problem than in the previous example there is problem that new instance is in the point that two o more nearest neighbor are near to him at the same distance and you can choose bet of them but you don't know on which



K=5 we can compute values only on 3 of 4 new instances

In particular we can see that for 1 instance we compute value "◇" another one we have 5 □ so this value should be "□" and then the last one has 3 • and 2 □ so we compute to it "●"

Only one instance cannot have a value because when we need to choose the last nearest neighbor there are two neighbors at the same distance, indeed it should not be a problem for classifier because we have already 3 ◇ and 1 □ so it should be "◇" but we don't know how to choose the last one nearest neighbors

---

Ex2

CNN is convolutional ~~neural network~~ layer composed by 3 stage: convolutional, detector and pooling

In convolutional stage we apply convolutional function

$$(I * K)(i, j) = \sum_{m} \sum_{n} I(i+m, j+n)\, K(m, n)$$

We can also apply some techniques such as sparse connectivity in which we consider that output depend of fewer inputs, we select a subset of input ~~neurons~~ that affect the output so in this way we can make less operations and increase the ~~#~~ efficiency

We use another technique : parameter sharing. this consist that all units of that layer share parameter with each other and with this we don't need to learn too much parameters for this layers and we can re-use them many times.

the last one thing in convolutional stage is the padding that it should be valid, so output contains only valid values, or same, so input layer is padded with zeros.

In the Detector stage we only decide which activation function we want to use for that layer, for example Relu or hyperbolic tangent.

the last one stage is pooling stage where we modify the output, in particular pooling function replace output of the net at certain location with a summary statistic of the nearby outputs

We have two different pooling : max pooling and average pooling Max pooling we consider maximum value of neighborhood regions, instead in average pooling we consider average value of neighborhood region.

IF we apply stride we reduce the size of output.

EX 3)

Using PCA we reduce dimensionality of our dataset consisting of many parameters correlated with each other while retaining the variation present in the dataset

We can do it + MAXIMIZE variance or MINIMIZE error.

IF we maximize variance we need to consider a dataset $D = \{x_n\}_{n=1}^{N}$ and our goal is to maximize variance after projection in some direction μ_1

Compute mean and variance

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n \quad \Rightarrow \text{projected mean } \mu_1^T \bar{x}$$

$$\frac{1}{N}\sum_{n=1}^{N}(\mu_1^T x_n - \mu_1^T \bar{x})^2 = \mu_1^T S \mu_1 \quad \text{projected variance where } S \text{ is variance}$$

Maximize $\mu_1^T S \mu_1$

We use lagrance so we obtain

$$\max \mu_1^T S \mu_1 + \lambda_1(1 - \mu_1^T \mu)$$

We set derivative to zero

$$S\mu_1 = \lambda_1 \mu_1$$

$$\mu_1^T S \mu_1 = \lambda_1$$

We can say that μ_1 is eigenvector and λ_1 is eigenvalue

To maximise variance we need to find eigenvector M_1 corrispondig to the largest eigenvalue λ_1, this is principle Components

EX4

1) In MDP we consider to have

$$\langle X, A, \delta, r \rangle$$

Where X is set of state, ~~[redacted]~~ in this case we have only 1 state, the initial state

- A is set of actions and in this case there are 3 possible paths
- δ transition function se the probability that actor a in state s at certain time t will lead state at time $t+1$
- r is reward function, in this case reward ~~[redacted]~~ that we want to MAXIMIZE

2) We don't know reward Function se

What we need to use is the Q-learning FUNCTION to find optimal hypothesis that maximize reward summary

time

training rule is

$$\hat{Q}_{\text{new}}(x, a) \leftarrow \bar{r} + \gamma \max_{a'} \left(\hat{Q}(x', a') \right)$$

where r is the immediate reward one x' is the state resulting from apply action a' in state x

3)

Exploitation we use to select best action that maximize $\hat{Q}(x, a)$ and exploration we select a random acte a with low value of $\hat{Q}(x, a)$

We prefer to use ε-greedy stated

We select random action with probability ε and then select best action with probability $1-\varepsilon$

When ε decrease over time

EXS

1) Confusion matrix is a performance measured used when we want to summary result of multi classes, in particular in the diagonal we have accuracy of each class and out of the diagonal we have the error of classification

2) $f: \{0,1\}^4 \rightarrow \{Low, Medium, High\}$

150 samples, 70 for low, 30 for medium, 50 for high

	Low	Medium	High	
Low	35	15	25	40
Medium	5	20	5	10
High	5	10	35	15
	10	25	30	90

65 (sum of the diagonal)

3) Accuracy of classification matrix is given by

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Where TP = true positive, TN = true negative, FP = false positive and FN = false negative

true positive are sample that are classified well for that class and true negative are value of the diagonal in which is not classified that particular class but the other class

For example for class low

TP = 35 TN = 20 + 35

False negative are sample of that class ~~predicted~~ predict some value in wrong way, for example For class low we ~~predicts 45 of low class~~ incorrectly,

predict as of low to be realu and is to be high

Finally False positive are true sample that are predicted in wrong way, for example

~~In the first colum we have that [illegible]~~
~~5 False low~~

In the first column we have 5 medium that are wrongly predicted to be low and 5 Hight that are wrongly predicted to be low

$$\text{Accuracy} = \frac{90+90}{180+65+65} = 0.58 \Rightarrow 58\% \text{ accuracy}$$

Ex 6

1) Loss function is

2) Yes the model used is in the form of linear regression and in particular estabilish relation between input and output

$y = \theta^T x \quad \theta = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{pmatrix} \quad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$

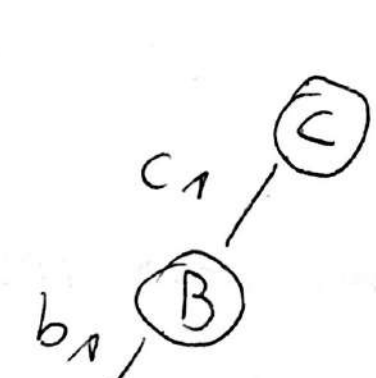
2) We can use the approach of sequential learning in which we use the stochastic gradient

ERASMUS

EX1

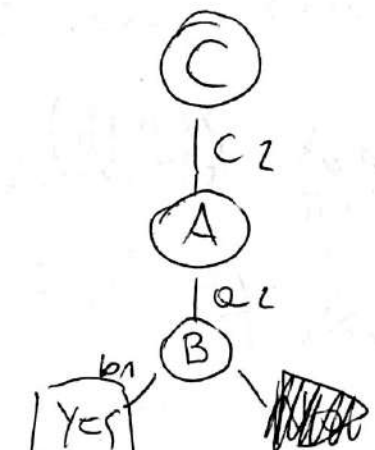
1) IF $C = C_1 \wedge B = b_1$ THEN NO
IF $C = C_1 \wedge B = b_2$ THEN YES
IF $C = C_2 \wedge A = a_1$ THEN YES
IF $C = C_2 \wedge A = a_2 \wedge B = b_1$ THEN YES
IF $C = C_2 \wedge A = a_2 \wedge B = b_2$ THEN NO
IF $C = C_2 \wedge A = a_3$ THEN NO
IF $C = C_3$ THEN NO

2) $S_1 = \langle a_1, b_1, c_1, NO \rangle$



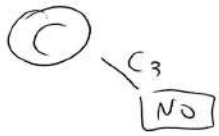
We can see that i drow part of tree given that value of set S1 and we can see that T is consistent with s_1 because NO of the set is else result of Tree for that values of attributs C, B, A

$S_2 = \langle a_2, b_1, c_2, Yes \rangle$



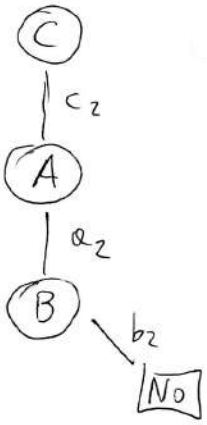
In this case T is ~~not~~ constant because with sequence c_2, a_2, b_1 We obtain value Yes that is the same of set

- $S_3 = \langle a_1, b_2, c_3^*, Yes \rangle$



In this case T is not constant because we should have "NO" value as we can see in the tree but S_3 set have value Yes so T is NOT consistent with S_3

- $S_4 = \langle a_2, b_2, c_2, Yes \rangle$



T IS NOT constant with S_4 for the same reason in which T is NOT constant with S_3

EX 2

1) MAXIMUM a posterior is used to compute most probable hypotheses

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(h|D) = \underset{h \in H}{\operatorname{argmax}} P(D|h) P(h)$$

We drop $P(D)$ because it is constant and independent of h

IF $p(h_i) = p(h_j)$

We have MAXIMUM LIKELIHOOD

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} P(h|D) = \underset{h \in H}{\operatorname{argmax}} P(D|h)$$

We drop $p(h_i)$ because it is constant

Bayes optimal classifier is classifier that we use to determine the most probable classification for new instance

given a dataset D and target function $f: X \to V$, $V = \{v_1, \dots, v_n\}$ and x new instance, $x \notin D$

$$V_{OB} = \underset{v_j \in V}{\operatorname{argmax}} \sum_{h_i \in H} P(v_j | x, h_i) P(h_i | D)$$

h_i is indipend from D so

$$P(v_j | x, h_i, D) = P(v_j | x, h_i)$$

and also h_i is indipendent from $x \notin D$

$$P(h_i | D, x) = P(h_i | D)$$

3) Bayes optimal classifier is optimal because it use prior probability hypothesis and to compute result but it is not pratical when we have huge Hypothesis space because we cannot have prior probability of each hypothesis, so in this case we use approximation using Naive Bayes Classifier

Ex3

Perceptron is a model used for liner classification

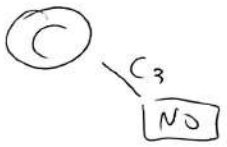
In particular we have function $\sigma(x) = \begin{cases} 1 & \text{IF } w^T x > 0 \\ -1 & \text{otherwise} \end{cases} \Rightarrow D = w^T x$

in which we have $y = w^T x$ and dataset $D = \{(x_n, t_n)\}$

the aim is to find good seperator to classify new instances.

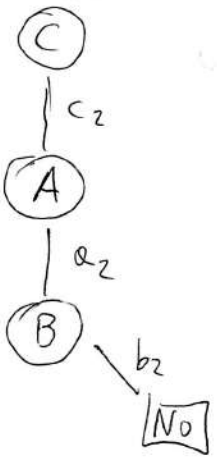
We start with value of w that is random and then at

- $S_3 = \langle a_1, b_2, c_3^*, Yes \rangle$



In this case T is not content because we should have "NO" value as we can see in the tree but S_3 set have value YES so it is NOT consistent with S_3

- $S_4 = \langle a_2, b_2, c_2, Yes \rangle$



T IS NOT content with S_4 for the same reason in which T is NOT content with S_3

EX 2

1) MAXIMUM a posteriori is used to compute most probable hypotheses

$$h_{MAP} = \underset{h \in H}{\arg\max}\ P(h|D) = \underset{h \in H}{\arg\max}\ P(D|h)\,P(h)$$

We drop $P(D)$ because it is constant and independent of h

IF $p(h_i) = p(h_j)$

We have MAXIMUM LIKELIHOOD

$$h_{ML} = \underset{h \in H}{\arg\max}\ P(h|D) = \underset{h \in H}{\arg\max}\ P(D|h)$$

We drop $p(h_i)$ because it is constant

)

Bayes optimal classifier is classifier that we use to determine the most probable classification for new instance

given a dataset D and target function $f: X \to V$, $V = \{v_1, \dots, v_n\}$ and x new instance, $x \notin D$

$$V_{OB} = \arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j | x, h_i) P(h_i | D)$$

~~x~~ h_i is indipendent from D so

$$P(v_j | x, h_i, D) = P(v_j | x, h_i)$$

and also h_i is indipendent from $x \notin D$

$$P(h_i | D, x) = P(h_i | D)$$

3) Bayes optimal classifier is optimal because it use prior probability hypothesis and to compute result but it is not practical when we have huge Hypothesis space because we cannot have prior probability of each hypothesis, so in this case we use approximation using Naive Bayes Classifier

Ex 3

Perceptron is a model used for linear classification

In particular we have function $\sigma(x) = \begin{cases} 1 & \text{IF } w^T x > 0 \\ -1 & \text{otherwise} \end{cases} \Rightarrow \theta = w^T x$

in which we have $y = w^T x$ and dataset $D = \{(x_n, t_n)\}$

The aim is to find good separation to classify new instances.

We start with value of w that is random and then at

each iteration this value change with stochastic gradient
in particular we minimize error:

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}(t_n - o_n)^2 = \frac{1}{2}\sum_{n=1}^{N}(t_n - w^T x_n)^2$$

$$\nabla E = -\sum_{n=1}^{N}(t_n - w^T x_n)\, x_{i,n} = \frac{\partial E}{\partial w}$$

We use stochastic gradient se

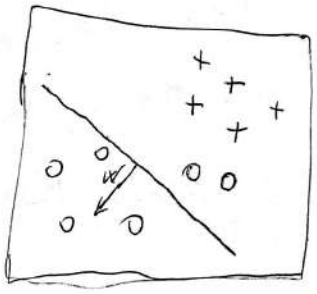
$$\hat{w} \leftarrow \hat{w} + \Delta w_i$$

$$\Delta w_i = -\eta \nabla E \qquad \text{when } \eta \text{ is learning rate}$$

$$= +\eta \sum_{n=1}^{N}(t_n - w^T x_n)\, x_{i,n}$$

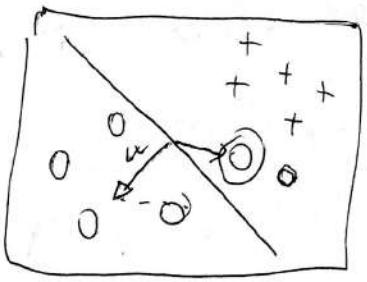
this means that the value of w change at each iteration when we find instance in the dataset that is MISSclassified this cycle termine after some FINITE ITeration IF dataset is linearly seperable and η is small otherwise it will never converge.

2) a)

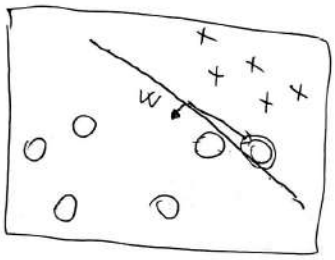


we divide with decision bendary input space because we choose w randomly, on the bottom should be instances of class "-" and on to instance of class "+"

b)

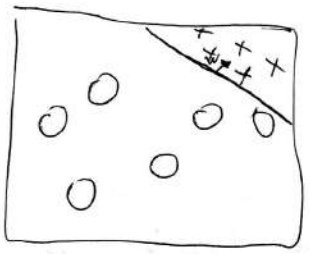


a class is MISSclassfier, so we need to update our weight and this mean that decision bendary change and also value of w



As we can see there is another instance that is missclasfier so we need to upgrade weights again

d)



Now we have that perceptron converge because there are no more instance missclassfied and dataset that we create is linearly seperable, and we consider lr small

EX4

1) W_1 We pass from 128 to 50 units so dimens of W_1 should be 128×50 and dimens of W_2 should be 50×10

2) we have RELU activation function for hidden layer so

$$h = g(x^T W_1 + b)$$

where $g = \max(0, z)$

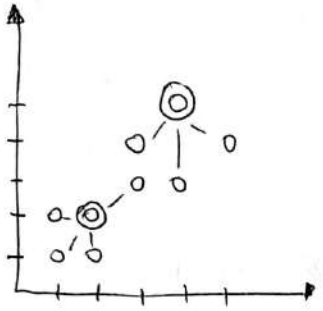
for output we have liner activation function so consider

$$y = W_2^T h + c = W_2^T \cdot \max(0, x^T W_1 + b) + c$$

EX 5

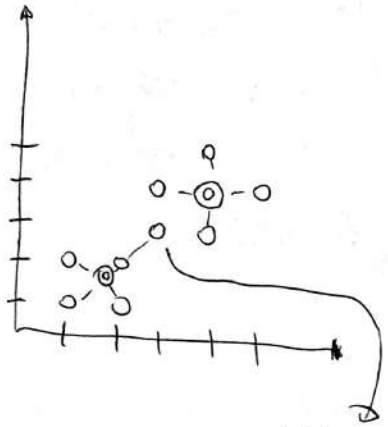
K = 2

a)



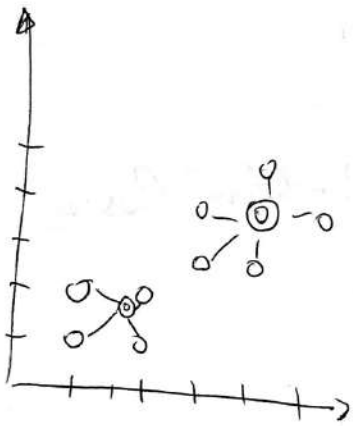
we understant that we want to update our centroid and I show in the picture samples belong to each cluster

b)



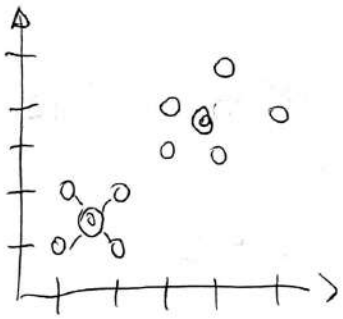
As we can see we Update our centroid and then we take in sequence each sample to check IF IT STAY IN cluster with nearest centroid
We see that there is a cluster that is not belong to the cluster with nearest centroid for it so we need to SWITCH cluster for this Sample

c)



Now we need to re compute centroids of the clusters involved

d)

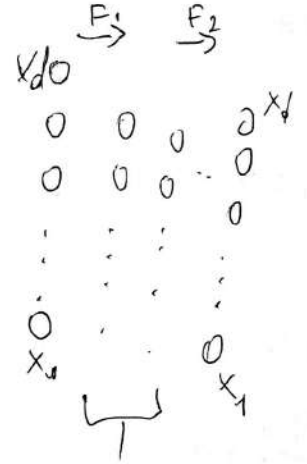


Now after update centroid we take each sample IN sequence and check again, We can see that all sample are in cluster with nearest centroid so K-means converge

Autoencoder is a neural network with reduced size of hidden layer and learn to reconstruct structure of inputs by minimize sum-of-squared error.

In perfect output and input have same size and hidden layers have a size minor of inputs and outputs

For example we have 3 for input and 2 for hidden

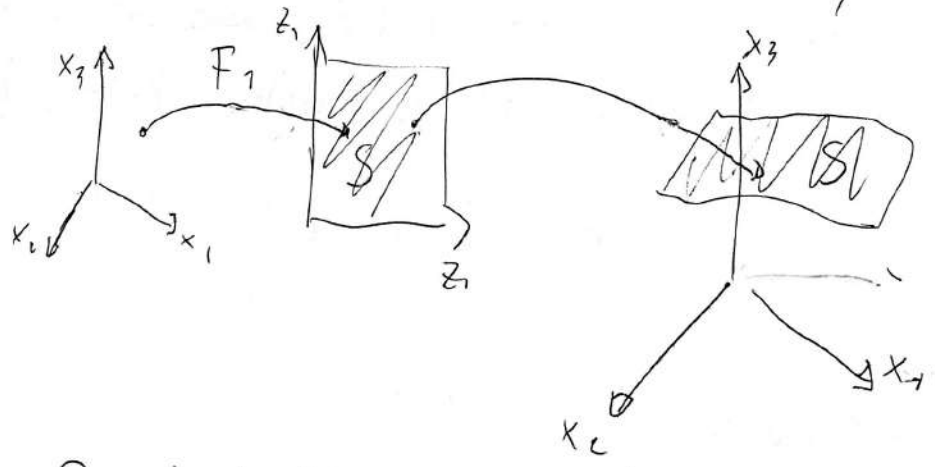


F_1 convert dimensionality of 3 to 2 and F_2 convert dimensionality of 2 to 3

these are non linear function for hidden layer

2)

We enable 3D - input, 2D - hidden, 3D - output



F_1 map from 3 to 2

F_2 map from 2 to 3

$\Delta = 3$

When $M = 2$

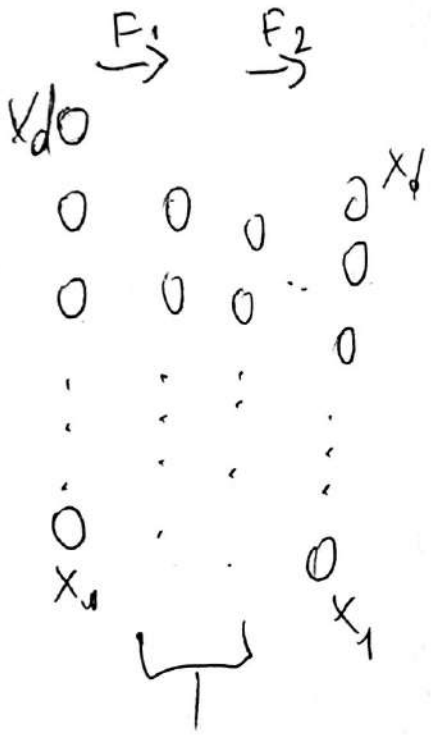
If F_2 is non linear we obtain in output a non planar function

6

Autoencoder is a neural network with reduced size of hidden layer and learn to reconstruct structure of inputs by minimize sum-of-squared error.

In particular output and input have same size and hidden layers have a size more of inputs and outputs

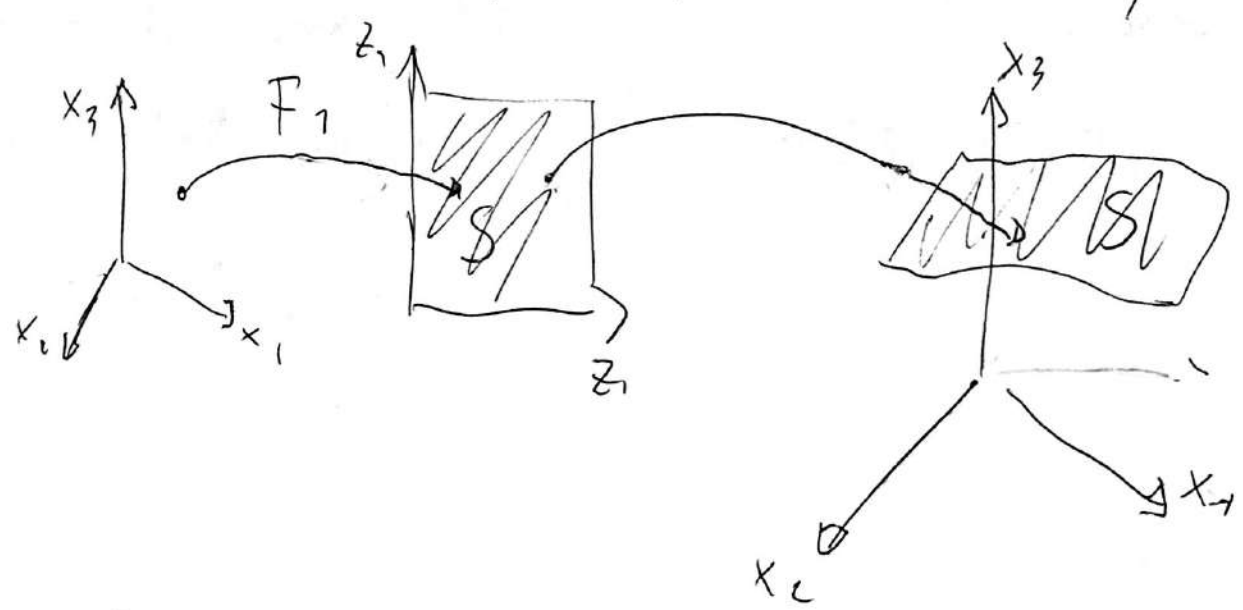
For example we have D for input and M for hidden



F_1 convert dimensionality of D to M
and F_2 convert dimensionality of M to D

2)

We enable 3D - input, 2D - hidden, 3D - output



F_1 map from D to M
F_2 map from M to D

D = 3
When M = 2

IF F_2 IS non linear
We obtain in output
a non planar
surface

Ex1

1) Given an Hypothesis space H, we can say that h ∈ H overfit training data IF EXIST h' ∈ H such that:

$error_S(h) < error_S(h') \Rightarrow error_D(h) > error_D(h')$

Where
$error_S(h)$ is Sample error
$error_D(h)$ is true error

2)
In Decision tree we consider to have overfitting when for example we have dataset that contains noisy data and ~~the decison tree~~ we have $|D| \geq 15$ and two decison tree T' and T make of different ID3.
We can have that T' has accuracy > than T' but this doesn't mean that T' is better than T because T' do overfitting causes of noisy data so there are different technique to avoid overfitting, in particular We can:

1) Stop growing tree, but it is not easy to do because it's difficult to estimate when to stop
2) Grow full tree then post prune

There are two different techique of post pru

1) Reduced Error Pruning
2) Rule post Pruny

- Reduced Error pruny we split dataset in training and validation set, then we until pruny is harmful we evaluate the impact on validation set of pruning each possible node then greedly remove node that most improves validation set accuracy

This method give use optimal subtree and Furthermore when trany sample is small it can give us bad results

- Rule post Pruning we infer the tree allowing overfitting then we convert tree in set of rules this because in this way we can distinguish each path. In this way we can prune each path independently to each other At the end after pruning we order set of rules as we want.

3

$D = \{ (\langle (q_1, s_1), p_1 \rangle, \dots, \langle (q_N, s_N), p_N \rangle) \}$

1) likelihood function

$$P(p \mid w) = \prod_{n=1}^{N} y_n^{p_n} (1 - y_n)^{(1-p_n)}$$

where $y_n = \sigma(w^T x_n)$

w - vector of weights

$$x = \begin{pmatrix} (q_1, s_1) \\ \vdots \\ (q_N, s_N) \end{pmatrix}$$

2) We should learn w, weights

In particular

$$w^* = \arg\min_w E(w)$$

3) $E(w)$ is cross entropy

$$E(w) = -\ln P(p \mid w) = -\sum_{n=1}^{N} p_n \ln y_n + (1-p_n) \ln (1-y_n)$$

Ex 5

1) In stochastic and non deterministic actions, state resulting of execution of an action is not know before execution of the action but it is fully observable after execution.

2) MDP is a process that use decision making and it is expressed by,

$\langle X, A, \delta, r \rangle$

X is set of states
A is set of actions
δ is transition function
r is reward function
and states are fully observable

Instead POMPD is combination of MDP and HMM so POMPD has also observation respect to MDP with partially observable

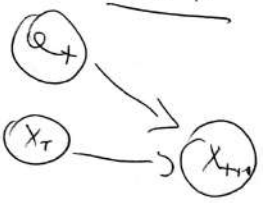
$\langle X, A, Z, \delta, r, O \rangle$

O observation function, this mean that states are

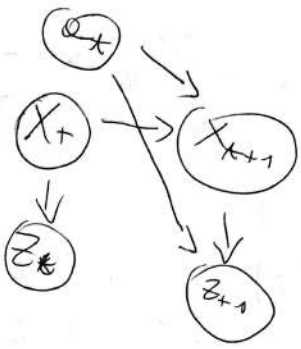
X set of states
A set of action
Z set of observation

δ is transfer fun
r is rew f
O is observ fun

MDP



POMDP



Ex 6

1) Boosting is a method usefy when we want to use different classifier and combine their result. In particular Boosting use different classifier in sequence. Each learner is trained in sequence and it is trained on weighted data. Each weighted data depends on performance of the previous classifier. Points that are missclassified give vs high weight. the end prediction , se final output is done with majority of votes.

2) AdaBoost is a kind of Boosting in particular the algorithm is const, given a dataset $D = \{(x_n, t_i)\}_{n=1}^{N}$, $x_n \in X$, $t_n \in \{-1, 1\}$

We initialize $w_n^{(1)} = \frac{1}{N}$ $n = 1, \dots, N$

then for $m = 1 \dots M$ do

We train weak learner $y_m(x)$ by minimize weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(x) \neq t_n)$$

With

$$I(e) = \begin{cases} 1 & \text{if } e \text{ is True} \\ 0 & \text{if } e \text{ is false} \end{cases}$$

the we evaluate ε_m and α_m

Update weighted coefficient

$$w_n^{(m+1)} = w_n^{(m)} \exp(\alpha_m I(y_m(x) \neq t_n))$$

Output of classifier is

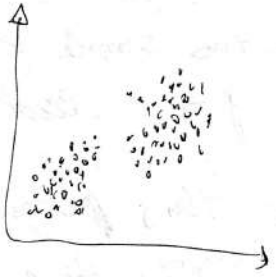
$$Y_M(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(x)\right)$$

EX

1) SVM we want to find MAXIMUM margin to provide better accuracy. In particular we find hyperplane with maximum margin separation on the training data, in this way we classify our points in good way respect margin that is too small

1) Unsupervised learning is learning that has a dataset $D = \{x_n\}_{n=1}^{N}$ we have input but value of target function corresponding to the input is not available. In unsupervised learning we want to find a structure of input using different method such as clustering

2)



$D = \{x_n\}$

$K = 2$

3) We want to find K-means provided by K gaussian distributions

Given a dataset $D = \{x_n\}$ and $K = 2$, such as previous problem

1) We consider $K = 2$ number of clusters

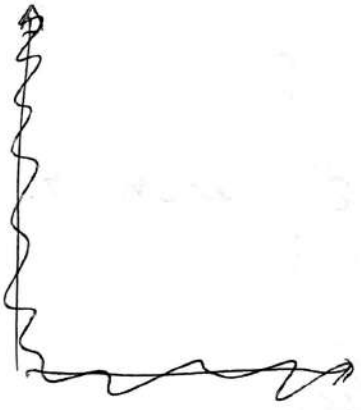
2) Now we need to do an initialization that classify data in K clusters

3) We take First K training samples as single clusters

4) Assign to remaining N-K training sample to cluster with Nearest centroid.
After each assignment we update centroid of clusters

5) Take each sample in sequence and we check if that sample is in the cluster with Nearest centroid
If sample is not in cluster with Nearest centroid we switch cluster for this sample and then We update centroids of two clusters involved

6) We repeat 5) UNTICE this algorithm converge, It converge when there are no more sample that need to switch cluster.



Y

Backpropagation is an algorithm that is use to compute gradient ~~Beacon~~ Respect to parameter θ of loss function; $\nabla_\theta J(\theta)$

We want to propage this gradient from output to input after computing loss function.

Backpropagation is algorithm that is used to compute only gradient, IT IS NOT a training algorithm for neural network.

Forward and backward pass are two steps of backpropagation algorithm

Forward pass we compute loss function starting from input and go to the output, then in backward pass we use loss function computed to compute gradient and propagate this in each layer from output layer to input.

Stochastic Gradient Discent is a training algorithm that want to improve performance of neural network using gradient.

In particular is an algorithm that update parameters θ using gradient: $\theta^{(k+1)} \leftarrow \theta^{(k)} + \eta g$

where g is gradient

2) Forward pass

Require: l depth of neural Network

Require: $W^{(i)}$ matrices of weights $i = 1, \dots, l$

Require: $b^{(i)}$ bias $i = 1, \dots, l$

Require: input vector x

Require: target vector t

$h^{(0)} = x$

for $k = 1, \dots, l$ do

$a^{(k)} = W^{(k)T} h^{(k-1)} + b^{(k)}$

$h^{(k)} = f(a^{(k)})$

end for

$y = h^{(l)}$

$J = L(y, t)$

where J is cost function; loss function

ckward Pass

$g \leftarrow \nabla_y J = \nabla_y L(T, y)$

For $k = l, \dots, 1$ do

Propagate gradient to pre-linear activations

$g \leftarrow \nabla_{a^{(k)}} J = g \odot f'(a^{(k)})$

$\odot$ — element-wise product

$\nabla_{b^{(k)}} = g$

$\nabla_{W^{(k)}} = g \left(h^{(k-1)}\right)^T$

Propagate gradient to the next lower-level hidden layer

$g \leftarrow \nabla_{h^{(k-1)}} J = \left(W^{(k)}\right)^T \cdot g$

End for

EX

1) We have a binary classification problem so for $t \in T = \{0, 1\}$ we can use for activation function in output sigmoid function

$y = \sigma(w^T h + b)$ and cost function should be binary cross entropy

$J(\theta) = \text{softplus}((1 - 2t)\alpha)$ $\alpha = w^T h + b$

2)

As we on the spiral forest we see that our units activate only when they give use the correct answer.

EX

1)

$W_{out} = W_{IN}$

$W_{out} = \frac{W_{IN} - 3_{kernel} + 2p}{S} + 1$

$W_{out} = \frac{W_{IN} - 3 + 2p}{S} + 1$

$W_{IN} \cdot S = W_{IN} - 3 + 2p + S$

$$S = \frac{W_{IN} - 3 + 2p}{W_{IN} - 1}$$

$h_{out} = \frac{h_{IN} - 3 + 2p}{S} + 1$

$h_{IN} \cdot S = h_{IN} - 3 + 2p + S$

$S = \frac{W_{IN} - 3 + S(h_{IN} - 1) + 6}{W_{IN} - 1}$

$2p = S(h_{IN} - 1) + 3$

$$p = \frac{S[h_{IN} - 1] + 3}{2}$$

$$S = \frac{W_{IN} - 3 + 2p_{IN} - S + 6}{W_{IN} - 1}$$

$$S(W_{IN} - 1) + S = W_{IN} + 2p$$

$$S(W_{IN} - 1 + 1 - h_{IN}) = 3$$

$$S(W_{IN} - h_{IN}) = 3$$

$$\boxed{S = \frac{3}{W_{IN} - h_{IN}}} \qquad \boxed{p = \frac{3(h_{IN} - 1) + 3}{2(W_{IN} - h_{IN})}}$$

EX

Convolutional layer is composed of 3 stages: convolutional stage, detector stage and pooly stage.

In Convolutional stage we have convolution functions between input and kernel

$$(I * K)(i, j) = \sum_m \sum_n I(i+m, j+n) K(m, n)$$

We can apply different Techniques.

- Sparse connectivity and parameter sharing

With Sparse connectivity we have output of layer depends only of few input in this way we can reduce number of operation and increase statistically efficiency

With Parameter sharing we have that ALL units of that layer share subset of parameter, this means that we don't learn more parameter and we can use some parameter ~~for other~~ with other functions.

Finally we have padding that could be same or valid, IF it same input is padding with zeros and output doesn't depend on kernel size

IF padding is valid output contains valid values and output depends of kernel size.

After convolutional stage we have linear function that in detector stage use some non linear Function, like ReLu, to transform those linear funct in non linear.

In pooling stage we modify the output of neural network, in particular pooling function replace ~~output~~ at certain position the output with some of statistically outputs nearby

are are two different Pooly funct:

Max pooling: We take MAXIMUM of neighbors outputs

Average Pool: we take average of neighbors outputs

With Stride we also reduce size of output

EX

PCA is technique reduce dimensionality of dataset with many parameter correlated with each other while keeping present variation.

PCA maximize variance or minimize error

When we maximize variance we need to consider points projected to perticular direction

we need to ~~[illegible]~~ consider direction vector u_1

We compute mean projected

and variance project

and after ~~[illegible]~~ maximize of it, se ~~[illegible]~~ use lagrance and then ~~[illegible]~~ setting derivate to zero We obtain

$S u_1 = \lambda u_1$ when S is variance, we know $u_1^T u_1 = 1$

$u_1^T S u_1 = \lambda$

to MAXIMIZE variance we need to chose u_1 as eigenvector corresponding to largest ~~[illegible]~~ eigen values this is principal components

EX

1) K Fold Cross validation is a method to estimate accuracy of learning algorithm

We consider dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$

We split our dataset IN K S_i with $|S_i| \geq 30$

for $i = 1, \ldots, K$

S_i is test set and T_i is training set where

$T_i = D - S_i$

$h_i = L(T_i)$ We compute predict using algorithm L

$\delta_i = error_S(h_i)$ sample error of h_i

$$error_S(h) = \frac{1}{K} \sum_{i=1}^{K} \delta_i$$

$$accuracy_{S,D}(h) = 1 - error_{S,D}(h)$$

2) K-fold is a validation method used also when you need to understand which algorithm is better IN terms of accuracy

We consider L_A, L_B

We split the dataset D IN K S_i $|S_i| \geq 0$

for $i = 1, \ldots, K$

S_i is Test set

T_i is traing set = $\{D - S_i\}$

$h_A = L_A(T_i)$

$h_B = L_B(T_i)$

we compute prediction of A and B over some traing set

$\delta_i = error_S(h_A) - error_S(h_B)$

$$\bar{\delta} = \frac{1}{K} \sum_{i=1}^{K} \delta_i$$

If $\bar{\delta} < 0$ we can say that L_A is better than L_B because $error_S(h_A)$ is minor of $error_S(h_B)$

EX

One state markov process we have

$\langle x_0, A, \delta, r \rangle$

where x_0 is unique state
A is set of actions
δ is transiton funcke
r is reward function

K-armed bandit problem is stocastic problem for one-state MDP
In pertucular we have K actions and

$r(a_i) = N(\mu_i, \sigma_i)$ gaussian distribution

K-armed Bandit use ε-greedy strategy

In which we choose ~~action~~ random action with probability ε and then we select best action with probability $1-\varepsilon$ and ε decreases during time

2) The trony rule is based on Q-learning

$$Q_n(a_i) \leftarrow Q_{n-1}(a_i) + \alpha [r_t(a_i) Q_{n-1}(a_i)]$$

$$\alpha = \frac{1}{V_{n-1}(a_i)}$$

where $V_{n-1}(a_i)$ N° of ~~execute~~ execute of a_i upto time $n-1$

Ex

1) Generative is a probabilistic model that use Bayes rule to estimate $P(C_i|x)$ for classification, instead discriminant probabilistic model use directly model to estimate value of $P(C|x)$, in details logistic regression

CNN

1) 2 layer

- Output of CNN is W_{out} and h_{out}

$$W_{OUT} = \frac{W_{IN} - W_{kernel} + 2 \cdot padding}{Stride} + 1$$

$$h_{out} = \frac{h_{IN} - W_{kernel} + 2 \cdot p}{S} + 1$$

- Output value is some of output in input based activation function also + layer dimension of output
- output of pool is

$$W_{out} = \frac{W_{IN} - W_{pool}}{Stride} + 1 \qquad h_{out} = \frac{h_{IN} - h_{pool}}{S} + 1$$

We consider first layer

we have after conv 1

$$W_{out} = \frac{1242 - 5 + 4}{1} + 1 = 1242 \qquad h_{out} = 378$$

⇒ output after conv 1 is $1242 \times 378 \times 64$

where 64 are features

• after relu dimension doesn't change and after pooly we have

$W_{out} = \frac{1242 - 2}{2} + 1 = 621 \qquad h_{out} = \frac{378-2}{2} = 189$

Output of first layer is $621 \times 189 \times 64$

this should be input of second layer

- Conv2 - 3x3 AND 128 feature pad = 0 stride = 2

OUTPUT IS given by $621 \times 189 \times 64$

$W_{out} = 310$

$h_{out} = 94$

We obtain $310 \times 94 \times 128$

- We can skip value 2 because dimensions don't change after pool 2

$W_{out} = 78$

$h_{out} = 24$

Final output $78 \times 24 \times 128$

2) N° of parameter depend only of convolutive stage this because pooly and Relu doesn't don't contribute of number of parameter

N_1 = N° parameter after first layer

$= W_K * h_K \cdot Feature_{IN} \cdot Feature_{out} + Feature_{out}$

$= 5 \cdot 5 \cdot 3 \cdot 64 + 64 = 4864$

$N_2 = 3.3.68.128+128 = 73856$

Ex

1) the dimensionality of data space is the input space, se in this ex 12x12 = 144 the intrisic dimensionality of given data is translation and rotation, since is the same image we have 2 traslation and 1 rotation se INTRINSIC dimensionality is 3

2) ~~$x_n = \sum_{i=1}^{M} z_{ni} \cdot \mu_i$~~ μ_i

$$\tilde{x}_n = \sum_{i=1}^{M} z_{ni} \cdot \mu_i \quad \text{where } z_{ni} = x_n^T \mu_i$$

means that z_{ni} depend on particular data point

3) Not necessary, usually M is usually differ of intrinsic dimension because PCA is linear so ~~open~~ principal compents of PCA are not latent variable

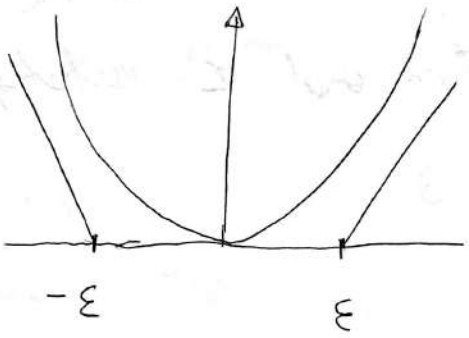
$$Q(x,a) \leftarrow \bar{r} + \gamma \max_{a'} \left(Q(x', a') \right)$$

$$Q_n(a_i) \leftarrow Q_{n-1}(a_i) + \alpha \left[r(a_i) + \gamma Q_{(n-1)}(a_i) \right]$$

$$\alpha = \frac{1}{V_{n-1}(a_i)}$$

EX3

1)



the problem to minimize J is that ϵ is NOT differentiable

2) ξ_n^+ and ξ_n^- slack variable with

$\xi_n^+ > 0 \quad \xi_n^- > 0$

the role of them is to de minimize of J without probl of not differentiable met by ϵ in practice we have

$$J(w, \xi) = C \sum_{i=1}^{N} (\xi_n^+ + \xi_n^-) + \frac{1}{2} \|w\|^2$$

15/10/2019

EX 1

1) CONFUSION MATRIX is a performance measure used to evaluate in a classification problem with many classes, in the diagonal we can have accuracy of each class.
Furthermore we compute how many times an instance of class C_i is classified in class C_j outside the diagonal we have errors.

EXAMPLE

true value \ predicted value	C_1	C_2	C_3
C_1			
C_2			
C_3			

2)

3 class classifier 100 sample of each class

2)

	C_1	C_2	C_3	FN
C_1	70	20	10	30
C_2	15	75	10	25
C_3	25	15	60	40
FP	40	35	20	205

	C_1	C_2	C_3	
C_1	0.7	0.2	0.1	
C_2	0.15	0.75	0.1	
C_3	0.25	0.15	0.60	

3) $TP = TN = 205 \quad FN = 95$
$FP = 95$

$$Acc = \frac{TP+TN}{TP+TN+FN+FP} = \frac{410}{410+190} = 0.68$$

EX2

1) We peform regression method chong good relationshion between inputs and outputs in particular in this case we can choose polynomial of degree 3

$$y(x,w) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

w_0, w_1, w_2, w_3 are weights

x input variable

2) In regression we use regularization to to avoid overfitting, in pattricular we add a

regularization term with error E_D

$$E_D(w) = \frac{1}{2}\sum_{n=1}^{N}(t_n - w^T x_n)^2$$

In particular we have

$$\min E_D(w) + \lambda E_w(w)$$

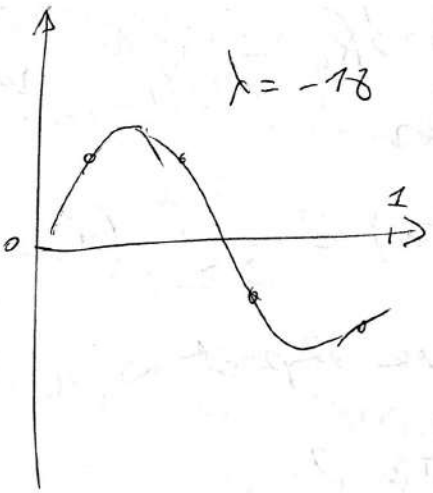
where λ is regularization factor, $\lambda > 0$

And $E_w(w)$ is regularization term

We can choose $E_w(w) = \frac{1}{2} w^T w$ for our model

3)

the plot is



EX3

1) Perceptron is model using in linear classification in particular it has aim to classify perfectly inputs of dataset updating weight in each iteration if it find some instances that are classified in bad way

IT update weights using learning rate and gradient of error, but we can say that after this operation the decision border change until we don't find any instances that are NOT Missclassified (IF IT possible usually depends on dataset)

2) we have $y = w^T x$, but in particular IN this case we have function $\theta(x) = \begin{cases} 1 & \text{IF } w^T x > 0 \\ -1 & \text{otherwise} \end{cases}$

$\theta = w^T x$

Given a dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$ we want to minimize loss function

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (t_n - \theta_n) = \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T x_n)$$

We compute derivative respect to w and we obtain

$$\frac{\partial E(w)}{\partial w_i} = -\sum_{n=1}^{N} (t_n - w^T x_n)^2 x_{i,n} = \nabla E$$

$w \leftarrow w + \Delta w_i$ this is formula for update weights

where

$$\Delta w_i = -\eta \nabla E = +\eta \sum_{n=1}^{N} (t_n - w^T x_n)^2 x_{i,n}$$

η is learning rate

, so this means that we
find a decision border that classify our instances
in a such way there is no instance that is
missclassified, if Dataset D is linearly separable
and if learny ~~rate~~ rate η is sufficiently small.
If η is too small, the convergence will be
too slow.

Ex4

1) In SVM we want to maximize margin to provide better accuracy.
Margin is the smallest distance between closest points to the hyperplane (decision border) and the hyperplan itself.
To find w^* and w_0^* we maximize that margine, so this means that we need to use Lagrange Method to compute w^* and w_0^*
w_0^* is compute using suppect vectors
at the end we have

$$sg(y(x')) = sign\left(\sum_{k \in SV}^{N} a_k t_k x_k^* x'^T + w_0^*\right)$$

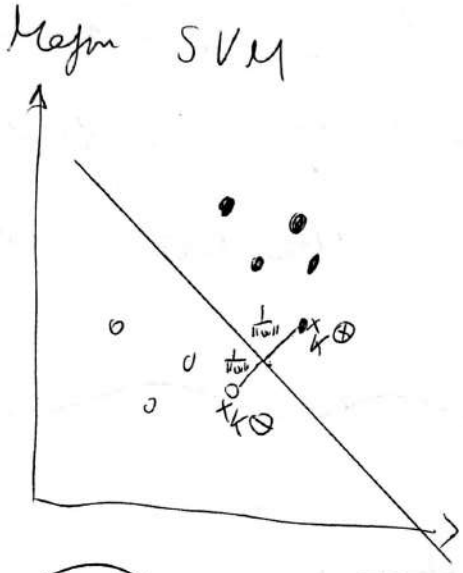
At the end we obtain that we have two closest points ~~[redacted]~~ $x_k^{\oplus}$ and $x_k^{\ominus}$ whe distance should

be the same $\frac{1}{\|w\|}$ INFACT $w^{*T} \cdot x_k^{\ominus} + w_0^{*} = 1$ or

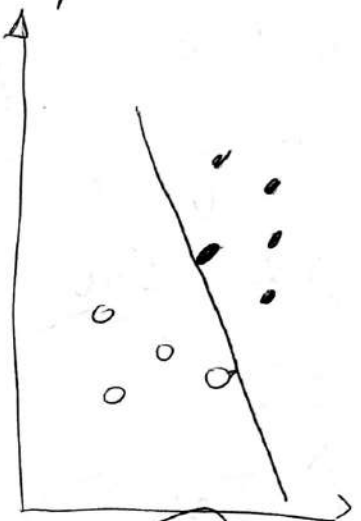
$w^{*T} x_k^{\ominus} + w_0^{*} = -1$

2)

ii)



i)



EX 5

1) K Nearest neighbor algorithm is algorithm used for non parametric model.

In particular we consider a dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$ and learning function $f: X \to$ and K, and $x \notin D$ the first steps is to compute K nearest neighbors to instance x then we assign to x the most common value among this K-nearest neighbor

In particular likelihood function is

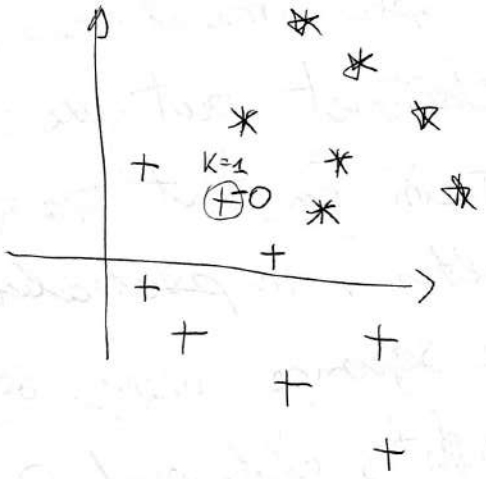
$$c(x, D, K) = \frac{1}{K} \sum_{N_K(x,D)} I(T_i = c)$$

$N_K(x, D)$ are K nearest neighbors of x

$$I(e) = \begin{cases} 1 & \text{if } e \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

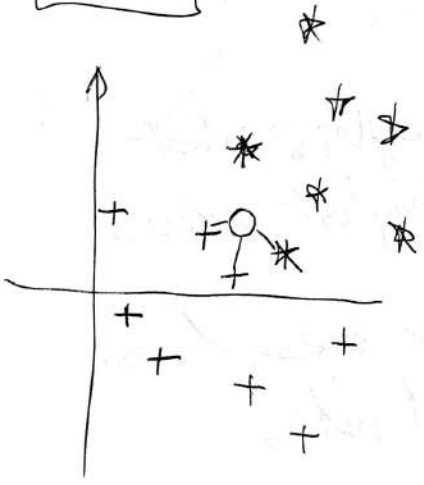
2) **K=1**

With K=1 we can see that "+" is nearest neighbor of "O" so in this case the value of this new instance should be plus (+).



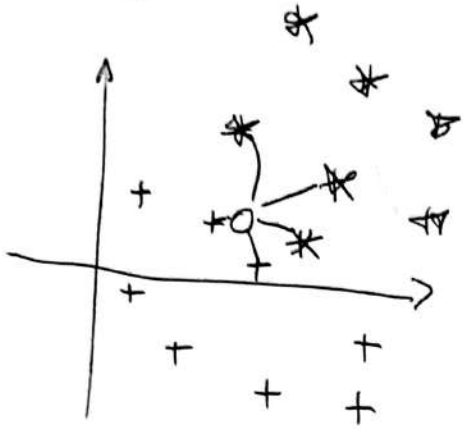
✓

K = 3



In this case with K=3 O has 3 neighbors : 2 plus, 1 star so also in this case we assign value "+", plus, to the new instance "O"

K = 5



IN this case neighbors are 3 stars and 2 plus stars are major than plus so the New instance "o" Will have value of stars (*)

EX6

1) Boostin is based on concept that we want to use NOT a single complex model/learner to train a particular dataset but we want more learners that train on that trainy dataset and then combine results / IN particular boosting made this a a sequence way, so each learner train on weights obtained from performance of previous learner

IF there are some error, ~~the~~ weights that next learner decide will have highst values.

The predictor of learner are based on weighted majority of votes

2) We create a weak learner, base learner,

(x), we train it minimizing error function than what we do is to evaluate it, so make prediction and after this we update the data weighting coefficient to obtain the output of the final classifier

$$Y_M(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m Y_m(x)\right)$$

$$\alpha_m = \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$$ ϵ_m is the evaluation

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} \left(Y_m(x_n) \neq t_n\right)}{\sum_{n=1}^{N} w_n^{(m)}}$$

I forgot to say that first we inizialize $w^{(1)} = \frac{1}{N}$ and for $m = 1, \dots M$ we train the weak classifier

12/04/2019

EX 1

1) Given Hypothesis space H, given $h \in H$ and with error on training set $error_S(h)$, while error on distribution D is $error_D(h)$

consider to have another $h' \in H$ in which we have that if

$$error_S(h) < error_S(h') \Rightarrow error_D(h) > error_D(h')$$

this mean that h overfit training data.

2)

We can have problem of overfring in decision tree in particular when we have noise data in our dataset

~~Further more~~ also ~~when~~ training data cannot represent ~~in a~~ good way ~~sample~~ or target function.

our when you create a tree ~~that~~ that is too deep and too specific for a particular training set

because also if you obtain an accuracy that is better than another decision tree that is more generic you risk to have a huge overfitt.

We can avoid overfit using two different methods:

we can stop tree or we build tree and then postprune.

estimate to stop the tree is difficult because usually you don't know exactly when you have to stop.

post-prune is ~~good~~ best idea.

THERE ARE TWO TYPE OF POST-PRUNE: REDUCED-ERROR PRUN
AND RULE POST-PRUNING

WITH REDUCED-ERROR PRUNING WE SPLIT DATA IN TRAINING TEST
AND VALIDATION TEST.

UNTIL PRUNE IS HARMFUL WE EVALUATE IMPACT ON VALIDATION
SET OF PRUNING EACH POSSIBLE NODE AND THEN WE SELECT
NODE TO REMOVE THAT IS THE MOST IMPROVES
VALIDATION SET ACCURACY

THIS REDUCED-ERROR PRUNING IS GOOD ONLY FOR SUBTREE OF
OUR DECISION TREE AND FURTHERMORE IF WE HAVE A SMAL TRAINING
SET IT CAN GIVE US BAD RESULTS.

RULE-POST PRUNING IS A VARIANT OF REDUCED POST-PRUNING
AND WE INFER THE DECISION TREE ALLOWING FOR OVERFITTING
AND THEN WE CONVERT OUR TREE INTO AN EQUIVALENT
SET OF RULES, THIS BECAUSE EACH PATH OF THE TREE
PRODUCE DIFFERENT RULE AND CONVERTING YOU CAN SEE
THIS DISTINCTION, SO YOU CAN PRUN IN DIFFERENT WAY.
AFTER CONVERTING YOU PRUNE EACH RULE INDIPENDENTLY OF OTHERS
FINALLY SORT FINAL RULES INTO A SEQUENCE THAT YOU WANT TO
USE.

EX2

NAIVE BAYES CLASSIFIER IS A CLASSIFIER TO FIND MOST
PROBABLE CLASSIFICATION FOR A NEW INSTANCE THAT IS NOT IN
THE DATA SET, IT USE VALUES OF ATTRIBUTE, IN
PARTICULAR INSTANCE IS DESCRIBED WITH VALUES OF

RIBUTE.

GIVEN $D = \{(x_n, t_n)\}_{n=1}^{N}$ dataset, TARGET FUNCTION

$f : X \to V$, $V = \{v_1, \ldots, v_K\}$

$$P(v_j \mid x, D) = P(v_j \mid a_1, \ldots, a_n, D)$$

WE NEED TO MAXIMIZE THIS TO COMPUTE NEIVE BAYES CLASSFI

class OF NEW INSTANCE x

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j \mid D) \prod_i P(a_i \mid v_j, D)$$

NAIVE BAYES RESPECT TO BAYES OPTIMIZER DON'T COMPUTE "OPTIMAL VALUE" BUT IT DO APPROXIMATION AND IT DOESN'T NEED HYPOTHESIS AND PRIOR PROBABILITY TO FIND MOST PROBABLE CLASSIFICATION FOR A NEW INSTANCE, AND THIS MEAN THAT NAIVE BAYES DON'T HAVE PROBLEM OF SIZE OF H THAT HAS BAYES OPTIMAL CLASSIFIER.

OF COURSE WE LOST OPTIMALITY BUT APPROXIMATION OF NAIVE BAYES CLASSIFIER ARE GOOD ENOUGH.

2)

3 CLASSE, SO I NEED TO USE NAIVE BAYES WITH MULTINOMIAL

WITH PROBABILITY $P(C_i)$, PROBABILITY OF THAT PARTICULAR CLASS BETWEEN: ML, KR, PL

WE MUST ESTIMATE $P(C_J)$ AND $P(w_i|C_J)$

where w_i is word i-esime and $P(w_i|C_J)$ is the probability to find word w_i given a class C_J

$$V_{NB} = \underset{C_J}{\arg\max}\ P(C_J) \cdot \prod_{i=1}^{len(d)} P(w_i | C_J, D)$$

d is document that we need to analize

EX3

1) giving a dataset $D = \{ \langle (a_1, s_1), p_1 \rangle, \dots, \langle (a_n, s_n), p_n \rangle \}$

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

$$P(T|W) = \prod_{i=1}^{N} y_i^{p_i} (1 - y_i)^{1-p_i}$$

with

$$y_i = \sigma(w^T x_i) = P(C_J | x_i)$$ posterior probability of class C_J

We use cross-entropy function

$$E(W) = -\ln P(T|W) = -\sum_{i=1}^{N} p_i \ln y_i + (1 - p_i) \ln (1 - y_i)$$

$$w^* = \arg\min_w E(w)$$

in this way we can estimate IF student pass or NOT exams, where w are weights

We know iterative option to minimize error: Newton-Raphson

Gradient error respect to w

$$\nabla E = \sum_{i=1}^{N} (y_i - p_i) x_i$$

adient descent step

$$w^{Nea} \leftarrow w^{old} - H^{-1}\nabla E$$

H^{-1} is hessian matrix of $E(w)$

$$H^{-1} = \nabla\nabla E = \sum_{i=1}^{N} y_i(1-y_i)\, x_i x_i^T = X^T R Y X$$

$$R = y_i(1-y_i)$$

$$\nabla E(w) = X^T(y-t) \qquad \text{↳ } w \leftarrow w - (X^T R Y)^{-1} \cdot X^T(y-t)$$

2) Parameter w that you have to learn are weight, so w and you update it using iterative method IRLS until then are termination condition

3) ERROR FUNCTION IS A CROSS ENTROPY

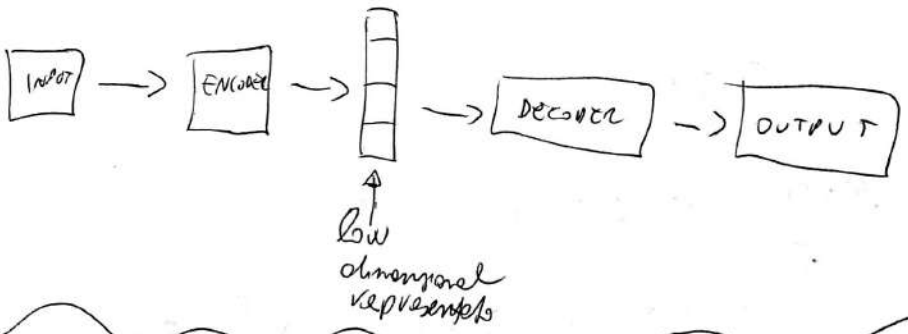
$$E(w) = -\ln p(t|w) = -\sum_{i=1}^{N} p_i \ln y_i + (1-p_i)\ln(1-y_i)$$

EX 4

1) Autoencoder are COMBINATION OF encoder and decoder, encoder take imges in input and return a vector such as CNN, decoder takes input a vector and return imges se it is compose by deconvolutional layers

Autoencoder Trained is base on reconstruction loss. and provides low dimensional representation.

2)



EX 5

1)

STATES ARE FULLY OBSERVABLE BECAUSE IN PARTICULAR IN PRESENCE OF NON DETERMINISTIC OR STOCHASTIC ACTIONS, THE STATE RESULTING FROM EXECUTION OF AN ACTION IS NOT KNOW BEFORE EXECUTION OF THE ACTION, BUT AFTER IS FULLY OBSERVABLE

2)

IN MDP WE HAVE STATES THAT ARE FULLY OBSERVABLE SO WE DON'T NEED OBSERVATIONS, INDEED IN POMDP WE HAVE COMBINATION OF NOT FULLY OBSERVABLE AND FULLY OBSERVABLE AND FURTHER MORE THEY HAVE DIFFERENCE OF STRUCTURE OF MODEL.

IN MDP WE HAVE

$\langle X, A, \delta, r \rangle$

Where X IS SET OF STATES, A IS SET OF ACTIONS, δ IS TRANSITION FUNCTION AND r IS REWARD FUNCTION

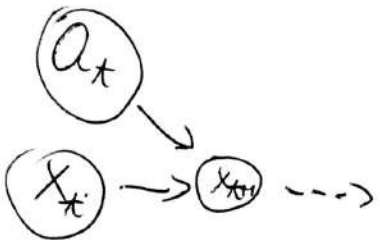
IN POMDP WE HAVE

$\langle X, A, Z, \delta, r, O \rangle$

X IS SET OF STATES, A IS SET OF ACTIONS, Z IS SET

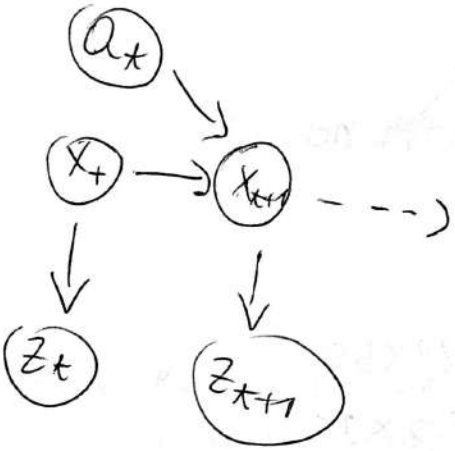
OBSERVATION, δ TRANSITION FUNCTION, V IS REWARD FUNCTION,
O IS PROBABILITY DISTRIBUTION OVER OBSERVATION.

3) MDP



AS WE CAN SEE
X_{t+1} DEPEND OFONLY OF CURRENT STATE AND WE DON'T HAVE ANYTHING THAT ACTIONS AND STATES

POMDP



HERE WE HAVE THAT X_{t+1} DEPEND OF COURSE OF CURRENT STATE BUT ALSO ACTIONS AND OBSERVATION BECAUSE WE HAVE A COMBINATION OF MDM HMM PROCESS (IN WITICH ONE IS FULLY OBSERVABLE AND THE OTHER ONE NO.

EX6

two layer ANN

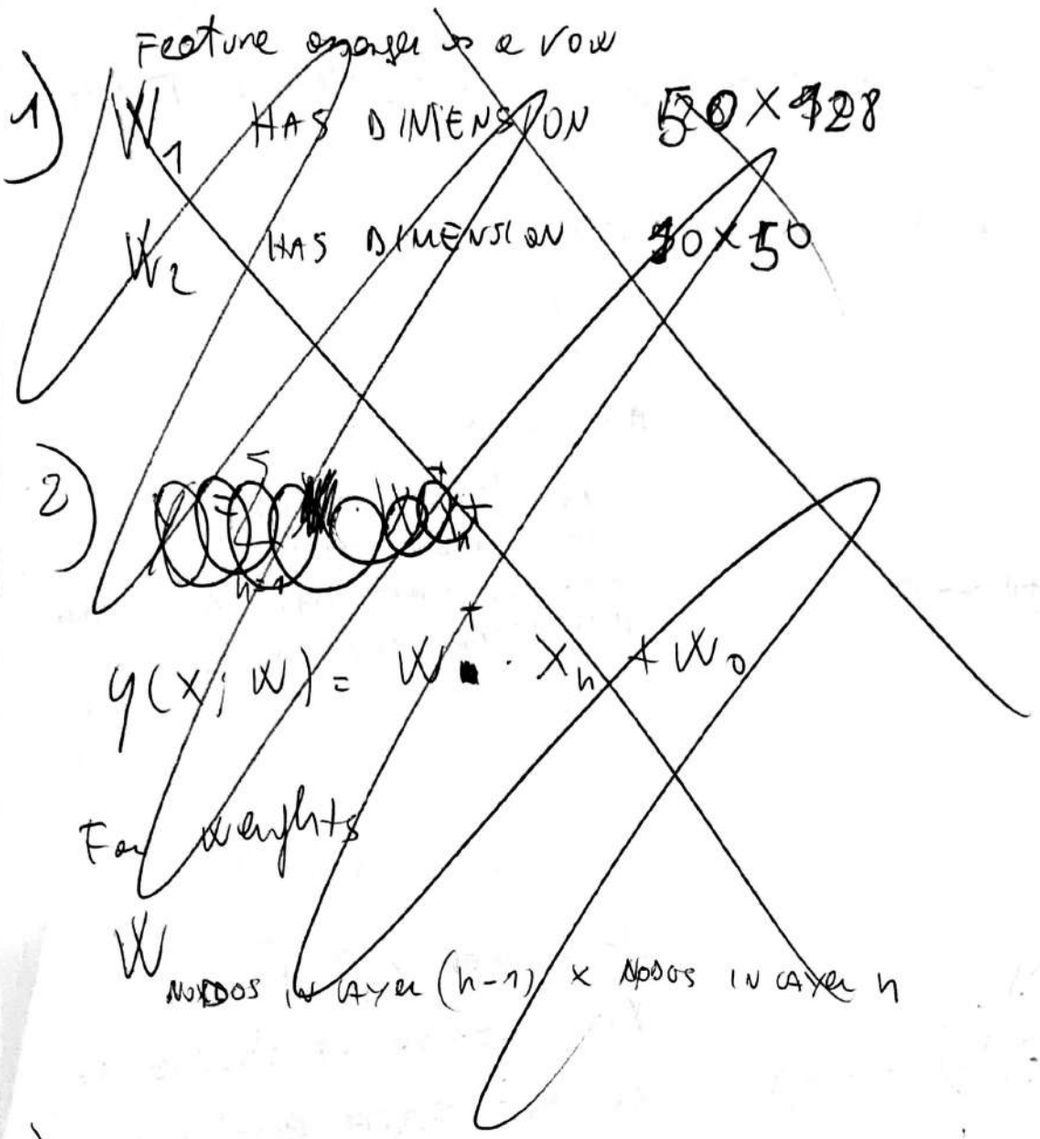
INPUT 128
OUTPUT 10
HIDDEN 50

4)

Feature organized in a row

1) W_1 HAS DIMENSION 50×128

W_2 HAS DIMENSION 50×50



2)

$y(x; W) = W^T \cdot X_h + W_0$

For weights

$W_{\text{NODOS IN LAYER } (h-1) \times \text{NODOS IN LAYER } h}$

1) DIMENSION OF $W_1 = 128 \times 50$

DIMENSION OF $W_2 = 50 \times 10$

2) W_0 we have value for activation function of hidden layer and linear for output

For weight DIMENSION IS = NODES IN LAYER $(h-1) \times$ NODES IN LAYER h

$h_1 = g(W_1^T \cdot X + W_{01})$ output for first layer

when $g(z) = \max(0, z)$

$y = W_2^T \cdot h_1 + W_{02}$ output of out layer using a linear function

EXAM TEST 18/01/2019

EXERCISE 1

1) $f : X \to Y$

is target Function which

$X = F \times NR \times NK$.

Choose by attribute of furniture as F, Nr room as NR, New Kitchen as NK

$Y = \{0, 1\} = \{Yes, No\}$

$F = \{Yes, No\}$

$NR = \{3, 4\}$

$NK = \{Yes, No\}$

NO CONSIDER NOT ACCEPTABLE

Yes Acceptable

$D = \{(x_n, t_n)_{n=1}^{5}\}$ dataset

x_n INPUT of dataset based of attribute, F, NR, NK of that specific n house and t_n is target value, so 0 OR 1 associated to that x_n

2)

You can choose attribute using INFORMATION gain, higher is this information but it is, so the attribute with the highest information gain is chose during each step of build a DECISION

tree.

Furthermore, Information gain is how good that particular attribute separed well the training data respect to a toget function

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{S} Entropy(S_v$$

$S_v = \{s \in S \mid A(s) = v\}$

Entropy is impurity of a collection of samples, in particular give a collection S, containing positive and negative sample of some toget function

$$Entropy(S) = - P_\oplus \log_2 P\oplus - P\ominus \log_2 P\ominus$$

$P\oplus$ = portion of POSITIVE samples

$P\ominus$ = portion of negative samples

Values (Fruit) = {Yes, No} $|S| = 5$

$S = [3+, 2-]$

this means that IN 3 time is acceptable so 3+ and 2 time IS NOT acceptable so 2-

$S_{FNo} = [2+, 3-]$ $S_{FYES} = [1+, 1-]$

VALUES (NR) = {3, 4}

$S_{NR3} = [1+, 2-]$ $S_{NR4} = [2, 0-]$

$NK) = \{Yes, No\}$

$= [1+, 0-]$ $\quad S_{NKNO} = [2+, 2-]$

$(S) = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right) = 0.9710$

$S, F) = 0.9710 - \frac{3}{5}\text{ENTropy}(S_{FNO}) - \frac{2}{5}\text{ENTropy}(S_{FYES})$

$= 0.9710 - 0.5509 - 0.4 = 0.0201$

$(S, NR) = +0.9710 - \frac{3}{5}\text{ENTropy}(S_{NR3}) - \frac{2}{5}\text{ENTropy}(S_{NR4})$

$= 0.9710 - 0.5460 = 0.425$

$\text{ain}(S, NK) = 0.9710 - \frac{1}{5}\text{Entropy}(S_{NKYES}) - \frac{4}{5}\text{ENTropy}(S_{NK\,NO})$

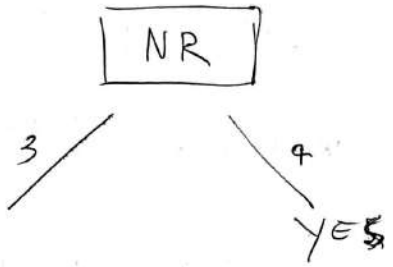
$= 0.9710 - 0 - 0.8 = 0.171.$

3) ID3

Examples are not all ~~positive~~ or all negative and attribute's list is not empty so we need to find attribute to assign to the root of tree, so to choose attribute we want to see the ~~gain~~ information gain and choose attribute with

highst information gain, in this case it is NR

so at first we have



Now when NR is equal to 3 so when $A(S) = 3$

we consider gain knowing 3

$|S_3| = 3$

$Gain(S_3, F) = Entropy(S_3) - \frac{1}{3} Entropy(S_{3FY}) - \frac{2}{3} Entropy(S_{3FN})$

$Entropy(S_3) = -\frac{1}{3}\log_2(\frac{1}{3}) - \frac{2}{3}\log_2(\frac{2}{3}) = 0.9183$

$S_3F_Y = [0, 1+]$

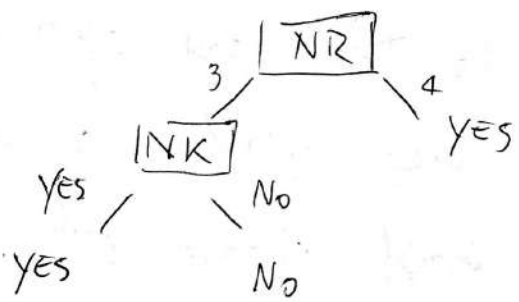
$S_3F_N = [2, -]$

$\Rightarrow Gain(S_3, F) = 0.9183 - 0 - 0 = 0.9183$

$Gain(S_3, NK) = 0.9183 - \frac{1}{3} Entropy(S_{3NKY}) - \frac{2}{3} Entropy(S_{3NKN})$

$= 0.9183 - 0 - 0 = 0.9183$

DOESN'T matter what you choose because they have highst Gain in both case



cise 2

1) We have a Dataset D and hypotesis space H
We want to compute probability distribution over H given D

$P(H|D)$ -> with Bayes

$$P(h|D) = \frac{P(D|h) \cdot P(h)}{P(D)} \quad \text{where } h \in H$$

where $P(h)$ prior probability of h
$P(D)$ prior probability of D
$P(D|h)$ is a probability of D given h, posterior probability of h

We want to find most probable hypothesis
and else we consider that $P(h_i) = P(h_j)$ with $i \neq j$

$$\Rightarrow h_{ML} = \arg\max_{h \in H} P(h|D)$$

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

$$= \arg\max P(D|h)$$

this because $P(D)$ and $P(h)$ are constant so we can not consider

2) No this statement is false because MAXIMUM LIKELIHOOD is NOT good method for ~~classification~~ is most probable classification of new istance this because: IF

For example we have

GIVEN NEW ISTANCE x we have

$h_1(x) = \oplus$ $\quad P(D|h_1) = 0.4$

$h_2(x) = \ominus$ $\quad P(D|h_2) = 0.3$

$h_3(x) = \ominus$ $\quad P(D|h_3) = 0.3$

IF we use MAXIMUM LIKELIHOOD we choose that the most probable classification for x is $\oplus$ because we DO argmax of $P(D|h_i)$ $i = 1,3$ and we obtain that is 0.4 so 0.4 compared to h_1 that is positive but this is not true, indeed x should be classify as $\ominus$ this because $\sum_{h_i \in H} P(\ominus|x,h_i) P(h_i|D) = 0.6$ and

$\sum_{h_i \in H} P(\oplus|x,h_i) P(h_i|D) = 0.4$

So the most probable classification for x is $\ominus$ and we can do using Bayes optimal classifier in which we can find optimal classification for new instance if we have

$$P(v_j|x,D) = \sum_{h_i \in H} P(v_j|x,h_i) P(h_i|D)$$

when probability of $h_i = v_j$ is indipendet from D given h_i and also doesn't depend of $x \in D \Rightarrow P(h_i|x,D) = P(h_i|D)$

3.1

In linear model we have target function $f \to X \to Y$

where $X \subseteq R^d$ and $Y = \{C_1, \ldots, C_K\}$

we assume that our dataset D is linearly seperable, this means that we can divide instance space in two regions in which differently classified instances are separated

We consider for example two classes

$y : X \to \{C_1, \ldots, C_K\}$

$y(x) = w^T x + w_0$

where w_0 is bias and w is weight vector

we have Dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$ and least square method so we need to find

$y(x) = \tilde{w}^T \tilde{x}$ where $\tilde{w} = (\tilde{w}_1, \ldots \tilde{w}_K]$

$\tilde{w}_K = \begin{pmatrix} \tilde{w}_{K0} \\ \tilde{w}_K \end{pmatrix}$

$\tilde{x} = \begin{pmatrix} 1 \\ x \end{pmatrix}$

So our aim is to MINIMIZE SUM-OF-SQUARES error

$$E(\tilde{w}) = \frac{1}{2} Tr\left\{(\tilde{X}\tilde{W} - T)^T (\tilde{X}\tilde{W} - T)\right\}$$

we obtain

$$\tilde{W} = (\tilde{X}^+ \tilde{X})^{-1} \tilde{X}^T . T \Rightarrow y(x) = T^T\left((\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T\right)$$

where this is a closed-form solution

this method has some problem with outliers, in p
when there are outlier the decision boundary could
be in a position to divide perfectly two classes in different
region. This because SUM-OF-SQUARE error is a sum
so also outlier give a contribution and when we want
to MINIMIZE it to find a $y(x)$, so then find a
decision boundary, we obtain wrong division of our instance
space and some instances are on decision boundary and
you cannot classify or in particular we do misclassify.

Ex 4

1) $a = (XX^T + \lambda I)^{-1} +$

$K = XX^T$ is a gram matrix

$$K = K(x_n, x_m^*) = \begin{bmatrix} K(x_1, x_1) & \dots & K(x_1, x_N) \\ \vdots & & \\ K(x_N, x_1) & \dots & K(x_N, x_N) \end{bmatrix}$$

is a $N \times N$ matrix

2) Linear model for Regression

$y = w^T x$ and $D = \{(x_n, t_n)\}_{n=1}^N$ We have that we
want to MINIMIZE loss function

$$J(w) = \sum_{n=1}^{N} E(y_n, t_n) + \lambda \|w\|^2 \qquad \text{where } y_n = w^T x_n$$

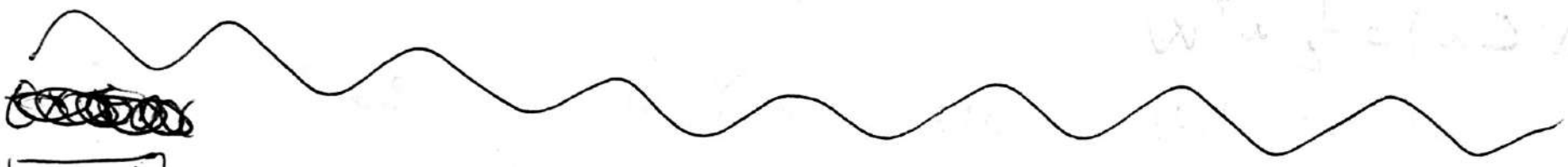
So we consider $E(y_n, t_n) = (y_n - t_n)^2$

$$\Rightarrow \tilde{w} = (X^T X + \lambda I)^{-1} X^T t = X^T \alpha$$

$$\Rightarrow y(x; w) = \sum_{n=1}^{N} \hat{\alpha}_n x_n^T x$$

We apply kernel trick, that consist to substitute inner product $x_n^T x$ with kernel function

$$\Rightarrow y(x, w) = \sum_{n=1}^{N} \hat{\alpha}_n K(x_n, x)$$



ex 5

1) We can perform regression based of some basis functions for example we know polynomial, Radial, Sigmoidal function and we want to find one of them, that is useful for this problem.

For this case we can use polynomial basis function of degree 3

$$y(x) = w_1 x + w_2 x^2 + w_3 x^3 + w_0 = w x \qquad w = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix}$$

2) For linear regression we can avoid overfitting using regularization. In particular in linear regression we want to maximize likelihood, so minimize the error

$$\min E_D(w) = \frac{1}{2} \sum_{n=1}^{N} (t_n - y_n)^2 = \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T \phi(x_n)$$

Regularization technique we have

$$\min E_D(w) + \lambda E_W(w)$$

where λ is regularization factor $\lambda > 0$

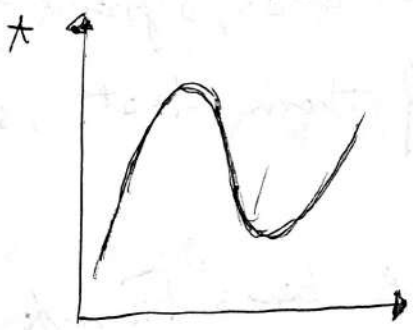
we can choose two different choice for $E_W(w)$ that is regularization term

(1) $E_W(w) = \frac{1}{2} w^T w$

or

(2) $E_W(w) = \sum_{j=0}^{M-1} |w_j|^q$

3)



$$\min E_D(w) + \frac{\lambda}{2} w^T w$$

EX6

1) Classificator of K nearest neighbors is an algorithm used for NON-parametric model, in particular we chose K nearest neighbor and that for model a classification of a new instance x and so we have

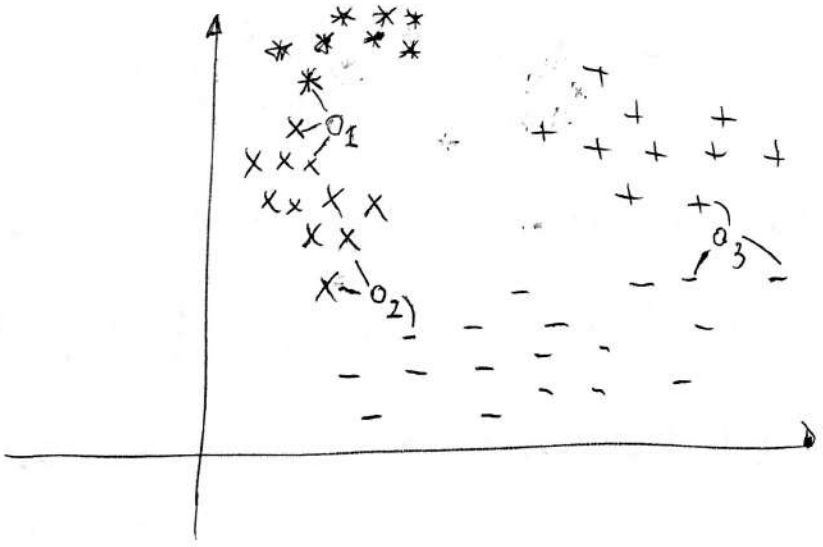
$S = \{(x_n, t_n)\}_{n=1}^{N}$ and $f: X \to C$ target function.

First we chose K nearest neighbors of new instance x, then we assign to x the most common value/label among majority of its K nearest neighbors chosen.

2)

(*, x, +, -) $K = 3$

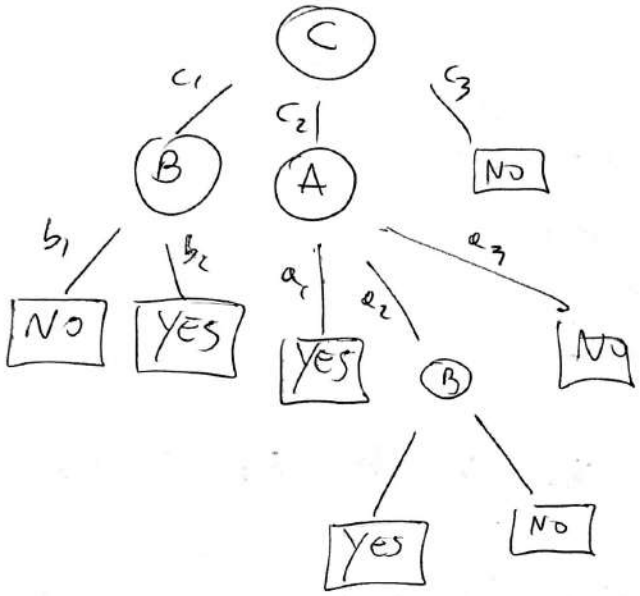
Assume that our instances are "o" that we want to classify, in particular (o_1, o_2, o_3)



o_1 is classify as "x"
o_2 is classify as "x"
o_3 is classify as "-"

18/01/2019

EX1



1)

IF $C = c_3$ THEN No

IF $(C = c_1) \wedge (B = b_2)$ THEN ~~Yes~~

IF $(C = c_1) \wedge (B = b_1)$ THEN NO

IF $(C = c_2) \wedge (A = a_1)$ THEN YES

IF $(C = c_2) \wedge (A = a_2) \wedge (B = b_1)$ THEN YES

IF $(C = c_2) \wedge (A = a_2) \wedge (B = b_2)$ THEN NO

IF $(C = c_2) \wedge (A = a_3)$ THEN NO

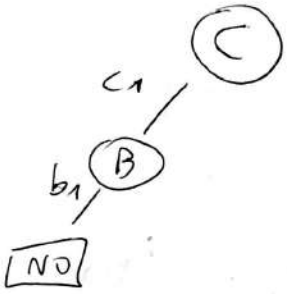
2) $S = \{S_1 = \langle a_1, b_1, c_1, NO \rangle, S_2 = \langle a_2, b_1, c_2, Yes \rangle,$

$S_3 = \langle a_1, b_2, c_3, YES \rangle, S_4 = \langle a_2, b_2, c_2, Yes \rangle\}$

We consider $S_1 = \langle a_1, b_1, c_1, No \rangle$

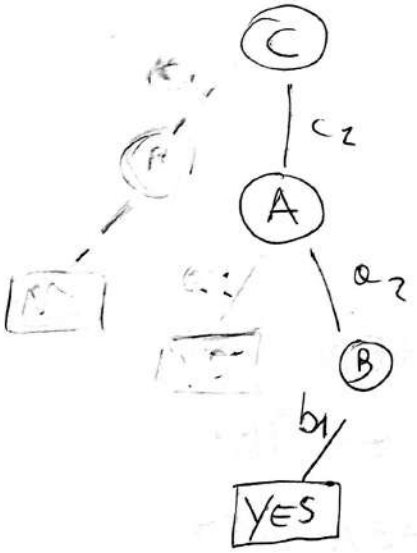
$(C = c_1) \wedge (B = b_1) \wedge (A = A_1)$

we can see that



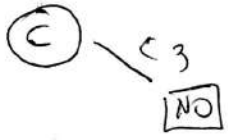
give us No value so with S_1 T is consistent

For $S_2 = \langle a_2, b_1, c_2, YES \rangle$



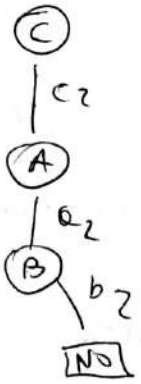
IT'S CONSISTENT With this set S_2

For $S_3 = \langle a_1, b_2, c_3, YES \rangle$



IT IS NOT CONSISTENT because IN S_3 We couple Yes as output but IN OUR T We have No value at the end

or $S_4 = \langle a_2, b_2, c_1, YES \rangle$



Also in this case we some problem because in set S_4 It said to have output yes but in our Tree we have No

⇒ We on conclude that T is consistent for S_1 and S_2 and NOT CONSISTENT for S_3 and S_4

EX2

1) We have Dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$, x_n ~~eat~~ input and t_n value of target function associated to that particular x_n. We have hypothesis $h \in H$ (Hypothesis space, so for bayes

$$P(h|D) = \frac{P(D|h) \cdot P(h)}{P(D)}$$

Maximum a posteriori ~~are~~ compute most probable hypothesis so we need to maximize $P(h|D)$

$$h_{MAP} = \arg\max_{h \in H} P(h|D) = \arg\max_{h \in H} P(D|h) \cdot P(h)$$

We DROP $P(D)$ because is a costant and h does depend on it.

IF we also consider $P(h_i) = P(h_j)$ with $i \neq j$

We compute MAXIMUM LIKELIHOOD

$$h_{ML} = \arg\max_{h \in H} P(h|D) = \arg\max P(D|h)$$

we drop $P(h)$ because is a costant for that particular hypothesis

2)

Bayes Optimal classifier is a classifier that Find most probable classification for a new instance

In particular, given a dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$ a target function $f: X \to V$, $V = \{v_1, \ldots, v_K\}$ and $x \notin D$ We have that a probability that x is classify for a particular v_j is

$$P(v_j|x, D) = \sum_{h_i \in H} P(v_j|x, h_i) P(h_i|D)$$

Where

$P(v_j|x, h_i)$ means that probability that $h_i = v_j$ is indipendent from D so $P(v_j|x, h_i D) = P(v_j|x, h_i)$

d else h_i doesn't depend on $x \notin D$

$$P(h_i | x, D) = P(h_i | D)$$

this mean that Bayes Optimal classifier is that

$$v_{OB} = \underset{v_j \in V}{\operatorname{argmax}} \sum P(v_j | x, h_i) P(h_i, D)$$

We maximize probability that new instance is classify correctly

Bayes optimal classifier find optimal value because no other classifier use same hypothesis space and same prior knowledge

3) Bayes Optimal classifier has problem that for Hypothesis space that are very large is very difficult To know prior probability of all hypotesis belong to that hypothesis space, so is powerful for Hypothesis space that are small enough.

EX3

1) Goal of linear regression is to find a relationship in which predictor y and input x are well defined, or a kind of curve that can represent

all point of class in a good way.
Define a model $y(x, w)$ with parameter w to approximate target f
Linear regression function

$$y(x,w) = w_0 + w_1 x_1 + \dots + w_d x_d = w^T x$$

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

2)

$D = \{(x_1, t_1), \dots, (x_N, t_N)\}$ x_n input value, t_n target values

To Find parameter w

we need to maximize maximum likelihood but

in particular we use feature space to

represent linear basis function model

$$y(x; w) = \sum_{j=0}^{M-1} w_j \cdot \phi_j(x) = w^T \phi(x)$$

$$w = \begin{bmatrix} w_0 \\ \vdots \\ w_{M-1} \end{bmatrix} \quad \phi(x) = \begin{bmatrix} \phi_0(x) \\ \vdots \\ \phi_{M-1}(x) \end{bmatrix} \quad \phi_0(x) = 1$$

the target value t is given by $y(x; w)$
affected by noise ε so

$$t = y(x; w) + \varepsilon$$

assume Gaussian noise

$$N(\varepsilon \mid 0, \beta^{-1}) = P(\varepsilon \mid \beta)$$

β is precision

$$P(t \mid x, w, \beta) = N(t \mid y(x; w), \beta^{-1})$$

Likelihood function is

$$P((t_1, \dots, t_n) \mid (x_1, \dots, x_n), w, \beta) = \prod_{n=1}^{N} N(t_n \mid w^T \phi(x_n), \beta^{-1})$$

MAXIMIZE likelihood means minimize error so we need to MINIMIZE likelihood, we compute $\ln$ at first

$$\ln\left(P((t_1, \dots, t_n) \mid (x_1, \dots, x_n), w, \beta)\right) =$$

$\Rightarrow$ we obtain $-\beta E_D - \frac{N}{2} \ln(2\pi\beta^{-1})$

$$E_D = \frac{1}{2} \sum_{n=1}^{N} (t - w^T \phi(x_n))^2$$

we need to minimize error

$$\min E_D = \min\left(\frac{1}{2} \sum_{n=1}^{N} (t - w^T \phi(x_n))^2\right)$$

$$= \min\left(\frac{1}{2} \sum_{n=1}^{N} (t - \phi w)^T (t - \phi w)\right)$$

optimal condition is when $\boxed{\nabla E_D = 0 \Leftrightarrow \phi^T \phi w = \phi^T t}$

where $\phi = \begin{bmatrix} \phi_1(x_1) & \dots & \phi_{M-1}(x_1) \\ \vdots & & \\ \phi_N(x_N) & \dots & \phi_{M-1}(x_N) \end{bmatrix}$

$$W_{ML} = \left[(\Phi^T \Phi)^{-1} \Phi^T \cdot t \right] = \Phi^+ t$$

For sequential learning, we can say that we use, when data set is sufficiently large so we obtain sequential learning using stochastic gradient descendent

$$\hat{w} \leftarrow \hat{w} + \eta \nabla E_n$$

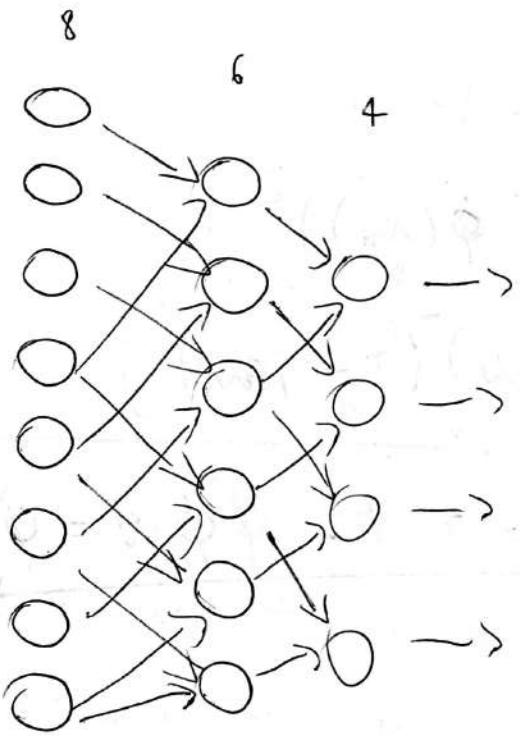
η is learning rate

so $\hat{w} \leftarrow \hat{w} - \eta \left[t_n - \hat{w}^T \cdot \phi(x_n) \right] \phi(x_n)$

And algorithm converge for small values of η

EX4

(1)



N° parameter = $(6 \times 8) + (6 \times 4) + (6 + 4)$

= 82

2) Backpropagation is an algorithm that only compute gradient, it is NOT a learning algorithm, for example a SGD (Stochastic Gradient Descendent) with Nementum suffer of problem of local minima because we we compute the gradient and then velocity, given by Nementum, we can change direction and get stuck in local MINIMA

3) No, Backpropagation is not effected by overfitting because is not a model or a learny algorithm so it doesn't have problem of overfitting, it just compute gradient

ex 5

1)

We ave 3 classes IN figure and you can separate them with a line, curve line because their are aggregated IN some speaple area so they are NOT mixed together, this is reason we can separate them in 3 different region of our figure

2) We can choose different type of kernel but we are consider SVM so for classification is useful to choose linear kernel because in SVM we have $w = \sum_{n=1}^{N} \alpha_n x_n$

$$\text{sign}(y(x; \alpha)) = \text{sign}\left(\sum_{n=1}^{N} \alpha_n x_n^T x_i + w_0\right)$$

So we can replace kernel function if we have inner product $x^T x$

$$\Rightarrow y(x; \alpha) = \text{sign}\left(\sum_{n=1}^{N} \alpha_n K(x_n, x_i) + w_0\right)$$

3) For SVM For classification we have

$$w = \sum_{k=1}^{K} \sum_{n=1}^{N} \alpha_{kn} t_{kn} x_{kn}$$ where K are k classes

And to find w_0 we need to use Lagrang method

$$\tilde{L}(\alpha) = \sum_{k=1}^{K} \sum_{n=1}^{N} \alpha_{kn} - \frac{1}{2} \sum_{k=1}^{K} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_{kn} \alpha_{km} t_{kn} t_{km} K(x_{kn}, x_{km})$$

So we can consider $w+n$ multiplies α_{kn} of lagrange

$$\hat{w}_k = \sum_{k=1}^{K} \sum_{n=1}^{N} \alpha_{kn} t_{nk} x_{kn}$$

and consider support vector we can have

$$w_0 = \frac{1}{|SV|} \sum_{x_i \in SV} \left(t_i - \sum_{k=1}^{K} \sum_{x_j \in SV} \alpha_{kj} t_{kj} x_{ki}^T x_{kj}\right)$$

26

Perceptron model is a linear classification model that ~~[illegible]~~ instance, we use we consider to have we want to classify new we have FUNCTION

$$O(x) = \begin{cases} 1 & \text{IF } w^T x > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$O = w_1 x_1 + \dots + w_d x_d = w^T x$$

Now we consider a dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$ and we want to MINIMIZE error where sum-of-squares error is

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (t_n - o_n)^2 = \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T x_n)^2$$

MINIMIZE means do derivative respect to w of $E(w)$

$$\frac{\partial E(w)}{\partial w_i} = \sum_{n=1}^{n} (t_n - o_n)(-x_{i,n})$$

After this the training rule of percept is to update weights in such to find a decision border that can divide instance space in two ~~[illegible]~~ where there are classes with different classification values that are separated.

regions where in one should be a class and ~~[illegible]~~ in the other other class with different classification value

IF and only IF Dataset is linearly separable.

So updating weights, that perceptron do when it find a instance that is misclassify, model will converge (IFF Dataset is linearly separable) in a FINITE number of steps

$W_i^{New} \leftarrow w_i^{old} + \Delta w$

$$\Delta w = -\eta \nabla E = -\eta \sum_{n=1}^{N} (t_n - o_n)(x_{i,n}) = \eta \sum_{n=1}^{N} (t_n - w^T x_n)(x_{i,n})$$

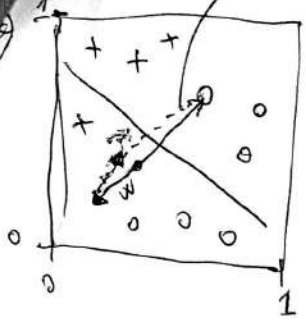
$$= \eta \sum_{n=1}^{N} (t_n - \text{sign}(w^T x_n))\, x_{i,n}$$

η is learning rate

Perceptron algorithm has aim to find a vector that can perfectly classify inputs in our dat and it garanteed to find solution in a FINITE number of step IF Dataset D is linearly seperable but after model converge IF initial values that we have to choose for w are good, but you choose them arbitrary so you don't know very well

2)

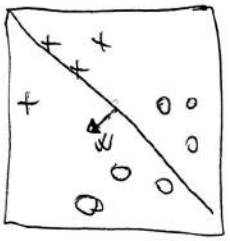
a)



We FIND a INSTANCE THAT IS MISCLASSIFY BECAUSE ON TOP we CHOOSE TO CLASSIFY INSTANCE OF CLASS "+" and on bottom INSTANCE OF CLASS "-"

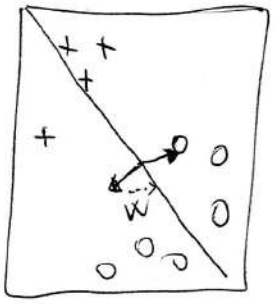
THIS MEANS THAT we need to update weight and w change

b)

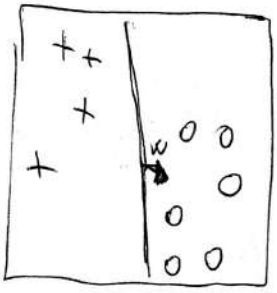


w is smaller than in the previous picture this because arrow that go to MISCLASSIFY point we proced to w to FIND lenght of vector w with update weights

c)



we find another point that is misclassify so we need to update weight and w change and also decision boundary



at the end we don't reach any points that is misclassified so we converge, also because our dataset is linearly seperable and we don't need to update weights anymore

12/02/2019

Ex-1

1) Reinforcement learning problem we have a dataset $D = \{x_1, a_1, r_1, \dots, x_n, a_n, x_n\}$

We want to maximize reward so we need to find optimal policy to obtain this

In Reinforcement learning we have only inputs but the input is a collection of state, action and reward and the output is not available.

2) Q-learning is an algorithm used to find optimal policy when transition function (δ) and reward function (r) are NOT known, so we need to do an approximate with Q-learning whose training rule of this algorithm is

$$\hat{Q}(x,a) \leftarrow \bar{r} + \gamma \max(\hat{Q}(x',a'))$$

where $\bar{r}$ is the immediate reward, $\hat{Q}$ is the approximation function of Q, γ is scaling factor and optimal policy is $\pi^* = \underset{a \in A}{\operatorname{argmax}} \hat{Q}(x,a)$

the algorithm is:

We initialize $\hat{Q}_{(0)}(x,a) \leftarrow 0$

then we observe current state x

for $t = 1, \dots, T$ (until terminal condition) do

choose action a

execute action a
observe resulting state x'
collect immediate reward r
update table entry of $\hat{Q}(x,a)$

$$\hat{Q}_{(t)}(x,a) \leftarrow r + \gamma \max_{a'} \left(\hat{Q}_{(t-1)}(x', a') \right)$$

$x \leftarrow x'$

end for

$$\pi^* = \arg\max_{a \in A} \hat{Q}_{(T)}(x,a)$$

We need to execute action and collect reward to find value of $\hat{Q}(x,a)$, but we notice how this learning doesn't need transition function and reward function to execute then optimal policy

EX2

CNN is a convolutional neural network, this means that use convolutional function between input and kernel. To Avoid overfitting in CNN we use parameter sharing, a method in which all unit of that layer share a subset of parameter in this way we don't need to learn more parameter and this also mean that a particular parameter can be used more than one time for a function. Another methods most common is Dropout, in which we eliminate randomly UNITS of network with a probability α

EX3

In SVM we want to maximize margin to provide better accuracy.

Margin is smallest distance between decision boundary and closest points of it (support vectors)

the main idea is to find w^* and w^* maximize the margin, in particular we have a dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$ $t_n \in \{-1, 1\}$

$y = w^T x + w_0$

$$\text{Margin is} = \min_{n=1,\dots,N} \frac{|y(x_n)|}{\|w\|}$$

We know that $y(x_n) \geq 0$ if $t_n = 1$

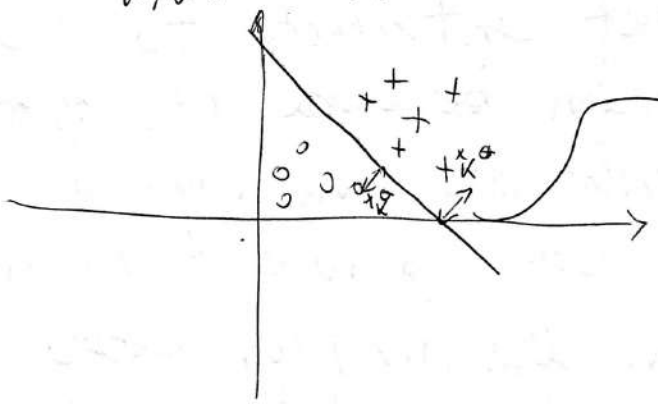
$y(x_n) \leq 0$ if $t_n = -1$

$\Rightarrow t_n y(x_n) \geq 0$

So $|y(x)| = t_n\, y(x_n)$

$$\text{Margin} \overset{!}{=} \min_{n=1,\dots,N} \frac{t_n\, y(x_n)}{\|w\|} = \frac{1}{\|w\|} \min_{n=1,\dots,N} t_n \cdot (w^T x_n + w_0)$$

$$w^*, w_0^* = \frac{1}{\|w\|} \arg\max_{w, w_0} \left(\min_{n=1,\dots,N} \left(t_n (w^T x_n + w_0) \right) \right)$$



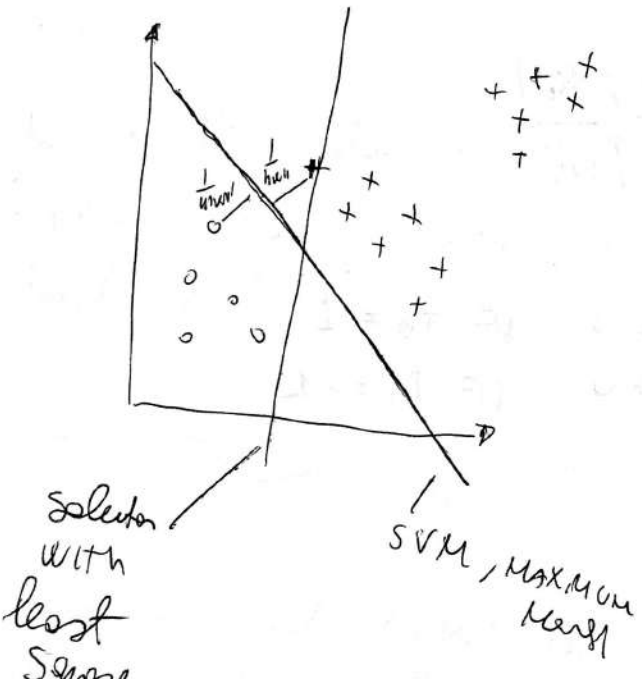
margin that we maximize

Perfect we notice that We have 2 closest point that are at some distance

in where one $x_k^{\oplus}$ $w^{*T} x_k^{\oplus} + w_0^{*} = 1$
(point in the Top.)

and other point $x_k^{\ominus}$ when $w^{*T} x_k^{\ominus} + w_0^{*} = -1$
distance between these two points from Hyperplane (decison boundary) is $\frac{1}{\|w\|}$

2)



Solution with least Square

SVM, Maximum Margin

3)

We can notice how least square is not good in this problem because it is sensitive to outliers, this because it's error is a function that is sum, so every points in dataset contribute to the solution but this create problem because it's decision boundary is not in right position and don't classify better solution, instead SVM is robust to the outliers so it better solution else in this case

Furthermore Maximum Margin solution is preferd because

is a method that create a good separation for classification of instances in this way we don't have risk ~~on instance is in~~ absent classification of instance in wrong way because we have a sufficient margin that can provide to classify better close instance near margin, this not happen in perceptron because we can have decision borders that has margin that is very small and so there is a risk to classify instance not in good way.

EX 4

1) Confusion matrix is a performance metric used for multiclass problem and it show how many time instance of class C_i is classified class C_j.
In the diagonal we have the accuracy of each class and outside it we have errors

2)

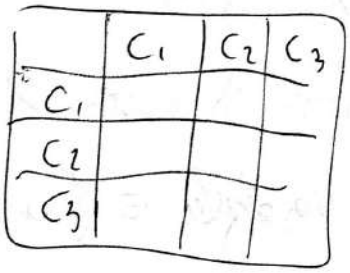
	C_1	C_2	C_3	
C_1	60	20	20	40
C_2	10	70	20	30
C_3	20	15	65	30
FP	70	35	40	195 = TP + TN

FN

ABSOLUTE VALUES

	C_1	C_2	C_3
C_1	0.6	0.2	0.2
C_2	0.1	0.7	0.2
C_3	0.2	0.15	0.65

PERCENTAGE VALUES



confusion matrix of 3 classes

3)

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} = 0.65 \Rightarrow 65\% \text{ accuracy}$$

TP = true positive = 195

TN = true Negative = 195

FP = false positive = 105

FN = false Negative = 105

EX 5

1) GMM

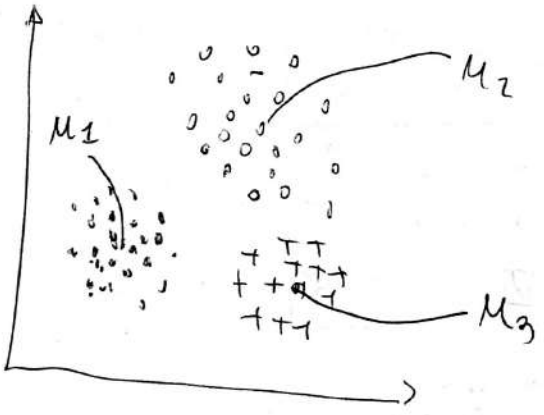
We have Dataset $D = \{x_n\}$, we are IN unsupervised learning and K is K different gaussian distribution

$$P(X) = \sum_{K=1}^{K} \pi_K \, N(x; \mu_K, \Sigma_K)$$

μ_K is mean, Σ_K is covariance

$P(z_K = 1) = \pi_K$ where z_K is a variable $\in \{0,1\}$

2)



Matrix covariance dim 2x2

MEAN has dim 2

but we have 3 clusters so, we obtain to have

$(2x2+2) \cdot 3 + 3 = 21$

(3 because we have 3 clusters)

- $2x2$ → MATRIX COVARIANCE
- $+2$ → MEAN of dim 2
- $\cdot 3$ → because we have 3 clusters
- $+3$ → 3 Π that we have to consider ("weights")

EX6

KNN is an algorithm for classification of non parametric model in which we have $f: X \to C$ and dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$

this algorithm has two steps:

1) Choose K nearest neighbors to new instance $x \notin D$
2) Assign to x most common value among the majority of neighbors

likelihood function

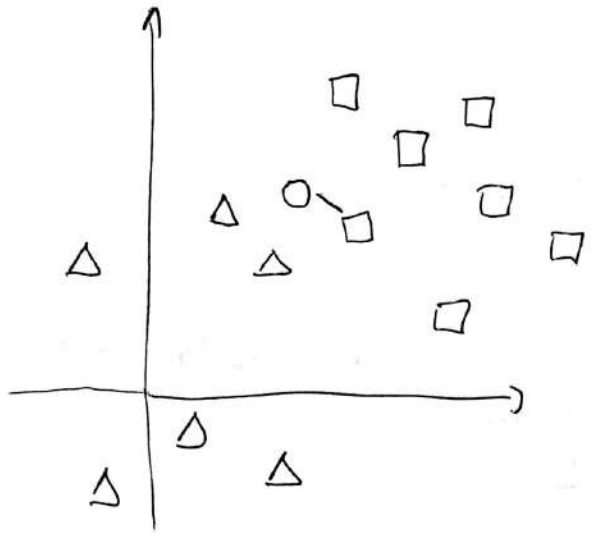
$$P(c \mid x, K, D) = \frac{1}{K} \sum_{i \in NK(x,D)} I(t_i = c)$$

when $N_K(x, D)$ are nearest K neighbor

and

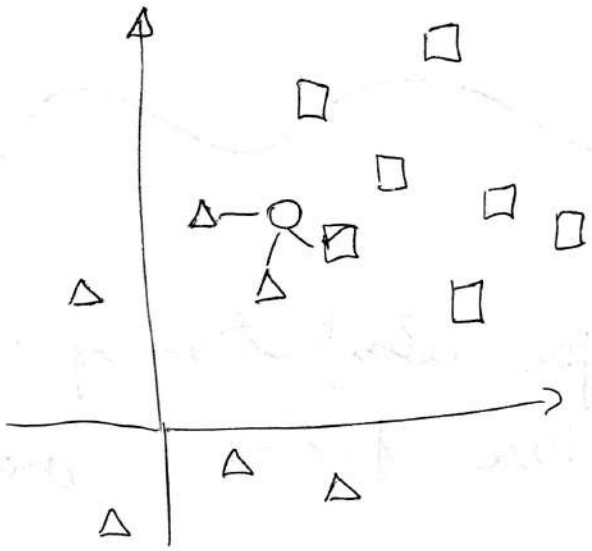
$$I(e) = \begin{cases} 1 & \text{IF } e \text{ is true} \\ 0 & \text{IF } e \text{ is false} \end{cases}$$

2) ■ K=1



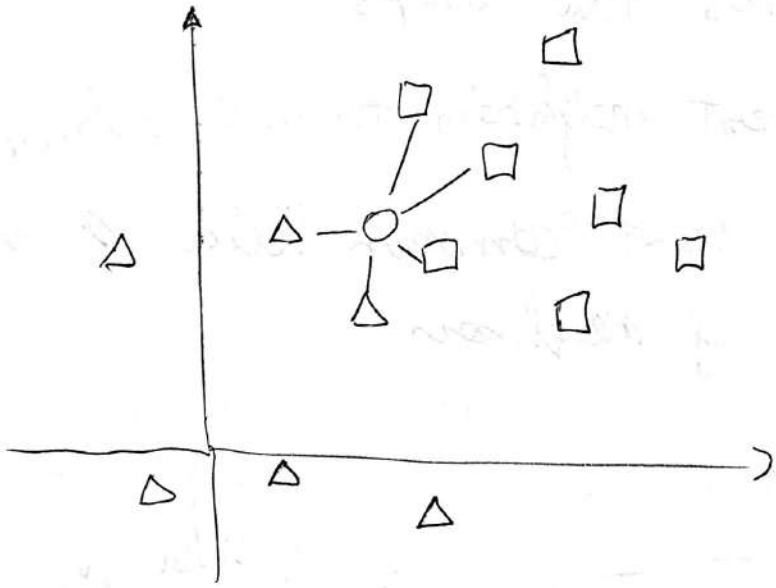
1- nearest neighbor is "□" this means that O should be classified as "□" For K=1

■ K=3



IN this one 3- nearest neighbors are 2 △ and 1 □ so we can say that O is classified "△" For K=3

■ K=5



IN this one 5- nearest neighbors are 3 "□" and 2 "△" so the new instance "O" is classified "□" for K=5

EXAM TEST 2

EX1

(a)

1) $a = (XX^T + \lambda I)^{-1} t$

$K = XX^T$ gram matrix

$$K = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_N, x_1) \\ \vdots & & \vdots \\ k(x_1, x_N) & \cdots & k(x_N, x_N) \end{pmatrix}$$

2) For linear regression

$$J(\theta) = \frac{1}{2} \sum_{n=1}^{N} E(y_n, t_n) + \lambda \|w\|^2$$

We want to minimize this so we have $E(y_n, t_n) = (y_n - t_n)^2$ and in closed form solution we have

$$w = \sum_{n=1}^{N} a_n x_n$$

So $y(x) = w^T x = \sum_{n=1}^{N} a_n x_n^T \cdot x$

We can use kernel trick to obtain Kernelized version for linear regression

$$y(x) = \sum_{n=1}^{N} a_n k(x_n, x)$$

EX2

1)

INPUT Vector DIM 128
output Vocter DIM 10
HIDDEN layer DIM 50 UNIT

dimenon of W_1 IS 128X50 and dimena of W_2 IS 50X10

2)

We hove hidden layer that use relu funcher, this is means that

$$h = g(x^T W_1 + c)$$

Whee $g(a) = max(0, a)$ and it is ReLU activation functi
for output we have linar funcion

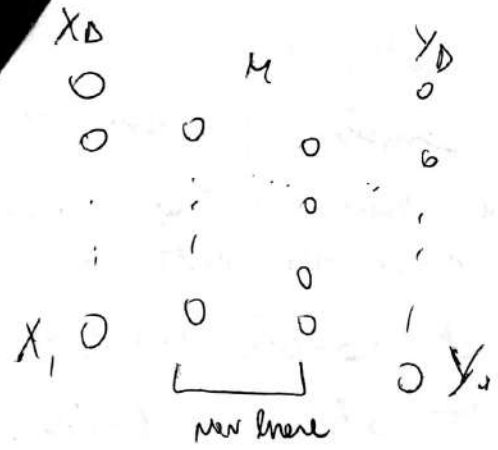
$$y = W_2^T h + b$$

$$y = W_2^T \cdot max(0, x^T W_1 + c) + b$$

EX3

Autoencoder is a neural network that ~~with~~ reduced size of hidden layer, which learn to reconstruct their inputs by MINIMize sum-of-squared error

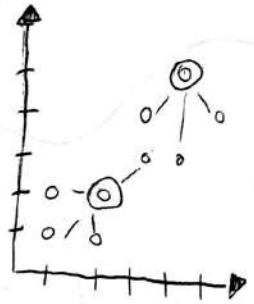
In pertcular input and output hose some dimenon that we consider D, and hidden layca M dimens with $M < D$



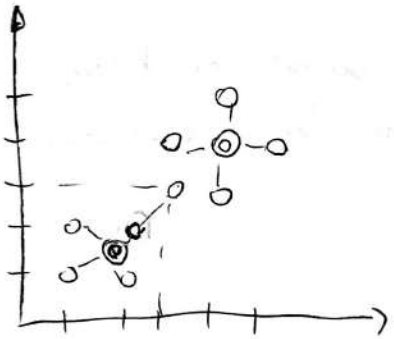
input and output are linear and hidden layer are non linear

this means that we have, if we create 3 D of input layer and 1 D of hidden layer we have function F_1 that maps 2D to R and function F_2 that maps M to D and if F_1 is not linear the output cannot be planar

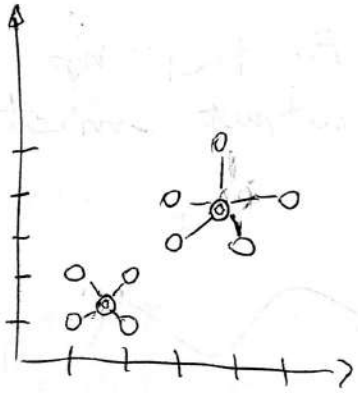
EX 4



In K-mean we need to find K means of K cluster, in this case we have 2 cluster
I draw in the picture samples of each cluster with specific centroids of the two cluster, now we can understand that we need to update our centroid because the cluster on the top of the picture doesn't have centroid in ~~good~~ right position and also centroid in the bottom



After computing centroids we have each samples of them IN sequence and we check if they are in the cluster with Neoast centroid. We notice that sample in (3, 3) position doesn't stay in cluster with nearest centroid, so we need to switch this centroid with the other and update centroids of two clusters involved



Now we can see that there is no more sample that is not in the cluster with nearest centroid. K-mean converge, with K = 2

Ex 5

1) KNN algorithm for classification is used for non parametric model

We have target function $f: X \to C$ and dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$

this algorithm is compose by two steps

(1) Find K neighbors of new instance x

(2) Assign to x most common value among the majority of neighbors

elihood function

$$P(c \mid x, K, D) = \frac{1}{K} \sum_{N_K(x,D)} I(t_i = c)$$

Where $N_K(x,D)$ is K nearest neighbor of x

$$I(e) = \begin{cases} 1 & \text{IF } e \text{ is true} \\ 0 & \text{IF } e \text{ is false} \end{cases}$$

2)

We can say that we choose $K = 3$ nearest neighbor of o_1, o_2, o_3

For o_1 we have 2 "+" and 1 "*" so this means that we classify o_1 as "+"

For o_2 we have 2 "x" and "+" so we classified o_2 as "x"

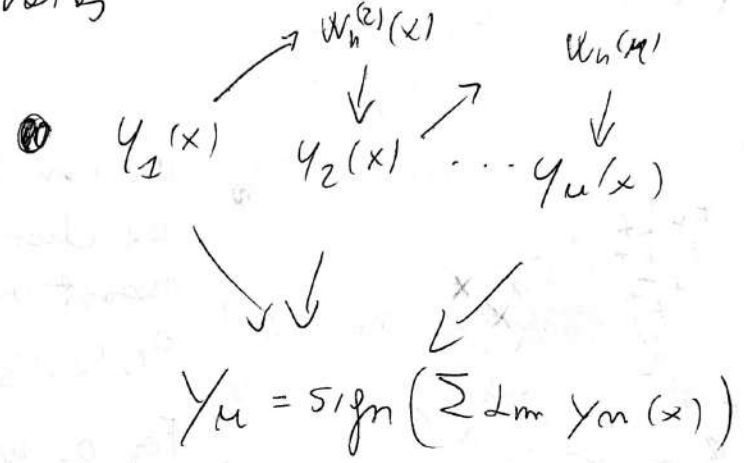
the last one o_3, in this case we have 2 "-" and 1 "x" so o_3 should be classified "-"

EX 6

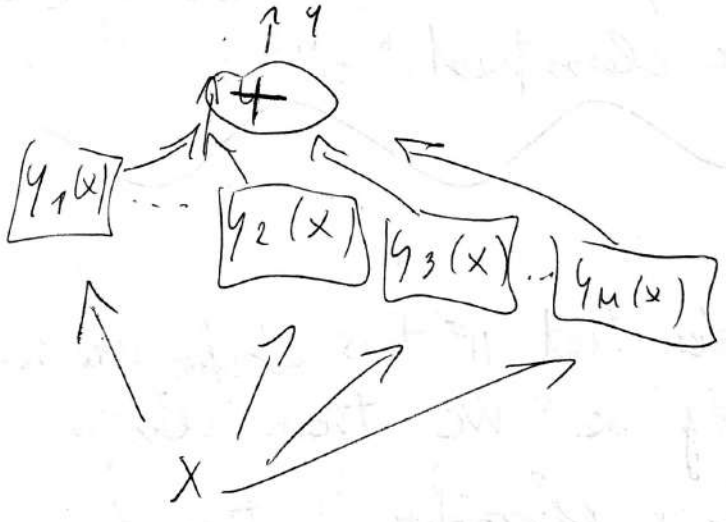
We need to consider that IF this classifier are not trained yet we can use boosting so We train classifier in sequence ~~this classifier~~ Each classifier is trained on weighted data.

Weighted data depends on performance of the previous classifier, so IF POINT IS misclassified we have high weight

then the prediction of instance are based on the majority of votes



Instead IF our classifier are already trained we consider to use method for multiplier learner such as voting in which the approach is to consider that each learner gives a result and all these results will be used to compute final output.



In Mixture of expert these classifier should be specific because gating model, which look at the input will choose classifier that is a responsible of final output But usually this depend, so we use these multiple learner method to understand and combine results of classifiers rather than considering only just one

18/01/2019 (B)

EX 1

1) Yes data are separable because exist a line that can divide 3 classes IN 3 different part of R region.

Data sets is not linearly separable but we can separate these classes

2) We can use RBF function or polynomel of degree three or Kernel function both cases should be right

$$K(x_i, x') = \varphi(\beta x^T x' + \gamma)^d \quad d = \{2, \dots\}$$ CASE OF POLYNOMIAL

$$K(x, x') = \exp(-\beta \|x - x'\|^2)$$ CASE OF RBF

3) IF We apply SVM for clasificator of multple classes We use Lagrange to obtain value of w^* and w_0^* that We obtain maximize margin

For K-classes

$$W^* = \sum_{n=1}^{N} \sum_{k=1}^{K} a_{kn} t_{kn} x_{kn}$$

$$SV = \{x_k \in X_D \mid t_k y(x_k) = 1\}$$

$$y(x) = sign(W^{*T} x + w_0^*)$$

and

$$w_0^* = \frac{1}{|SV|} \sum_{x_i \in SV} \sum_{k=1}^{K} \left(t_i - \sum_{x_j \in S} a_{kj} t_{kj} x_{kj}^T x_{kj} \right)$$

EX 2

Perceptron is a model for classification that want to find good separation to classify instance.
In particular perceptron works with function $\sigma(x)$
where $\sigma(x) = \begin{cases} 1 & \text{if } w^T x > 0 \\ -1 & \text{otherwise} \end{cases}$ that we know

$y = w^T x$ so we want to find w to have good separation of our classes.

Given a dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$ we want to MINIMIZE sum of squared error

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (t_n - o_n)^2 = \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T x_n)^2$$

Need to compute derivative respect to w

$$\frac{\partial E(w)}{\partial w_i} = -\sum_{n=1}^{N} (t_n - w^T x_n)(x_{i,n})$$

For this the main concept of perceptron is to update weights at each iteration with this training rule

$$W \leftarrow W + \Delta w_i$$

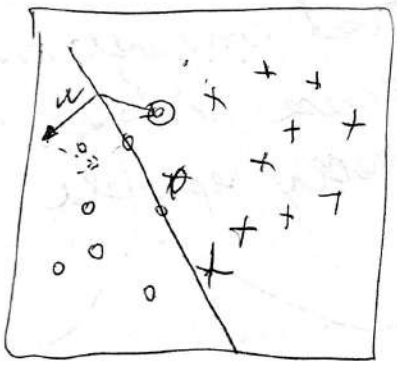
$$\Delta w_i = -\eta \nabla E = +\eta \sum_{n=1}^{N} (t_n - w^T x_n)$$

η is learning rate

Perceptron update weights when it find an instance that is misclassified, so it update weights and the decision boundary changes, we move this until termination condition.

In pocket Perceptron converge after a FINITE number of steps IF dataset is linearly separable otherwise it will never converge.

2) a)



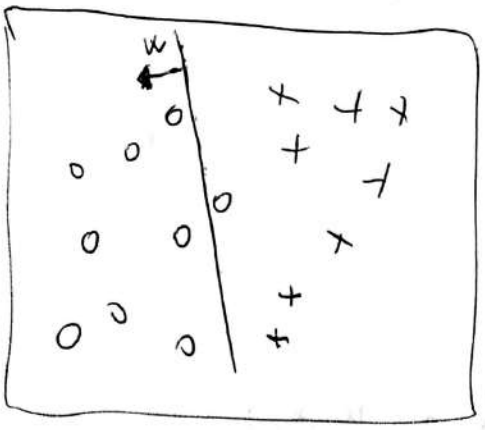
We choose w randomly, then we see at each iteration IF there is an instance that is misclassified

On the bottom we have class "o" and on top class "+"

We can see that we find an instance of class "o" over the decision boundary, we need to

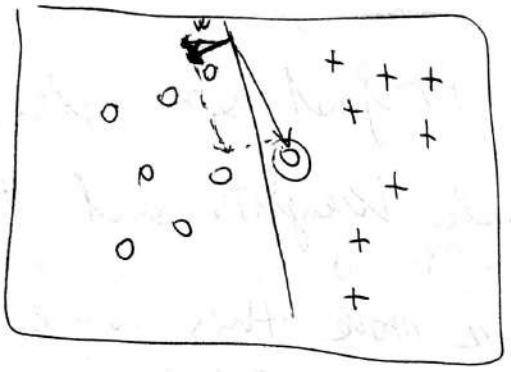
update our weights

b)



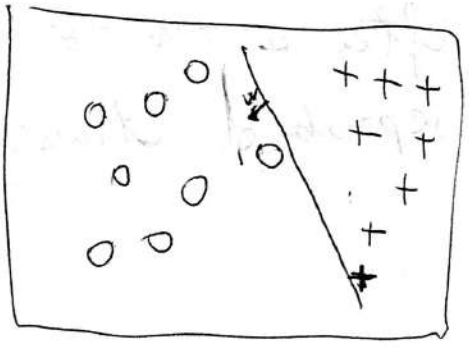
w is smaller than before because $\Delta w_i = -\eta \nabla E$ reduce value of w

c)



We find another instance that is missclassified so we want to update weights again and thus decision bounday change

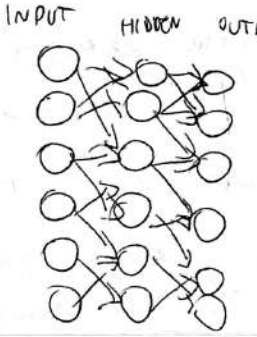
d)



w is reduced again but we can see now that there is no instance that is MISS classify so method converge and this also because dataset is linearly separable

Ex3

1)



regression problem we have output linear so activate function of output is linear and this means that loss function is mean squared error.

Furthermore for hidden layer we can choose Relu function because it's similar of linear function so we prefer for this model

N° parameters $(6 \times 6) + (6 \times 6) + (6 + 6) = 36 + 36 + 12$
$= 84$

2) No because Backpropagation is algorithm that only compute the gradient is NOT an algorithm the is a traing algorithm so it doesn't suffer of local MINIMA, stochastic gradient also not suffer of local MINIMA

3) No because Backpropagation IS NOT a traing algorithm so it doesn't suffer problem of over fitting, because it only propage gradient value From loss function, computed IN Forward pass to the INput in the backward pass.

EX 4)

1) Linear regression describe linear relationship between input variable and output variable, in particular learning function $f: X \to Y$, where $X \subseteq R^d$ and $Y = R$, given $D = \{x_n, t_n\}_{n=1}^{N}$

$$y = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_d x_d = w^T x$$

$$w = \begin{pmatrix} w_0 \\ w_d \end{pmatrix} \qquad x = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{pmatrix}$$

the goal is to ~~predict~~ define a model that can be used to predict the values of y depend of x

2) In linear regression we can use also non linear function se

$$y = \sum_{i=0}^{M-1} w_i^T \cdot \phi_i(x_n) = w^T \phi(x)$$

$$w = \begin{pmatrix} w_0 \\ \vdots \\ w_{M-1} \end{pmatrix} \qquad \phi(x) = \begin{pmatrix} \phi_0(x) \\ \vdots \\ \phi_{M-1}(x) \end{pmatrix} \quad \text{where } \phi_0(x) = 1$$

We want to minimize sum of squared error

$$E_D(w) = \frac{1}{2} \sum_{n=1}^{N} \left(t_n - w^T \phi(x_n) \right)^2$$

min $E_D(w)$

but in particular $t = \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix}$ $\Phi = \begin{pmatrix} \phi_0(x_1) & \cdots & \phi_{M-1}(x_1) \\ \vdots & & \\ \phi_0(x_N) & \cdots & \phi_{M-1}(x_N) \end{pmatrix}$

$$E_D(w) = \frac{1}{2}(t - \Phi w)^T (t - \Phi w)$$

optimal condition when:

$$\nabla E = 0 \iff \Phi^T t = \Phi^T \Phi \, w$$

thus

$$w_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \cdot t$$

IF we have dataset that is very huge we can think to do sequential learning we need to use stochastic gradient so

$$\hat{w} \leftarrow \hat{w} + \eta \nabla E$$

$$\hat{w} \leftarrow \hat{w} + \eta \sum_{n=1}^{N} (t_n - w^T \phi(x_n)) \phi(x_n)$$

Algorithm converge for small value of η, learning rate

Ex 5

1) $h_{MAP} = \underset{h \in H}{\operatorname{argmax}} \; P(h|D) = \underset{h \in H}{\operatorname{argmax}} \; P(D|h) \cdot P(h)$

We can drop $P(D)$ because is constant and independent of h

(if $P(h_i) = P(h_j)$)

$h_{ML} = \arg\max_{h \in H} P(D|h)$

We use Maximum a posteriori and maximum likelihood to compute the most probable hypotheses

2) Given a dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$ and target function $f: X \to V$, $V = \{v_1, \ldots, v_m\}$ and a new instance $x \notin D$ that is described with values of attribute $\{a_1, \ldots, a_n\}$

$$v_{NB} = \arg\max_{v_j \in V} P(v_j|D) \prod_i P(a_i|v_j, D)$$

IF we assume that

$$P(a_1, \ldots, a_n | v_j, D) = \prod_i P(a_i|v_j, D)$$

Naive Bayes use ~~optimal~~ conditional independence to approximate $P(a_i|v_j, D)$ ~~this~~ this because we don't have prior probability of hypothesis and in this case we have values of attribute so we don't compute "optmal" value for most probable classification of new instance but the approximation give us good result.

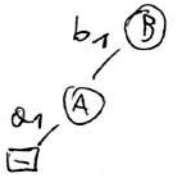
3) In Naive Bayes classifier we don't find optimal

ch as Bayes Optimal Classifier but impractical when we have huge hypothesis space because we cannot have all prior probabilities of hypothesis so we need to use approximation to compute most probable classifier for new instance

EX6

1) IF $B = b_1 \wedge A = a_1$ THEN -
IF $B = b_1 \wedge A = a_2 \wedge C = c_1$ THEN +
IF $B = b_1 \wedge A = a_2 \wedge C = c_2$ THEN +
IF $B = b_1 \wedge A = a_2 \wedge C = c_3$ THEN -
IF $B = b_1 \wedge A = a_3$ THEN +
IF $B = b_2 \wedge C = c_1$ THEN -
IF $B = b_2 \wedge C = c_2$ THEN -
IF $B = b_2 \wedge C = c_3$ THEN +

2)

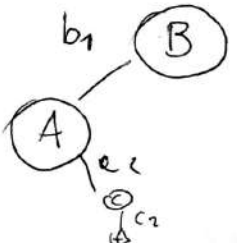


$S_1 = \langle a_1, b_1, c_1, - \rangle$

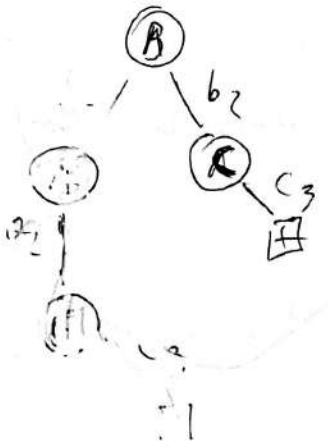
T is consistent because of First rule we have that IF $B = b_1 \wedge A = a_1$ THEN -

$S_2 = \langle a_2, b_1, c_2, + \rangle$

T is consistent with this subset of sample for third rule



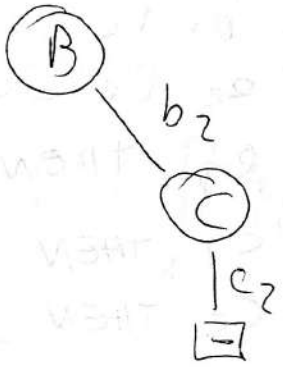
- $S_3 = \langle a_1, b_2, c_3, + \rangle$



As we can see T is consistent with set S_3

- $S_4 = \langle a_2, b_2, c_2, + \rangle$

T is not consistent with S_4 because we expect - but set S_4 has '+'



7/06/2019 (ok)

EX 1

1) We have L, is an algorithm
Split dataset D in K set S_i $i = 1, \dots, K$
$|S_i| \geq 30$

for $i = 1 \dots K$

S_i IS TEST SET

$T_i = \{D - S_i\}$ IS TRAINING SET

$h_i = L(T_i)$ is hypothesis obtain applying that algorithm L

$\delta_i = error_{S_i}(h_i)$

$$ERROR_{L,D}(h) = \frac{1}{K} \sum_{k=1}^{K} \delta_i$$

$$ACCURACY_{L,D} = 1 - error_{S,L}(h)$$

With K-Fold Cross validation we can estimate accuracy of particular learning algorithm split dataset IN Test and training set and train your model on training set that give to us hypothesis that we can use to determine the sample error, so the accuracy of that particular learning algorithm, better is the accuracy better should be our algorithm.

2)

We can also use K-Fold Cross Validation to compare two different learning algorithms L_A AND L_B

Pseudocode

We split dataset D IN K S_i when $|S_i| \geq 30$

for $i = 1 \ldots K$

S_i is TEST set

T_i is TRAINING set $= \{D - S_i\}$

$h_A = L_A(T_i)$

$h_B = L_B(T_i)$

$\delta_i = error_{S_i}(h_A) - error_{S_i}(h_B)$

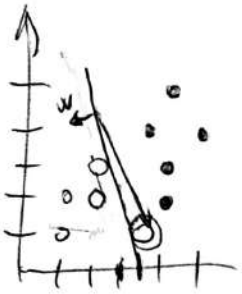
$$\bar{\delta} = \frac{1}{K} \sum_{K=1}^{K} \delta_i$$

IF $\bar{\delta} < 0$ THIS MEANS that L_A is better than L_B

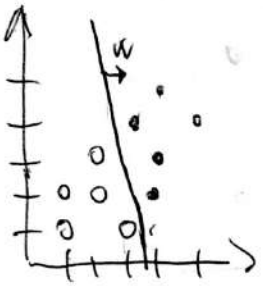
So we evaluate two learning algorithms consisting some training dataset T_i and we generate two hypothesis and we consider difference between two error of two hypothesis of two learning algorithms and if $error_{S_i}(h_A) < error_{S_i}(h_B) \Rightarrow \delta_i < 0$

so Accuracy of h_A is better than accuracy of h_B or viceversa.

notice that we found instance that is misclassified, we update weights and change our decision boundary



another instance misclassified, update weights



Data set is linearly separable so we converge IN a FINITE N° of steps

b) SVM we want to MAXIMIZE margin, distance between decision boundary and closest points from it closest points are support vectors, and at first we can have only just one, but when we maximize margin we have 2

$x_{1\oplus}$ and $x_{K\ominus}$ where $w^T x_{K\oplus} + w_0 = 1$

and $w^T x_{K\ominus} + w_0 = -1$

and distance is $\frac{+}{\|w\|}$ where $y(x) = w^T x + w_0$

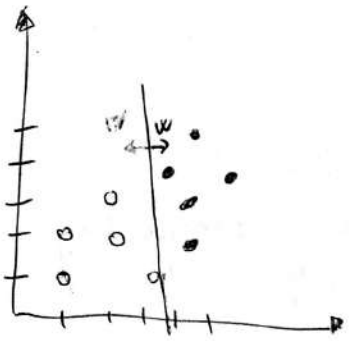
this is the reason why we obtain that value for SVM

3) SVM is prefer respect to perceptron because IF a Dataset is NOT linearly separable, perceptron will never converge, instead SVM use some slack

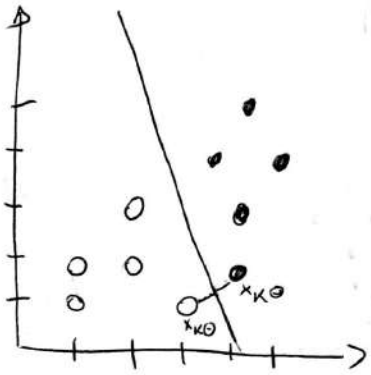
EX2

1)

a) PERCEPTRON



b) SVM



2) Both method wants to classify correctly INPUTS OF DATES

Perceptron use weights and in particular a FUNCTION

$$\sigma(x) = \begin{cases} 1 & \text{IF } w^T x > 0 \\ -1 & \text{otherwise} \end{cases} \qquad y = w^T x \qquad \sigma = w^T x$$

When start choosing random value of w and we update weight until there is a terminal condition

$$w^{New} \leftarrow w^{old} - \eta \nabla E$$

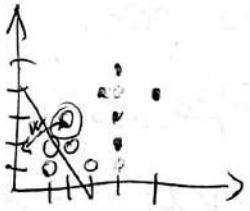
η = learning rate

$$\nabla E = -\frac{1}{2} \sum_{n=1}^{N} (t_n - w^T x_n)(x_{i,n})$$

= gradient error

We update our weight only if we found INSTANCE that is MISSclassified.

For example



on bottom we have INSTANCES classified IN "0" and on top INSTANCES of class "1"

ble to try to find solution of w and w_0, so find maximum margin.

EX3

ONE STATE MDP

$\langle \{x_0\}, A, S, r \rangle$

x_0 is unique state

A is set of actions

$\delta(x_0, a_i)$: $\forall a_i \in A$ transition function

$r(x_0, a_i, x_0)$ reward function

We need to determine optimal policy

but in K-armed bandit problem you don't know

δ and r so you need to find some strategy

to find this optimal policy.

In particular here we use value function iteration,

more specific Q-learning $Q(x_0, a)$, after compute this

$\pi^*(x_0)$ = optimal policy $= \arg\max_{a \in A} Q^*(x_0, a)$

2)

We have K actions $a_1, \dots, a_K$ $r(a_i) = N(\mu_i, \sigma_i)$

gaussian distributed (consider stochastic case)

We choose ε-greedy strategy so we have uniform

random choice with probability ε and best

choice with probability $1-\varepsilon$

the trainy rule is

$$Q_n(a_i) \leftarrow Q_{n-1}(a_i) + \alpha [r_{t+1}(a_i) - Q_{n-1}(a_i)$$

when $\alpha = \frac{1}{V_{n-1}(a_i)}$

$V_{n-1}(a_i)$ = n° of executions of action a_i up to the n-t

$a^* = \text{argmax}_a Q_n(a_i)$

EX4

1) Both probabilist model want to compute (estimate)

$P(C_i | X)$ when $i = 1, \dots, n$

the probability that given new istance is belonged of a class C_i

Generative probabilist model using bayes to estimate this value, in particular

$$P(C_i | X) = \frac{P(X|C_i) \cdot P(C_i)}{\sum_{i=1}^{n} P(X|C_i) P(C_i)}$$

se we compute $P(X|C_i)$ and we have prior probability of that particular class.

Instead with discriminative we use method of logistic regression to estimate $P(C_i|X)$, se we estimate that value directly from a model

2D Dataset of C_1 and C_2

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

We assume $P(x|C_1) = N(x|\mu_1, \Sigma)$ and $P(x|C_2) = N(x|\mu_2, \Sigma)$

they have gaussian distribution with same covariance, and N_1 are samples of C_1 and N_2 samples of C_2, $N = N_1 + N_2$

$P(C_1) = \pi$

$P(C_2) = 1 - \pi$

We have likelihood funct

$$P(t|\pi, \mu_1, \mu_2, \Sigma) = \prod_{n=1}^{N} \pi N(x_n|\mu_1, \Sigma)^{t_n} (1-\pi) N(x_n, \mu_2, \Sigma)^{(1-t_n)}$$

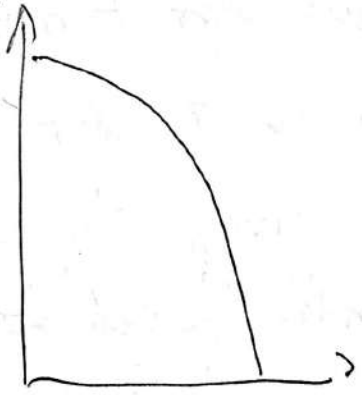
Maximize likelihood and we obtain

$$\mu_1 = \frac{1}{N_1} \sum_{n=1}^{N} t_n x_n \qquad \mu_2 = \frac{1}{N_2} \sum_{n=1}^{N} (1-t_n) x_n$$

$$\Sigma = \frac{N_1}{N} S_1 + \frac{N_2}{N} S_2 \qquad \pi = \frac{N_1}{N}$$

$$S_i = \frac{1}{N_i} \sum_{n \in C_i} (x_n - \mu_i)^T (x_n - \mu_i) \qquad i = 1, 2$$

$P(C_1|x)$



EX 5

1) CNN is a neural network that use a convolution function between input and kernel
It is compose by 3 stage ; the FIRST one is convolution stage.
Convolution Stage is

2) $(I * K)(i, j) = \sum_m \sum_n I(i+m, j+n) K(m, n)$ this is cross correlation

In this stage we apply sparse connectivity in which output depends only of a few input, this beacuse K is much smaller than input, in this way we can reduce memory requirement and improve statistically efficend reducing number of operations.
Another thing that is done in this stage is parameter sharing in which units of layer share a set of parameter (in this way We increase efficient because we don't need to learn more parameter, such as in Traditional neural network
the last thing to conde is the padding which We can conder "valid" so outputs contains only valid value (this depend of Kernel) or "same" so input

er is padden with zero and output size independent of Kernel size.

EX6

1) In a supervised learning we have dataset

$D = \{(x_n, t_n)\}_{n=1}^{n}$ when x_n are ~~output~~ input and t_n is the target value associated to that input.

In a unsupervised learning we have a DataSet

$D = \{(x_n)\}_{n=1}^{N}$ we have only input of our dataset, we don't know target value of them

2)

Unsupervised learning we can model problem to ~~define~~ partition of dataset in K cluster In particular with K-means we can obtain means of this cluster using a gaussian distribution Unsupervised learn use technique such as clustering, representation and density estimation, with them we hope to learn the inherent structure of our input data with the ~~same~~ target value of them