* K-Nearest neighbour.

→ Also known as lazy learner/ instance-based learning.

→ Used when data is distributed in non-linear manner.

K => how many nearest values we need to consider. In terms of distance.

↳ Euclidean

↳ Manhattan.

① for classification use-case

→ The answer for the new point would be the one with majority of category. and thus gets classified to that group

② Regression use-case.

→ Here all values will be of same group

→ We will find the mean of all k nearest neighbours.

→ KNN will get impacted by outliers., because if there are a lot of outliers, then it will be difficult for a classification. prediction.

→ same will happen if the data is imbalanced. and leads to bias prediction.

- static
- RIP
- OSPF
- Subnetting
- Virtual LAN
- Wireless LAN.

SE -

ML - 14

DAA -

→ when dataset is too big, classification time is too high.

→ when K is high, the less similar data (one who value is 0 if done with cosine similarity) will contribute which leads to wrong prediction.

→ when K is too small. wrong prediction may lead as if one outlier of one class is in other coordinate where other class has so many points, this causes wrong prediction.

→ k is small ⇒ overfit

k is large ⇒ underfit

Thumb rule ⇒ $k = \sqrt{n}$

where n = no. of training examples.

→ * k should be odd.

* k value must not be multiple of no. of classes.

* Should not be too small or too large

too small ⇒ greater influence to outlier/noise.

too big ⇒ bias towards highly probable class.

→ In KNN regressor, we take avg. of k near values.

* Naive - Bayes classifier.

→ Used bayes theorem, i.e., probabilistic approach to class the data.

→ They are generally used for text classification or medical image related classification.

→ A fast machine learning model for quick prediction.

→ Two assumptions.

(a) features given are independent of each other.

(b) All the features contribute equally.

$$P(C|x) = \frac{P(x|C) \cdot P(C)}{P(x)}$$

$P(C|x)$ → posterior probability

$P(x|C)$ → likehood

$P(C)$ → prior probability

$P(x)$ → marginal probability.

C = Class

x = feature / test sample

→ Here $P(X)$ is constant $(=1)$ for all classes

→ Let D be no. of training set of tuples and has n features
∴ we will for n-D attribute vector $X = (x_1, x_2, \ldots x_n)$

→ Suppose there are m classes $C_1, C_2, \ldots, C_m$
Classification is to derive max. posteriority, i.e., the max $P(C_i|X)$

$$\therefore P(C_i|X) = \frac{P(X|C_i)\,P(C_i)}{P(X)}$$

But $P(X)$ = constant = 1

$$P(C_i|X) = \underbrace{P(X|C_i) * P(C_i)}_{\text{needs to be maximized}}$$

Now, $P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \cdot P(x_2|C_i) \cdot P(x_3|C_i) \ldots P(x_n|C_i)$

→ If A_k/feature (k) is categorical, $P(x_k|C_i)$ is the no. of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$

→ If A_k is continuous value, $P(x_k|C_i)$ is computed based on gaussian distribution with mean μ and S.D σ.

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

and $P(x_k|C_i)$ is $= g(x_k, \mu_{C_i}, \sigma_{C_i})$.

✗ Avoiding Zero Probability Problem.
→ We use laplacian correction.
Suppose for 1000 tuple, income = low(0), medium(990), high(10)
∴ we add 1 to each case
So, now, tuple income = low(1), medium(991), high(11)
and no. of tuple will be 1003..

* Weighted KNN

→ first find the k-nearest neighbor. based on distance.

→ Then, inverse them

→ Then, sum the inversed (done in step 2) k neighbours.

→ Then, redefined the weights i.e, inversed (step 2)/sum for each of the k-neighbor.

→ Then sum them based on their classes.
like class 0 = sum$_1$; class 1 = sum$_2$ - - -

→ Then one with highest sum, the test data is assigned that class.

* Model Evaluation and Selection.

→ Methods for estimating a classifier's accuracy.
(a) Holdout method, random subsampling
(b) cross-validation
(c) Bootstrap.

→ Validation techniques.
* Resubstituting
* Hold out
→ LOOCV (Leave-One-Out-Cross-Validation)
→ Random subsampling
→ Bootstrapping

* Classifier Evaluation Metrics = Confusion Matrix.

AC | PC	C_1	$\neg C_1$	
C_1	TP	FN	P
$\neg C_1$	FP	TN	N
	P'	N'	

CM_{ij} ⇒ no. of tuples in class i that were labelled by the classifier as class j.

→ Accuracy = (TP + TN) / All

→ Error rate = 1 − (accuracy) = (FN + FP) / All

→ Sensitivity = True positive recognition rate = TP / P

→ Specificity = True negative recognition rate = TN / N

→ Precision = what % of tuple classifier labelled +ve are actually positive

= $\frac{TP}{TP + FP} = \frac{TP}{P'}$

→ Recall = what % of +ve tuples did the classifier label as +ve?

= $\frac{TP}{TP + FN}$

→ Note = Inverse relation b/w precision & recall

→ f-measure (f1 score or f score) = Harmonic mean of precision & recall.

$$F = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

→ f_β : weighted measure of precision and recall.

$$f_\beta = \frac{(1 + \beta^2) * \text{precision} * \text{recall}}{\beta^2 * \text{precision} + \text{recall}}$$

* Naive Bayes Classifier = Multinomial.

→ compulsory to do laplacian correction.

→ Generally used for document classification., where we are interested in finding the frequency of occurence of that specific term in that document.

i, m = terms
j, n = documents.

$P(d_j | C)$

↓

Probability of document belonging to class C.

$$P(d_j | C) = \left(\sum_{i=1}^{m} n_{ij} \right)! \prod_{i=1}^{m} \frac{P(t_i | C)^{n_{ij}}}{n_{ij}!}$$

$\left(\sum_{i=1}^{m} n_{ij} \right)!$ ← Length of document j

$\frac{P(t_i | C)^{n_{ij}}}{n_{ij}!}$ ← no. of appearance of term i in document j.

$$P(t_i | C) = \frac{\sum_{j=1}^{n} n_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} n_{ij}}$$

$\sum_{j=1}^{n} n_{ij}$ ← no. of appearance of ith term in all documents of class C.

$\sum_{i=1}^{m} \sum_{j=1}^{n} n_{ij}$ ← sum of length of all documents of class C.

→ Resubstitution error

When we are using the set on which we train our model and we are testing the same set again for accuracy. Thus the error we get is called resubstitution error.

→ Bootstrapping (0.632 bootstrap)

from ppt.

* Decision Tree.

→ Decision tree induction is the learning of decision trees from class-labeled training tuples.

→ * Each internal node denotes the test on an attribute.
* Each branch/edge represents outcome of test.
* Each leaf node holds a class label.

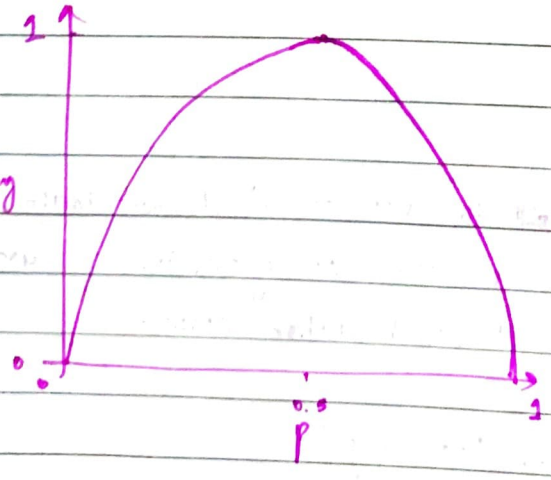
→ DT has good classification accuracy.

→ * When outcomes are certain, we will get less information
* When outcomes are uncertain, we will get information.

→ Entropy is the measure of impurity of dataset D.

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

* As the data becomes purer and purer, the entropy value becomes smaller and smaller.



* Classification by Decision Tree Induction - ID3.

→ See lecture of 20/09/2021

→ The problem with ID3 classification is that it is biased towards tests with many outcomes., i.e., it prefers to select attributes having a large no. of values.

→ Thus this will help in classification better, but such a partitioning used could be useless.

* Classification by Decision Tree Induction - C4.5.

→ C.4.5 is successor of ID3 which overcome this bias of ID3, using gain ratio.

→ Gain ratio

- Kind of normalization to information gain using a split information value defined analogously with Info(D) as

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \log_2\left(\frac{|D_j|}{|D|}\right)$$

(v → no. of attributes.)

$$Gain\ ratio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

- The attribute with max. gain ratio is selected as splitting attribute.

Eg

WB	F	Fur	S	Lays Egg
Y	Y	N	No	Y
N	N	N	Y	Y
Y	Y	N	N	Y
Y	Y	N	N	Y
Y	N	N	Y	N
Y	N	Y	N	N

Using ID 3

$$\text{Info}(D) = -\frac{4}{6}\log_2\left(\frac{4}{6}\right) - \frac{2}{6}\log_2\left(\frac{2}{6}\right)$$

$$= 0.9182 \text{ bits}$$

$$\text{Info}_{WB}(D) = \frac{5}{6}\left[\frac{-3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log\left(\frac{2}{5}\right)\right] + \frac{1}{6}\left[\frac{-1}{1}\log\left(\frac{1}{1}\right)\right]$$

$$= 0.8091 \text{ bits}$$

$$\text{Info}_{F}(D) = \frac{3}{6}\left[\frac{3}{3}\log_2\left(\frac{3}{3}\right)\right] + \frac{3}{6}\left[\frac{-1}{3}\log_2\left(\frac{1}{3}\right) - \frac{2}{3}\log\left(\frac{2}{3}\right)\right]$$

$$= 0.4591 \text{ bits}$$

$$\text{Info}_{Fur}(D) = \frac{1}{6}[0] + \frac{5}{6}\left[\frac{-1}{5}\log\left(\frac{1}{5}\right) - \frac{4}{5}\log\left(\frac{4}{5}\right)\right]$$

$$= 0.6016 \text{ bits}$$

$$\text{Info}_{S}(D) = \frac{2}{6}\left[\frac{-1}{2}\log\left(\frac{1}{2}\right) - \frac{1}{2}\log\left(\frac{1}{2}\right)\right] + \frac{4}{6}\left[\frac{-3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right)\right]$$

$$= 0.8741 \text{ bits.}$$

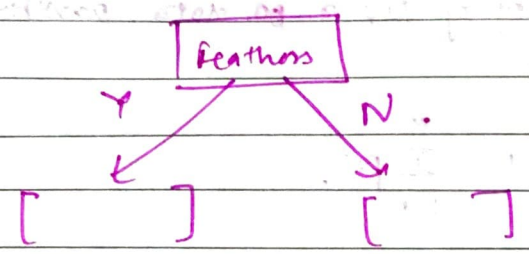
$\therefore$ Gain(WB) = 0.9182 − 0.8091 = 0.109 bits

Gain(F) = 0.9182 − 0.4591 = 0.4591 bits

Gain (thin) = 0.9182 - 0.6016 = 0.3166

Gain (S) = 0.9182 - 0.8741 = 0.0441.

∴ Max gain is of f.



* Computing Information - Gain for continuous-valued attributes.

→ Let attribute A be continuous valued attribute.

* discretization (break attribute into ranges in advance)
* using thresholds for splitting nodes.

→ Select point with min. expected information required.

Length	10	15	21	24	32	40	50
Class	N	Y	Y	N	Y	Y	N

$A_1 = \frac{10+15}{2} = 12.5$ $A_2 = \frac{21+24}{2} = 24.5$ $A_3 = \frac{32+24}{2} = 30$ $A_4 = \frac{40+50}{2} = 245$

no. of samples in partition

$$Info_{A_1}(D) = \frac{1}{7}\left[\frac{-1}{1}\log\frac{1}{1}\right] + \frac{6}{7}\left[\frac{-4}{6}\log\frac{4}{6} - \frac{2}{6}\log\frac{2}{6}\right]$$

No in partition 1 → Yes in partition 2 → no is partition 2

$$Info_{A_2}(D) = \frac{3}{7}\left[\frac{-2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3}\right] + \frac{4}{7}\left[\frac{-2}{4}\log\frac{2}{4} - \frac{2}{4}\log\frac{2}{4}\right]$$

$Info_{A_3}(D)$, $Info_{A_4}(D)$ ∴ Select point with min info.

* Classification by Decision Tree Induction - CART

→ Gini index.

- Used when we want to have binary partition
- measure of impurity of D, a data partition or set of training tuples. as

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2.$$

→ For discrete value attribute, all possible combination of value except empty set & power set.

$$Gini_A(D) = \frac{|D_1|}{D} Gini(D_1) + \frac{|D_2|}{D} Gini(D_2) \dots$$

→ Subset that gives min. gini index is selected as splitting subset.

→ for continuous, each possible split point must be considered.

→ Reduction in impurity that would be incurred by a binary split on a discrete/continuous attribute A is,

$$\Delta Gini(A) = Gini(D) - Gini_A(D).$$

→ Attribute that max. the reduction impurity is selected as splitting attribute.

→ Information gained is biased towards multivalued attribute.

→ Pruning of decision tree is used to avoid overfitting.
Overfitting when no. of features are high.

* Support Vector Machine (SVM)

→ Used for binary classification.
→ Used for high dimensional data.
→ SVM is linear learning system that builds two class classifier.

$\{(x_1, y_1), (x_2, y_2), (x_3, y_3) \dots (x_n, y_n)\} \Rightarrow D$

$x_i \Rightarrow (x_{i1}, x_{i2}, \dots x_{ir})$ $\quad x \in R^r$

↳ this are the features

y_i is a class label $\quad y_i \in \{-1, 1\}$

SVM finds linear function which is of the form

$f(x) = \langle w.x \rangle + b$

If $f(x_i) \geq 0$, it results in class label 1. $\therefore y_i = 1$

$f(x_i) < 0$, Then class label is -1 $\therefore y_i = -1$.

$$\therefore y_i = \begin{cases} 1, & \text{if } f(x_i) \geq 0 \\ -1, & \text{if } f(x_i) < 0 \end{cases}$$

$f(x_1, x_2, \dots x_r) = \underbrace{w_1x_1 + w_2x_2 + \dots + w_rx_r + b}$.

↓

this is a hyperplane.

Hyperplane in 2D ➡ line
Hyperplane in 3D ➡ plane.

→ Disadvantage of having min distances or margin is that we want to avoid false prediction

→ We select a hyperplane which is away from both the classes equally. (or max margin)

A large margin effectively corresponds to a regularization of SVM weights which prevents overfitting. Hence, we prefer a large margin (or the right margin chosen by cross-validation) because it helps us generalize our predictions and perform better on the test data by not overfitting the model to the training data.

→ $wx + b = 0$

w → defines a direction ⊥ to the hyperplane. (normal vector).

→ Maximum margin hyperplane

It gives you maximum margin b/w the +ve and -ve class sample.

$H_+ = \langle w.x^+ \rangle + b = 1$

$H_- = \langle w.x^- \rangle + b = -1$

such that $\langle w.x_i \rangle + b \geq 1$ if $y_i = 1$

$\langle w.x_i \rangle + b \leq -1$ if $y_i = -1$

$(d_+) + (d_-) =$ max. margin.

and $(d_+) + (d_-) = \frac{2}{\|w\|}$

we reverse this,

thus, minimize $\frac{\langle w.w \rangle}{2}$ / $\frac{\|w\|^2}{2}$ → this is slightly better

and $y_i(\langle w.x_i \rangle + b) \geq 1$ for all i → same as this equation

⇒ Standard Lagrangian Multiplier Method.

minimize $\frac{\langle w.w \rangle}{2}$

subject to $y_i(\langle w.x_i \rangle + b) \geq 1$, $i = 1, 2, \ldots, n$

$$\therefore L_p = \frac{1}{2}\langle w.w\rangle - \sum_{i=1}^{n} \alpha_i [y_i(\langle w.x_i\rangle + b_i) - 1]$$

Constraints $w_1, w_2, w_3, \ldots, w_r$ and b.

Now, $\frac{dL_p}{dw_j} = w_j - \sum_{i=1}^{n} y_i \alpha_i x_{ij} = 0$ where $j = 1, 2, \ldots r$

$$\frac{dL_p}{db} = -\sum_{i=1}^{n} y_i \alpha_i = 0.$$

$(y_i(\langle w \cdot x_i\rangle + b) - 1) \geq 0$ for $i = 1 \ldots n$.

$\alpha_i \geq 0$ for $i = 1, 2 \ldots n$

$\alpha_i(y_i(\langle w.x_i\rangle + b) - 1) = 0$ for $i = 1, \ldots n$

↑ only those points which are on marginal planes for which $\alpha_i > 0$.

$\therefore$ those points (samples) are called support vectors.

→ $\therefore$ for SVM, not all points / data is used for identifying the hyperparameter.

See lectures for better understanding.

Kuch samaj nahi aata SVM me.

$L_D = \sum_{i=1}^{n} \alpha_i$ [illegible] $- \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j x_i x_j \alpha_i \alpha_j$.

↑ wolfe dual

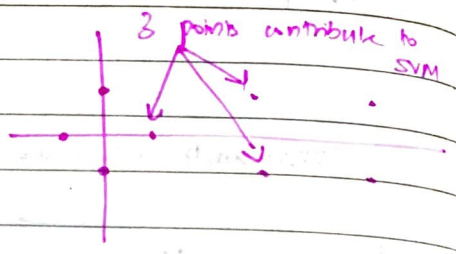
subject to : $\sum_{i=1}^{n} y_i \alpha_i = 0$

$\alpha_i \geq 0$; $i = 1 \ldots n$

SVM

Support Vector Machine

Eg:- SVM



+ve labelled points in R^2.

$\{(3,1), (3,-1), (6,1), (6,-1)\}$

-ve labelled points in R^2.

$\{(1,0), (0,1), (-1,0), (0,-1)\}$

$\therefore x_1 = (1,0)$ } class -1.

$x_2 = (3,1)$

$x_3 = (3,-1)$ } class 1

The vector is augmented with 1 as "bias input".

$\therefore x_1 = (1,0,1)$

$x_2 = (3,1,1)$

$x_3 = (3,-1,1)$

$\therefore$ Objective function.

$$w(\alpha) = \sum_{i=1}^{l} a_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\vec{x_i} \cdot \vec{x_j})$$

kernel function is identity.

$\therefore K(\vec{x_i} \cdot \vec{x_j}) = \vec{x_i} \cdot \vec{x_j}$

and $l = 3$ = no. of points contribute to SVM.

$$\therefore w(\alpha) = \sum_{i=1}^{3} \alpha_i - \frac{1}{2} \sum_{i=1}^{3} \sum_{j=1}^{3} \alpha_i \alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}).$$

$$= \sum_{i=1}^{3} \alpha_i - \frac{1}{2} \left[\alpha_1 \alpha_1 y_1 y_1 (\vec{x_1} \cdot \vec{x_1}) + \alpha_1 \alpha_2 y_1 y_2 (\vec{x_1} \cdot \vec{x_2}) + \ldots \right.$$

$$\left. \alpha_3 \alpha_2 y_2 y_3 (\vec{x_3} \cdot \vec{x_3}) \right].$$

$$(\vec{x_1} \cdot \vec{x}) = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = 2 \qquad \vec{x_1} \cdot \vec{x_2} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix} = 4.$$

$(\vec{x_1} \cdot \vec{x_3}) = 4$; $(\vec{x_2} \cdot \vec{x_1}) = 1$; $(\vec{x_2} \cdot \vec{x_3}) = 9$; $(\vec{x_3} \cdot \vec{x_3}) = 11$.

→

$$\therefore W(\alpha) = \sum_{i=1}^{3} \alpha_i - \frac{1}{2}\left[2\alpha_1^2 - 4\alpha_1\alpha_2 - 4\alpha_1\alpha_3 - 4\alpha_1\alpha_2 + 11\alpha_2^2 + 9\alpha_2\alpha_3 - 4\alpha_1\alpha_3 + 9\alpha_2\alpha_3 + 11\alpha_3^2\right].$$

$$= \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}\left[2\alpha_1^2 - 8\alpha_1\alpha_2 - 8\alpha_1\alpha_3 + 11\alpha_2^2 + 18\alpha_2\alpha_3 + 11\alpha_3^2\right].$$

∴ we need to max. this equation w.r.t α.

$$\therefore \frac{dW(\alpha)}{d\alpha_1} = 1 - \frac{1}{2}\left[4\alpha_1 - 8\alpha_2 - 8\alpha_3\right] = 0.$$

$$2\alpha_1 - 4\alpha_2 - 4\alpha_3 = 1.$$

$$\frac{d(W(\alpha))}{d\alpha_2} = 4\alpha_1 - 11\alpha_2 - 9\alpha_3 = -1$$

$$\frac{dW(\alpha)}{d\alpha_3} = 4\alpha_1 - 9\alpha_2 - 11\alpha_3 = -1.$$

Solving the above 3 equation

$\therefore \alpha_1 = -3.5, \quad \alpha_2 = 0.75, \quad \alpha_3 = 0.75.$

Now we have α_i, we find hyperplane from :

$$\vec{w} = \sum_{i=1}^{3} \alpha_i y_i \vec{x_i}$$

$$= 3.5\begin{bmatrix}1\\0\\1\end{bmatrix} + 0.75\begin{bmatrix}3\\1\\1\end{bmatrix} + 0.75\begin{bmatrix}3\\-1\\1\end{bmatrix}$$

$$= \begin{bmatrix}1\\0\\-2\end{bmatrix}$$

Since the vectors are augmented with bias, the last entry in $\vec{w}$ is value of bias b, i.e., b = -2 and $\vec{w} = [1\ 0]$

Now hyperplane is

$\vec{w} \cdot \vec{x} + b = 0$.

Since dataset is 2D,

then $w_1 x_1 + w_2 x_2 + b = 0$.

$1 \cdot x_1 + 0 \cdot x_2 - 2 = 0$

$\boxed{x_1 = 2}$

Now, if we want to classify $x = (4\ 1)$ then.

$f(x) = \text{sgn}(\vec{w} \cdot \vec{x} + b)$

$f\left(\begin{bmatrix} 4 \\ 1 \end{bmatrix}\right) = \text{sgn}\left(\begin{bmatrix} 4 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} - 2\right) = \text{sgn}(4 - 2)$

$= \text{sgn}(2)$

$=$ +ve class.

Eg-2 Shortcut method

$S_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$ → class = -1 $\quad S_2 = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}$ → class = -1 $\quad S_3 = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$ → class = 1.

$\alpha_1 S_1 S_1 + \alpha_2 S_2 S_1 + \alpha_3 S_3 S_1 = -1$

$\alpha_1 S_1 S_2 + \alpha_2 S_2 S_2 + \alpha_3 S_3 S_2 = -1$

$\alpha_1 S_1 S_3 + \alpha_2 S_2 S_3 + \alpha_3 S_3 S_3 = +1$

} 3 equations when differentiated wrt α_i

$6\alpha_1 + 4\alpha_2 + 9\alpha_3 = -1$

$4\alpha_1 + 6\alpha_2 + 9\alpha_3 = -1$

$9\alpha_1 + 9\alpha_2 + 17\alpha_3 = 1$

$\alpha_1 = -3.25$, $\alpha_2 = -3.25$, $\alpha_3 = 3.5$

$\therefore \vec{w} = \sum_{i=1}^{3} \alpha_i y_i \vec{x}_i$

$$\vec{w} = -3.25\begin{bmatrix}2\\1\\1\end{bmatrix} - 3.25\begin{bmatrix}2\\-1\\1\end{bmatrix} + 3.5\begin{bmatrix}4\\0\\1\end{bmatrix}$$

$$= \begin{bmatrix}1\\0\\-3\end{bmatrix}$$

$\therefore \vec{w} = \begin{bmatrix}1\\0\end{bmatrix} \quad b = -3.$

$\therefore$ Hyperplane

$$\vec{w}.\vec{x} + b = 0$$

Since dataset is 2D.

$\therefore$ $w_1x_1 + w_2x_2 + b = 0.$

$1.x_1 + 0.x_2 - 3 = 0.$

$\boxed{x_1 = 3}$

* SVM - Inseparable case.

Due to noise/outlier, we need to modify the equation. Thus we add some error term.

$\therefore$ $\langle w.x_i \rangle + b \geq 1 - \varepsilon_i$ for $y_i = 1$

$\langle w.x_i \rangle + b \leq -1 + \varepsilon_i$ for $y_i = -1.$

$\therefore$ $y_i(\langle w.x_i \rangle + b) \geq 1 - \varepsilon_i$ for $i = 1, 2, \ldots n$

$\varepsilon_i \geq 0, \quad i = 1, 2, \ldots n.$

Also, Minimize $= \frac{\langle w.w \rangle}{2} + C\left(\sum_{i=1}^{n} \varepsilon_i\right)^k$

where $C \geq 0$ is user specified parameter.

Here we will discuss $k = 1$ case only.

Now equation becomes,

$$L_P = \frac{1}{2}\langle w.w\rangle + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i(\langle w.x_i\rangle + b) - 1 + \xi_i\right] - \sum_{i=1}^{n}\mu_i\xi_i$$

where $\alpha_i, \mu_i \geq 0$ are lagrange multiplier.

And, we can find value of b as.

$$b = \frac{1}{y_i} - \sum_{i=1}^{n} y_i\alpha_i\langle x_i.x_j\rangle$$

Again, due to numerical errors, we can compute all possible b's and take average b as final value.

* Non-Linear SVM.

→ The basic idea is to map data in the input space X, to a feature space F via a nonlinear mapping ϕ.

$$\phi : X \to F$$
$$x \mapsto \phi(x).$$

After mapping, original dataset $\{(x_1, y_1), (x_2, y_2) \ldots, (x_n, y_n)\}$ becomes $\{(\phi(x_1), y_1), (\phi(x_2), y_2), \ldots, (\phi(x_n), y_n)\}$

* Perceptron model.

$a = \text{hardlim}(wp + b)$

∴ $a = 1$ if $net\ (wp+b) \geq 0$.

else $a = 0$ if $\ldots < 0$

- $w_{new} = w_{old} + \eta e p$ → error; p → input samples; η → learning rate.

  $b_{new} = b_{old} + \eta e$, where e = target − actual.

Eg: input {0.0, 0.17, 0.33, 0.50, 0.67, 0.83, 1.0}

First four are of class 0 (desired output is 0).

Remaining are of class 1 (desired output is 1).

Learning rate = 0.1

Initial w = −0.36

Initial b = −0.1.

→ The value of the bias must be such that it guarantees it does not cross the origin.

→ Issues

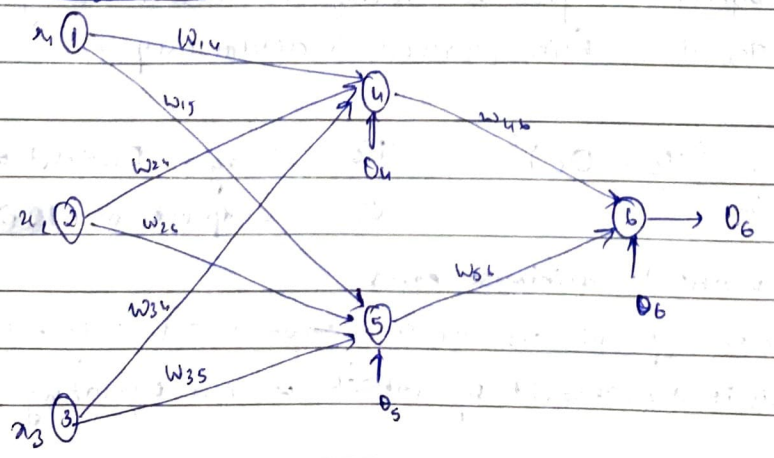
→ Terminating condition ⇐ What is the tolerable amount of error.

→ Learning rate

→ Non-numeric inputs

→ Epoch.

Backpropogation



Above is example of multilayer feed forward neural network.

Initial values

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

Class label = 1.

net(4) = input given to neuron 4.

$= w_{14}x_1 + w_{24}x_2 + w_{34}x_3 + \theta_4$

$net(5) = w_{15}x_1 + w_{25}x_2 + w_{35}x_3 + \theta_5.$

- O(4) = Output by neuron 4.

$= f(net(4))$

$= \frac{1}{1+e^{-net(4)}}$ (Sigmoid function).

$net(6) = w_{46}O_4 + w_{56}O_5 + \theta_6$

$= (-0.3)f(net(4)) + (-0.2)f(net(5)) + 0.1.$

Backpropagation algorithm, which is generalization of the least mean squared algo. that modifies network weights to minimize the MSE b/w desired & actual output.

$Error = \frac{1}{2}(T_6 - O_6)^2$ T_6 = Target / Desired output.

O_6 = Output at neural 6.

∴ We want to minimise error.

∴ We take differentiation of this error w.r.t weight w.

and this is the amount by which we are updating the weight

∴ update in network parameters.

w_{14} w_{15} w_{46}

w_{24} w_{25} w_{56}

w_{34} w_{35} θ_4

θ_5

θ_6

∴ In backpropogation we update θ_6, then $w_{46}, w_{56}, \theta_4, \theta_5, w_{\dots}$ then propogate it backwards.

$$\theta_6 = \theta_6 - \eta \left(\frac{dE}{d\theta_6}\right)$$

(learning rate)

Now, $\frac{dE}{d\theta_6} = \frac{dE}{dO_6} \times \frac{dO_6}{dnet_6} \times \frac{dnet_6}{d\theta_6}$.

$$\frac{dE}{dO_6} = \left[\frac{1}{2} \times 2 \times (T_6 - O_6)(-1)\right] = -(T_6 - O_6) = O_6 - T_6$$

Now $O_6 = f(net(6))$

↳ sigmoid.

$$\therefore \frac{d\, f(net(6))}{d\, net(6)} = f(net(6)) * (1 - f(net(6)))$$

$$\frac{dO_6}{dnet_6} = O_6(1 - O_6).$$

and $net(6) = w_{46} O_4 + w_{56} O_5 + \theta_6$.

$$\frac{d(net(6))}{d\theta_6} = 1.$$

$$\therefore \frac{dE}{d\theta_6} = (O_6 - T_6) \times O_6(1 - O_6) \times 1 = -(T_6 - O_6) O_6(1 - O_6).$$

Now, for w_{46}.

$$\frac{dE}{dw_{46}} = \frac{dE}{dO_6} \times \frac{dO_6}{dnet_6} \times \frac{dnet_6}{dw_{46}}$$

$$= (-1)(T_6 - O_6) O_6(1 - O_6) \times (O_4)$$

$$\frac{dE}{dw_{56}} = (-1)(T_6 - O_6) O_6 (1 - O_6) \times O_5$$

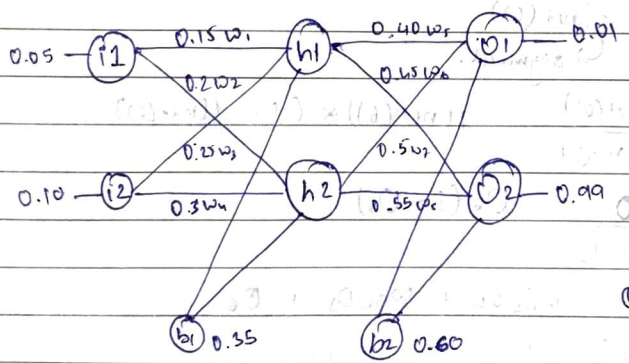
$$\frac{dE}{dO_4} = \frac{dE}{dO_6} \times \frac{dO_6}{dnet_6} \times \frac{dnet_6}{dO_4} \times \frac{dO_4}{dnet_4} \times \frac{dnet_4}{dO_4}$$

$$= -(T_6 - O_6) O_6 (1 - O_6) \times w_{46} \times O_4 (1 - O_4) \times 1$$

$$\frac{dE}{dw_{14}} = -(T_6 - O_6) O_6 (1 - O_6) \times w_{46} \times O_4 (1 - O_4) \times x_1$$

$$\frac{dE}{dO_5} = -(T_6 - O_6) O_6 (1 - O_6) \times w_{56} \times O_5 (1 - O_5) \times 1$$

Eg



Given input 0.05 and 0.10, we want neural network to output 0.01 & 0.99

Calculate w_1, w_2, w_3, w_4.

$$\therefore net\ h_1 = i1 * w_1 + i2 * w_2 + b_1$$

$$= (0.05 * 0.15) + (0.1 * 0.20) + 0.35$$

$$= 0.3775$$

$$net\ h_2 = i1 \times w_3 + i2 \times w_4 + b_1$$

$$= (0.05 \times 0.25) + (0.1 * 0.3) + 0.35$$

$$= 0.393$$

$$out\ h1 = \frac{1}{1 + e^{-net_4}} = \frac{1}{1 + e^{-0.3775}} = 0.593$$

$$\text{Out } h_2 = \frac{1}{1+e^{-\text{net}h_2}} = \frac{1}{1+e^{-0.393}} = 0.597.$$

$$\text{net } O_1 = \text{out } h_1 * w_5 + \text{out } h_2 * w_6 + b_2$$
$$= (0.593 * 0.4) + (0.597 * 0.45) + 0.6$$
$$= 1.1059$$

$$\text{out } O_1 = \frac{1}{1+e^{-1.1059}} = 0.751$$

$$\text{net } O_2 = \text{out } h_1 * w_7 + \text{out } h_2 * w_8 + b_2$$
$$= (0.593 * 0.5) + (0.597 * 0.50) + 0.6$$
$$= 0.2965 + 0.3283 + 0.6 = 1.2248.$$

$$\text{out } O_2 = \frac{1}{1+e^{-1.2248}} = 0.773.$$

$$E O_1 = \frac{1}{2}(0.01 - 0.751)^2 = 0.275$$

$$E O_2 = \frac{1}{2}(0.99 - 0.773)^2 = 0.02356$$

$$\therefore E_{\text{total}} = E_1 + E_2 = 0.275 + 0.02356 = 0.29837$$

$$\frac{dE_{\text{Total}}}{dw_5} = \frac{dE_{\text{Total}}}{d\text{out } O_1} * \frac{d\text{out}(O_1)}{d\text{net}(O_1)} * \frac{d\text{net}(O_1)}{dw_5}$$
$$= -(T_1 - O_1)\, O_1 (1 - O_1) * \text{Out}(h_1).$$
$$= 0.5564 * 0.249 * 0.593 = 0.08215$$

$$\frac{dE_{\text{Total}}}{dw_6} = -(T_1 - O_1)\, O_1 (1 - O_1) * \text{out}(h_2).$$
$$= 0.0827$$

$$\frac{dG_{Total}}{dw_7} = \frac{dE_{Total}}{dout_2} * \frac{dout_2}{dnet\ O_2} * \frac{dnet\ (O_2)}{dw_7}$$

$$= -(T_2 - O_2)O_2(1-O_2) * out(h1).$$

$$= -0.0225$$

$$\frac{dE_{Total}}{dw_8} = -(T_2 - O_2)O_2(1-O_2) * out(h2).$$

$$= -0.217 * 0.773 * 0.227 * 0.597$$

$$= -0.0227$$