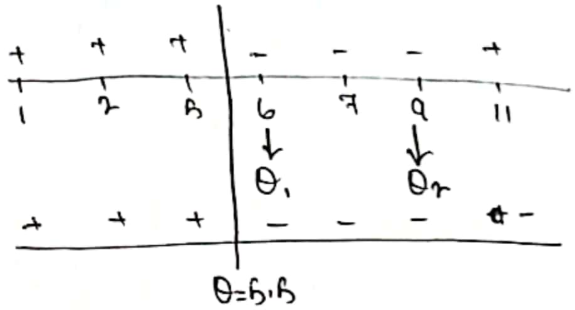
BOOSTING

Interval function :-

- $h_{\theta_1,\theta_2,b}(x) = \begin{cases} -b & \text{if } \theta_1 \le x \le \theta_2 \\ +b & \text{otherwise} \end{cases}$

Example :-

(1,+), (2,+), (5,+), (6,−), (7,−), (9,−), (11,+) → arranged in asc



$\theta = \frac{5+6}{2} = 5.5$, $b = 1$ → for threshold function

min Error = 1

Threshold function:-

- $h_{\theta,b}(x) = \text{sign}(\theta - x) \cdot b$

$$\theta = \frac{\theta_i + \theta_{i+1}}{2}$$

$b \in \{-1, 1\}$

Weak learners:-

A learning algorithm A is said to be λ- weak learner with $0 \le \lambda \le \frac{1}{2}$ for a hypothesis class H if there exists a sample size $m \ge \text{poly}(\frac{1}{\delta})$ such that for any $\delta \in (0,1)$, for any distribution D over X, for any labelling function $f: X \to \{1,-1\}$, if the realizability holds, then when the algorithm is run with a sample size of at least m, then the following holds:

- $P_r[R_D(h) \le \frac{1}{2} - \lambda] \le 1 - \delta$

AdaBoost

input:

training set $S = (x_1, y_1), \ldots\ldots, (x_m, y_m)$

weak learner WL

number of rounds T

initialize $D^{(1)} = \left(\frac{1}{m}, \ldots\ldots, \frac{1}{m}\right) \rightarrow$ initially, $D^{(1)}$ এর importance same থাকে

$\xrightarrow{\text{Distribution}}$

for $t = 1, \ldots\ldots, T$:

invoke weak learner $h_t = WL(D^{(t)}, S)$

generalisation Error

compute $\epsilon_t = \sum_{i=1}^{m} D_i^{(t)} 1_{y_i \neq h_t(x_i)} \rightarrow$ $y_i \neq h(x_i)$ হলে $\epsilon_t = \sum_{i=1}^{m} D_i^{(t)}$ । হবে $\left[\epsilon_t \leq \frac{1}{2} - \lambda\right]$

let $w^t = \frac{1}{2}\log\left(\frac{1}{\epsilon_t} - 1\right) \rightarrow w^t$ ছোট হবে $\left[w^t \propto \frac{1}{\epsilon_t}\right]$

update $D_i^{(t+1)} = \dfrac{D_i^{(t)} \exp(-w_i y_i h_t(x_i))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_j y_j h_t(x_j))}$ for all $i = 1, \ldots, m$

$\downarrow$

D_i update হবে

Output:

the hypothesis $h_S(x) = \text{sign}\left(\sum_{t=1}^{T} w_t h_t(x)\right)$

$\hookrightarrow$ weighted aggregate of the individual predictions of all the returned hypothesis.

* A data gets high probability if it had high probability in the previous round and also if the returned hypothesis makes a mistake on this data instance is identified as a problematic data instance.

* Empirical Risk bound of AdaBoost:-

• $R_S(h_{AB}) = \frac{1}{m}\sum_{i=1}^{m} 1_{h_{AB}(x_i) \neq y_i} \leq e^{-2\lambda^2 T}$

* Generalisation error:-

• $\sum_{i=1}^{m} D_i 1_{h(x_i) \neq y_i} \rightarrow R_D(h)$

• $R_{D,\theta}(h) = \sum_{i:y_i=1} D_i 1_{\theta < x_i} + \sum_{i:y_i=-1} D_i 1_{\theta \geq x_i}$ | • $R_{D,\theta_{i+1}} = R_{D,\theta_i} - D_i y_i$

4] SUPPORT VECTOR MACHINES

Convex function:-

- $f(\lambda u + (1-\lambda) v) \leq \lambda f(u) + (1-\lambda) f(v)$

  Condition:-

  At minimum x, $\nabla f(x) = 0$

Concave function:-

- $f(\lambda u + (1-\lambda) v) \geq \lambda f(u) + (1-\lambda) f(v)$

Tangent:-

- $f(x) \geq f(u) + \nabla f(u)^T (x-u)$

Sub-Gradient:-

$f(y) \geq f(x) + a^T (y-x)$

* যে x এর জন্য $f(x)$ কে differentiate করা possible, that is tangent & the remaining are sub-gradients.

SVM Objective function

$$\min_{w,b} \frac{1}{2} w^T w + C \sum \max(0, 1 - y_i(w^T x_i + b))$$

Here,

$w^T w$ = regularization form → maximise margin

C = hyper-parameter → controls trade-off bet^n a large margin & small hinge loss

$C \sum \max(0, 1 - y_i(w^T x_i + b))$ = hinge/empirical loss → penalizes weight vectors that make mistakes

Modify,

$w_0 \leftarrow w$; $w \leftarrow [w_0, b]$, $x_i \leftarrow [x_i, 1]$

$$\min_{w,b} \frac{1}{2} w_0^T w + C \sum \max(0, 1 - y_i w^T x_i)$$

Gradient Descent for SVM

(1) Initialize w^0

(2) For $t = 0, 1, 2 \ldots$

(i) Compute gradient of $J(w)$ at w^t. Call $\nabla J(w^t)$

(ii) Update w^t by taking w^{t+1} by taking a step in the opposite direction of the gradient.

$$w^{t+1} = w^t + \eta \nabla J(w^t)$$

Perceptron V/S SVM

$L_{perceptron}(y, x, w) = \max(0, -yw^T x)$

[no regularization]

Perceptron does not maximise margin width.

$L_{Hinge}(y, x, w) = \max(0, 1 - yw^T x)$

[regularization]

SVM optimises hinge loss

Stochastic Gradient Descent for SVM

A training set $S = \{(x_i, y_i)\}$, $x \in \mathbb{R}^n$, $y \in \{-1, 1\}$ is given.

(1) Initialize $w^0 = 0 \in \mathbb{R}^n$

(2) For Epoch $= 1 \ldots T$

(i) Pick a random example (x_i, y_i) from S.

(ii) Repeat (x_i, y_i) to make a full dataset and take derivative of the SVM objective at w^{t-1} to be $\nabla J^t(w^{t-1})$

$$J^t(w) = \frac{1}{2} w_0^T w_0 + CN \max(0, 1 - y_i w^T x_i)$$

(iii) Update:

$$w^t \leftarrow w^{t-1} - \gamma_t \nabla J^t(w^{t-1})$$

(3) Return w

Stochastic Sub-Gradient Descent for SVM

A training set $S = \{(x_i, y_i)\}$, $x \in \mathbb{R}^n$, $y \in \{-1, 1\}$ is given.

(1) Initialize $w^0 = 0 \in \mathbb{R}^n$

(2) For Epoch $= 1 \ldots T$

(i) For each training example $(x_i, y_i) \in S$:

if $y_i w^T x_i \leq 1$:

$w^{t+1} \leftarrow (1 - \gamma_t) w^t + \gamma_t C N y_i x_i$

else:

$w^t \leftarrow (1 - \gamma_t) w^t$

(3) Return w

* sub-gradient দিলে কোনো ধাপে up w কে update করা possible, but it takes time.

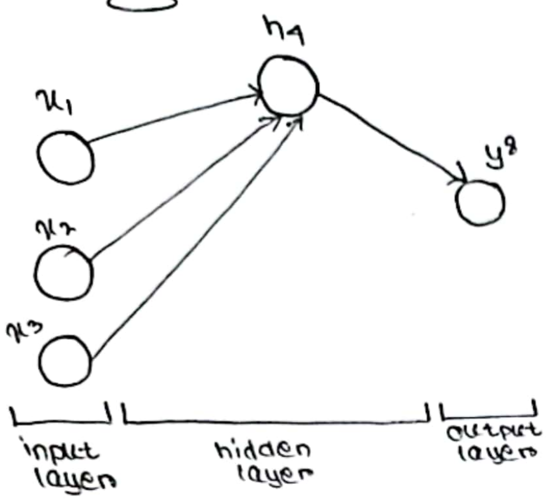
$\gamma_t = \dfrac{1}{1+t}$, $t \uparrow, \gamma_t \downarrow$; $t \downarrow, \gamma_t \uparrow$

Here, $\gamma_t \rightarrow$ learning rate

$t \rightarrow$ time

NEURAL NETWORK

Configuration

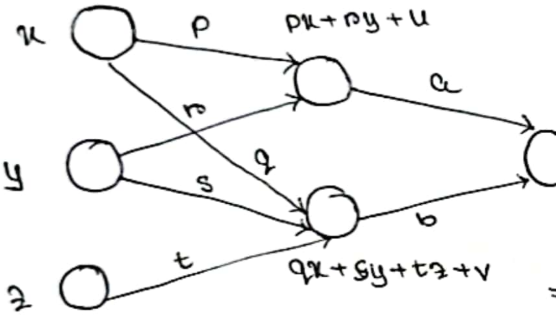


$$h_1 = \underset{\downarrow\, x_1}{w_{1,1} a_1} + \underset{\downarrow\, x_2}{w_{2,1} a_2} + \underset{\downarrow\, x_3}{w_{3,1} a_3} + b_1$$

$$a_1 = \frac{1}{1+e^{-h_1}}$$

$$y^8 = w_{h,8}\, a_1$$

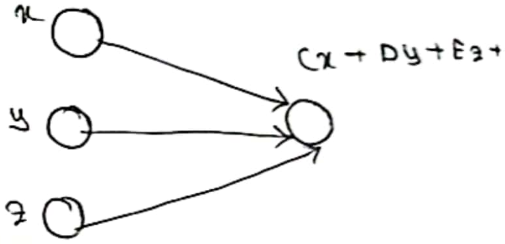
- identity function:- $f(x) = x$
- logistic function:- $f(x) = \frac{1}{1+e^{-x}}$



$$a(px+ry+u) + b(qx+sy+tz+v) + w$$

$$\Rightarrow x\underbrace{(ap+bq)}_{\to C} + y\underbrace{(ar+bs)}_{\to D} + \underbrace{zt}_{\to E} + \underbrace{v}_{\to F}$$

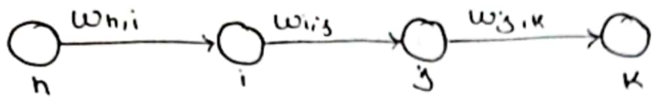
$\longrightarrow$



: input layer থেকে hidden layer, then hidden layer থেকে outer layer পাঠানোর জন্য activation function লাগে।

- Input layer has identity function as the activation function.
- Output layers have logistic function as the activation function.

Back Propagation



$$E = (t_k - a_k)^2$$

$$\frac{\partial E}{\partial w_{j,k}} = \frac{\partial}{\partial w_{j,k}} (t_k - a_k)^2$$

$$= \frac{\partial}{\partial w_{j,k}} (-2(t_k - a_k))$$

$$\frac{\partial}{\partial w_{j,k}} = \frac{\partial in_k}{\partial w_{j,k}} * \frac{\partial g(in_k)}{\partial in_k}$$

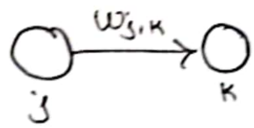
$$= a_j\, g(in_k)(1 - g(in_k))$$

$$\frac{\partial E}{\partial w_{j,k}} = a_j\, g(in_k)(1 - g(in_k))(-2(t_k - a_k))$$

$$g'(in_k) = g(in_k)(1 - g(in_k))$$

$$\Delta k = g'(in_k)(-2(t_k - a_k))$$

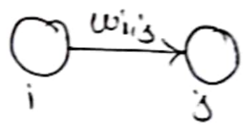
$$w_{j,k} = w_{j,k} - \lambda a_j \Delta k$$



$$\frac{\partial E}{\partial w_{i,j}} = -2 \sum_k \Delta k\, a_i\, w_{j,k}\, g'(in_j)$$

$$\Delta j = -2 \sum_k \Delta k\, w_{j,k}\, g'(in_j)$$

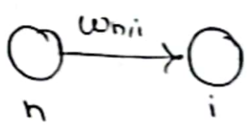
$$w_{i,j} = w_{i,j} - \lambda a_i \Delta j$$



$$\frac{\partial E}{\partial w_{h,i}} = -2 \sum_j \Delta j\, w_{i,j}\, a_h\, g'(in_i)$$

$$\Delta i = -2 \sum_j \Delta j\, w_{i,j}\, g'(in_i)$$

$$w_{h,i} = w_{h,i} - \lambda a_h \Delta i$$



Adding momentum to Back-Propagation

$$w_{i,j,n} = w_{i,j,n-1} + \alpha (w_{i,j,n-1} - w_{i,j,n-2}) + \lambda a_{i,n} \Delta_{j,n}$$

* $a_{i,n} \Delta_{j,n} = 0 \rightarrow$ GD doesn't stop

1] K-NEAREST NEIGHBOURS

* value of K is given

* distance between testing data and training data is calculated. $d = \sqrt{(x_0 - x_t)^2}$

* Sorted.

* K-number of shortest distance is checked for Yes/1/+ve and No/0/-ve.

* If there are more Yes/1/+ve than No/0/-ve, then the testing data falls under Yes/1/+ve class.

* If the value of K is odd, then there is no possibilities of no. of Yes to be equal to no. of No. However, if there's a tie, we can flip a coin to decide whether to decide to give the testing data a class or not.

* There's no correct way to determine best value for K. A small value (such as K=1 and K=2) can be noisy and subject to outliers.
A large value can smooth over things so can avoid noise.
But, a larger value can rule over results from small samples.