DEEP LEARNING (CCS6005)

DATE

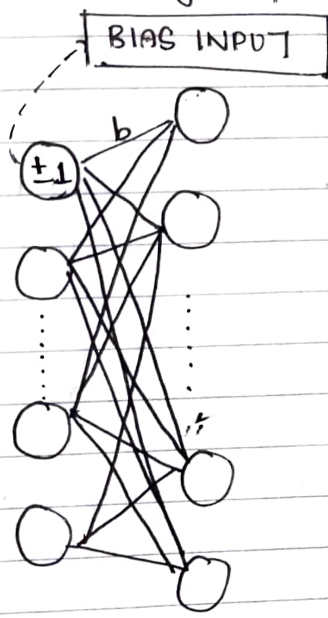
* PERCEPTRON: (SINGLE LAYER PERCEPTRON)

→ A linear model used for binary classification

→ Step Function for activation i.e $f(x) = \begin{cases} 0, & x \le \alpha \\ 1, & x > \alpha \end{cases}$

$\alpha \Rightarrow$ THRESHOLD VALUE.

→ Contains only input layer and output layer.



used to control whether the Neuron should fire or not fire if all the inputs are '0'

(m input)

(n output).

At each output neuron K we have,

$x_0 = \pm 1$ and $w_{0K} = b$ which is weight of bias

$$\boxed{\hat{y}_K = f\left(\sum_{i=0}^{m} x_i w_{iK}\right)} \Rightarrow \text{output of Kth neuron.}$$

⇒ The weights of the connections are update through perceptron learning algorithm.

At the K^{th} output neuron the error E_K is Given by

$$\boxed{E_K = \hat{y}_K - y_K}$$

where $\hat{y}_K$ = predicted value

y_K = actual value.

if $E_K > 0 \Rightarrow$ weights should be reduced

if $E_K < 0 \Rightarrow$ weights should be increased

$$\Rightarrow \boxed{\Delta w_{iK} = - x_i (\hat{y}_K - y_K)}$$

Therefore the weight updation is given by.

$$\boxed{w_{iK} = w_{iK} + \eta \Delta w_{iK} = w_{iK} - \eta \cdot x_i (\hat{y}_K - y_K)}$$

The parameter η is called as learning Rate. It determines how fast should the perceptron learn by controlling the amount of change in weight

if η is too high $\Rightarrow$ network becomes unstable

If η is too low $\Rightarrow$ learning will take large amount of time

$$\boxed{0.1 < \eta < 0.4} \Rightarrow \text{optimal Range}$$

* MULTILAYER PERCEPTRON (MULTILAYER FEED FORWARD NETWORK):

→ Neurons are arranged into groups called as layer

→ Contains 3 main layers: ① Input layer
② Hidden layers.
③ Output layer

⇒ On a given layer all the Neurons will have same activation function

⇒ For input layer, input will be the actual data vector. For other layers, input will be activation of previous layer

⇒ Number of neurons in the input layer will be equal to number of features in the input.

⇒ we have two main steps:

① Forward Propogation (Predicting Output)

② Backward Propogation (Backpropogation of error) (Reducing Error by tuning weight and bias value).

⇒ each neuron can be given a seperate bias value (OR) each layer can have a common bias value, This is based on the implementation

⇒ BACKPROPOGATION: it is the method of calculating and propagation error from output layer to all Hidden layer.

→ To reduce the error we using "optimisation algorithm"

⇓ ⓧ

$\boxed{\text{error is calculated using "loss function"} \Rightarrow \text{The goal of optimiser to to minimise the loss function}}$

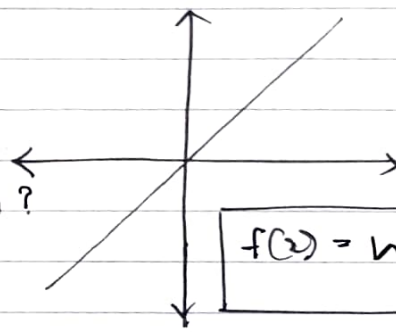
linear   Non linear

⇒ ACTIVATION FUNCTION : used to introduce non-linearity in the decision boundary (Because not all problems are linearly Seperable)

→ without [redacted] activation function the neural network will just perform linear weighted Sum (Similar to linear Regression).

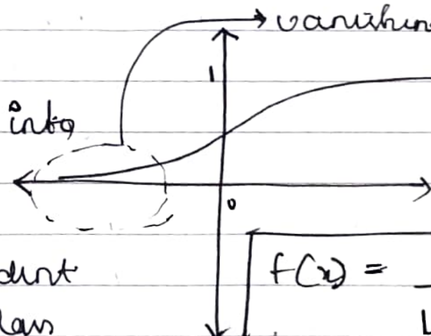
① LINEAR : $f(x) = Wx$.

⇒ Identity function
⇒ Used in input layer ?



$\boxed{f(x) = Wx}$

② SIGMOID :



→ vanishing gradient

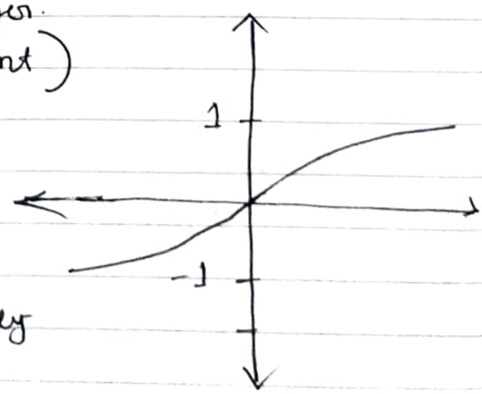
⇒ Reduce the input into the Range 0 - 1.

⇒ Outputs independent probability for each class

$\boxed{f(x) = \frac{1}{1 + e^{-x}}}$

[redacted]: → used in binary classification output layer.

③ TANH : (Hyperbolic Tangent)



⇒ Reduces input to the Range -1 to 1

⇒ it can deal more easily with negative values.

$\boxed{f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}}$

Both Sigmoid and tanh can cause vanishing Gradient problem (For extreme high / low value) ⇒ So we use ReLU.
But it can cause exploding Gradient
↳ need to scale input

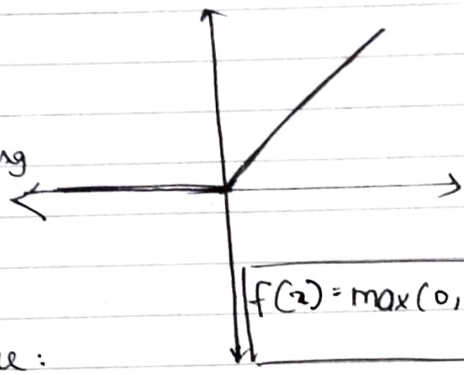
④ SOFTMAX:

→ provides the probability distribution of each class.
→ used in output layer for multi-class classification

$\cancel{f(x) = \frac{e^x}{\sum_i e^{x_i}}}$ ⇒ if output layer contains M neurons and X be the output value/activation

⇒ $\boxed{f(x_j) = \frac{e^{x_j}}{\sum_{i}^{M} e^{x_i}}}$ (Sum of all probability will be 1)
(class with maximum probability will be the Result).

⑤ RECTIFIED LINEAR UNIT (ReLU):



$\boxed{f(x) = \max(0, x)}$

⇒ overcomes vanishing / exploding gradient problem.
⇒ ReLU is highly used in hidden layer
⇒ Some variant of ReLU are:

(i) Leaky ReLU = $f(x) = \max(0.01x, x)$.

(ii) Parametric ReLU = $f(x) = \max(\alpha x, x)$.

(iii) Exponential ReLU = $f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha(e^x - 1), & \text{if } x < 0 \end{cases}$

⇓

$\max(\alpha(e^x - 1), x)$.

classmate

PAGE

DATE

LOSS FUNCTION: It quantifies how close the predicted Value is to the Real Value.

For Given weights W and Bais B we have $L(W,B)$ as loss function

$\hat{Y}_i = h_{W,B}(X) \Rightarrow$ predicted value for ith Sample
$Y_i \Rightarrow$ Real value for ith Sample

LOSS FUNCTION FOR REGRESSION

① Mean Squared Error

$$L(W,B) = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2$$

if more than one feature is dim in output

$$\Rightarrow \frac{1}{N}\sum_{i=1}^{N}\left(\sqrt{\sum_{j=1}^{M}(\hat{y}_{ij}-y_{ij})^2}\right)^2$$

$$\Rightarrow \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}(\hat{y}_{ij}-y_{ij})^2$$

② Mean Absolute Error

$$L(W,B) = \frac{1}{N}\sum_{i=1}^{N}|\hat{Y}_i - Y_i|$$

$$\Rightarrow \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}|\hat{y}_{ij}-y_{ij}|$$

③ Root Mean Squared

$$L(W,B) = \sqrt{MSE}$$

LOSS FUNCTION FOR CLASSIFICATION

Cross entropy loss.
(OR)
Negative Log Likelihood

↳ Binary
↳ Categorical.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} Y_i \log \hat{Y}_i + (1-Y_i)\log(1-\hat{Y}_i)$$

$$CCE = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} Y_{ij}\log(\hat{Y}_{ij})$$

Hinge loss:
↳ designed for SVM

Result should be $-1/1$.

$$L(W,B) = \frac{\sum_{i=1}^{N}\max(0, 1-Y_i \times \hat{Y}_i)}{N}$$

classmate

PAGE

DATE

$$\sum_{i=1}^{N}\sum_{j=1}^{M} \max(0, y_{ij} \times \hat{y}_{ij}) \Rightarrow$$ only used for binary classification

MSE and MAE are prone to outliers due to which we can use $\boxed{\text{HUBER LOSS}}$ → For Regression

↳ $$L(W,B) = \begin{cases} \frac{1}{2}(Y_i - \hat{Y}_i)^2, & |Y_i - \hat{Y}_i| \le \delta \\ \delta\,|y_i - \hat{Y}_i| - \frac{1}{2}\delta^2, & \text{else.} \end{cases}$$

Here δ is a hyperparameter which depicts the tolerance to outliers.

Huber loss is like a combination MSE and MAE

* GRADIENT DESCENT :

→ An optimisation algorithm for ~~...~~ minimising the cost / loss function

(i) Batch Gradient Descent
(ii) Stochastic Gradient Descent
(iii) Mini Batch Gradient Descent

⇒ In (i) the complete training set is used for every iteration/epoch.

⇒ In (ii) Only single sample is used for one iteration (i.e. Number of iterations/epoch = Number of samples in training set minimum).

⇒ In (iii) A Batch of Samples used for each iteration from the training dataset.

⇒ Some other optimisers are : RMSProps, ADADelta, ADAGrad, Adam.

* HYPERPARAMETERS :

⇒ Configuration options ~~that~~ can be varied or altered to improve the performance of model

1. No. of. Hidden Layers
2. No. of. Neurons in each layer
3. Activation Functions
4. Loss Function
5. Optimizer Function
6. No. of. ephochs / Iterations
7. Type of Regularisation
8. Weight initialisation
9. Learning Rate & Momentum.

* BIAS AND VARIANCE : (Trade off).

⇒ when a model is trained it captures the patterns in the training data and ~~try~~ to generalize it.

⇒ BIAS is the measure of our models inability to learn patterns from data set.

⇒ when model has HIGH BIAS ⇒ did not train well in training set

when model has LOW BIAS ⇒ it has identified necessary patterns from Training Set properly

⇒ VARIANCE is the measure of models fluctuations in performance when there is a change in data set.

DATE

⇒ UNDERFITTING: when a model does not perform well in training data.
(HIGH BIAS).

⇒ OVERFITTING: when a model perform well in train data but it lacks performance in testing data
(LOW. BIAS, HIGH VARIANCE)

⇒ A Good model should have LOW BIAS and LOW VARIANCE.

MACHINE LEARNING: A Algorithm is said to be learning from experience E with Respect to Some task T and performance measure P if the performance P increases with increase in experience E.

TYPES OF MACHINE LEARING:

① SUPERVISED → CLASSIFICATION, → REGRESSION

② UNSUPERVISED → CLUSTERING.

③ SEMI-SUPERVISED

④ REINFORCEMENT.

LINEAR SEPERABILITY:

A data is said to be linearly Seperable if only a single line / Plane / Hyperplane can partition the data into 2 classes.

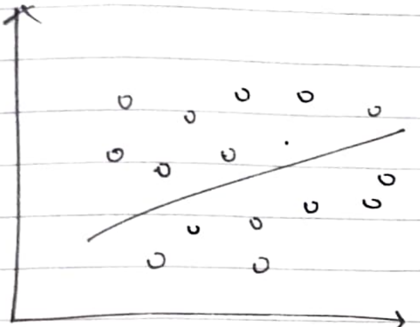
DATE

Eg: Perceptron can only work on a data set that is linearly seperable.

→ One of the main reason for MLP improvement

LINEAR REGRESSION :

consider the following equation $y = \theta_0 x + \theta_1$



Linear regression tries to fit a line of this form on the given dataset.

$$E = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

⇒ where $\hat{y}_i$ is the predicted value and y is the actual value

The error E is called as MEAN SQUARED ERROR. we have to Reduce this hence this is called LEAST SQUARE REGRESSION.

$$E = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(\theta_0 x_i + \theta_1 - y_i)^2.$$

$$\frac{\partial E}{\partial \theta_0} = \cancel{2} \times \frac{1}{N} \times \sum_{i=1}^{N}(\theta_0 x_i + \theta_1 - y_i)(x_i + 0 - 0) = 0$$

$$= \theta_0 \overline{x^2} + \theta_1 \bar{x} - \overline{x \cdot y} = 0. \quad -(1)$$

$$\frac{\partial E}{\partial \theta_1} = \cancel{2} \times \frac{1}{N} \times \sum_{i=1}^{N}(\theta_0 x_i + \theta_1 - y_i)(0 + 1 - 0) = 0$$

$$= \theta_0 \bar{x} + \theta_1 - \bar{y} = 0$$

$$\Rightarrow \quad \theta_1 = \bar{y} - \theta_0 \bar{x} \quad -(2)$$

From (1) and (2) we have.

$$\theta_0 \overline{x^2} + (\bar{y} - \theta_0 \bar{x}) \cdot \bar{x} - \overline{x \cdot y} = 0.$$

$$\theta_0 \overline{x^2} + \bar{x} \cdot \bar{y} - \theta_0 \bar{x}^2 - \overline{x \cdot y} = 0$$

$$\theta_0 = \frac{\overline{x \cdot y} - \bar{x} \cdot \bar{y}}{\overline{x^2} - \bar{x}^2}$$

$\Rightarrow$

$$\boxed{\theta_0 = \frac{\overline{x \cdot y} - \bar{x} \cdot \bar{y}}{\overline{x^2} - \bar{x}^2}}$$

$$\boxed{\theta_1 = \bar{y} - \theta_0 \cdot \bar{x}}$$

$\Rightarrow$ why Huber loss for Regression?

MSE is Highly Sensitive to outliers because we Square the over all error values. MAE just take the absolute value of error it give same weightage to all errors due to which outlier can be completely Neglected. Huber Loss Provide Balance (δ).

DATE

* BAYESIAN STATISTICS :

↳ (methods that used Bayes Theorem)

⇒ BAYES THEORM : Finds the probability of an event occuring, given the probability of another event that has already occured

↳ (called as conditional probability

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

⇒ Here, •) $P(B)$ is called as evidence.

•) $P(A)$ is called as prior probability (i.e, probability occurance of event A without considering event B).

•) $P(A \mid B)$ is called as posterior Probability (i.e, probability of occurance of event A with considering event B)

⇒ we can $P(Y \mid X)$ using above method.

Y ⇒ Target Class and X = Feature Vector.

⇒ $P(Y_j \mid X_1, X_2, \dots, X_M)$ =

$$= \frac{P(Y_j \mid X_1) \cdot P(Y_j \mid X_2) \cdots P(Y_j \mid X_M) \cdot P(Y_j)}{P(X_1) \cdot P(X_2) \cdots P(X_M)}$$

since evidence is Same. for all target we can ignore this

⇒ Result will be class j having max $P(Y_j | X)$.

→ There are various variation of Naive Bayes algorithms like

(i) Gaussian / Normal NB
(ii) Multinomial NB
(iii) Bernoulli NB
⋮

⇒ Important assumption for Naive Bayes algorithm is all the features are independent.

$\boxed{\text{i.e} \quad P(x_i | x_j) = P(x_i)}$ X

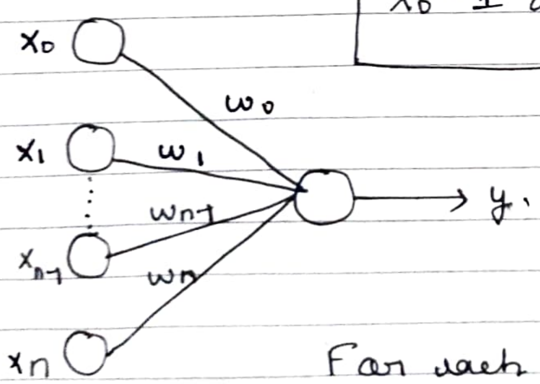
— x —

* HEBBIAN LEARNING RULE (HEBB NETWORK)

→ inputs can only be 1 and −1

→ initialise all weights and bias to 0. in starting

$\boxed{x_0 = 1 \text{ and } w_0 = 0 \text{ (Bias)}}$



$\boxed{w_1 \text{ to } w_N = 0}$

For each training sample we do the following

$\Rightarrow$ $\Delta w_i = x_i y$ and $w_i = w_i + \Delta w_i$

— x —

DATE

* FORMULAS

① PERCEPTRON WEIGHT UPDATION:

$$W_{ij} = W_{ij} + \eta(\hat{Y}_j - Y_j)(x_i)$$

② LOSS FUNCTIONS:

(i) REGRESSION:

⇒ MEAN SQUARED ERROR:

$$E = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(\hat{y}_{ij} - y_{ij})^2}{N} \quad (\text{OR}) \quad E = \frac{\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2}{N}$$

⇒ MEAN ABSOLUTE ERROR.

$$E = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}|\hat{Y}_{ij} - Y_{ij}|}{N} \quad (\text{OR}) \quad E = \frac{\sum_{i=1}^{N}(\hat{Y}_i - Y_i)}{N}$$

⇒ ROOT MEAN SQUARED ERROR

$$E = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(\hat{Y}_{ij} - Y_{ij})^2}{N}} \quad (\text{OR}) \quad E = \sqrt{\frac{\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2}{N}}$$

$$\Rightarrow \boxed{E_{RMSE} = \sqrt{E_{MSE}}}$$

⇒ HUBER LOSS (A Balance between MSE and MAE).

$$E = \begin{cases} \frac{1}{2}(Y_i - \hat{Y}_i)^2, & \text{if } |Y_i - \hat{Y}_i| \le \delta \\ \delta|Y - \hat{Y}_i| - \frac{1}{2}\delta^2, & \text{otherwise} \end{cases}$$

classmate

X: δ value controls the behaviour of huber loss (Hyperparameter).

PAGE

DATE

(ii) CLASSIFICATION :

⇒ HINGE LOSS :

⇒ CROSSENTROPY LOSS :

$$E = -\sum_{i=1}^{N}\sum_{j=1}^{M} Y_{ij} \log \hat{Y}_{ij}$$ (Multiple Class ⇒ Categorical CE).

$$E = -\sum_{i=1}^{N} Y_i \log \hat{Y}_i + (1-Y_i) \log (1-\hat{Y}_i).$$

(Binary Class ⇒ Binary CE).

③ SOME ACTIVATION FUNCTION :

(i) Sigmoid :

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

(ii) TanH :

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

(iii) ReLU :

$$g(x) = \max(0, x)$$

→ Parametric : $g(x) = \max(ax,$

→ Leaky : $g(x) = \max(0.01x, x)$

→ Exponential : $g(x) = \max(\alpha(e^{x}-1), x$

(iv) Softmax:

$$g(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{M} e^{x_j}}$$

(v) LINEAR:

$$g(x) = \alpha x.$$

④ NAIVE BAYES:

For M target classes we have to find max of

$$P(Y_j \mid x_1, x_2, x_3, \ldots, x_k) = \frac{P(x_1 \mid Y_j) \cdot P(x_2 \mid Y_j) \cdots P(x_k \mid Y_j) \cdot P(Y_j)}{\boxed{P(x_1) \cdot P(x_2) \cdots P(x_k)}}$$

Same for all So can be ignored.

⑤ MULTILAYER PERCEPTRON:

$$E_k = O_k (1 - O_k)(O_k - T_k). \quad \text{(Output layer)}$$

$$E_j = O_j (1 - O_j)\left(\sum_{k=1}^{N} W_{jk} E_k\right). \quad \text{(Hidden layer)}.$$

$$W_{ij} = W_{ij} + l \, O_i E_j$$