Date:

Sun Mon Tue Wed Thu Fri Sat

Perceptron: Basic building block of an ann.
- used for binary classification.
- Supervised. learning.

Components:

1- **Inputs:** Perceptron takes binary inputs or real valued inputs.

2- **Weights:** parameters that the perceptron learns. It determine influence of each input on output

3- **Summation function:** perceptron computes the weighted sum

$$z = \sum_{i=0}^{n} w_i x_i$$

4- **Activation function:** The weighted sum is passed to activation function to get output in binary. Threshold or sign function.

-weight update: During training, Based on diffrence in true & predicted

$$w_i = w_i + (\eta \cdot y_{act} \cdot x_i)$$

Limitations:
- Only learn linear Seprable data : bcz it finds single linear decision boundry
- only binary outputs.
- Sensitive to noise
- Single layer so unable to learn complex relations
- Original weight update lacks adaptive mechanism.

Bias: - Shifts the activation function
- Adjust the decision boundary.
- handle assymetry in data
- improve flexibily

Learning rate: Controls the size of step during weight update

Date:

Sun Mon Tue Wed Thu Fri Sat

Multilayer perceptron:

- type of ANN made by multiple layers of neurons.
- Include input layer, 1 or more hidden layers and an output layer.
- capable of learning complex data

How Better than Single layer:

- layers learn progressively
- Network extract meaningful features from data.
- Shared parameters aid generalization.

Gradient Learning:

- Gradient is a vector that points to steepest ascent
- We want the minimum loss so we descent to the minima.
- Magnitude of gradient tells the rate of change of function
- In gradient decent the gradient guides the updates of parameters.
- As the gradient becomes smaller tells that model is reaching a minimum.

Gradient Descent:

It is an optimizing algorithm. used for finding minimum of a function.

- goal is to move towards the minimum of loss function by updating parameters.

Date:

Sun Mon Tue Wed Thu Fri Sat

Steps: 1- Initialize parameters.
2- Define the cost functions
3 ~~Update Parameters~~. 3- compute Gradient
4- update weights
5- iterate until minimum

Loss function = $\frac{1}{m} \sum_{i=0}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2$

new_parameter = oldparameter - LR × $\frac{\partial Loss}{\partial parameter}$.

Learning Rate: - hyperparameter
- Determine size of step taken toward descent

Too Small LR: Converge slow, stuck at local minima.

Too big LR: Diverge, miss out minima.

Convergence: occurs when:
1- All point correctly classified
2- Error reaches global minima.
3- optimal weights are set
4- LR is balanced.

Pattern Detection:
- Perceptron can identify black & white patterns.
- Struggles with grey pixels.

Date:

Sun Mon Tue Wed Thu Fri Sat

Multi Layer Neural Network

- forward pass : A learning example is fed to network and the output is compared with ground truth

- Backward pass: compute the derivative with respect to weights then weights are updated to make an optimized network.

Activation Function:

- Sign / Threshold function not differentiable.
- Discontinous at 0
- Sigmoid & tanh symmetric at 0
- tanh in hidden & Sigmoid at output.

$$\text{Sigmoid} = \frac{1}{1+e^{-x}} \qquad S'(x) = S(x)\cdot(1-S(x))$$

$$\tanh = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \qquad \tanh'(x) = 1-\tanh^2(x)$$

Backpropagation:

- combines forward pass & backward parts to compute & propagate gradients.

- Iteratively adjusts weights to minimize error.

Date:

Sun Mon Tue Wed Thu Fri Sat

Steps:

1- Initialization: initialize weight & biases with random values

2- forward pass: 1- Input training set to network
2- Apply activation function at each layer
3- Compute predicted outputs.

3- Compute loss: 1- Compare predicted output with actual output
2- Use a loss function

4- Backward pass: 1- Compute gradient of loss wrt. activation function
2- Compute gradient at each layer.

5- Optimization: 1- Update the weights & biases

6- Iterative: Repeat 2, 3, 4 until minimum loss, or no. of epochs.

function Composition:

1- NN are compositions of functions with each acting as a function applied to output of previous layer

2- differentiating error wrt parameter helps minimize loss

3- Chain rule breaks computation of gradient layer by layer

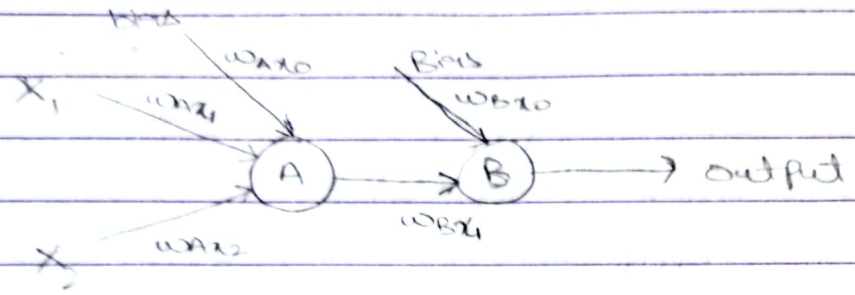
4- ~~Chain rule~~ By applying chain rule errors can be efficiently, guiding parameter update.

3

5-

Date:

Sun Mon Tue Wed Thu Fri Sat



Neuron A = tanh Neuron B = Sigmoid

error : MSE

Weights: Neu-A : W_{AX0}, W_{AX1}, W_{AX2}

Neu-B : W_{BX0} , W_{BX1}

$$MSE = \frac{1}{n} \sum (y_{pred} - y_i)^2$$

$$Neu\_A = \tanh\underbrace{\left(W_{AX0} + X_1 \cdot W_{AX1} + X_2 W_{AX2}\right)}_{Z_A}$$

$$Neu\_B = S\underbrace{\left(W_{BX0} + Neu\_A \cdot W_{BX1}\right)}_{Z_C} \rightarrow y_{pred}\ or\ f$$

$$Error = \left(y_{act} - S\left(W_{BX0} + W_{BX1} \cdot \tanh\left(W_{AX0} + X_0 W_{AX1} + X_2 W_{AX2}\right)\right)\right)$$

$$Error = \frac{(y - \hat{y})^2}{2}$$

idk 2 kahan se aya.

Error :

Date:

Sun Mon Tue Wed Thu Fri Sat

→ $\frac{\partial error}{\partial W_{B,0}} = \frac{\partial error}{\partial ypred} \cdot \frac{ypred}{Z_B} \cdot \frac{Z_B}{W_{B,0}}$

→ $\frac{\partial error}{\partial W_{B,1}} = \frac{\partial error}{\partial ypred} \cdot \frac{ypred}{Z_B} \cdot \frac{Z_B}{W_{B,1}}$

→ $\frac{\partial error}{\partial W_{A,0}} = \frac{\partial error}{\partial ypred} \cdot \frac{ypred}{Z_B} \cdot \frac{Z_B}{New\_A} \cdot \frac{New\_A}{Z_A} \cdot \frac{Z_A}{W_{A,0}}$

→ $\frac{\partial error}{\partial W_{A,1}} = \frac{\partial error}{\partial ypred} \cdot \frac{ypred}{Z_B} \cdot \frac{Z_B}{New\_A} \cdot \frac{New\_A}{Z_A} \cdot \frac{Z_A}{W_{A,1}}$

→ $\frac{\partial error}{\partial W_{A,2}} = \frac{\partial error}{\partial ypred} \cdot \frac{ypred}{Z_B} \cdot \frac{Z_B}{New\_A} \cdot \frac{New\_A}{Z_A} \cdot \frac{Z_A}{W_{A,2}}$

$\frac{\partial error}{\partial ypred} \cdot \frac{\partial ypred}{\partial Z_B}$ common in all

f, ypred same cheez

* $\frac{\partial error}{\partial ypred} = \frac{\partial \frac{(y-f)^2}{2}}{\partial ypred} = \frac{-2(y-f)}{2} = -(y-f)$

* $\frac{\partial ypred}{Z_B} = \frac{\partial S'(Z_B)}{\partial Z_B} = S'(Z_B)$

$\frac{\partial Z_B}{\partial New\_A}$, $\frac{\partial New\_A}{\partial Z_A}$, ∂ common in last 3

$S' = S.(1-S)$

$\tanh' = (1 - \tanh^2)$

Date:

Sun Mon Tue Wed Thu Fri Sat

* $\frac{\partial Z_B}{\partial NeuA} = \frac{\partial (W_{B0} + NeuA \cdot W_{B1} x_1)}{\partial NeuA} = W_{B1} x_1$

* $\frac{\partial NeuA}{\partial Z_A} = \frac{\partial (\tanh(Z_B))}{\partial Z_A} = \tanh'(Z_B)$

Calculating:

→ $\frac{\partial Z_A}{\partial W_{B0}} = \frac{\partial (W_{B0} + NeuA \cdot W_{A1} x_1)}{\partial W_{B0}} = 1$

→ $\frac{\partial Z_A}{\partial W_{B1}} = \frac{\partial (W_{B0} + NeuA \cdot (W_{A1} x_1))}{\partial W_{B1}} = NeuA$

→ $\frac{\partial Z_B}{\partial W_{A0}} = \frac{\partial (W_{A0} + X_1 W_{A1} + X_2 W_{A2})}{\partial W_{A0}} = 1$

→ $\frac{\partial Z_B}{\partial W_{A1}} = X_1$ → $\frac{\partial Z_B}{\partial W_{A2}} = X_2$

Weight updating:

→ $\frac{\partial e}{\partial W_{B0}} = -(y-f) \cdot S'(Z_B) \cdot 1$

→ $\frac{\partial e}{\partial W_{B1}} = -(y-f) \cdot S'(Z_B) \cdot NeuA$

→ $\frac{\partial e}{\partial W_{A0}} = -(y-f) \cdot S'(Z_B) \cdot W_{B1} \cdot \tanh'(Z_A) \cdot 1$

→ $\frac{\partial e}{\partial W_{A1}} = -(y-f) \cdot S'(Z_B) \cdot W_{B1} \cdot \tanh'(Z_A) \cdot X_1$

Date:

Sun Mon Tue Wed Thu Fri Sat

Data Sets:

- good data set must contain over representation (diversity) of the real world

1- Handle missing data: 1- Imputation 2- Deletion

2- Handle categorical data: 1- One hot encoding 2- Label encoding
3- Ordinal encoding

3- Scaling Normalization: 1- MinMax Scaling 2- Standardization.

4- Handle outliers: 1- Identification 2- transformation (log, sqrt)
3- Removal

5- Feature Engineering: 1- Create New feature 2- Reduce feature

6- Handle Imbalance data: 1- Resampling 2- Sythetic Data Generation

Overfitting:

Memorizing → Overfitting

1- Increase training set

2- Early Stopping

Hyperparameter techniques:

1- Grid Search: Grid of parametrs

2- Random Search: Choose random combinations

3- Bayesian Optimization: probabilistic model for parameters.

4- Gradient-based: Use Gradient based algos

5- Ensemble: Combine different models into one.

6- Automated: in

Date:

Sun Mon Tue Wed Thu Fri Sat

Training: Coming up with weights for the network is done before deploying into production

inference: using network without updating weights

Optimal Model.

- Make data not biased
- Preprocess the data
- Split train, test, validation
- Proper loss functions
- Suitible optimizer
- Proper hyperparameter tuning.

Batch Size :- faster Convergence
- Efficient memory usage

Bias:

- Simple model complex data
- Don't capture patterns
- leads to under fitting

* Use complex model

* Add features
* Add polynomial feats
* Decrease λ

Variance:

- Model's sensitivity to noise
- overly complex model captures noise
- leads to over fitting

* Simplify model

* Reduce feat
* Reduce poly feat
* Increase λ

Regularization: introduces penalty terms to reduce influence of inputs on outputs

Lasso L1: - adds sum of absolute values to loss function
- Encourages sparsity, Some coefficient may become 0

Ridge L2: - adds sum of squared values to the loss function
- Spreads out the impact of each feature across all features, reducing impact of any single feature

λ → Regularization Strength: controls tradeoff between underfit & overfit

→ Prevent of overfit, Encourage Simplicity, Improve Generalization