/

DL Assignment

Unit 1

- Bishal Saini.

Q1. Discuss various norm form for Deep Learning?

→

- In Deep Learning, norms are mathematical tools used to measure the size, magnitude, or length of vectors and matrices.
- They play a crucial role in optimization, regularization and defining loss function.
- Different types of norms provide different properties, which can influence the learning behaviour of neural networks.
- The most commonly used norms forms in Deep learning are as follow

1. L0 Norm :- - The L0 norm is not true norm in the mathematical sense, but it counts the number of non-zero elements in a vector

Formula :- $||x||_p = \left(\Sigma |x_i|^p\right)^{1/p}$

2. L2 Norm :- - It is also known as euclidean norm as it measures the straight lines or euclidean distance from the origin ~~part~~ point x
- It is easy for ~~the~~ optimization and penalize large values more strongly.

Formula :-

$$||x||_2 = \sqrt{\Sigma_i x_i^3}$$

3. L1 norm :- - It is also known as Manhattan norm
- It measures distance by summing absolute values
- It is robust to outliers compared to L2

Formula :-

$$||x||_1 = \Sigma_i |x_i|$$

4. L∞ Norm :- - It is known as max norm
- It takes the largest absolute value among all elments
- It represent the least case deviation in any direction.

Formula :-

$$||x||_\infty = \max |x_i|$$

5. Lp Norm :- The most general family of norms is the Lp norm
- It measures the length of size of a vector in p-dimensional space

Formula :-

$$||x||_p = (\Sigma_i |x_i|^p)^{1/p}$$

2. Illustrate the use of Jacobian and Hessian matrix. What is gradient based matrix.

→

i) Jacobian Matrix :- - The Jacobian Matrix is the collection of first order partial derivatives of vector-valued function

- It also generalized the gradient for vector valued functions

- For a function $f: R^n \to R^m$;

$$J_{ij} = \frac{\partial f_i}{\partial x_j}$$

For E.g:- In backpropagation the chain rules relies on jacobian matrices to propogate gradient efficiently through layers.

ii) Hessian Matrix :- - The hessian captures second-order derivates

- It is a squared matrix of second-order partial derivates.

- For a scalar function $f(x)$, the hessian is an $n \times n$ symetric matrix:

$$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

- If hessian eigen values are large, optimization may suffer from sharp minimal or sadle points.

Gradient Based Optimization :-

- The core principle of trainning neural network is to minimize a loss function $J(\theta)$ with respect to parameter

- Gradient Descent :- Updates parameter in the opposite direction of the gradient

$$\theta \leftarrow \theta - \eta \nabla_\theta J(\theta) \quad (\eta \text{ is learning rate})$$

- First order methods :- Use gradients
- 2^{nd} order methods :- Use curvature to adapt step sizes

Unit 2

1 What is Dropout
->

- Dropout is a regularization techniques used in Deep learning to prevent overfitting.
- The core idea is simple, during trainning randomly "dropout" (deactivate) a fraction of neurons in the network

- Dropout Working :- - At each training step, every neuron has a ~~proab.~~ probability p of being set to zero (dropped)
- Now it creates different subnetwork for each iterations
- During testing all neurons are used but their outputs are called $(1-p)$ so that the expected activation remain constant

Mathematical Formula :-
If h is a vector of hidden activations :-
$$h' = h \cdot r, \quad r \sim \text{Bernoulli}(p)$$
where r is an random binary used

At test time
$$h_{test} = (1-p)h$$

E.g :- Suppose we have a hidden layer with 6 - neuron's
- Without Dropout :- All six are active every iteration
- With Dropout :- $(p = 0.5)$:- Only 3 neurons are active iteration and the choosen neuron changes randomly across steps.

2. Explain the Algorithm
→

a] Stochastic Gradient Descent :- Instead of using the entire dataset, SGD updates parameters using one sample or mini batch at a time

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t; x_i; y_i)$$

- Advantage :- Faster, good for large datasets and helps to escape local minima

- Disadvantage :- It can be noisy and unstable

b] Momentum Algorithm :- It adds velocity term to update, which accumulates past gradients to smooth out oscillations.

- $v_i = \beta v_{i-1} + \eta \nabla L(\theta_t)$

$$\theta_{t+1} = \theta_i - v_i$$

- Like a ball rolling downhill, momentum helps accelerate in the right direction

c] Nesterov Accelerated Gradient (NAG) :- It is an improvent over momentum that looks ahead befor updating.

$$v_i = \beta v_{i-1} + \eta \nabla L(\theta_i - \beta v_{t-1})$$
$$\theta_{t+1} = \theta_t - v_t$$

- It provides more accurate update by considering the future positions.

d] AdaGrad :- It adjust learning rate individually for each parameter, scaling by past squared gradients.

$$g_t = \nabla L(\theta_t)$$
$$G_t = G_{t-1} + g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} g_t$$

- It works well for sparse data
- The learning rates keeps shrinking - may stop learning too early.

e] RMS Propogation (Root Mean Square)- It's a refinment of AdaGrad that uses an exponentially decaying average of past square gradients instead of summing them forever.

- It prevents learning rate from shrinking too much

Formula :- $E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta) g_t^2$

$$\theta_{t+1} = \theta_t = \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}}$$

f) Adam (Adaptive Moment Estimation) - It Combines Momentum + RMS (Root Mean Square) propagation

- It tracks both :-
  - 1st moment (means of gradient)
  - 2nd moment (variance of gradient)

Rule:-

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g^2_t$$

g) RMS Prop with Momentum:- It combines RMS Prop's adaptive learning rate with Momentum acceleration

Rule:-

$$V_t = \beta v_{t-1} + \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} g_t$$

$$\theta_{t+1} = \theta_t - v_t$$

Unit 3

1. What are RNN. Explain the various representation.

→

- A RNN (Recurrent Neural Network) is a class of neural network designed to process sequential data by maintaining an internal hidden state or memory.
- unlike tradition FFN network which assume inputs and outputs are independents, an RNN uses information from previous steps in the sequence to influence the processing of the current steps.

- Mathematical Representation

$$h_t = f(W_h h_{t-1} + W_x x_t + b)$$
$$y_t = g(W_y h_t + c)$$

x_t = input at time step t
h_t = hidden ~~con~~ state at time t
y_t = output at time t
W = weight matrices
f, g = activation functions

- Various Representation of RNNs

a] One to One :- A single i/p produces a single o/p. This is the standard configuration of ffn

Use Case :- Image ~~Generati~~ Classification
$T_x = 1$, $T_y = 1$

Page No.
Date

b] One to Many :- A single i/p is processed to generate a sequence of o/p. The hidden state is updated and passed along to generate subsequent o/p.

- Use Case :- Music Generation
- $T_x = 1$, $T_y > 1$

c] Many to One :- A sequence of i/p processed and a single o/p is generated at very end, which summarized the entire sequence.

- Use Case :- Sentiment Analysis
- $T_x > 1$, $T_y = 1$

d] Many to Many (with different lengths) :-
- I/p sequence length ≠ o/p sequence length

- Use Case :- Video Captioning

- $T_x = T_y$ (Synchronised OR
- $T_x \neq T_y$ Asynchronised)

Page No.
Date

2 → Describe LSTM model

- A Long Short term Meomery is a model is a type of RNN designed to overcome the limitations of traditional RNN, particularily the problem of learning long-term dependencies in sequential dates, which is often hindered by the vanishing gradient problem.

- Structure of LSTM:-

a] Cell State (C_t) - Carries long-term memory
b] Hidden state (h_t) · ~~Short term information for output~~
c] Gates :- Control the flow of information
- Forget Gate
- I/p gate
- o/p gate

- Working of LSTM

a] Forget Gate :- Decides what information to discard from cell size.

$$f_t = \sigma(w_f \cdot [h_{t-1}, x_t] + b_t]$$

b] Input Gate :- Decides what new information to store

$$i_t = \sigma(w_i \cdot [h_{t-1}) [t] + b_i)$$

$$C_t = \tanh(w_c \cdot [h_{t-1}, x_t] \; b_c]$$

c] update cell Gate :- Combines old memory with new information

$C_t = f_t + C_{t-1} + i * t)$

d] o/p Gate :- Decides what part of the cell state to o/p

$o_t = \sigma(w_o.[h_{t-1}, x_t] + b_o)$

$h_t = O_t + \tanh(C_t)$

- The cell state acts like a conveyor belt that carries long-term memory with minor modifications.
- Gates mechanism allows sensitive rembering or forgetting.
- It's application are Speech Recognition, Time Series Forecasting, video processing.

4/10/25