ML ASSIGNMENT-1
FRACTAL-3

NAME-RAVI CHOUDHARY
ROLL NO-M20AIE288

Problem 3: MISC

Q-3.1-Compute derivative of following activation functions: (3 points)
(i) Sigmoid
(ii) tanh
(iii) ReLU

Solution:

Date.....

Problem - 3:

1. Derivative of Activation function.

1) Sigmoid → Logistic function

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

$$\frac{d}{dx}\text{sigmoid}(x) = \frac{d}{dx}(1+e^{-x})^{-1}$$

$$= (-1)(1+e^{-x})^{-2}\frac{d}{dx}(1+e^{-x})$$

$$= (-1)(1+e^{-x})^{-2}\left(\frac{d}{dx}1 + \frac{d}{dx}e^{-x}\right)$$

$$= (-1)(1+e^{-x})^{-2}\left(0 + e^{-x}\frac{d}{dx}(-x)\right)$$

$$= (-1)(1+e^{-x})^{-2}(-1)e^{-x}$$

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

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

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

$$= \frac{1}{(1+e^{-x})}\cdot\left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right)$$

$$= \frac{1}{(1+e^{-x})}\cdot\left(1 - \frac{1}{(1+e^{-x})}\right)$$

$$\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$$

$\sigma(x)$ → sigmoid function.

2. tanh(x) → Hyperbolic tangent

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

We know that

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)}$$

$$\frac{d}{dx}\tanh(x) = \frac{d}{dx}\frac{\sinh(x)}{\cosh(x)}$$

$$\frac{d}{dx}\frac{\sinh(x)}{\cosh(x)} = \frac{\cosh(x)\frac{d}{dx}\sinh(x) - \sinh(x)\frac{d}{dx}\cosh(x)}{\cosh^2(x)}$$

$$= \frac{\cosh^2(x) - \sinh^2(x)}{\cosh^2(x)}$$

$$= 1 - \frac{\sinh^2(x)}{\cosh^2(x)}$$

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

3. Relu(x): → Rectified linear unit

$$\text{Relu}(x) = \max(0, x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

$$\frac{d}{dx}\text{relu}(x) = \frac{d}{dx}\max(0, x) = \frac{d}{dx}\begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

$$= \begin{cases} 0, & x < 0 \\ 1 & x \geq 0 \end{cases}$$

3.2:What are the strategies you will follow to avoid over-fitting in a neural network.

Solution:

Q.2

What are the strategies you will follow to Avoid over-fitting in a neural network.

Solution:

1 Simplifying the model: To decrease the complexity, we can simply remove layers, or reduce the number of neurons to make network smaller.

2) Early Stopping: This is form of Regularization. This method update the model so as to make it better fit the training data with each iteration Early stopping rules provide guidance as to how many iterations can be run before the model begin to overfit

3) Use data Augmentation: It simply means increasing the size of the data that is increasing the number of images present in the dataset.

4) Use Regularization: It Add a penalty term to loss function. The most common techniques are L1 & L2

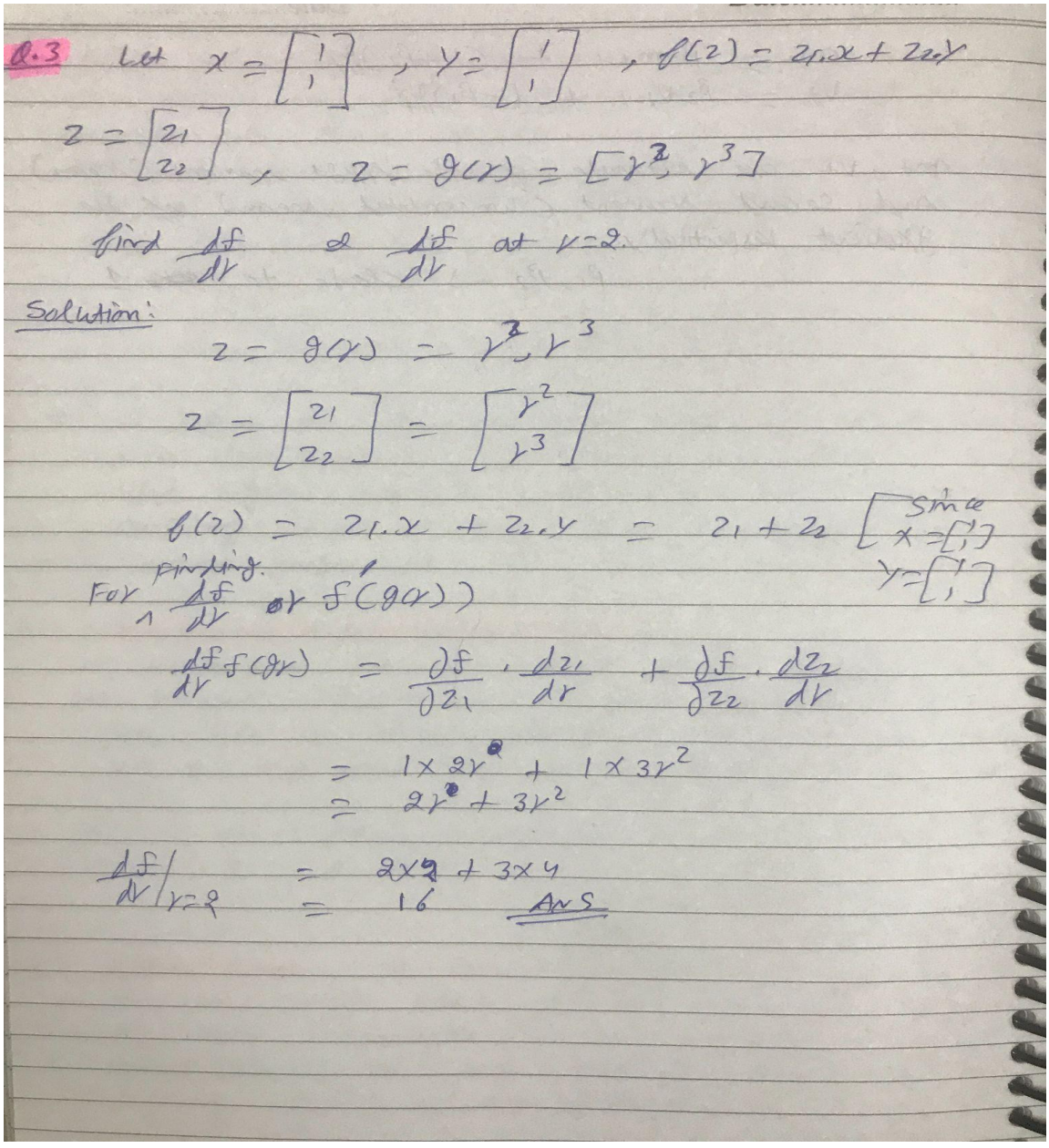
L1 → penalty aim to minimize the Absolute value of W.

L2 → penalty aim to minimize the squared magnitude of weight.

5) Use Dropout: It randomly drop neurons from the neural network during training in each iteration. When we drop different set of neurons, it's equivalent to training different neural network.

Q-3.3:Let x = [1, 1]T, y = [1, 1]T ∈ R2and let f : R2 =⇒ R2 with f(z) =z1.x + z2.y for anyz = [z1, z2]T ∈ R2. Further, z = g(r)=[r2, r3] where r ∈ R . Show how chain rule is applied here giving major steps of the calculation, write down the expression for ∂f/∂r ,and also evaluate ∂f/∂r at r=2.

Solution:



3.4. Define following loss functions in brief. (3 points)
(i) Mean Squared Error (MSE)
(ii) Binary Cross-entropy (BCE)
(iii) Categorical Cross-entropy (CC)

Solution:

Q4. Loss function

1) Mean Squared Error (MSE)

Mean squared error is calculated as the Average of the square difference between the Predicted and actual values. The result is always positive regardless the sign. perfect value is 0.0.
The squaring means that larger mistakes result in more error than smaller mistakes. i.e model punished for larger mistakes.

$$MSE = \frac{1}{m} \sum_{i}^{n} (y_i - \hat{y}_i)^2$$

2) Binary Class entropy

- Binary Class entropy compare each of the Predicted Probabilities to actual class output which be either 0 or 1. It than calculates the scores that penalize based on the distance from the expected value. That means how close or far from the actual value.

$$Loss = \frac{1}{N} \sum_{i=1}^{N} -(y_i \log(P_i) + (1-y_i) \log(1-P_i))$$

P_i → class 1 Probability
$1-P_i$ → class 0 Probability

3) Categorical cross entropy (CCE): When we have Multi-class classification task, we used CCE. It has same number of output nodes as the classes. And the final layer output must pass through a Softmax Activation. So that each node output is probability value between (0-1)

$$Loss = -\sum_{i}^{o/p\ size} y_i \log \hat{y}_i$$

Spiral

Q-3.5:Explain the following variants of Gradient Descent in brief: (10 points)

(i) Batch Gradient Descent

(ii) Stochastic Gradient Descent

(iii) Mini-batch Gradient Descent

(iv) Momentum based Gradient Descent

(v) Adam solver

Q.5: Explain the following variants of Gradient Descent.

1) Batch Gradient Descent:-

Let 'm' observation in a dataset And we use all these observation to calculate the cost function J, then this Batch gradient Descent.

Batch gradient descent can be very slow And sometimes don't fit in memory. Batch gradient descent also doesn't allow us to update our model online.

$$\theta = \theta - \eta . \nabla_\theta J(\theta)$$

- Gradient of the cost function w.r.t to the parameters θ for the entire training dataset.

2) Stochastic Gradient Descent:

If we use single observation to calculate the cost function it is known as Stochastic Gradient Descent. We pass a single observation at a time, calculate the cost and update the parameters. and each observation

Let we have 5 observation has three feature. Then we will pass one by one each observation And update parameter.

$$\theta = \theta - \eta . \nabla_\theta J(\theta; x^{(i)}, y^{(i)}) \quad x^{(i)} \rightarrow \text{Input}, y^{(i)} \rightarrow \text{Label}$$

→ Batch gradient descent performs redundant computation, As it recomputes gradient for similar examples before each parameter update but SGD does not have redundancy. It is usually faster And can be used to learn online.

3) Mini-batch Gradient Descent:-
It takes a subset of the entire dataset to calculate the cost function. So if there are 'm' observation then the number of observation in each subset or mini-batches will be more than 1 and less than 'm'

i.e $1 < x < m$

$$\theta = \theta - \eta . \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

- It reduce the variance of the parameter updates, which can lead to more stable convergence.
- Used highly optimized matrix that make gradient computing with respect to mini-batch very efficient
- common range is (50 — 256)

4) Momentum based Gradient Descent
- Momentum is a method which help SGD in the relevant direction and dampens oscillation. It does this by adding fraction γ of the update vector of the past time step to the current update vector

$$V_t = \gamma V_{t-1} + \eta \nabla_\theta J(\theta)$$
$$\theta = \theta - V_t$$
$$\gamma \rightarrow \leq 0.9$$

5) Adam Solver:- It is a method that computes Adaptive learning rates for each parameters. It also Store exponentially decaying Average of past squared gradient V_t and keep an exponentially decaying Average of past gradients m_t

Date.....

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

m_t, v_t are estimate of the first moment (mean) and second moment (uncentered mean) of the gradient respectively.

$\beta_1, \beta_2 \rightarrow$ close to 1