CSE-5368 Neural Networks Spring 2024 Exam_01

Last Name	First Name	Student ID Number
Solution		

Prob #	**1**	**2**	**3**	**4**	**Total**
Points	**21**	**29**	**25**	**25**	

Time: 80 Minutes

Seat Number:

Last Name	First Name	Student ID Number
Solution		

$$F(\mathbf{x}) = F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*)$$

$$+\frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^T\nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*) + \cdots$$

$$\frac{\mathbf{p}^T\nabla F(\mathbf{x})}{\|\mathbf{p}\|} \qquad \frac{\mathbf{p}^T\nabla^2 F(\mathbf{x})\mathbf{p}}{\|\mathbf{p}\|^2} \qquad \alpha_k = -\frac{\mathbf{g}_k^T\mathbf{p}_k}{\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k\mathbf{g}_k \qquad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$$

$$L_i = \sum_{j\neq i} max(0, y_j - y_i + \Delta)$$

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

$$H(p,q) = -\sum_x p(x)log(q(x))$$

$$L_i = -log\left(\frac{e^{y_i}}{\sum_j e^{y_j}}\right)$$

CSE-5368 Neural Networks Spring 2024 Exam_01

Last Name	First Name	Student ID Number
Solution		

1. Consider a convolutional neural network.
   Note: **Do NOT consider Biases**.

Input layer:

Input to this CNN are color images of size **100x70x3** with the **batch size = 30**

Note: Input image has **different horizontal and vertical** resolution.

Next layer is Conv2D layer:

Number of filters: **15**, filter size: **9x9**; stride: **3x3**; padding: **4x4**

Q1: What is the shape of the weight matrix for this layer? **Q1:** __9x9x3x15__

Q2: What is the shape of the output (tensor) of this layer? **Q2:** 30x34x24x15

Next layer is Conv2D layer:

Number of filters: **10**, filter size: **4x4**; stride: **2x2**; padding: **1x1**

Q3: What is the shape of the weight matrix for this layer? **Q3:** __4x4x15x10__

Q4: What is the shape of the output (tensor) of this layer? **Q4:** 30x17x12x10

Next layer is Flatten layer:

Q5: What is the shape of the output (tensor) for this layer? **Q5:** __30x2040____

Next layer is Dense layer:

Number of nodes: **50**

Q6: What is the shape of the weight matrix for this layer? **Q6:** __2040x50____

Q7: What is the shape of the output (tensor) for this layer? **Q7:** ___30x50____

Last Name	First Name	Student ID Number
Solution		

Problem 1 Continued

CSE-5368 Neural Networks Spring 2024 Exam_01

Last Name	First Name	Student ID Number
Solution		

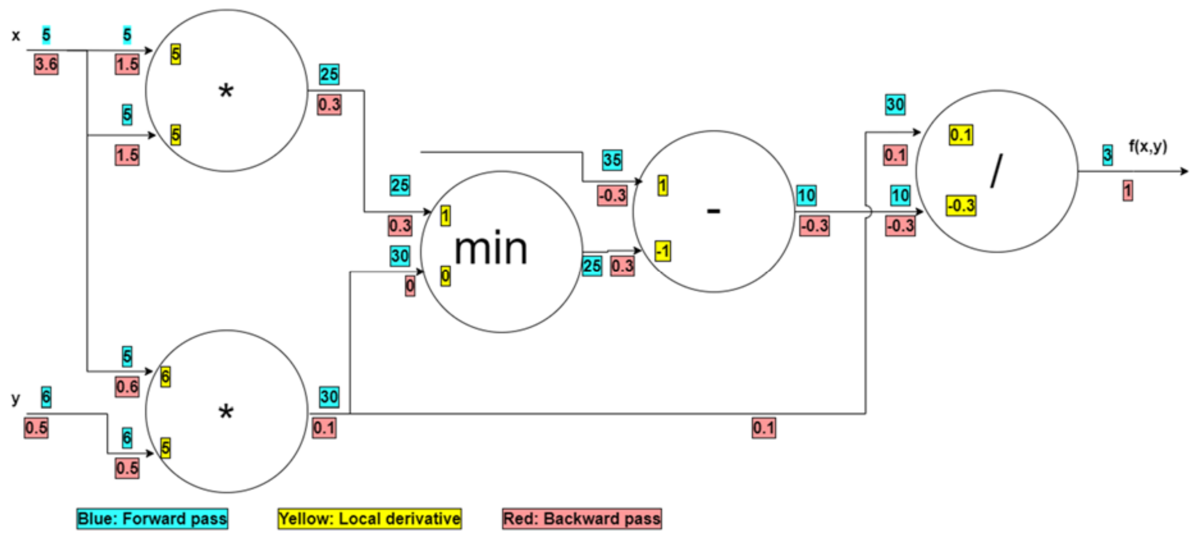
2. Consider the expression:

$$f(x,y) = \frac{xy}{35 - min(xy, x^2)}$$

given the inputs: $x = 5, \ \ y = 6$

Draw the computational graph and calculate the $\frac{\delta f(x,y)}{\delta x}$ and $\frac{\delta f(x,y)}{\delta y}$

For proper credit, you MUST SHOW all the numerical values **for each node** as they flow in the forward and backward path in the computational graph.



Last Name	First Name	Student ID Number
Solution		

Problem 2 Continued

Last Name	First Name	Student ID Number
Solution		

3. Using tensorflow, complete the following function to create and train a **two-layer** neural network. The **first layer** has **7 sigmoid** nodes. The **output layer** has **linear nodes**. Loss function should be **MSE**. Anything not specified in the description should be **inferred from the function's parameters and not hardcoded.**

   Code should include initializing weights, training loop with forward pass, gradient calculation, and weight updates.

   You may assume the entire dataset is one batch.

   DO NOT USE Keras

```
import numpy as np
import tensorflow as tf
def create_and_train_nn(X, Y, epochs, alpha):
"""
:param X: Array of input [n_samples,input_dimensions]
:param y: Array of desired outputs [n_samples , target_dimension].
:param epochs: number of epochs
:param alpha: Learning rate:
:return w1, w2 Weight matrices."""

    w1=tf.Variable(np.random.randn((X.shape[1],7)))
    b1=tf.Variable(np.random.randn((7)))
    w2=tf.Variable(np.zeros((7,Y.shape[1])))
    b2=tf.Variable(np.random.randn((Y.shape[1])))

    for epoch in range(epochs):
        with tf.GradientTape() as tape:
            y1=tf.sigmoid(tf.matmul(X,w1)+b1)
            y2=tf.matmul(y1,w2)+b2
            loss=tf.reduce_mean(tf.square(Y-y2))
            dw1,dw2,db1,db2=tape.gradient(loss,[w1,w2,b1,b2])
    w1.assign_sub(alpha*dw1)
    w2.assign_sub(alpha*dw2)
    b1.assign_sub(alpha*dw1)
    b2.assign_sub(alpha*dw2)

    return w1,w2
```

Last Name	First Name	Student ID Number
Solution		

Problem 3 Continued

Last Name	First Name	Student ID Number
Solution		

4. Complete the code for the following function.

USE numpy only. DO NOT USE tensorflow or keras

```
import numpy as np
def calculate_svm (yhat,yt):
# This function calculates the SVM error for the entire data set
# yhat: Array of actual outputs [num_of_samples,num_of_classes]
# yt: Array of desired outputs [num_of_samples]
# Each element of yt array is the index of the true class.
# return: SVM for the entire data set (A single float number).
# Return value is the average of all the SVMs for the samples.
# Assume delta is equal to 1

    # Detail solution
    number_of_samples=yhat.shape[0]
    number_of_classes=yhat.shape[1]
    total_loss=0
    for sample_index in range(number_of_samples):
        target_class_index=yt[sample_index]
        yi=yhat[sample_index,yt[sample_index]]
        sample_loss=0
        for class_index in range(number_of_classes):
            if target_class_index==class_index:
                continue
            yj=yhat[sample_index,class_index]
            sample_loss=sample_loss+np.maximum(0,yj-yi+1)
        total_loss=total_loss+sample_loss
    total_loss=total_loss/number_of_samples
    return total_loss

def calculate_svm_v2(yhat, yt):
    # More compact solution
    total_loss = 0
    for k in range(yt.shape[0]):
        loss = np.maximum(0, yhat[k] - yhat[k][yt[k]]+1)
        loss[yt[k]] =0
        total_loss=total_loss+np.sum(loss)
    return total_loss / yhat.shape[0]

def calculate_svm_v3(yhat, yt):
    # Another solution
    total_loss = 0
    for sample,target_index in zip(yhat,yt):
        margins = np.maximum(0, sample - sample[target_index] + 1)
        total_loss = total_loss + np.sum(margins) - 1
    return total_loss / yhat.shape[0]


def calculate_svm_v4(yhat, yt):
    # Most compact solution with numpy
    c=yhat.shape[0]
    return (np.sum(np.maximum(0,yhat-[yhat[np.arange(c)[:,None],yt.reshape(c,1)]] + 1))-c)/c
```

Last Name	First Name	Student ID Number
Solution		

Problem 4 Continued