CSE-5368 Neural Networks Fall 2024 Exam_02

Last Name	First Name	Student ID Number

Prob #	**1**	**2**	**3**	**4**	**Total**
Points	**18**	**21**	**21**	**40**	

Time: 80 Minutes

,

CSE-5368 Neural Networks Fall 2024 Exam_02

Last Name	First Name	Student ID Number

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

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

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

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

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

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

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

$$KL\ Divergence = -\frac{1}{2}\sum(1 + \log(\sigma^2) - \mu^2 - \sigma^2)$$

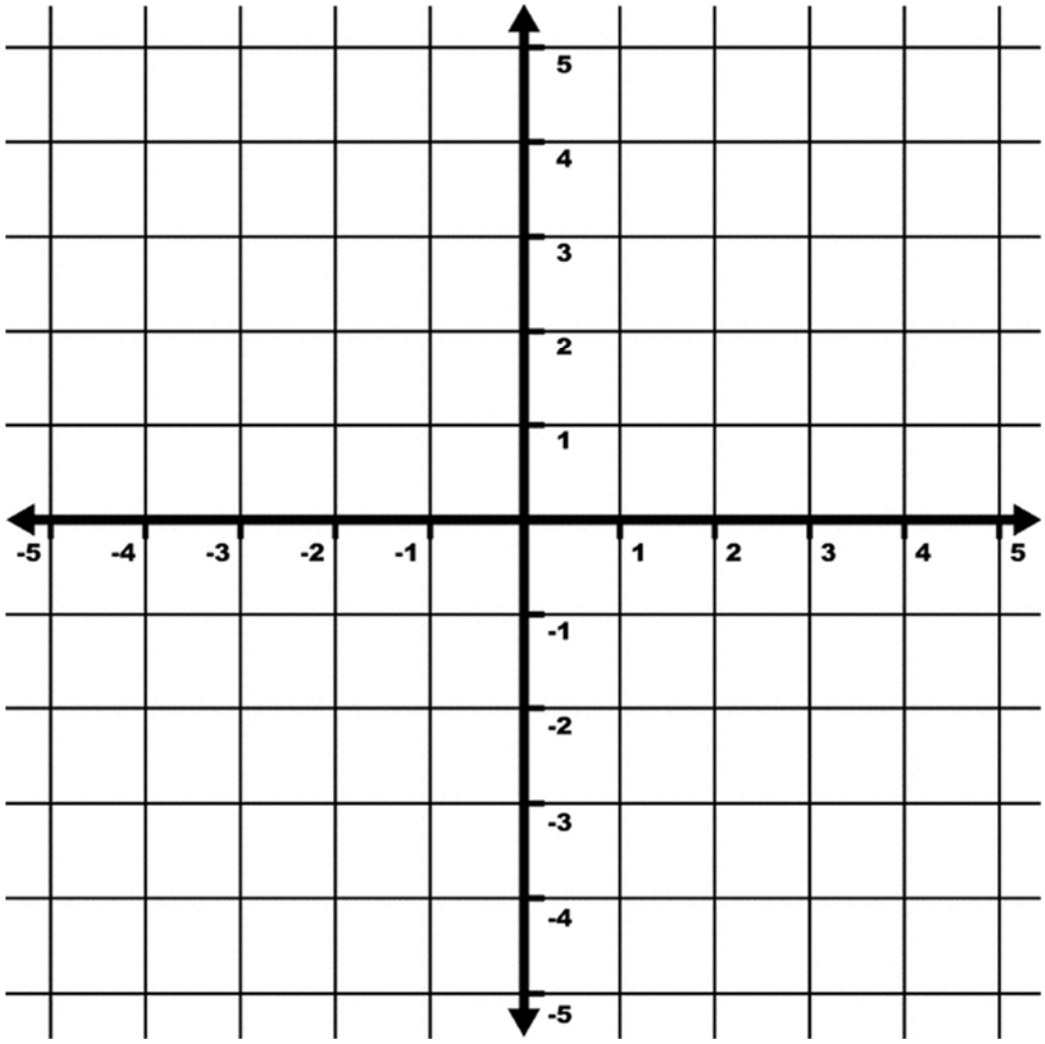
CSE-5368 Neural Networks Fall 2024 Exam_02

Last Name	First Name	Student ID Number

1. Consider the following data where p is the input and t is the desired output

$$\left\{p_1 = \begin{bmatrix}4.5\\0\end{bmatrix}, t_1 = 0\right\}, \left\{p_2 = \begin{bmatrix}-3.5\\0\end{bmatrix}, t_2 = 0\right\}, \left\{p_3 = \begin{bmatrix}0\\2.5\end{bmatrix}, t_3 = 0\right\},$$
$$\left\{p_4 = \begin{bmatrix}3.5\\0\end{bmatrix}, t_4 = 1\right\}, \left\{p_5 = \begin{bmatrix}-2.5\\0\end{bmatrix}, t_5 = 1\right\}, \left\{p_6 = \begin{bmatrix}0\\1.5\end{bmatrix}, t_6 = 1\right\}$$

DESIGN a two-layer Perceptron neural network which will correctly classify the input data. Assume the activation function for all the nodes are hardlimit with the output of 0 and 1. Show the weight matrices and biases for both layers. Biases should be included in the weight matrix in the first column.



Last Name	First Name	Student ID Number

Problem 1 Continued

CSE-5368 Neural Networks Fall 2024 Exam_02

Last Name	First Name	Student ID Number

2. Consider a convolutional neural network.

Note: **Do NOT consider Biases**.

Input layer:

Input to this CNN are color images of size **67x67x3** with the **batch size = 90**

Next layer is Conv2D layer:

Number of filters: **20**, filter size: **7x7**; stride: **2x2**; padding: **2x2**

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

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

Next layer is Conv2D layer:

Number of filters: **30**, filter size: **5x5**; stride: **3x3**; padding: **4x4**

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

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

Next layer is Flatten layer:

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

Next layer is Dense layer:

Number of nodes: **50**

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

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

Last Name	First Name	Student ID Number

Problem 2 Continued

Last Name	First Name	Student ID Number

3. Assuming that the actual output and the desired output of a neural network are given.

   Complete the code for the following function to calculate the **mean cross entropy** loss for a batch of data.

   Note:

   Assume that actual output and desired output are logits. This means you have to apply **softmax** to both **actual outputs** and **desired outputs** before cross entropy calculation.

Notes:

Only use numpy. Do NOT use PyTorch or any other package.

You may use the numpy helper functions such as `np.log()` and `np.sum()`

```
def calculate_mean_cross_entropy_loss(y_hat,y):
Import numpy as np
""" This function calculates the mean cross entropy loss.
:param y_hat: actual output [number_of_samples,number_of_classes].
:param Y: Desired output [number_of_samples,number_of_classes].
:return: mean cross-entropy loss
"""
```

Last Name	First Name	Student ID Number

Problem 3 Continued

Last Name	First Name	Student ID Number

4. Complete the following function. This function should initialize a variational autoencoder and return the mean loss over all the samples. **Only use numpy.**

Notes:
Ignore biases. Initialize all weights to **ones**. Assume all layers are **fully connected**.
Activation function for layers are **Sigmoid**.
Loss is: **MSE + KL divergence**
Decoder has the same structure as the encoder in reverse order. #

Complete the incomplete functions

```
import numpy as np

def variational_autoencoder(input, encoder_layers):
# input: numpy array of inputs [nof_train_samples, input_dimensions]
# encoder_layers: list of integers representing number of nodes in
# each layer of the encoder. The last number represents the dimension of latent
space
# return: mean_loss

# Step 1 (8 points)
  list_of_encoder_weights,list_of_decoder_weights =
initialize_encoder_and_decoder_layers(encoder_layers,input)
# Step 2 (8 points)
   mu, var = encoder_forward_pass(input, list_of_encoder_weights)
# Step 3 (8 points)
  z = sampling_trick(mu,var)
# Step 4 (8 points)
  reconstructed_input = decoder_forward_pass(z, list_of_decoder_weights)
# Step 5 (8 points)
  mean_loss = calculate_loss(input,reconstructed_input, mu,sigma)
  return mean_loss
```

CSE-5368 Neural Networks Fall 2024 Exam_02

Last Name	First Name	Student ID Number

def initialize_encoder_and_decoder_layers(encoder_layers,input):

```
# encoder_layers: list of integers representing number of nodes in
# each layer of the encoder. The last number represents the dimension
of latent space
# return list_of_encoder_weights, list_of_decoder_weights # as list of
numpy arrays
```

Last Name	First Name	Student ID Number

Problem 4 Continued

```
def encoder_forward_pass(input, list_of_encoder_weights):
# input: numpy array of inputs [nof_train_samples, input_dimensions]
# return mu, var
```

```
def sampling_trick(mu,var)
# return z  # sampled latent variable
```

Last Name	First Name	Student ID Number

Problem 4 Continued

```
def decoder_forward_pass(z, list_of_decoder_weights):
#z:numpy array of latent varibales [nof_train_samples,latent_dimension]
# return reconstructed_input [nof_train_samples, input_dimensions]
```

```
def calculate_loss(input, reconstructed_input, mu, sigma):
# input: numpy array of inputs [nof_train_samples, input_dimensions]
# return mean_loss (MSE + KL divergence) it should be a single number
```