0.1 Softmax [20 points]

I) [10 point] Prove that softmax is invariant to constant sifts in the input, i.e., for any input vector x and a constant scalar c, the following holds:

softmax(x) = softmax(x -+-c) ,

where softmax(xh —4— and x -kc means adding c to every dimension of x.

2) [10 point] Let z = W x -F c, where W and c are some matrix and vector, respectively. Let

J = log .

① Given $\text{softmax}(x)_i = \dfrac{e^{x_i}}{\sum_{i'} e^{x_{i'}}}$

$$\text{softmax}(x+c)_i = \frac{e^{x_i+c}}{\sum_{i'} e^{x_{i'}} e^{c}} = \frac{e^{x_i} e^{c}}{\sum_{i'} e^{x_{i'}} e^{c}} = \frac{e^{x_i} \cancel{e^{c}}^{\text{canceled}}}{\underbrace{\cancel{e^{c}}}_{\text{canceled}} \sum_{i'} e^{x_{i'}}}$$

Calculate the derivatives of J w.r.t. W and c, respectively, i. e. , calculate _uu- and $\frac{\partial J}{\partial c}$.

$$= \frac{e^{x_i}}{\sum_{i'} e^{x_{i'}}} \equiv \text{softmax}(x)$$

② $J = \sum_i \text{Log softmax}(z)_i$

if $\text{softmax}(x)_i \triangleq \dfrac{e^{x_i}}{\sum_{i'} e^{x_{i'}}}$ then,

$\text{softmax}(z)_i \triangleq \dfrac{e^{z_i}}{\sum_{i'} e^{z_{i'}}}$ and,

$\boxed{j = i'}$

$$J = \cancel{\sum_i} \log \frac{e^{z_i}}{\cancel{\sum_{i'}} e^{z_{i'}}} = \log \frac{e^{z_i}}{i' e^{z_{i'}}} = \log \frac{e^{z_i}}{j^2 e^{z}}$$

$$\frac{dJ}{dw} = \frac{d}{dw}\left(\log \frac{e^{z_i}}{j^2 e^{z}}\right) = 0$$ since all term are constants

then $j = i'$

$$\frac{dJ}{dc} = \frac{d}{dc}\left(\log \frac{e^{z_i}}{j^2 e^{z}}\right) = 0$$ since all term are constants

0.2 Logistic Regression with Regularization [20 points]

1) [10 point] Let the data be (Xi, where Xi G IR^d and Yi G {O, 1}. Logistic regression is a binary classification model, with the probability of Yi being I as:

$$p(y_i = 1 \mid \mathbf{x}_i, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^T \mathbf{x}_i) \triangleq \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}_i}},$$

where is the model parameter. Assume we impose an 102 regularization term on the parameter, defined as:

$$\mathcal{R}(\boldsymbol{\theta}) = \frac{\lambda}{2} \boldsymbol{\theta}^T \boldsymbol{\theta}$$

with a positive constant A. Write out the final objective function for this logistic regression with regularization model.

2) [10 point] If we use gradient descent to solve the model parameter. Derive the updating rule for O. Your answer should contain the derivation, not just the final answer.

0.3 Derivative of the Softmax Function [30 points]

I) [10 point] Define the loss function as

$$J(\mathbf{z}) = -\sum_{k=1}^{K} y_k \log \tilde{y}_k \ ,$$

where , and , • • • , !/K) is a known probability vector. Derive the DJ(z)

Note z — ZK) is a vector so DJ(z) is in the form of a vector. Your answer should contain the derivation, not just the final answer.

2 [10 point] Assume the above softmax is the output layer of an FNN. Briefly explain how $\frac{\partial}{\partial \mathbf{w}}$ the derivative is used in the backpropagation algorithm.

① $J(z) = -\sum_{k=1}^{K} Y_k \log \tilde{Y}_k$ & $\tilde{Y}_k = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}$ & $\frac{dJ(z)}{dz} = \begin{bmatrix} dJ/dz_1 \\ \vdots \\ dJ/dz_K \end{bmatrix}$

so

$$\frac{dJ(z)}{dz} = -\frac{d}{dz}\sum_{k=1}^{K} Y_k \log \tilde{Y}_k$$

$$= -\frac{d}{dz}\sum_{k=1}^{K} Y_k \log \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}$$

3) [10 points] Let z = h + b), where a(•) is the sigmoid function, W is a matrix, b and

$$= -\frac{d}{dz}\sum_{k=1}^{K} Y_k z_k + \frac{d}{dz}\sum_{k=1}^{K} Y_k \log \sum_{k'} e^{z_{k'}}$$

h are vectors. Use the chain rule to calculate the gradient of W and b, i.e., and Db,

$$= -Y_k + \frac{d}{dz}\sum_{k=1}^{K} Y_k \log \sum_{k'} e^{z_{k'}}$$

respectively (it is enough to derive the gradients for one element of the matrix/vector

$$= -Y_k + \sum_{k=1}^{K} Y_k \cdot \frac{d}{dz} \log \sum_{k'} e^{z_{k'}}$$

parameter).

$$= -Y_k + \sum_{k=1}^{K} Y_k \cdot \frac{1}{\sum_{k'} e^{z_{k'}}} \cdot e^{z_{k'}}$$

$$= -Y_k + \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}} \left(\overbrace{\sum_{k=1}^{K} Y_k}^{1} \right)$$

$= -y_k + \tilde{y}_k$ this means $\frac{dJ(z)}{dz} = \begin{bmatrix} -y_1 + \tilde{y}_1 \\ -y_2 + \tilde{y}_2 \\ \vdots \\ -y_k + \tilde{y}_k \end{bmatrix}$

(2) The derivative is used in backpropigation to update the weights of a FNN during training to help minimize loss.

(3) $z = \sigma(W^T h + b)$

$\frac{dJ}{dW} = ?$ $\frac{dJ}{db} = ?$

$\frac{dJ}{dW} = \frac{dJ}{dz} \cdot \frac{dz}{dW}$ $\frac{dJ}{dz} = \begin{bmatrix} -y_1 + \tilde{y}_1 \\ -y_2 + \tilde{y}_2 \\ \vdots \\ -y_k + \tilde{y}_k \end{bmatrix}$

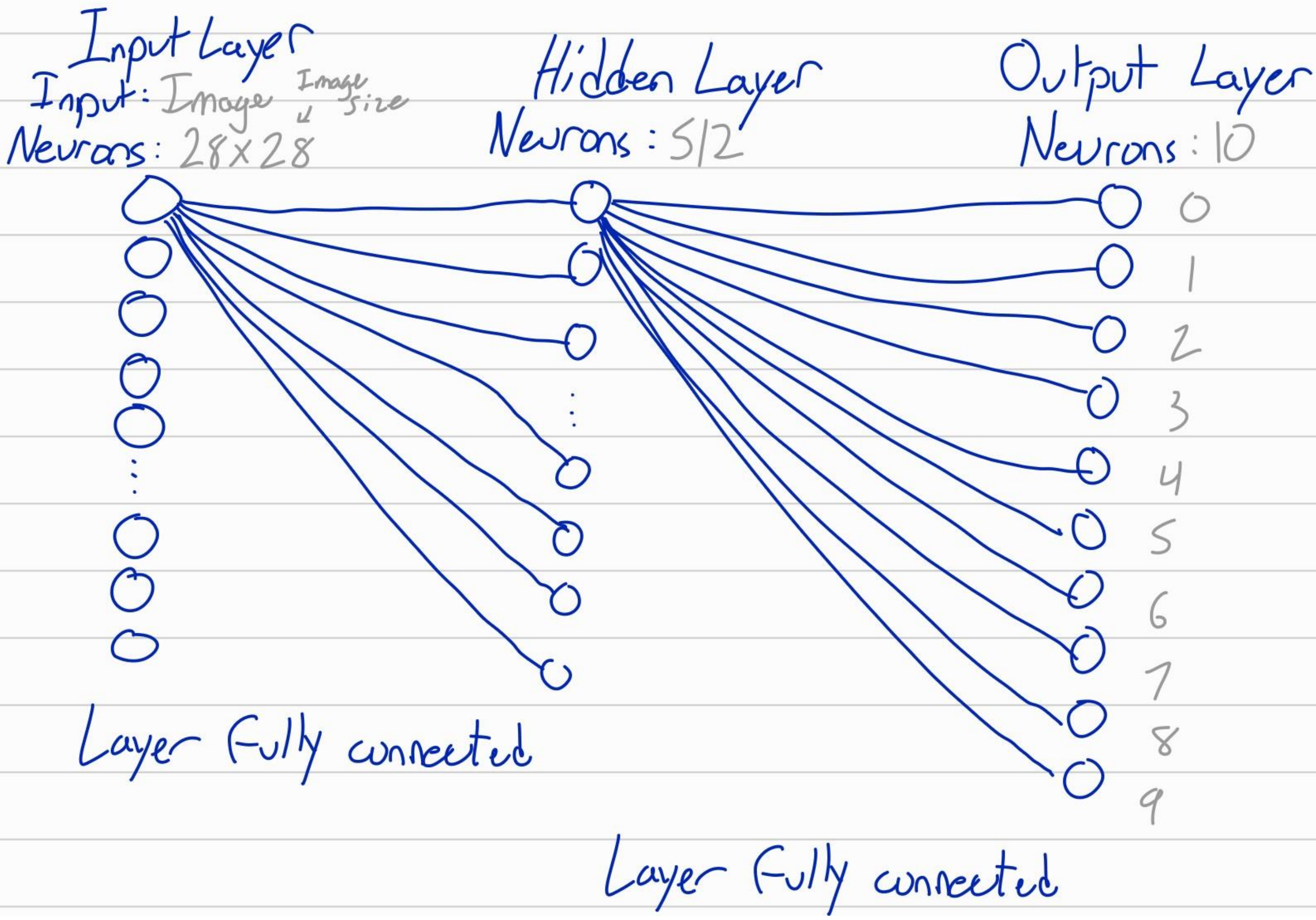
0.4 MNIST with F NN [30 points]

1) CIO points] Design an FNN for MNIST classification. Draw the computational graph of your model.

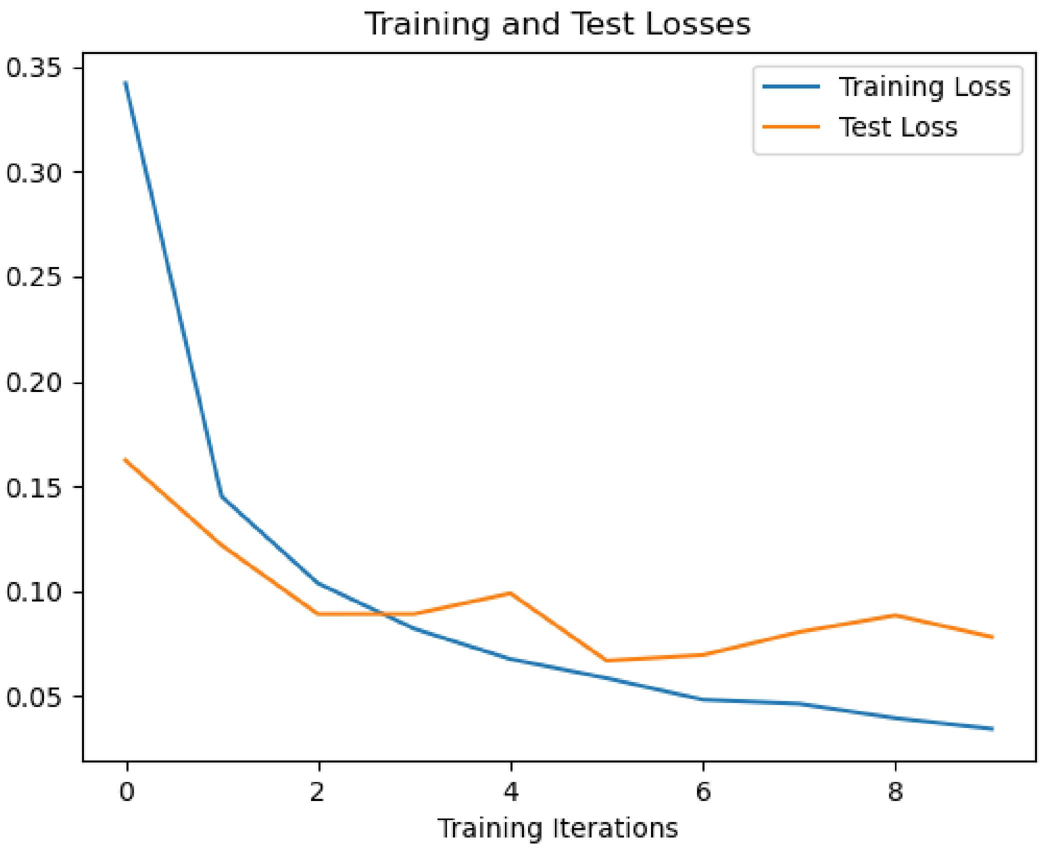
2) [20 points] Implement the model and plot two curves in one figure: i) training loss vs. training iterations; ii) test loss vs. training iterations.

   – You can use any packages; but Pytorch is recommended.

1



References: Code generated from chat.openai.com


Training and Test Losses
Training Loss
Test Loss
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
2
4
6
8
Training Iterations