CSCI3230 / ESTR3108: Fundamentals of Artificial Intelligence 2023-2024 Term 1
The Chinese University of Hong Kong

Assignment 3

Due date: 17 November 2023 (Thu) 23:59 Full mark: 100
Expected time spent: 5-7 hours

Aims: 1. Understand the knowledge about PCA and have hands-on practice about optimizations.
2. Understand basic knowledge about neural networks and hands-on programming of simple neural networks, including training and testing.

Description:

In Assignment 3, you will conduct some calculations and proofs related to PCA using elementary linear algebra and calculus knowledge. Furthermore, you will practice on essentials of neural network, including neural network construction, backpropagation, model training and evaluation, PyTorch programming, etc.

Questions:

1. Recall that the geometric formulation of PCA for dimensionality reduction is the optimization problem as shown below:

$$U^* = \arg\min_{U \in \mathbb{R}^{D\times d}, U^T U = I} \| \mathbf{X} - UU^T\mathbf{X} \|_F$$

where $\mathbf{X} \in \mathbb{R}^{D\times m}$ is a zero-mean data matrix with D-dimensional features and m samples, and $d \ll D$ is the target lower dimension. When $d = 1$, we say U^* is the first principal axis and $U^{*T}\mathbf{X}$ is the first principal component.

Suppose we have zero-mean data matrix: $\mathbf{X} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$, which contains 3 samples with 4-dimensional features. We can observe there are some redundant information in the data matrix. In this regard, we would like to perform PCA on the data matrix to reduce the dimensionality of the original data.

(a) Find all eigenvalues of $\mathbf{X}\mathbf{X}^T$. Show all of your calculation steps. (9%)

Solution:

Firstly, we can compute $\mathbf{X}\mathbf{X}^T = \begin{bmatrix} 2 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 \\ -2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$. (1%)

Then, we find those λ's that vanish the characteristic polynomial $p(\lambda) = \det(\mathbf{X}\mathbf{X}^T - \lambda I)$:

(2%)

$$\det(\mathbf{X}\mathbf{X}^T - \lambda I) = 0$$

$$\det\begin{pmatrix} 2-\lambda & 0 & -2 & 0 \\ 0 & -\lambda & 0 & 0 \\ -2 & 0 & 2-\lambda & 0 \\ 0 & 0 & 0 & -\lambda \end{pmatrix} = 0$$

$\Longrightarrow \lambda = 0$ or $\lambda = 4$.

(b) Find the first principal axis and the first principal component of $\mathbf{X}$. Then explain how dimensionality of the data matrix is reduced by PCA. (8%)

Solution:

The first principal component is the projection of original data onto the first principal axis, i.e., the orthonormal eigenvector associated with the largest eigenvalue of $\mathbf{X}\mathbf{X}^T$. From (a), we know the largest eigenvalue is 4 (1%). Then we need to solve those v's which satisify $\mathbf{X}\mathbf{X}^T v = 6v$. This equation is equivalent to finding the nullspace of $\mathbf{X}\mathbf{X}^T - 6I = \begin{bmatrix} -2 & 0 & -2 & 0 \\ 0 & -4 & 0 & 0 \\ -2 & 0 & -2 & 0 \\ 0 & 0 & 0 & -4 \end{bmatrix}$ (2%). We can easily see $v = [-1,0,1,0]$ is a basis vector of the nullspace (or solve it via Gaussian elimination) (2%). Thus, $\bar{v} = normalize(v) = \left[\frac{-\sqrt{2}}{2}, 0, \frac{\sqrt{2}}{2}, 0\right]$ is the orthonormal eigenvector associated with the largest eigenvalue. Thus, the first principal axis is $\left[\frac{-\sqrt{2}}{2}, 0, \frac{\sqrt{2}}{2}, 0\right]$. The first principal component is $\bar{v}^T\mathbf{X} = \left[-\sqrt{2}, 0, \sqrt{2}, 0\right]$ (3%).

(c) In real-word use case, the order of data collecting is random. For example, suppose we have another data matrix

$$\mathbf{X}' = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

which contains the same data in $\mathbf{X}$ but store them in different orders. Do you think the order of data collecting will affect the first principal axis? Please justify your answer with mathematical proofs or counterexamples. (8%)

Solution:

The order of data collecting will not affect the first principal axis (2%). The relationship between $\mathbf{X}'$ and $\mathbf{X}$: $\mathbf{X}' = \mathbf{X}P$ where P is a column-permutation matrix (2%). Note that permutation matrices are orthogonal. Thereby $\mathbf{X}'\mathbf{X}'^T = \mathbf{X}PP^T\mathbf{X}^T = \mathbf{X}\mathbf{X}^T$ (4%).

2. Recall the statistical view of the first principal component. For a multi-variate random variable $X \in \mathbb{R}^D$ (zero-mean), the first principal component is defined as a unit vectors $u_1 \in \mathbb{R}^D$, $u_1^T u_1 = 1$ such that $y_1 = u_1^T X$ and $\mathrm{Var}(y_1)$ are maximized.

Note that we have: $\mathrm{Var}(y_1) = \mathrm{E}(y_1 y_1^T) = \mathrm{E}(u_1^T X X^T u_1) = u_1^T \mathrm{E}(XX^T) u_1$. Thus, we can write this problem in an optimization form:

$$u_1{}^* = \arg\max_{u_1 \in \mathbb{R}^D, u_1^T u_1 = 1} u_1^T \mathrm{E}(XX^T) u_1$$

In this question, we aim to prove the optimal solution $u_1{}^*$ is the eigenvector of $\mathrm{E}(XX^T)$ associated with the largest eigenvalue, which is consistent to the geometric view of PCA.

(a) Since $\mathrm{E}(XX^T)$ is symmetric, it must have D orthonormal eigenvectors $[v_1, \dots, v_D]$ spanning the entire ambient vector space $\mathbb{R}^D$. Thus, we can write any $u_1 \in \mathbb{R}^D$ as $u_1 = \sum_{j=1}^D c_j v_j$, for some $c_1, \dots, c_D$. Try to prove $\sum_{j=1}^D c_j^2 = 1$ when $u_1^T u_1 = 1$. (Hint: begin with rewriting $u_1^T u_1 = 1$ in terms of c_j's and v_j's) (12%)

Solution:

Firstly, we expand $u_1^T u_1$:

$$u_1^T u_1 = 1 = \sum_{j=1}^{D} c_j \, v_j^T \sum_{j=1}^{D} c_j \, v_j = \sum_{j=1}^{D} \sum_{k=1}^{D} c_j \, c_k v_j^T v_k$$

(4%)

Since $v_1 \ldots v_n$ are orthonormal vectors, then we have:

$$v_j^T v_k = \begin{cases} 1, j = k \\ 0, j \neq k \end{cases}$$

(4%)

$$u_1^T u_1 \quad = \sum_{j=i}^{D} c_j \, c_j v_j^T v_j = \sum_{j=i}^{D} c_j^2 = 1$$

(4%)

(b) Denote eigenvalues of $\mathrm{E}(XX^T)$ as $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$ associated with eigenvectors $v_1, \ldots, v_D$. $\mathrm{E}(XX^T)$ admits a spectral resolution:

$$\mathrm{E}(XX^T) \; = \sum_{k=1}^{D} \lambda_k \, v_k v_k^T$$

Using the results of (a) and spectral resolution, show that $u_1^T \mathrm{E}(XX^T) u_1 \leq \lambda_1$ for any $u_1 \in \mathbb{R}^D$, $u_1^T u_1 = 1$. Thereby show the eigenvector v_1 can achieve $u_1^T \mathrm{E}(XX^T) u_1 = \lambda_1$. (Hint: begin with substituting $u_1 = \sum_{j=1}^{D} c_j \, v_j$ into the inequality) (13%)

Solution:

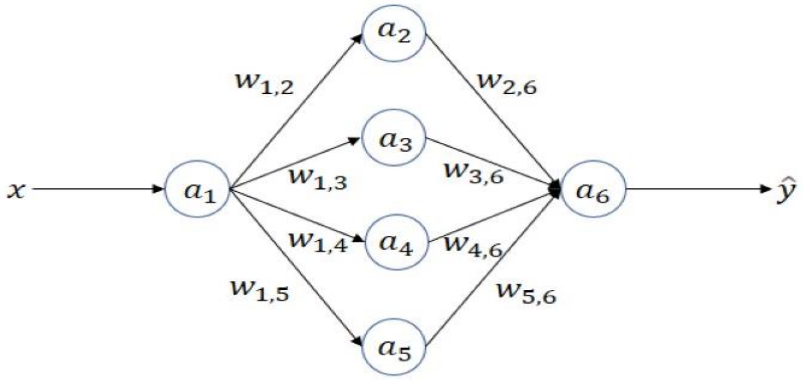
For any $u_1 \in \mathbb{R}^D$, $u_1^T u_1 = 1$, we have $u_1 = \sum_{j=1}^{D} c_j \, v_j$ and $\sum_{j=i}^{D} c_j^2 = 1$. (2%)

$$\begin{aligned} {u_1}^T \mathrm{E}(XX^T) u_1 \quad &= \sum_{j=1}^{\boldsymbol{D}} c_j \, v_j^T \sum_{k=1}^{\boldsymbol{D}} \lambda_k \, u_k u_k^T \sum_{l=1}^{\boldsymbol{D}} c_l \, v_l \\ &= \sum_{j=1}^{\boldsymbol{D}} \sum_{k=1}^{\boldsymbol{D}} \sum_{l=1}^{\boldsymbol{D}} c_j \, c_l \lambda_k v_j^T v_k v_k^T v_l = \sum_{j=1}^{\boldsymbol{D}} c_j^2 \, \lambda_j \\ &\leq \left(\sum_{j=1}^{\boldsymbol{D}} c_j^2 \right) \lambda_{\boldsymbol{1}} = \lambda_{\boldsymbol{1}} \end{aligned}$$

(each key step takes 2%, total: 8%)

Furthermore, by taking $u_1 = v_1$, we can see the equality will be reached. (3%)

3. Consider a simple neural network for binary classification as the following:



In this neural network, the input is x and the output is $\hat{y}$. There is 1 input neuron (a_1), 4 hidden neurons (a_2, a_3, a_4, a_5) and 1 output neuron (a_6) in this network. The weights of connections between a_i and a_j is defined as $w_{i,j}$. For the input neuron, no activation function is used. For every hidden neuron, we use *ReLU* activation function. For the output neuron, we use *Sigmoid* activation function. To simplify the problem, we can set , $W_1 = [w_{1,2}, w_{1,3}, w_{1,4}, w_{1,5}]^T, W_2 = [w_{2,6}, w_{3,6}, w_{4,6}, w_{5,6}]^T$.

(a) Write the specific formula of $\hat{y}$, which is composed of $x, W_1 = [w_{1,2}, w_{1,3}, w_{1,4}, w_{1,5}]^T, W_2 = [w_{2,6}, w_{3,6}, w_{4,6}, w_{5,6}]^T$, *ReLU* and *Sigmoid*. (Hint: a_2 can be represented by $a_2 = ReLU(w_{1,2}a_1)$.) (5%)

Solution:

$$\hat{y} = Sigmoid(w_{2,6}ReLU(w_{1,2}x) + w_{3,6}ReLU(w_{1,3}x) + w_{4,6}ReLU(w_{1,4}x) + w_{5,6}ReLU(w_{1,5}x))$$

(b) Suppose we use the loss function of Binary Cross Entropy (BCE) to optimize this network, with input x, its corresponding ground-truth label y, and its corresponding network output $\hat{y}$, please write down the specific loss function based on the formula in (a), and calculate the gradient for $w_{1,3}$. (10%)

Solution:

The BCE loss:

$$\ell = -y ln\hat{y} - (1-y)\ln(1-\hat{y})$$

(5%)

Suppose $z = 1 + e^{-(w_{2,6}ReLU(w_{1,2}x)+w_{3,6}ReLU(w_{1,3}x)+w_{4,6}ReLU(w_{1,4}x)+w_{5,6}ReLU(w_{1,5}x))}$

$$\frac{\partial \ell}{\partial w_{1,3}} = \begin{cases} \frac{w_{3,6}x(1-yz)}{z}, & w_{1,3}x > 0 \\ 0, & otherwise \end{cases}$$

(5%)

(c) Given a training set and test set listed in *train_q3.csv* and *test_q3.csv*, write PyTorch code to train the above network, and present the followings: 1) plot the curve of training loss, 2) give the final optimized weights for (W_1, W_2), 3) test the network on the given test set and report the testing accuracy.

Please randomly initialize the weights W_1 and W_2 using torch.nn.init.kaiming_uniform_ and initialize the bias term as zero. Try to tune the learning rate, training epochs and the batch size to make the test accuracy $\geq 95\%$. What's your findings according to these training accuracy and test accuracy with different training epochs? For binary classification, we typically set the threshold as 0.5, i.e., if the output probability is larger than 0.5, then the output class is 1. Otherwise, the prediction is 0. (Hint: you can tune the learning rate from set {0.01, 0.1, 1}, training epochs from {10, 50, 150} and batch size from set {64, 128, 256}.) (10%)

Solution:

Here, we set the learning rate as 0.1, the batch size as 256 and training epochs as 50.

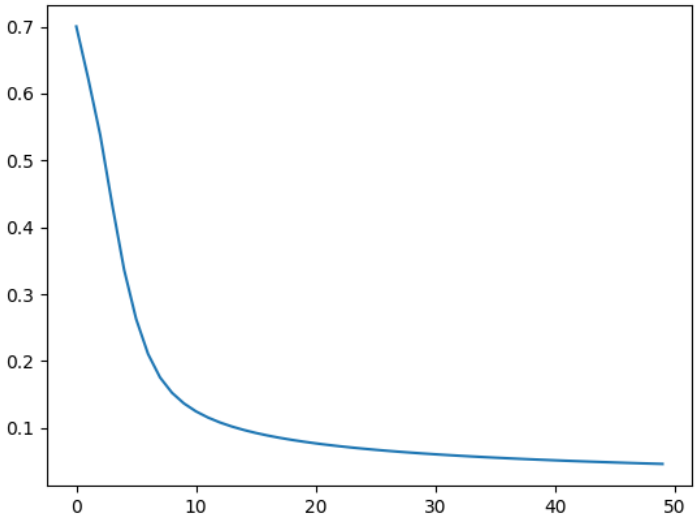
$$W_1 = [-2.811, 4.2131, 0.4031, -3.3807]^T$$

(2%)

$$W_2 = [-2.0665, 4.2729, -0.3982, -3.4644]^T$$

(2%)

Training loss curve:

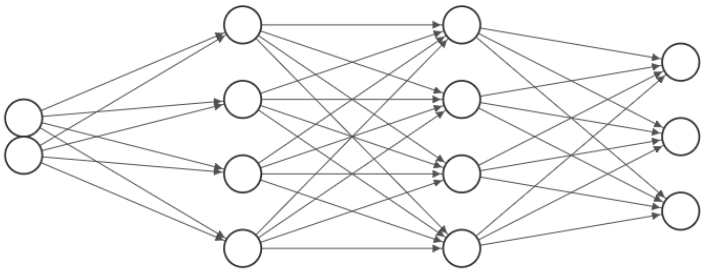


(4%)

Test Accuracy: 98.35%

The accuracy of models on the test data would peak after training for a number of epochs. Then the test accuracy may start to decrease or stand still. (2%)

4. In this question, we consider a multi-class classification problem, i.e., to discriminate C ($C \geq 3$) classes. We use softmax function in the output layer to produce probability predictions for each class. In specific, consider the following neural network with 2 input neurons (see the figure below), 2 hidden layers and 3 output neurons. Each hidden layer has 4 neurons. For each hidden neuron, we use *ReLU* activation function. The input and output of this neural network is represented by $\boldsymbol{x} \in R^2$ and $\hat{\boldsymbol{y}} \in R^3$. The weight matrix from the input layer to the first hidden layer is denoted by $W_1 \in R^{2\times4}$. The weight matrix from the first hidden layer to the second hidden layer is denoted by $W_2 \in R^{4\times4}$ and the weight matrix from the second hidden layer to the output layer is denoted by $W_3 \in R^{4\times3}$.



Here, we use one-hot encoding to convert ground truth labels into "one-hot" vectors, where only an entry is 1 and other entries are 0. For example, suppose there are C classes in total, the one-hot encoding of a label c (an integer) is a n-dimensional vector where only the c-th entry is equal to 1 while other entries are 0. We use loss function of cross entropy loss to optimize the network. Given the ground truth label in the one-hot encoding format $y = [y_1, y_2, y_3]$ and the probability prediction of the network $p = softmax([\hat{y}_1, \hat{y}_2, \hat{y}_3]) = [p_1, p_2, p_3]$, the cross entropy loss between the ground truth label and probability prediction is defined as $\ell = -\sum_{i=1}^{3} y_i log p_i$.

(a) Given a training set and test set listed in *train_q4.csv* and *test_q4.csv*, write PyTorch code to learn the above network. Please randomly initialize the weights matrix with torch.nn.init_kaiming_uniform. Please use the torch.optim.SGD optimizer for training, tune the learning rate from set {0.4, 0.1, 0.001}, set the batch size as 256 and the total training epochs as 100. Also, fix the seed using code as following:

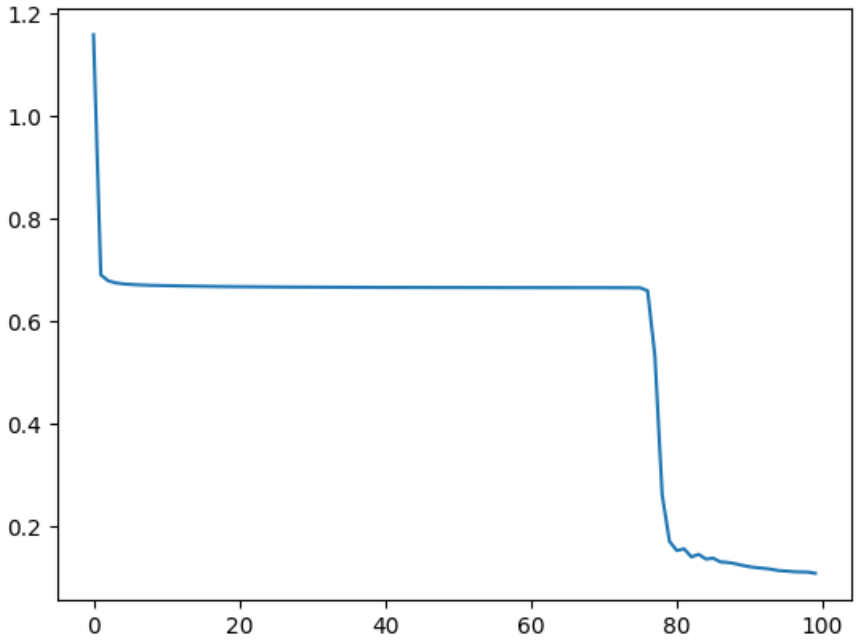
```
import torch
import torch.backends.cudnn as cudnn
import random
import numpy as np
seed = 1443
cudnn.benchmark = False
cudnn.deterministic = True
random.seed(seed)
np.random.seed(seed)
torch.manual_seed(seed)
torch.cuda.manual_seed(seed)
```

Present the followings: 1) plot the curve of training loss, 2) test the network on the given test set and report the testing accuracy. What's your findings according to these training loss curves and testing accuracy with different learning rates? (20%)

Solution:

1)

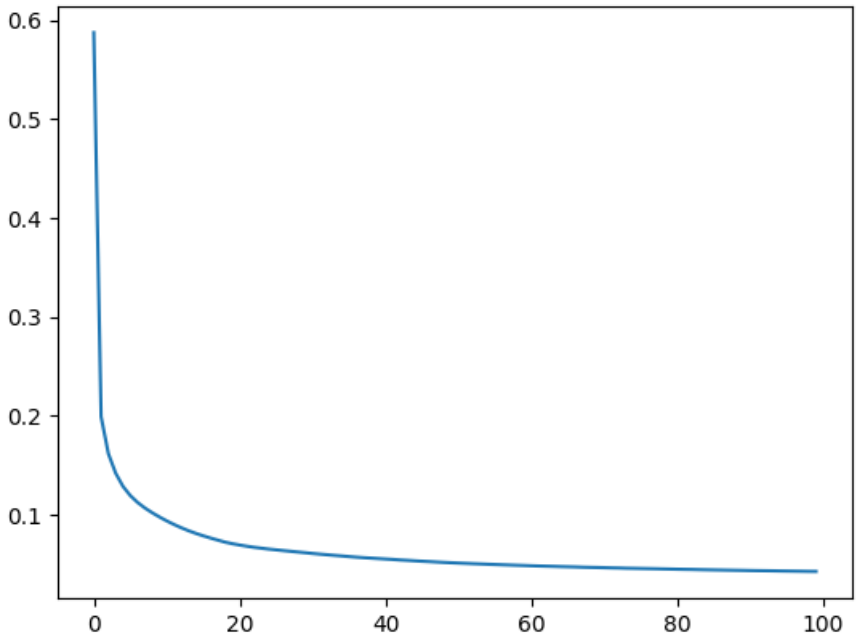
Learning rate = 0.4:



Test Accuracy: 80.80%

(5%)

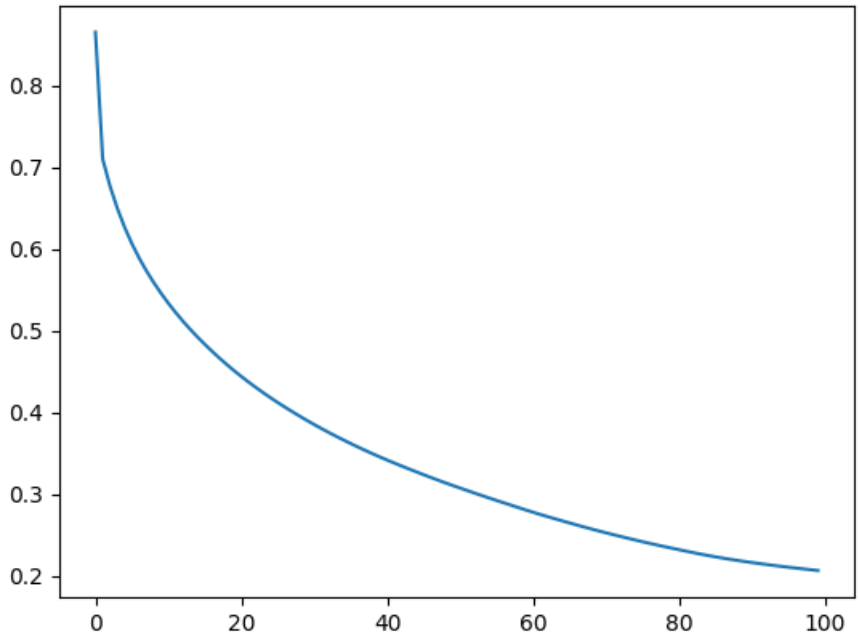
Learning rate = 0.1:



Test Accuracy: 94.50%

(5%)

Learning rate = 0.001:



Test Accuracy: 86.70%

(5%)

2) A learning rate that is too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal solution quickly or an unstable training process..

(5%)

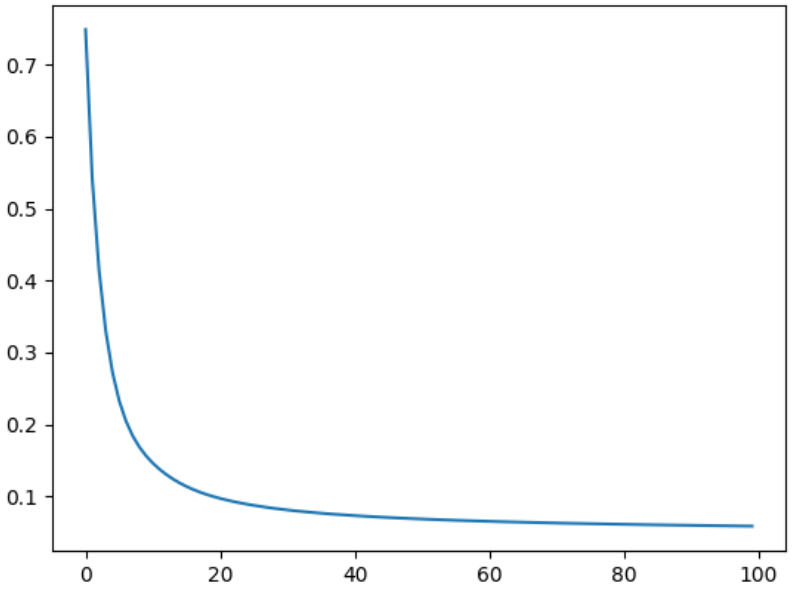
(b) Let's further study the impact of weight initialization. Torch.nn.init provides multiple methods for weight initialization. Please try your best to tune the weight initialization schemes to make the test accuracy ≥ 95%. Please try at least two methods and show your results. For fair evaluation, please use the torch.optim.SGD optimizer for training, set the learning rate as 0.1, the batch size as 256 and the total training epochs as 100. Also, fix the seed using code as following:

```
import torch
import torch.backends.cudnn as cudnn
import random
import numpy as np
seed = 1443
cudnn.benchmark = False
cudnn.deterministic = True
random.seed(seed)
np.random.seed(seed)
torch.manual_seed(seed)
torch.cuda.manual_seed(seed)
```

After that, please present the followings: 1) plot the curve of training loss, 2) test the network on the given test set and report the testing accuracy. (Hint: Find the documentation for torch.nn.init at https://pytorch.org/docs/stable/nn.init.html) (5%)

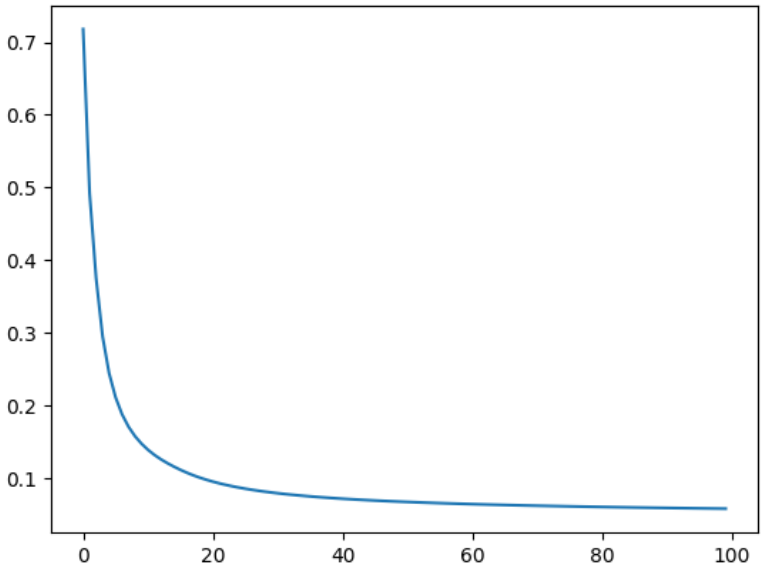
Solution:

Xavier_normal



Test Accuracy: 95.95%

Kaiming normal



Test Accuracy: 96.35%

(5%)

Submission:

Submit a single file named <ID>_asmt3.pdf, where <ID> is your student ID.
Your file should contain the following header. Contact Professor Dou before submitting the assignment if you have anything unclear about the guidelines on academic honesty.

```
CSCI3230 / ESTR3108 2023-24 First Term Assignment 3

I declare that the assignment here submitted is original except for source
material explicitly acknowledged, and that the same or closely related material
has not been previously submitted for another course. I also acknowledge that I
am aware of University policy and regulations on honesty in academic work, and
of the disciplinary guidelines and procedures applicable to breaches of such
policy and regulations, as contained in the following websites.

University Guideline on Academic Honesty:
http://www.cuhk.edu.hk/policy/academichonesty/
Faculty of Engineering Guidelines to Academic Honesty:
http://www.erg.cuhk.edu.hk/erg-intra/upload/documents/ENGG_Discipline.pdf

Student Name: <fill in your name>
Student ID  : <fill in your ID>
```

Submit your files using the Blackboard online system.

Notes:

1. Remember to submit your assignment by 23:59pm of the due date. We may not accept late submissions.
2. If you submit multiple times, **ONLY** the content and time-stamp of the **latest** one would be considered.

University Guideline for Plagiarism

Please pay attention to the university policy and regulations on honesty in academic work, and the disciplinary guidelines and procedures applicable to breaches of such policy and regulations. Details can be found at http://www.cuhk.edu.hk/policy/academichonesty/. With each assignment, students will be required to submit a statement that they are aware of these policies, regulations, guidelines and procedures.