CS464 Introduction to Machine Learning
Fall 2024
Homework 2

Due: November 30, 2024 23:59

Instructions

- For this homework, you may code in any programming language of your choice.
- You are NOT allowed to use any machine learning packages, libraries or toolboxes for this assignment (such as scikit-learn, tensorflow, keras, theano, MATLAB Statistics and Machine Learning Toolbox functions, e1071, nnet, kernlab etc.) unless otherwise stated.
- Submit a soft copy of your homework to Moodle.
- Upload your code and written answers to the related assignment section on Moodle (.TAR or .ZIP). Submitting hard copy, handwritten or scanned files is NOT allowed.
- The name of your compressed folder must be "`CS464_HW2_Section#_Firstname_Lastname`" (i.e., `CS464_HW2_1_sheldon_cooper`). Please do not use any Turkish characters in your compressed folder name.
- Your code should be in a format that is easy to run and must include a driver script serving as an entry point. You must also provide a README file with clear instructions on how to execute your program.
- This is an individual assignment for each student. That is, you are NOT allowed to share your work with your classmates.
- If you do not follow the submission routes, deadlines and specifications (codes, report, etc), it will lead to significant grade deduction.
- If you have any questions about the questions, you can contact:
  - ipek.oztas@bilkent.edu.tr

1 PCA Analysis [50 pts]

In this task, you will analyze fake face images using Principal Component Analysis (PCA). Specifically, you will work with a subset of the 140k Real and Fake Faces dataset[1] that contains images generated by StyleGAN. You are required to use only the test set of fake images, provided in the file `StyleGAN/fake.zip`[2]. This dataset contains $10,000$ images of fake faces. For this question, you may not use any libraries to perform PCA calculations. Instead, you are expected to implement the PCA algorithm yourself. To find eigenvalues and eigenvectors, it is recommended to use the `numpy.linalg.eig` function as part of your implementation.
The images are resized to 64×64 pixels using bilinear interpolation[3]. Before analysis, each image (originally of size $64 \times 64 \times 3$) must be flattened to a 4096×3 matrix. Note that the PIL library reads image files in the `uint8` format. Since unsigned integer values cannot be negative, this format may lead to issues in later calculations. To avoid this, consider converting the data type to `int` or `float32`.

Note that all images are 3-channel **RGB**. Create a 3-D array, X, of size $10000 \times 4096 \times 3$ by stacking flattened matrices of the images provided in the dataset. Slice X as $X_i = X[:,:,i]$, where i corresponds to the three indexes (0: **R**ed, 1: **G**reen, and 2: **B**lue), to obtain color channel matrix (10000×4096) of all images for each channel.

Question 1.1 **[15 pts]** Apply PCA on X_i's to obtain first 10 principal components for each X_i. Report the proportion of variance explained (PVE) for each of the principal components and their sum for each X_i. **Discuss your results and find the minimum number of principal components that are required to obtain at least 70% PVE for all channels.**

Question 1.2 **[15 pts]** Using the first 10 principal components found for each color channel, reshape each principal component to a 64×64 matrix. Later, normalize the values of each of them between 0 and 1 using the min-max scaling method[5]. After scaling them, stack corresponding color channels (R, G, and B) of each principal component to obtain 10 RGB images of size $64 \times 64 \times 3$, which are the visuals of eigenvectors. Display all and discuss your results.

Question 1.3 **[20 pts]** Describe how you can reconstruct a face image using the principal components you obtained in Question 1.1. Use the first k principal components to analyze and reconstruct the first image [6] in the dataset where $k \in \{1, 50, 250, 500, 1000, 4096\}$. In order to reconstruct an image, you should first calculate the dot product with principle components and the image. Later, you project the data you obtained back onto the original space using the first k eigenvectors. Discuss your results in the report.

Hint: Do not forget to add up the mean values at the end of the reconstruction process if you subtracted them from the data in Question 1.1 to calculate the principle components.

[1] https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces/data

[2] https://drive.google.com/file/d/1NLfnWvmIlP9dvQugOugxAKXgzI2qK_i7/view?usp=sharing

[3] Implemented with `PIL.Image.open(image_path).resize((64, 64), Image.BILINEAR)` in the PIL library[4]

[5] https://www.oreilly.com/api/v2/epubs/9781788627306/files/assets/ffb3ac78-fd6f-4340-aa92-cde8ae0322d6.png

[6] The order of the images may differ in different operating systems. The name of the second image is `image_9577.png`

2 Logistic Regression [50 pts]

For this question, you are asked to develop a Multinomial Logistic Regression Classifier model to classify fashion images extracted from the FASHION MNIST dataset[7]. The FASHION MNIST database is 70,000 grayscale images of fashion articles, including 10 classes given in Figure1. It comprises 70,000 examples, 60,000 being the training data and the remaining 10,000 being the test data. The images are size-normalized and centered in a 28x28 image. Since the dataset only contains training and test data, you must create your own validation dataset by separating the first 10,000 images from your training data and their corresponding labels. Ultimately, you will have 50,000 training, 10,000 test, and 10,000 validation images. You are provided with a script to upload and read this data. Please check the script for Question 2 given in Moodle. The corresponding files are as follows:

- `train-images-idx3-ubyte`
- `train-labels-idx1-ubyte`
- `t10k-images-idx3-ubyte`
- `t10k-labels-idx1-ubyte`

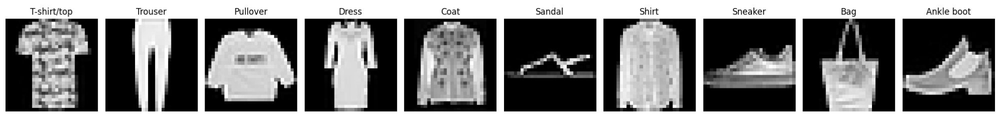

Figure 1: FASHION MNIST Dataset

Since you are asked to implement multinomial classification, you need to turn the labels into their one-hot-encoded version and initialize as many weight vectors as the number of your unique labels. Also, unlike in the Binomial Logistic Regression, you will use Softmax as the activation function instead of Sigmoid. The formula for Softmax activation function is provided in equation (2.1), where z is the input vector, i is the index of the element in the output vector, and K is the total number of classes.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.1}$$

Following this formulation, the update rule for weights is based on the derivative of the cross-entropy loss. It can be calculated using the difference between the target values and softmax outputs. For a detailed mathematical derivation of softmax, please visit the link provided[8]. While updating the weights of your model, remember to add the $L2$ regularization term given in (2.2). You need to take its derivative and combine it with the gradient in your loss formula.

$$L2_{\text{reg}} = \frac{\lambda}{2} \sum_{i=1}^{N} w_i^2 \tag{2.2}$$

As in the first question, you need to flatten your images of size 28x28 to get a 784 dimensional vector for each image. Also, in the dataset, feature scales are significantly different from each other. You need to normalize the data to train a model not influenced by feature scales. You can apply min-max normalization to scale the features in the range [0,1]. Formulation of this normalization is provided in equation (2.3). Since images consist of arrays of integers ranging from 0 to 255, X_{min} here is 0, and X_{max} is 255. So, according to equation (2.3), you need to divide your flattened image arrays into 255 to normalize them.

$$\hat{x} = \frac{x - X_{min}}{X_{max} - X_{min}} \tag{2.3}$$

[7] https://www.kaggle.com/code/ohwhykate/fashion-mnist-classification
[8] https://peterroelants.github.io/posts/cross-entropy-softmax/

Note: Use the same data split in all parts of the assignment to perform a fair split between classifiers for parameter selection. Also, you should train each model for 100 epochs for your experiments

Hint: Try to use as much as vectorized operations using numpy instead of for loops.

Implement a Logistic Regression Classifier for the aforementioned task. For this part, initialize your weights from a Gaussian distribution, where the weights are initialized as $\mathcal{N}(\mu = 0, \sigma = 1)$. Also, you should initialize the batch size as 200, the learning rate as 5x10^{-4}, and the L2 regularization coefficient (λ) as 10^{-4}, this will be your default model. Afterward, you will experiment with these hyperparameters to find the best model. While doing your experiments, you will only change the requested hyperparameters and keep the others as their mentioned default values.

Question 2.1 [15 pts] Train the default model described above. Display the test accuracy and confusion matrix for that case.

Question 2.2 [15 pts] For this part of the question, you will do separate experiments on the hyperparameters mentioned. Remember, you only need to change the mentioned hyperparameter types according to the given values and keep the other default ones. You will change one hyperparameter at a time. Try your model using the hyperparameters given below and compare their performances:

- Batch size: 1, 64, 50000
- Weight initialization technique: zero initialization, uniform distribution, normal distribution
- Learning rate: 0.01, 10^{-3}, 10^{-4}, 10^{-5}
- Regularization coefficient (λ): 10^{-2}, 10^{-4}, 10^{-9}

To exemplify, you need to evaluate your model performance based on batch sizes 1, 64, and 3000 by keeping other default hyperparameter values. After running your model with these values, you need a graph with epochs at the x-axis and resulting accuracies at the y-axis. You need to use legends to show the individual performances of given batch sizes. Do not forget to adjust the titles of the tables accordingly. You should perform this procedure for each hyperparameter given above.

Question 2.3 [5 pts] After you perform the above experiments, you need to select the best values for each of the hyperparameters (and the best-performing initialization technique for weights) and create the optimal model. You need to display the test accuracy and confusion matrix for the best model.

Question 2.4 [10 pts] As mentioned in the earlier parts of this section, you have initialized 10 (number of labels) weight vectors for your classification task. In this part, you need to visualize your finalized weight vectors (after your best model is trained) and print them as images. One line of code is provided for you to visualize your weights; please check the script that you are given in Moodle. Keep in mind that we expect some blurriness in the finalized weight images. After you obtain your results, comment on their look and what they might represent.

Question 2.5 [5 pts] Using the best model, calculate precision, recall, F_1 score and F_2 score for each class. Comment on the results using the confusion matrix you obtained in Question 2.3 and the weight images you obtained in Question 2.4.