CS464 Introduction to Machine Learning
Fall 2022
Homework 2

Due: Dec 5, 2022 11:59 PM

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 regarding this homework, you can contact:
  - a.yildirim@bilkent.edu.tr for Q1
  - yusuf.dalva@bilkent.edu.tr for Q2

1 PCA & Dogs [60 pts]

In this question, you are expected to analyze dog images using PCA. In this part of the assignment, you will be working on only the dog images inside both the validation and the training sets of the Animal Faces dataset[1]. You are requested to use the combined versions of these images `afhq_dog.zip`[2] instead of combining them by yourself. The provided dataset is composed of 5239 images of dogs. For this question, the use of any library for PCA calculations is not allowed. You are requested to implement the PCA algorithm by yourself. It is suggested to use the `numpy.linalg.eig` or `numpy.linalg.svd` functions to find the eigenvalues and the eigenvectors in your calculations.

Before the analysis, resize the images to 64×64 pixels by using the bilinear interpolation method[3] implemented in the PIL library[4]. Then, flatten all images of size $64 \times 64 \times 3$ to get 4096×3 matrix for each image. Remember that the PIL library reads the image files in uint8 format. Since unsigned integer values cannot be negative, the calculations in the following parts may fail. In such a case, the problem can be solved by converting the data type to int or float32.

Note that all images are 3-channel **RGB**. Create a 3-D array, X, of size $5239 \times 4096 \times 3$ 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) for obtaining each color channel matrix (5239×4096) independently for all images.

Question 1.1 [20 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 [20 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 an original dog image using the principal components you obtained in Question 1.1. Use 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, which you subtracted from the data to calculate the principle components in Question 1.1, at the end of the reconstruction process.

[1] https://www.kaggle.com/datasets/andrewmvd/animal-faces

[2] https://drive.google.com/file/d/1qniybe5sME_UzgYFc0G0cqw_TqPdHe7v/view?usp=sharing

[3] PIL.Image.open(image_path).resize((64,64), Image.BILINEAR)

[4] https://pypi.org/project/Pillow/

[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 first image: `flickr_dog_000002.jpg`

2 Logistic Regression [40 pts]

For this question, you are asked to develop a model to detect whether a smart grid is unstable. To briefly describe, "The grid" refers to an infrastructure that enables delivering electricity from power plants to homes and businesses. Among these grids, "Smart Grids" can respond to changing demand over time [7]. As a data scientist, you can determine whether a given smart grid has problems in terms of stability so that the respective partners can make further improvements.

You will use the dataset file `dataset.csv`, which includes a preprocessed version of the Smart Grid Stability dataset for this task, originally hosted at the UCI Machine Learning Repository [8]. This dataset is produced as a result of a simulation involving a producer and four customers. Your data involves 12 continuous values and 1 binary value corresponding to the class label, indicating whether or not the subjected power plant is stable. In our representation, `unstable` power plants are labeled as 0, and `stable` power plants are labeled as 1. Class 0 represents the positive class here, considering your objective of detecting unstable smart grids. For your convenience, the meanings of the variables are provided to you are given as follows:

- `tau{1, 2, 3, 4}`: Reaction Time for {producer, consumer 1, consumer 2, consumer 3, consumer 4}
- `p{1, 2, 3, 4}`: Power Balance for {producer, consumer 1, consumer 2, consumer 3, consumer 4}
- `g{1, 2, 3, 4}`: Price Elasticity Coefficient (γ) for {producer, consumer 1, consumer 2, consumer 3, consumer 4}
- `label`: Stability label of the smart grid (0: unstable, 1: stable)

To solve the task of detecting unstable smart grids, you will train a logistic regression classifier using the dataset provided.

Maximum Conditional Likelihood Estimators (MCLE) for Logistic Regression classifier with weights w are provided as follows:

$$P(Y = 0|X, w) \approx \frac{1}{1 + e^{w_0 + \sum_i w_i X_i}} \tag{2.1}$$

$$P(Y = 1|X, w) \approx \frac{e^{w_0 + \sum_i w_i X_i}}{1 + e^{w_0 + \sum_i w_i X_i}} \tag{2.2}$$

Following this, the update rule for w is provided below. In the given notation, j denotes a data sample and $\forall j, x_0^j = 1$. Alternatively, you can interpret w_0 as the bias term b, where feature indices start from 1. For the weights w, w_i^t denotes weight i at iteration t. α is the learning rate for the update.

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \alpha \sum_j x_i^j [y^j - \hat{P}(Y^j = 1|x^j, w^{(t)})] \tag{2.3}$$

To get a clue on what score you should go for, you perform a training with a Logistic Regression classifier provided by a library (eg. `scikit-learn` for Python). Your implementation must be your own work (these packages cannot be used in your submitted work), but to set a baseline for yourself you can follow such an approach. As these implementations make some implicit assumptions in their code, you do not have to exactly match the metrics but your results should be close.

In this assignment, we did not provide you with a train-validation-test split. Thus, you should create your splits by yourself. The distribution of your data to the splits is as follows:

- Training set: 70% of the data
- Validation set: 10% of the data

[7] https://www.smartgrid.gov/the_smart_grid/smart_grid.html
[8] https://archive.ics.uci.edu/ml/datasets/Electrical+Grid+Stability+Simulated+Data+#

- Test set: 20% of the data

In the dataset, feature scales are significantly different from each other. You need to normalize the data to train a model that is 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 2.4. While selecting the X_{min} and X_{max}, use only the samples from the training set since you will apply this normalization to unseen samples if the classifier is used in a real-world scenario.

$$\hat{x} = \frac{x - X_{min}}{X_{max} - X_{min}} \tag{2.4}$$

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.

Question 2.1 **[15 pts]** Implement a Logistic Regression Classifier for the aforementioned task. For this part, initialize your weights from a Gaussian distribution, where for the initial weights $w \sim \mathcal{N}(\mu = 0, \sigma = 1)$. Additionally, experiment on training with stochastic gradient ascent, mini-batch gradient ascent (batch size = 64), and full-batch gradient ascent. Provide the confusion matrix for your best model. How does your model performance change depending on the batch size? To analyze this behavior, plot curves based on validation accuracy at every epoch. In your plot, the x-axis should indicate the number of epochs, and the y-axis should indicate the accuracy value. You can show the batch size on a legend. Does it affect the training time? Simple insights will suffice here.

Question 2.2 **[7 pts]** By using mini-batch gradient ascent (batch size = 64), now experiment with the effect of the initialization technique. In addition to initializing from a Gaussian distribution, try to initialize your weights from a uniform distribution, and lastly, initialize your weights as zeros. Do different initialization techniques make a difference for the Logistic Regression model? Provide a plot similar to the previous question, where you assess this behavior on the validation set. Provide the confusion matrix for your best model.

Question 2.3 **[8 pts]** As another hyperparameter, now experiment on different learning rates. For this question, you will initialize your weights where $w \sim \mathcal{N}(\mu = 0, \sigma = 1)$ and use the batch size that you found optimal. For the learning rate values, try training your model with $\alpha \in \{1, 10^{-3}, 10^{-4}, 10^{-5}\}$. Provide a plot to summarise the change in validation accuracy depending on the learning rate. Provide the confusion matrix for your best model. Which learning rate value would you prefer, considering you should enable continuous learning in the given number of iterations? How does the learning rate affect the effectiveness of the gradient ascent algorithm? Explain.

Question 2.4 **[10 pts]** Considering all the hyperparameters you have inspected, train your optimal model with the best set of hyperparameters you have found. Then, report the confusion matrix for that model on the test set. Additionally, you should report accuracy, precision, recall, F_1-score, F_2-score, $F_{0.5}$-score, and false positive rate (also known as false alarm rate). Considering that your task was to correctly classify whether a smart grid is unstable, which metrics are most important for you? Should you consider the relative importance of detecting positives and negatives the same? Explain.