University of Nebraska Lincoln

Practical Machine Learning
ECEN 478/878

Assignment 2

Fall 2024

Training Linear Neural Networks Using Gradient Descent

Assignment Goals

The objective of this assignment is to learn how to apply the gradient descent algorithm for training Linear Neural Networks (LNNs). This will involve implementing logistic regression for binary classification and softmax regression for multi-class classification, both framed as LNN models. Through this assignment, you will gain practical experience in data preprocessing, training LNN models using gradient descent, optimizing performance through hyperparameter tuning and regularization, and evaluating model performance.

- **Part A**: Binary Classification with Logistic Regression via an LNN
- **Part B**: Multi-class Classification with Softmax Regression via an LNN

Assignment Instructions

Note: You must use TensorFlow Keras API to create the LNN models. You are allowed to use Python libraries such as Scikit-Learn, Pandas, NumPy, and Matplotlib.

i. The code should be written in two Jupyter Notebooks. Use the following naming convention.
   <lastname_1>_<lastname_2>_ assignment2a.ipynb
   <lastname_1>_<lastname_2>_ assignment2b.ipynb
ii. For each experiment, use unique names for your machine learning models.
iii. If you need to partition the data multiple times for training different models, use distinct names for each training and testing subsets.
iv. The Jupyter notebooks should be submitted via Canvas.

The programming code will be graded on **implementation, correctness, and the style of presentation**.

Score Distribution

ECEN 478: 100 points
ECEN 878: 110 points

Part A: Binary Classification with Logistic Regression

In this part, you will classify the binary income class from the UCI Adult Income Dataset. The dataset (*adult.csv*) comprises 14 attributes, including both categorical and numerical features. The target variable is "income", which is a binary class (<=50K, >50K). Your task is to predict whether a person earns over 50K annually.

You will implement this using a Logistic Regression via an LNN model in TensorFlow Keras.

Setup your environment

- Load the CSV file as a Pandas DataFrame object. Name it "df_raw". Display the first five rows and a summary of dataset information. [**1 pts**]

Data Preprocessing

- Check for missing values in the dataset (display using the print method) and handle them using appropriate techniques. Finally, display whether missing values exist. [**2 pts**]
- Encode categorical variables into numerical format. [**1 pts**]
- Create a new DataFrame "df" that includes both numeric and encoded categorical columns without redundancy. [**2 pts**]
- Create a deep copy of this DataFrame "df_copy" for use in Experiment 2. [**1 pts**]

Data Separation

- Create a "target" DataFrame containing the target variable and a "features" DataFrame containing all feature columns. [**1 pts**]
- From the "features" and "target" DataFrame objects, create a NumPy ndarray for the feature matrix X and a 1D array for the target y. [**1 pts**]

Train-Test-Validation Split [**3 pts**]

- Split the dataset into training and test subsets (20% for testing).
- Then, further split the training set into training and validation subsets (20% for validation).
- Display the shape of each subset for both the feature matrix and target array.

Standardization

- Standardize the three data subsets. Ensure that there is no data leakage. [**2 pts**]

Model Construction

- Create an LNN model for binary classification. Initially, the Dense layer should have the "kernel_regularizer" set to None. Later you will change this value as instructed below. **[3 pts]**
- Display the model summary. **[2 pts]**

Experiments: Conduct the following experiments. For each experiment:

- Display learning curves (accuracy vs. epochs, and loss vs. epochs).
- Clearly annotate your code.
- Report performance: Train and test accuracy, test confusion matrix, and test classification report

Hyperparameter tuning: For the following two experiments, you will perform hyperparameter tuning to enhance the performance of your LNN models. This tuning should be conducted using the Keras Tuner library, where you can choose either the RandomSearch or Hyperband algorithms to efficiently explore the hyperparameter space. Refer to the following tutorial on hyperparameter tuning using Keras Tuner: https://github.com/rhasanbd/Hyperparameter-Tuning-with-Keras-Tuner-A-Practical-Guide

Specifically, you will tune the learning rate, number of epochs, and mini-batch size. Additionally, you must develop your own heuristic to define the lower and upper ranges for these hyperparameters and briefly state your heuristic in no more than a few lines.

For the learning rate, you might define your heuristic as follows: "*We set the lower and upper range for the learning rate empirically by conducting several initial experiments with values of 0.2 and 0.9, respectively, while keeping the other hyperparameters at their default or standard values. Additionally, we looked at the hyperparameter tuning range for similar models on analogous datasets to gain an initial idea of the appropriate range.*"

Experiment 1: Optimal Logistic Regression LNN **[4 pts]**

- Tune hyperparameters, including learning rate, number of epochs, and mini-batch size. You may also apply regularization (both weight-based and early stopping) as needed to optimize performance.

Experiment 2: Comparing with No Standardization

- Using the deep copy of the DataFrame "df_copy", repeat the steps to create the target and feature DataFrames. **[1 pts]**
- Split the dataset into training and test subsets (20% for testing) and then into training and validation subsets (20% for validation) without standardizing the dataset. **[1 pts]**

- Create an optimal logistic regression LNN model. Select optimal hyperparameters and regularizers to ensure that the test performance of this experiment is comparable to that of Experiment 1. You must optimize the model's performance to align it closely with Experiment 1. **[5 pts]**

Use the following table in your written report to present the results. Adjust the table width as necessary.

	Train Accuracy	**Test Accuracy**	**Test Confusion Matrix**	**Test Classification Report**
Experiment 1				
	Justification for the hyperparameter ranges			
Experiment 2				
	Justification for the hyperparameter ranges			

Questions:

- Q1) What differences did you observe in the learning curves between experiments 1 and 2? Explain the performance differences. Which hyperparameter had the most significant impact on performance in Experiment 2? Why? **[5 pts]**
- Q2) Between Experiment 1 and your optimal k-NN model from Assignment 1 (Part A), which model performed better? Report the k-NN test accuracy and test confusion matrix from the previous assignment. Provide a clear analysis of the two models, discussing the advantages and disadvantages of the learning approaches used. **[5 pts]**

Part B: Multi-Class Classification with Softmax Regression

In this part, you will classify the three varieties of wheat seeds from the UCI Seeds Dataset: Kama, Rosa, and Canadian. The Seeds Dataset (*seeds.csv*) contains 210 samples of wheat seeds, each characterized by 7 features, which are geometrical measurements of the seeds:

- Area: The area of the seed.
- Perimeter: The perimeter of the seed.
- Compactness: Calculated as Area / Perimeter^2.
- Length of Kernel: The length of the seed kernel.
- Width of Kernel: The width of the seed kernel.
- Asymmetry Coefficient: A measure of the seed's asymmetry.
- Length of Kernel Groove: The length of the groove on the kernel.

You will use Softmax Regression via an LNN model implemented in TensorFlow Keras. Use a separate Jupyter Notebook for Part B.

Set Up Your Environment

- Load the Seeds dataset CSV file as a Pandas DataFrame object. Name it "df_seeds". Display the first five rows and a summary of dataset information. [**2 pts**]

Data Preprocessing

- Check for missing values in the dataset and handle them using appropriate techniques. [**2 pts**]
- The last column of the DataFrame is the target variable. Create a "target_seeds" DataFrame with one column for the target variable and a "features_seeds" DataFrame with all feature columns. [**2 pts**]
- From the "features_seeds" and "target_seeds" DataFrame objects, create a NumPy ndarray for the feature matrix X and a 1D array for the target y_labels. [**2 pts**]
- Adjust the label array y_labels (which contains values 1, 2, and 3) to a new array y such that y=y_labels −1. The new labels would be: 0, 1, and 2. This ensures the effective use of the sparse_categorical_crossentropy loss function. [**2 pts**]

Train-Test-Validation Split [**3 pts**]

- Split the dataset into training and test subsets (20% for testing).
- Then, further split the training set into training and validation subsets (20% for validation).
- Display the shape of each subset for both the feature array and target array.

Model Construction

- Create an LNN model for performing 3-class classification. Initially, the Dense layer should have the "kernel_regularizer" set to None. Later you will change this value as instructed below. [**3 pts**]
- Display the model summary. [**2 pts**]

Standardization

- Standardize the three datasets. Ensure that there is no data leakage. [**2 pts**]

Conduct the following experiments. For each experiment:

- Display learning curves (accuracy vs. epochs, and loss vs. epochs).
- Clearly annotate your code.
- Report performance: Train and test accuracy (for experiments 3 – 6).
- Report performance: Train and test accuracy, test confusion matrix, and test classification report (for experiments 7 and 8).

Experiment 3: Baseline Softmax Regression LNN

- Set kernel_regularizer=None, learning rate=0.001, number of epochs=200, and batch size=64. [**3 pts**]

Question

- Q3) Based on the learning curves for accuracy and loss (which should be included in your written report along with this answer), does the model exhibit overfitting, underfitting, or neither? Provide a brief explanation of your observations. [**3 pts**]

Experiment 4: L2 Regularization

- Use L2 regularization for kernel_regularizer, learning rate=0.001, number of epochs=200, and batch size=64. [**3 pts**]
- Find a suitable value for the L2 regularizer that maximizes validation and test performance. [**4 pts**]

Question

- Q4) Based on the learning curves for accuracy and loss (which should be included in your written report along with this answer), does the model exhibit overfitting, underfitting, or neither? Provide a brief explanation of your observations. [**3 pts**]

Experiment 5: L1 Regularization

- Use L1 regularization for kernel_regularizer, learning rate=0.001, number of epochs=200, and batch size=64. [**3 pts**]
- Find a suitable value for the L1 regularizer that maximizes validation and test performance. [**4 pts**]

Question

- Q5) Based on the learning curves for accuracy and loss (which should be included in your written report along with this answer), does the model exhibit overfitting, underfitting, or neither? Provide a brief explanation of your observations. [**3 pts**]

Experiment 6: High Learning Rate [**3 pts**]

- Set kernel_regularizer=None, learning rate=100.0, number of epochs=200, and batch size=64.

Question

- Q6) What behavior do you observe from the learning curves (accuracy and loss)? Include the learning curves along with this answer and explain. [**3 pts**]

Experiment 7: Optimal Model Creation [**8 pts**]

- Create an optimal model by varying hyperparameters: learning rate, number of epochs, and mini-batch size. If using a weight-based regularizer (L1 or L2), report the regularizer and its strength along with the optimal hyperparameters.

Experiment 8 [**required only for graduate students**]: **Optimal k-NN Model**

Create an optimal k-NN model for the Seeds classification problem by hyperparameter tuning. Apply suitable any optimization techniques. [**6 pts**]

Question:

- Q7) [**required only for graduate students**] Between Experiments 7 and 8, which model performed better? Provide a clear analysis of why one model outperformed the other, discussing the differences between the two learning approaches. [**4 pts**]

Use the following table in your written report to present the results. Adjust the table width as necessary.

	Setting (hyperparameters, regularizers, etc.)	**Train Accuracy**	**Test Accuracy**	**Test Confusion Matrix**	**Test Classification Report**
Ex 3	kernel_regularizer=None learning rate=0.001 number of epochs=200 batch size=64			Not required	Not required
Ex 4	kernel_regularizer=L2 learning rate=0.001 number of epochs=200 batch size=64			Not required	Not required
Ex 5	kernel_regularizer=L1 learning rate=0.001 number of epochs=200 batch size=64			Not required	Not required
Ex 6	kernel_regularizer=None learning rate=100.0 number of epochs=200 batch size=64			Not required	Not required
Ex 7					
Ex 8					

Deliverables:

You will submit two artifacts.

- Two Jupyter Notebooks containing all experiments for Parts A and B, respectively. For each experiment, display the required performance measures. Ensure your code is clearly annotated and includes header descriptions for each block. ***Note that if the notebook lacks annotations or has poor annotation, 5 points will be deducted***.
- A **PDF** copy of the written report, including a cover page (available on Canvas). The report must include the following items.

a) Use the two tables to report results of the experiments.

b) Answer to all questions. For each question, include the pair of learning curves as instructed.

The PDF file should be named as follows: <lastname_1>_<lastname_2>_ assignment2.pdf