UTSA

CS 6243 Machine Learning

EE 6363: Advanced Machine Learning

Assignment: HW3

I. Assignment Instructions

Implement all tasks.

Submit your report in the dedicated folder on CANVAS.

Submit your report by the deadline. Delayed reports will not be graded.

Deadline: 11/08/2023, 11:59pm.

Reports must be typed and uploaded in PDF format. Handwritten reports will not be graded.

Any requested code must be presented at the end of your report in a dedicated appendix, titled “Code.” All codes must have adequate comments.

All display equations must be numbered. All figures must be numbered and captioned.

You can use any source (notes, books, online), but it must be cited in a References List at the end of your report.

Do not outsource this assignment (or parts of it) to another intelligent entity (whether human or AI). Except for what you explicitly cite, what you submit must be your own intellectual work.

Grade: This assignment corresponds to 6% of your final grade. You will be graded based on 1) correctness, 2) completeness, 3) clarity (equal weight).

Teams: Work in teams. Each team member must submit the exact same report on CANVAS (only students that submit a report will receive a grade). It is expected that all team members will put equal effort into this assignment. The instructor may determine each student’s grade on this assignment by individual examination during course office hours.

II. Objective

The main objective of this assignment is to implement Softmax Regression with Mini-Batch Stochastic Gradient Descent and K-Fold Cross-Validation from scratch. You'll also evaluate the performance using different metrics and visualizations.

III. Dataset

You will work with the Wine dataset, a classic dataset for classification tasks. The dataset contains 178 instances and 13 features. Your task is to classify the wine into one of the three classes. The dataset can be found here: https://archive.ics.uci.edu/dataset/109/wine.

IV. Tasks

Task 1: Dataset Preparation [10%]

1. Load the Wine dataset and extract the features (`X`) and labels (`y`).
2. Filter the dataset to keep only the 40 first samples from each class.
3. Perform feature scaling using z-score normalization. Z-score normalization is a technique to transform each feature so that it has a mean of 0 and variance of 1 (by “mean” and “variance”, we mean the sample-average estimates thereof, using all available samples).

Task 2: Basic Functions [30%]

1. Implement the `softmax` function that takes the logits as input and returns the softmax probabilities. In the softmax function, subtract the max value of the logit vector to prevent numerical instability during the exponentiation step.
2. Implement the `kfold_split` function for K-Fold Cross-Validation.
3. Implement the `accuracy` function to manually calculate the accuracy of predictions. Accuracy is defined as the number of correct predictions divided by the total number of predictions.
4. Implement the `confusion_matrix` function to manually create a confusion matrix. The confusion matrix is a table that describes the performance of a classification model. Each row represents the actual class, and each column represents the predicted class.

All helper functions such as `softmax`, `kfold_split`, `accuracy`, and `confusion_matrix` should be implemented manually and should not rely on built-in functions for their core functionality.

Task 3: Main Loops [30%]

Your code will be computing results for various mini-batch sizes.

1. **For each mini-batch size**, use your `kfold_split` function to generate train and test data indices for each of the K folds.
2. **For each fold**, initialize W matrix and run a fixed number of epochs.
1. **For each epoch**, shuffle the training data using `np.random.permutation()`. Then split them in a sequence of non-overlapping mini batches. Then use mini-batch stochastic gradient descent to sequentially update the model on all the defined mini-batches. The gradient of each mini batch has to be calculated based on the formula derived in the class, not a built-in function. At the end of each epoch, capture the computation time using `time.time()` and the accuracy using your function.

   Hyperparameters:
   1. SGD learning rate (Step-Size): 0.1.
   2. Number of epochs per fold: 500.
   3. Mini-batch sizes: Use three different mini-batch sizes: 1, 36, 72.
   4. Number of folds: 10.
   5. Initialize weights at each fold: all zeros.
   6. Regularization parameter: 0.

Task 6: Result Visualization [10%]

1. **Figure 1:** Training and Testing Accuracy vs Epoch Index for Different Mini-Batch Sizes. This figure shows the average training and test accuracies across epochs for different mini-batch sizes. The x-axis represents the epoch index, and the y-axis represents the accuracy. Solid lines indicate training accuracy, and dashed lines indicate test accuracy.
2. **Figure 2:** Training Times vs Epoch Index for Different Mini-Batch Sizes. This figure shows the computational time taken for each epoch for different mini-batch sizes. The x-axis represents the epoch index, and the y-axis represents the time in seconds.
3. **Figure (set) 3:** Final Confusion Matrix for Different Mini-Batch Size. This set of figures shows the average confusion matrix after the last epoch for each distinct mini-batch sizes. Each cell in each confusion matrix represents the number of instances for the predicted label vs. the true label.

Task 7: Discussions [20%]

1. For each figure discuss the results and how they relate to code and theory.

Task 8: Bonus [5%]

1. Experiment with different values of step-size, regularization parameter, mini-batch size, number of epochs and present figures and comparisons.

Must correctly experiment with at least 2 of the hyperparameters above, and present discussions/comparisons, to claim the bonus points.

V. Deliverables

1. Complete Python code covering all tasks, with complete comments.
2. A report including all requested visualizations and presenting all requested discussions.