CIS590/890 DL - Assignment 1 (updated)

1 Exercise 1

In this exercise, you will train and evaluate a linear regression model. You will perform feature scaling. You will also analyze learning curves to gain insights into how the performance (i.e., mean square error) varies with the learning rate and the number of iterations used in the gradient descent algorithm.

The linear regression model will be trained and tested on data collected from homes in suburbs of Boston, Massachusetts, and can potentially be used to make predictions about the monetary value of a home.

1.1 Dataset

The dataset you will use is available in sklearn: `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html`. You can also find it on Canvas (housing.csv). The csv file can be read as a DataFrame using `pandas.read_csv`. The dataset consists of 506 house instances x, and each instance is represented using 13 numeric features, in addition to the target y, which is denoted by `medv` (median value of owner-occupied homes in thousand dollars).

1.2 Sample code

The following TensorFlow tutorial, `https://www.tensorflow.org/tutorials/keras/basic_regression`, together with the linear regression code discussed in class, may be useful in writing your code and answering the questions in this assignment.

1.3 Tasks

1. Load the housing data and randomly split it into training and test subsets (use 70% of the data for training and 30% for test). You can use the `train_test_split` in sklearn to do that. Fixing the *random_state* will ensure you get the same splits every time you run your code.

2. Normalize the features to bring them on a similar scale.

3. Create a linear regression model. The model should have one layer, the output layer with one linear unit, and should take as input instances x representing houses. Use a separate validation set to tune the learning rate and the number of iterations. The validation set can be created when fitting the model, e.g. `model.fit(X_train, y_train, epochs=100, X_val = 0.2)`. Experiment with 5 values for the learning rates.

4. Plot learning curves that show the variation of performance (i.e., mean squared error) with the number of iterations for both training and validations sets for the 5 values of the learning rate that you used.

5. Use the best model identified based on the validation data and run it on the test data.

6. Compare the performance on the training data with performance on the validation data and also performance on the test data. What learning rate gave the best results? How many iterations are needed for convergence? What features contribute the most to the price prediction?

2 Exercise 2

In this exercise, you will train and evaluate a logistic regression model. As for linear regression, you will perform feature scaling. You will also analyze learning curves to gain insights into how the performance (i.e., accuracy) varies with the learning rate and the number of iterations used in the gradient descent algorithm.

The logistic regression model will be trained and tested on breast cancer data collected by Dr. William H. Wolberg, physician at the University Of Wisconsin Hospital at Madison, Wisconsin, USA. Specifically, the model will be trained to classify instances (i.e., fluid samples taken from patients with solid breast masses) as benign or malignant.

2.1 Dataset

The dataset you will use is available in sklearn: `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html`. You can also find it on Canvas (breast-cancer.csv). The csv file can be read as a DataFrame using `pandas.read_csv`. The dataset consists of 569 instances x, and each instance is represented using 30 numeric features, in addition the class variable y (target), which is called `diagnosis` (and takes discrete value B = benign or M = malignant). The first feature/aatribute is a sample `id` and should not be included in the training data when building the model.

2.2 Sample code

The following TensorFlow tutorial, `https://www.tensorflow.org/tutorials/keras/basic_classification`, together with the linear regression code dis-

cussed in class, may be useful in writing your code and answering the questions in this assignment.

2.3 Tasks

1. Load the breast cancer data (don't forget to remove the `id` variable) and randomly split it into training and test subsets (use 70% of the data for training and 30% for test). You can use the `train_test_split` in sklearn to do that. Fixing the *random_state* will ensure you get the same splits every time you run your code. *Note*: You will need to convert the class variable y from categorical, B or M, to numbers like 0 or 1. You can use `LabelEncoder` in `sklearn.preprocessing` to do that.

2. Normalize the features to bring them on a similar scale.

3. Create a logistic regression model that optimizes the cost/loss given by the `sparse_categorical_crossentropy`. Use the `accuracy` as the performance metric (the accuracy is defined as the number of correctly classified instances divided by the number of all instances). The model should have one layer, the output layer with one sigmoid unit, and should take as input instances x representing samples from patients. Use a separate validation set to tune the learning rate and the number of iterations. The validation set can be created when fitting the model, e.g. `model.fit(X_train, y_train, epochs=100, X_val = 0.2)`. Experiment with 5 values for the learning rates.

4. Plot learning curves that show the variation of the loss and also variation of the accuracy metric with the number of iterations, for both training and validations sets, for the 5 values of the learning rate that you used.

5. Use the best model identified based on the validation data and run it on the test data.

6. Compare the accuracy on the training data with the accuracy on the validation data and also accuracy on the test data. What learning rate gave the best results? How many iterations are needed for convergence? What features contribute the most to the prediction?

3 What to submit (updated)

- For each exercise, please submit a Jupiter Notebook containing your code and answers to the corresponding questions. If you use Google Colaboratory, please export your code as a notebooks and upload the files on Canvas, so that we have a timestamp for your submission (links to the Colaboratory notebooks, in addition the Jupyter Notebook files, would also be useful).