Data-X Spring 2018, Homework-03 Author: Sana Iqbal

Due: April 4th, 2018 [Hard deadline]

Part 1: Regularization: [60 points]

In this section, you will get some practice working with regularisation and hyperparameter tuning in prediction models. You will use a python DIGITS dataset to complete this homework.
Load the dataset in your ipython notebook and split it into Training Set (80%) and Test Set (20%).
You will use training and validation sets to make sure that after training the classification algorithm is able to generalize well to new data. You will also use the Validation set performance to tune your hyperparameters. Note that the training and validation sets are created from 80% of the original data using Stratified K-fold cross-validation. Report your model performance on both sets.

Use this data to train a Logistic Regression classifier that predicts what number the image of the digit is (categories 0-9). You will train two different Regularized models, and tune hyper parameters using K-fold Cross validation (use scikit learn inbuilt method of Stratified K-Fold cross-validation
scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html).
Note that the K-fold cross-validation is what creates the validation set for each of the folds. Choose a suitable K, for example 3.

Choose the optimal hyperparameters based on accuracy metrics and confusion matrix, give reasons for your selection.

1. Use L2 regularization i.e set `'penalty'` coefficient equal to `"l2"` and tune regularisation coefficient i.e hyperparameter `'C'`, plot a graph between C values and validation accuracy.
2. Use L1 regularisation i.e set `'penalty'` coefficient equal to `"l1"` and tune regularisation coefficient i.e hyperparameter `'C'`, plot a graph between C values and validation accuracy.

Quick Note:
Regularization does not improve the performance on the data set that the algorithm used to learn the model parameters. It can improve the generalization performance of the model , i.e., the performance on new, unseen data.

You can load and split the dataset using this code block:

```
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt
digits= load_digits()
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(digits.data,
```

```
digits.target, test_size=0.20, random_state=0)
```

You can view the images using this code snippet:

```
plt.imshow(np.reshape(x_train[1], (8,8)), cmap=plt.cm.gray)
```

Part 2. Neural Networks: [140 points]

You are now familiar with Tensorflow. You understand how the complexity of prediction models varies in simple machine learning models and deep neural networks. To reinforce the concept that neural networks are not a black-box, but rather a stream / flow of functions put together, in this exercise you will compare the construction and training of a simple logistic regression model with a Dense Neural Network model.

1. Train a multiclass logistic regression model (softmax regression is a good choice) on the DIGITS dataset in tensorflow. You will create input and output placeholders, define the model initializing weight and bias variables, then define a loss function for the model. Once you create a blueprint of the model, compile or train the model to minimize the loss function. Use minibatch gradient descent to train the model, and use the hyperparameters: `batch_size=100, epochs=200, learning rate=.001`.
Report accuracy on 20% validation data. Plot the training accuracy vs epoch and plot validation accuracy vs epoc. Show your network graph as seen on tensorboard. [40p]

2. Now using the same technique as in the above multiclass logistic regression model, train a vanilla Dense Neural Network using the DIGITS dataset, with the characteristics listed below. Observe that the complexity of a Neural Network depends on the additional layers called 'Hidden Layers', which can extract relevant features and latent information in the data.
Compare the number of weights and bias terms (model parameters) in the DNN with the parameters in the simple multiclass logistic regression model that you trained in the above question.[100p]

The DNN model should have the following characteristics:

a. **Input layer** of size 8*8=64: This layer ingests image data in the form of a vector. Since the images are of size 8X8 pixels, it should have 64 ingesting neurons.
b. Three hidden layers of size 300,200,100**:** Hidden Layers are layers of neurons stacked in between inputs and outputs, allowing neural networks to extract more complex patterns and learn more complicated features.
c. **Output layer** of size 10 (ten categories is needed in order to classify digits 0-9): This layer will give us the probability of the image being a specific number.
d. **TANH activation function** in hidden layers: Activation functions are applied to each node's value to squeeze the output from that node. This produces a new representation of the original data, and allows for non-linearity. Tanh squishes the output from each node to fit between (-1,1) which helps in overcoming the vanishing gradient problem.
e. **The Dropout** ratio should be set to 10% (Ref: tensorflow, dropout): Dropout is a technique where randomly selected neurons are ignored during training.

They are “dropped-out” randomly so that the model does not overfit to training data.

f. Use the cross entropy loss (Ref: tensorflow) and minibatch gradient descent as the optimization function.

g. Hyperparameters:

   i. Batch size: 100

   ii. Epochs: 1000

   iii. Learning rate =.001

Report accuracy on 20% validation data. Plot training accuracy vs epoch. Present your network graph (could be a screenshot from Tensorboard). State the difference in model complexity between the Dense Neural Network model and the multiclass logistic regression model in terms of number of (trainable) model parameters -- i.e., weights + biases.

Part 3: Extra Credit (Mandatory for graduate students) [100 points]

1. Neuron Saturation slows down the training of neural networks, batch normalization is a method of preventing neuron saturation by scaling and centering the inputs to each layer in a neural network. Explain in 100 words what Neuron Saturation is and why it slows down the training process. Explain in 500 words how Batch Normalization is done. (20p)

2. Why do we use activation functions in Neural Networks? List three activation functions used in Neural Networks and derive their first order derivatives. Explain the difference among these functions. (20p)

3. Customize the Neural Network in Part 2 to include however many layers and neurons you want and use batch normalization that works for the train and test settings of the model. You will need to find the batch mean and variance, as well as keep an estimate of the population mean and variance (this is commonly done using the moving average method).
   Limit epochs to 2000 or less, plot the loss function in Tensorboard for every 50th epoch. Report accuracy on 20% validation data. Plot training accuracy vs epoch curve. Show your network graph as seen on tensorboard. Describe how this model is different from the one you used above and how do you expect these changes to manifest in the trained model. (60p)