AIGS538 Deep Learning - Programming Assignment 1

Oldenburg, Lukas (49003807)

March 24th, 2022

1 Run the Code

All experiments can be started by running the `main.py` file from the zip folder.

2 Generate Moon Datasets

The generated moon datasets are attached in the zip file. I created three datasets for a respective noise ratio of 0.2, 0.3 and 0.9. The train dataset was used to evaluate the training progress of the model. I used the validation dataset to adapt significant model/training parameters like learning rate, training epochs and number of neurons in the hidden layer. The final accuracy was measured on a pre-trained model, which was inferred on the test dataset.

3 Logistic Regression Classifier

The class `LogisticRegression` in the `logistic_regression.py` file includes all necessary computation steps for performing logistic regression on a certain dataset. The console output of an example training run for every dataset is stored in the `log_file.txt`. As one can see, the losses are decreasing while the accuracy's are increasing during the training process which is the desired behaviour of the logistic regression. Furthermore the accuracy decreases as the noise ratios increases. This makes sense because the linear separability of the data is harder with increasing noise ratios. However, as the test accuracy's slightly distinguish from the training accuracy's the model probably overfits a little. This can be fixed e.g. by adding a regularization term within the loss function. In general the accuracy's on the test set `[82.0%, 83.0%, 71,5%]` for noise ratios 0.2, 0.3, 0.9 are acceptable results. It is not expected that we reach perfect results as the underlying nature of the data is not linearly separable.

4 One Hidden Layer Neural Network Classifier

All necessary computations for the neural network can be found in the `NeuralNetwork` class in the `nn.py` file. This class is implemented to specify wether a single or two hidden layers should be used in training. One could also choose the number of hidden neurons for each layer respectively.

For the number of neurons `[1, 3, 5, 7, 9]` and for each dataset the results are also shown in the `log_file.txt`. Here, we can see that the accuracy on the test set generally increases with a higher number of neurons. This make sense, as this one hidden layer network is basically a logistic regression with adaptable number of parameters (number of neurons) and more parameters could improve the model fitting. However, one can see that even with one or three neurons in the hidden layer, we are able to reach a high accuracy of `84%` and `85.5%` for a 0.2 noise ratio. This is comparable to the logistic regression accuracy, as this neural network fits a linear model to the data just as the logistic regression does.

In addition I achieved better results for the one-hidden-layer neural network with 50 neurons (see `log_file.txt`).

5 Two Hidden Layer Neural Network Classifier

As previously said the desired parameters of the network were determined by using the validation set for evaluating the model. For the two hidden layers `[20, 20]` neurons are used. The model was trained over 200 epochs with a learning rate of 0.5. As you can see in the `log_file.txt` the network can achieve accuracy's on the test set `[95.5%, 92.0%, 72.0%]` for respective noise ratios 0.2, 0.3, 0.9. This results are better as for the logistic regression or the single-hidden-layer network, as the two hidden layer neural network is able to create nonlinear decision boundaries and therefore find a better fit for the dataset. However, its difficult to say if this chosen parameters are perfect. One could perform a grid search to find better fitting hyper-parameters and e.g. improve the compared bad result for the 0.9 noise ratio.

During this project I realized how beneficial and efficient Deep Learning Frameworks like PyTorch are implemented, since the backward pass is automatically computed there. Its a hard and error-prone task to implement the backward pass by hand, even for small and simple networks like the two-hidden-layer feed-forward network. Further I noticed the importance of hyper-parameters, especially the learning rate as it has a big influence on the final model performance.