Name(s): Chengjun Lu
NetID(s): cl78
Team name on Kaggle leaderboard: Chengjun Lu

For each of the sections below, your reported test accuracy should approximately match the accuracy reported on Kaggle.

Perceptron

Briefly describe the hyperparameter settings you tried. In particular, you should list the different values for learning rate and number of epochs you tried. You should also mention whether adding a learning rate decay helped and how you implemented this decay. Report the optimal hyperparameter setting you found in the table below. Report your training, validation, and testing accuracy with your optimal hyperparameter setting.

RICE DATASET

Optimal hyperparameters:	Learning rate=0.01, #epoch=10, decay rate=2
Training accuracy:	99.835
Validation accuracy:	99.808
Test accuracy:	99.835

Fashion-MNIST DATASET

Optimal hyperparameters:	Learning rate = 0.01, #epoch = 20, decay rate = 2
Training accuracy:	84.526
Validation accuracy:	83.190
Test accuracy:	82 .130

The hyperparameters for Perceptron include learning rate, number of epochs and learning decay rate. For MNIST, I started from what is given and tuned the learning rate, including {0.001, 0.005, 0.01, 0.1}. I also added the learning decary rate to help improve the accuracy. The

learning rate is reduced every epoch and is calculated as self.lr = self.lr * (1 / (1 + self.decay_rate * self.epochs)). A bias term was also added to the weight matrix, leading to around 0.01 increase in accuracy. The hyperparameters for the RICE dataset were inherited from MNIST and performed reasonaly well, except that I reduced the number of epochs to 10 because it converged very quickly.

SVM

Describe the hyperparameter tuning you tried for learning rate, number of epochs, and regularization constant. Report the optimal hyperparameter setting you found in the table below. Also report your training, validation, and testing accuracy with your optimal hyperparameter setting.

RICE DATASET

Optimal hyperparameters:	Learning rate=0.001, #epochs=1000, batch size=1024, regularization constant=0.01, decay rate=batch size$^{0.5}$-0.12
Training accuracy:	84.496
Validation accuracy:	82.660
Test accuracy:	81.940

Fashion-MNIST DATASET

Optimal hyperparameters:	Learning rate=0.001, #epochs=1000, batch size=256, regularization constant=0.01, decay rate=batch size$^{0.5}$-0.1
Training accuracy:	99.716
Validation accuracy:	99.725
Test accuracy:	99.560

The hyperparameters for SVM include learning rate, number of epochs, batch size, regularization constant and decay rate. Batch size is suggested to be power of two for computation purposes and I chose a smaller batch size for MNIST to save training time. Some other batch sizes I tried

include {64, 128, 512, 1024, 2048}. Increasing batch size doesn't necessarily guarantee an increase in accuracy. In terms of numbers of epochs, I tried {20, 50, 100, 200, 500, 1000, 2000}. More training epochs increase training accuracy significantly but suffer from marginal diminishment when it goes beyond 1000. So I decided to cut it right at 1000 epochs. Some learning rates I tried include {0.0001, 0.001, 0.005, 0.01, 0.5, 0.1}. The learning rate has to be tuned carefully according to number of epochs. It turned out 0.001 is just right for 1000 epochs with learning rate decay. The learning rate decay is calculated as self.lr = self.lr * (self.decay_rate / np.sqrt(self.batch_size)). Strictly speaking, the learning decay rate is 1 - self.decay_rate / np.sqrt(self.batch_size). Sorry for the bad naming convention. The general idea is to make self.decay_rate / np.sqrt(self.batch_size) a little bit less than one to reduce the learning rate every epoch. How much less than one is determined experientally by multiple runs of the model. The accuracy osciallates up and down initially and reaches steady state after 1000 epochs. A plot is included in the notebook. The regularization constant, in my opinion, does not have a strong impact on accuracies, and I kept it fixed at 0.01. A bias term was also added to the weight matrix, leading to around 0.01 increase in accuracy. The hyperparameters for the RICE dataset were inherited from MNIST, except that batch size was increased to 1000 as we can afford more computation time with the smaller RICE dataset.

Softmax

Once again, describe the hyperparameter tuning you tried for learning rate, number of epochs, and regularization constant. Report the optimal hyperparameter setting you found in the table below. Also report your training, validation, and testing accuracy with your optimal hyperparameter setting.

RICE DATASET

Optimal hyperparameters:	Learning rate=0.01, #epochs=1000, batch size=256, regularization constant=None, decay rate=batch size$^{0.5}$-0.1
Training accuracy:	99.698
Validation accuracy:	99.698
Test accuracy:	99.560

Fashion-MNIST DATASET

Optimal hyperparameters:	Learning rate=0.01, #epochs=1000, batch size=256, regularization constant=None, decay rate=batch size$^{0.5}$-0.09
Training accuracy:	85.020
Validation accuracy:	83.270
Test accuracy:	82.260

The hyperparameters for Softmax include learning rate, number of epochs, batch size, regularization constant (not used), and learning decay rate. Some batch sizes I tried include {64, 128, 256, 512, 1024, 2048}. Increasing batch size doesn't necessarily guarantee an increase in accuracy so I chose a medium one. In terms of numbers of epochs, I tried {20, 50, 100, 200, 500, 1000, 2000}. More training epochs increase training accuracy significantly but suffer from marginal diminishment when it goes beyond 1000. So I decided to cut it right at 1000 epochs. Some learning rates I tried include {0.0001, 0.001, 0.005, 0.01, 0.5, 0.1}. The learning rate has to be tuned carefully according to number of epochs. Learning rate = 0.01 turns out to be the most suitable rate for softmax, which is ten times larger than that of SVM. The learning rate decay formula is the same as that of SVM, `self.lr = self.lr * (self.decay_rate / np.sqrt(self.batch_size))`. Strictly speaking, the learning decay rate is `1 - self.decay_rate / np.sqrt(self.batch_size)`. Sorry for the bad naming convention. The general idea is to make `self.decay_rate / np.sqrt(self.batch_size)` a little bit less than one to reduce the learning rate every epoch. How much less than one is determined experientally by multiple runs of the model. The accuracy osciallates up and down and reaches steady state after 1000 epochs. A plot is included in the notebook. The regularization constant is not used in my implmentation of Softmax. A bias term was also added to the weight matrix, leading to around 0.01 increase in accuracy. The hyperparameters for the RICE dataset were inherited from MNIST and performed quite well.

Logistic

Once again, describe the hyperparameter tuning you tried for learning rate, number of epochs, and threshold. Report the optimal hyperparameter setting you found in the table below. Also report your training, validation, and testing accuracy with your optimal hyperparameter setting.

RICE DATASET

Optimal hyperparameters:	Learning rate=0.5, #epochs=10, threshold=0.5, decay rate=2
Training accuracy:	99.844
Validation accuracy:	99.835
Test accuracy:	99.835

The hyperparameters for logistic regression include learning rate, number of epochs, threshold (not used), and learning decay rate. The first hyperparameter I tuned is number of epochs. Unlike SVM and Softmax which uses mini-batch gradient descent, logistic regression updates its weights for every training sample. It converges very quickly and 10 epochs are sufficient. The learning rate is not that important. I tried {0.01, 0.05, 0.1, 0.3, 0.5} and all performed well. The learning rate is updated according to self.lr = self.lr * (1 / (1 + self.decay_rate * self.epochs)).The decay rate is determined experimentally including {0.1, 0.5, 0.8, 1.2, 1.5, 2}.

References

https://medium.com/analytics-vidhya/learning-rate-decay-and-methods-in-deep-learning-2cee564f910b
https://shaktiwadekar.medium.com/how-to-avoid-numerical-overflow-in-sigmoid-function-numerically-stable-sigmoid-function-5298b14720f6