ECE521 Assignment2 Report

Shengjie Xu (1001175186) (43% Contribution)
Yu Cheng Gu (1001202067) (42% Contribution)
Shihan Zhang (1002055795) (15% Contribution)

August 25, 2020

1 Linear Regression

1.1 Tuning the learning rate

Figure 1 is the curves of training MSE loss over number of epoch, with mini-batch size 500 and zero weight decay. The loss is from the last mini-batch used for training in that epoch. From the plot, learning with $\eta = 0.005$ gives the fastest convergence and lowest MSE. Among these three learning rates, larger learning rate gives faster convergence of loss function.

1.2 Effect of the mini-batch size

Table 1 is the final training MSE with zero weight decay, and learning rate of 0.005. The MSE loss is comparable, however a small mini-batch size is much faster for training, and the best size is 500. In our current implementation, time needed for training with mini batch size 500 is less than half of time needed for size 1500 and 3500.

1.3 Generalization

Table 2 contains the classification accuracy results when using different λ. $\lambda = 0.1$ gives the best validation set accuracy (98%), and the test accuracy in this case is 97.241%.

Weight decay can help to prune unnecessary weights and improve classification accuracy. However, when it gets too large, it will start to penalize useful weights too much and the

Table 1: Final Training MSE for three mini-batch sizes

Mini-batch size	Final Training MSE
500	0.06964706629514694
1500	0.06858054548501968
3500	0.07116446644067764

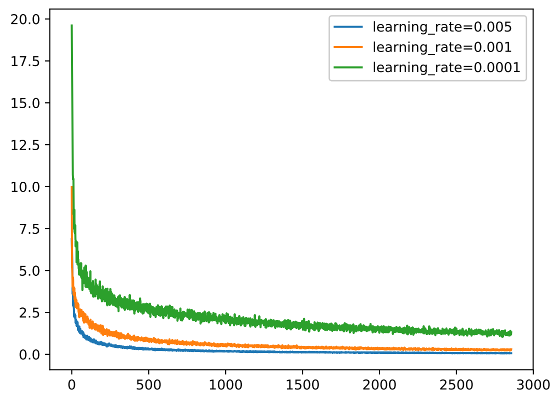


Figure 1: Training MSE versus number of epoch

Table 2: Classification accuracy under different weight decays

λ	Train	Valid	Test
0.0	88.629%	89%	82.759%
0.001	91.886%	86%	87.586%
0.1	97.771%	98%	97.241%
1.0	97.257%	97%	96.551%

accuracy will drop.

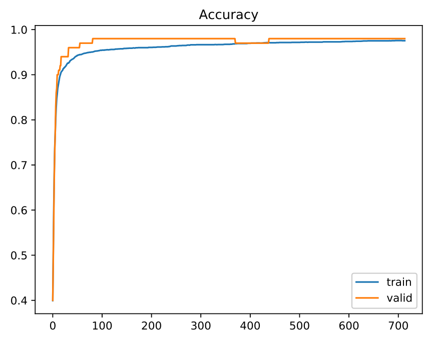
If the hyper parameter λ is tuned using training set, then the training process is prone to over-fitting. Tuning λ is similar to selecting range for hypothesis sets, and the data for this selection should not be used for training, since the result will probably do well only in training set and will have a large generalization error. The selection must be done using a separate validation set.

1.4 Comparing SGD with normal equation

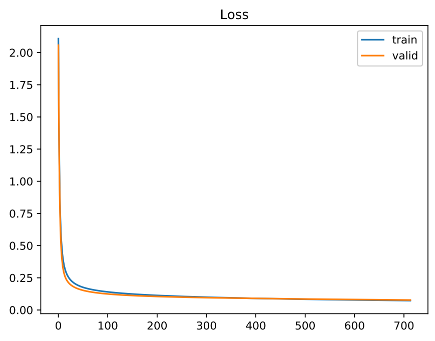
Table 3 is the result of using normal equation. The result is much better (lower MSE and higher accuracy) than that from SGD with zero weight decay, and is not much worse than the result from SGD with tuned λ. However, unlike SGD, using normal equation is very fast, since there is no iterative improvements and it involves merely one set of linear algebra calculations at its core.

Table 3: Normal equation MSE and accuracy

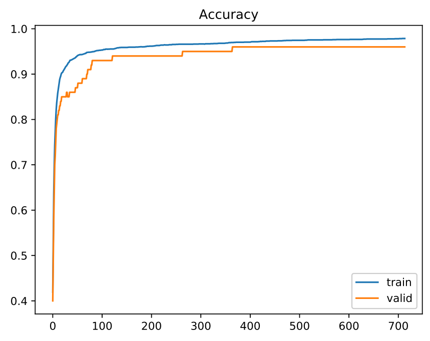
	MSE	Accuracy
Training	0.01876926469613493	99.343%
Validation	0.04085485030466457	97.000%
Test	0.05189571693650368	95.862%



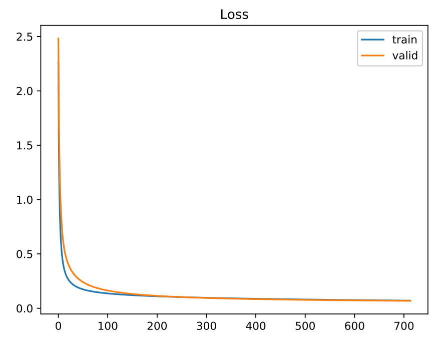
(a) Run 1 accuracy



(b) Run 1 loss



(c) Run 2 accuracy



(d) Run 2 loss

Figure 2: Training and validation accuracy and loss for two runs of $\eta = 0.008$ using logistic regression

SGD is more practical than using normal equation when a complex loss function is used and it is hard to compute the analytical form of optimized solution, or when such an analytical form do not exist. When this happens, normal equation becomes impractical. SGD only require a way to calculate the gradient of loss function, which is more versatile.

2 Logistic Regression - Binary cross-entropy loss

2.1 Learning

Figure 2 is the result for learning rate of 0.008. Both loss and accuracy are from the the entire training or validation set, evaluated at end of an epoch. Although the results fluctuate between each run, $\eta = 0.008$ have better chance of making better results than others. In the first run in figure 2, it gives 98% accuracy in validation set and 97.931% in test set, and in second run, it gives 96% in validation set but 98.621% in test set.

2.2 Beyond plain SGD

Figure 3 is the training cross entropy plot for Adam and GD Optimizer, trained in lock step with same training batches. Cross entropy losses are from entire training set, evaluated at

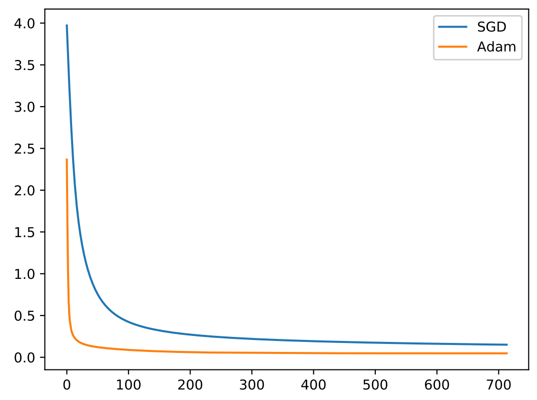


Figure 3: Training cross entropy plot for Adam and GD optimizer

Table 4: Accuracy comparison between linear regression using normal equation and optimal logistic regression

	training	validation	test
linear	3477 (99.343%)	97 (97%)	139 (95.862%)
logistic	3499 (99.971%)	99 (99%)	142 (97.931%)
total(100%)	3500	100	145

end of epoch.

Adam Optimizer use gradients to estimate first and second order momentum, and use them for training variable updates. The learning rate is also boosted at beginning of training, and decays to normal learning rate over iterations. These properties make training using Adam Optimizer to converge faster than normal SGD in general, especially when there are a lot of parameters, like the case of *notMNIST* dataset.

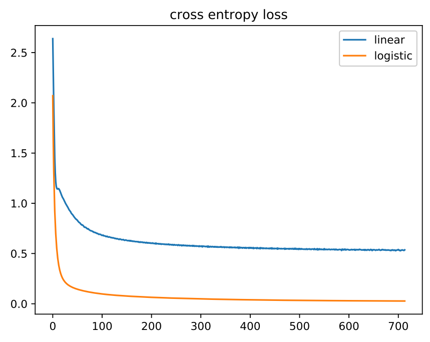
2.3 Comparison with linear regression

2.3.1 Comparing with normal equation

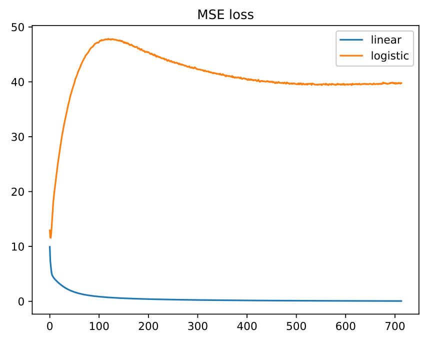
Table 4 lists accuracy results over all data sets for linear regression using normal equation and logistic regression using $\eta = 0.008$, mini-batch size of 500, and 5000 iterations, both with $\lambda = 0$. In this tests logistic regression out-performs linear regression significantly, obtaining very high accuracy over all datasets.

2.3.2 Comparing with linear regression on GD

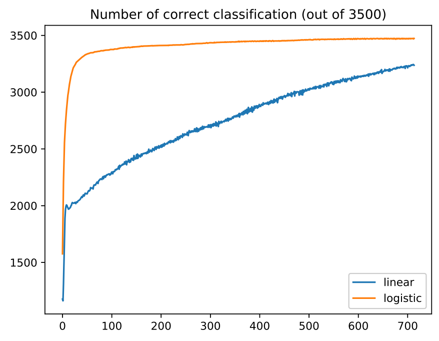
Figure 4 is the comparison of results from linear and logistic regression, trained in lock step with identical training batches. All data are from entire datasets.



(a) cross entropy

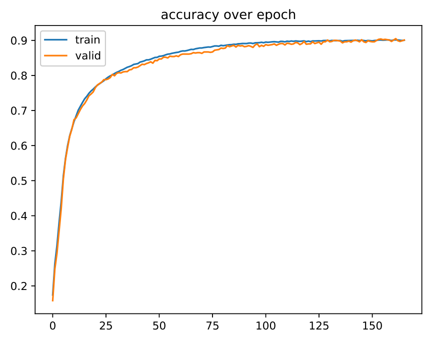


(b) MSE

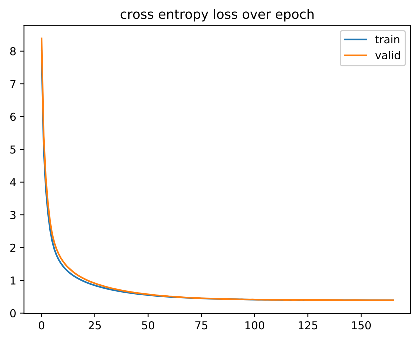


(c) Accuracy

Figure 4: Comparison of training losses and accuracy on linear and logistic regression



(a) accuracy



(b) cross entropy loss

Figure 5: Accuracy and cross entropy loss plot over *notMNIST* dataset

Cross entropy loss makes the convergence faster than that with MSE. It penalizes more when misclassification happens, and minimizing this loss makes correcting misclassified targets faster for classification tasks.

3 Logistic Regression - Multi-class classification

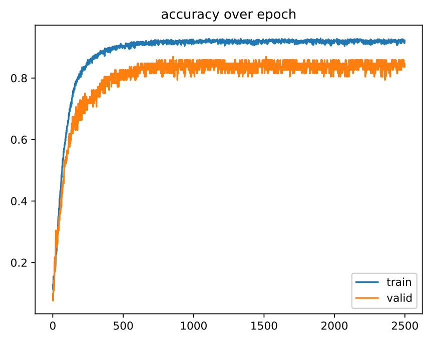
3.1 notMNIST dataset

Figure 5 is the plot of accuracy and cross entropy loss on training and validation set from *notMNIST*, using learning rate $\eta = 0.0005$. Values are using the entire set of data at the end of each epoch. The best test set accuracy achieved is 2432/2724, which gives 89.280% accuracy. This is much lower than the accuracy obtained in binary classification.

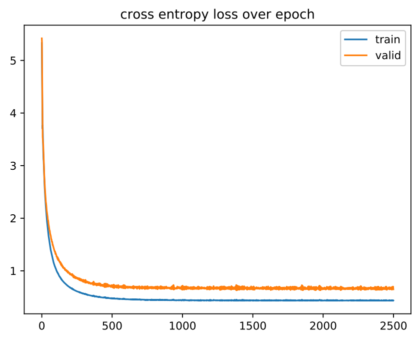
Intuitively speaking, this is most likely due to the fact that this 10-class problem requires the regression to choose the correct answer from 10 choices instead of 2. There are now 10 sets of weights, and each of them can go wrong to mis-classify a data point. Also, in the old problem, a set of weight only needs to correctly produce low probability for one class of data (the opposite class), but in this case, one set of weight needs to also produce low probability for remaining 8 classes of data. These challenge makes multi-class classification inherently more difficult.

3.2 FaceScrub dataset

Figure 6 is the plot of accuracy and cross entropy loss on training and validation set from *FaceScrub* dataset, using $\eta = 0.002$ and $\lambda = 0.01$. Values are using the entire set of data at the end of each epoch. The best test set accuracy achieved is 82/93, which gives 88.172% accuracy. This is comparable but slightly lower than the result from k-NN. Single level logistic regression is not powerful enough for face recognition.



(a) accuracy



(b) cross entropy loss

Figure 6: Accuracy and cross entropy loss plot over *FaceScrub* dataset