Assignment 3: Accelerated Methods for Stochastic Gradient Descent

Nima Leclerc (nl475), Ryan Gale (rjg295)
Cornell University
Principles of Large Scale Machine Learning Systems (CS 4787)

March 10, 2019

1 Part I

In this work, multinomial logistic regression is implemented to perform multi-class classification on images from the MNIST data set. Several optimization routines were used during training to perform the optimization of the cross-entropy loss function with softmax logistic regression. Minibatch Stochastic Gradient Descent (SGD) with sequential sampling, Gradient Descent (GD), GD with Nersterov's momentum, SGD with momentum, and Adam were implemented during training with variation of hyperparmeters within each. Hyperparameters considered here were the l_2 regularization parameter γ, the momentum parameter β, and learning rate α for both GD with and without Nersterov's momentum. Hence, these two algorithms were trained with $\gamma = 0.0001$ and $\alpha = 1.0$ and for $\beta = 0.9, 0.99$. Each were trained with 100 epochs/iteration and the training/test errors and training loss were computed over all iterations. The computed training/test errors and training losses are plotted over the 100 epochs for these algorithms in Figure 1, 2, and 3.

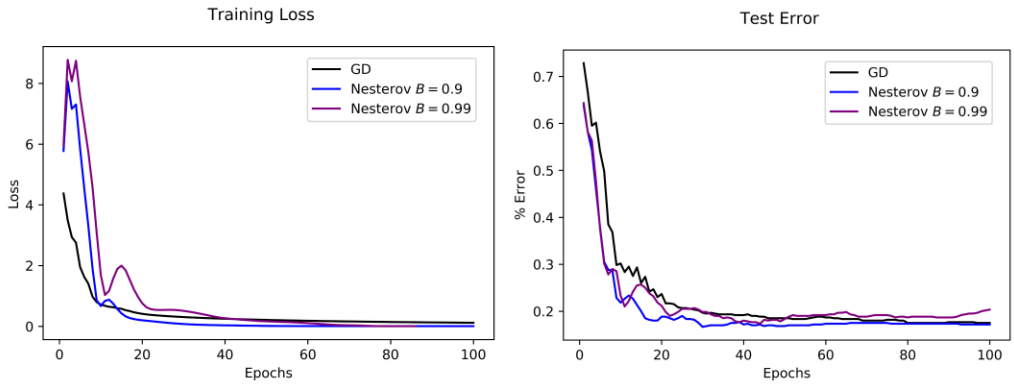
2 Part II

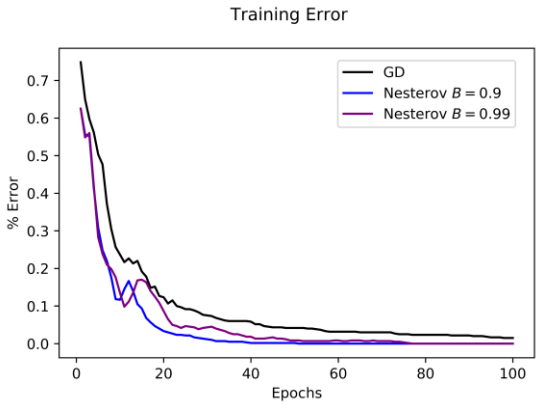
The relative training errors in the attached figure suggest that the inclusion of momentum in GD significantly improves the training performance, in comparison with its normal GD counterpart. In particular, it appears that the training error improves by nearly 10 % with momentum and this also reveals that the algorithm also converges more rapidly with momentum. The training losses for implementations with momentum also demonstrate small oscillations, suggesting that perhaps a decrease in training loss due to momentum addition has been reinforced by an increase in loss in the subsequent step and vice-versa. This behavior is not observed in normal GD. Generally, it appears that the test error for these algorithms tend to be nearly 10% higher than their corresponding training errors which is sensible given that the models are biased towards the data sets they were trained on. Moreover, a lower training and test error is observed for a smaller β which is also sensible given that for larger β, a greater penalty is added to the gradient update step for smaller shift in direction of the training parameter w which gives rise to a greater accumulation of error over iteration number. This is also justified by the fact that for larger β, larger amplitudes are observed in peaks of the training loss which attributes to the fact that greater error accumulation occurs in iteration steps that differ significantly in direction. The runtimes for GD and GD with Nesterov momentum were, $t_{run} = 223.18s, 232.61s$. These results suggest that Nesterov's addition slows the computation marginally which is expected given the extra step needed to compute the prior gradient step. For these two algorithms, the the α and β parameters were varied during training with values $\alpha = 1.5, 2.0$ and $\beta = 0.95, 0.85$. Increasing α shows an increased training error for GD and longer time to achieve convergence. Increasing β shows shorter time to achieve convergence. Both values of β reveal similar behavior in loss function.

3 Part III

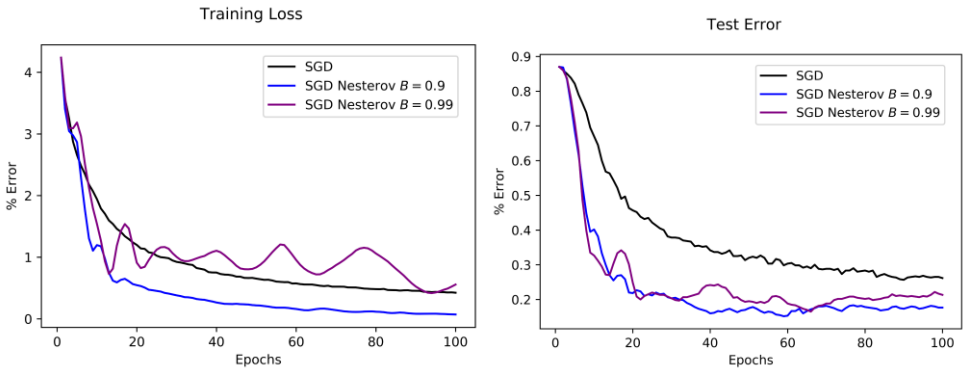
The MNIST data set was also used to train ADAM with sequential sampling. Hyperparameters considered here were the l_2 regularization parameter γ, batch size B, the two moment decay rates $\rho 1$ and $\rho 2$, and learning rate α. The algorithm was trained with $\gamma = 0.0001$ and $\alpha = 0.01$, $B = 600$, $\rho 1 = 0.9$, and $\rho 2 = 0.999$. It was trained with 10 epochs and the training/test errors and training loss were computed over all iterations. The computed training/test errors and training losses are plotted over the 100 epochs for these algorithms in attached figures. The relative training errors suggest that the inclusion of ADAM vs SGD significantly improves the training performance. The training error improves by nearly 10 % with ADAM and this also reveals that the algorithm also converges more rapidly with ADAM. The training loss is a fairly smooth decreasing plot as opposed to the oscillations seen with momentum. The runtime for training this algorithm was $t_{run} = 18.69s$. There is not much of a penalty for overhead on top of the standard SGD runtime. For SGD and ADAM, α, $\rho 1$, and $\rho 2$ were varied during training with values $\alpha = 0.3, 0.25, 0.23$, $\rho 1 = 0.85, 0.95, 0.95$, and $\rho 2 = 0.9, 0.95, 0.9$ for the 1^{st},2^{nd}, and 3^{rd} experiments respectively. Decreasing α from 0.3 shows an increased training error for ADAM. The test errors were all very similar for the 3 trials, only the first one seemed to have slightly more variance in the noise ball for both ADAM and SGD.

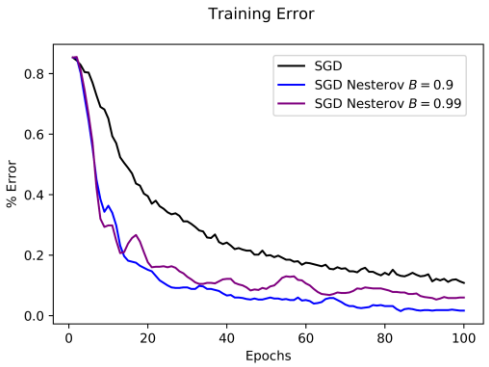
Overall momentum and ADAM improved performance in each of their applications. The slight increase in runtime for performing either optimization is greatly outweighed by the faster convergence and better test error. In conclusion, it is best to consider Nesterov momentum and ADAM, in addition to SGD, when designing large-scale ML applications.



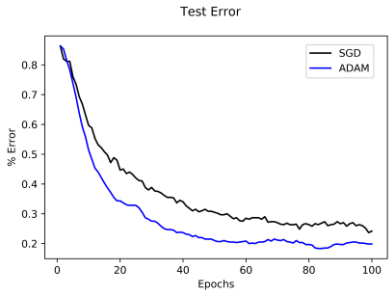


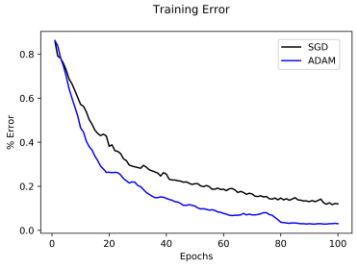
Part I

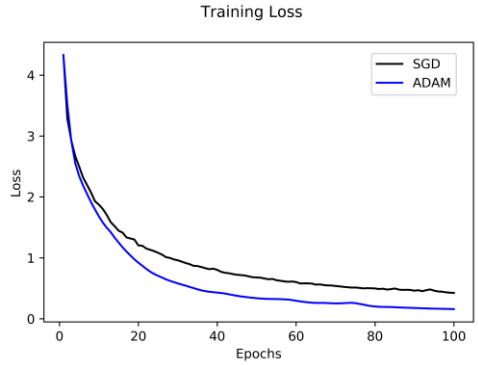




Part II


Test Error
SGD
ADAM
% Error
Epochs
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
20
40
60
80
100


Training Error
SGD
ADAM
% Error
Epochs
0.0
0.2
0.4
0.6
0.8
0
20
40
60
80
100


Training Loss
SGD
ADAM
Loss
Epochs
0
1
2
3
4
0
20
40
60
80
100

Part III