CS 594: Advanced Machine Learning	Due: 13:00, April 12, 2018

Project 2: Convex Optimization

Student: George Maratos, Ashwani Khemani *Email: gmarat2@uic.edu, akhema2@uic.edu*

Question P1

```
iter func.val gap time feval.num train_lett_err train_word_err test_lett_err test_word_err
0 24.5949329 4.55704418 0.521893978 1 92.309174 100.000000 93.091357 100.000000
1 20.2619527 5.09500228 2.1071229 3 71.379031 100.000000 72.095711 100.000000
2 17.4280532 3.65686478 2.90276504 4 66.662814 99.825480 67.009594 99.883687
3 15.3063722 2.66873138 3.70153904 5 53.442762 97.643979 54.005317 97.702821
4 13.2139426 2.13033316 4.48281908 6 45.382037 96.160558 46.075598 95.347485
5 10.8951126 2.51806306 5.28561425 7 39.675567 91.826643 40.808384 92.294272
6 9.33821858 1.17139586 6.07544017 8 33.791855 87.056428 34.978615 87.699913
7 8.76213396 0.87446294 6.86494994 9 31.568605 84.293194 32.632066 85.228264
8 7.90650424 0.849892316 7.6622529 10 28.570878 80.424666 29.638192 81.942425
9 7.31265051 1.54717585 8.45745993 11 27.129812 77.370564 28.385928 78.016865
10 6.80867555 0.727934637 9.26300693 12 24.594459 73.007563 25.892960 74.498401
:
:
95 3.33946396 0.00445979302 76.2979987 98 10.858090 40.517743 14.499287 49.520209
Optimization converged with status CONVERGED_GTTOL.
```

Question P2

See figure 1 for the plot. Yes, using a larger value of λ will allow for faster convergence. As λ increases, the upper bound on the largest eigenvalue will decrease. Smaller eigenvalues will lead to smaller updates over time, which will lead to faster convergence.

Question P3

See figure 2 for the plot. We observe that while test error is higer than train error, the model does a good job of generalizing well for letter wise error. Word error is much higher in general, but this is expected because getting the entire word correct is much harder than getting a single letter. The objective value drops dramaticaly during training for the first 20 seconds. We conclude that a drop in objective value is correlated with a drop in error, because we observe a similar drop in the same time frame.

Question P4

See figure 3 for the plot. The curve is not linear, as can be observed in the plot. Linearity breaks around 5 cores. We cannot expect a perpetual linear speed up from increased levels of parallelization. Consider an arbitrary task that has 100 independent atomic operations. Parallelizing this task will speed up the task, but there will be no benefit from going to 101 cores from 100 because the last core will remain idle and contribute nothing.

Question P5

No we do not need to store C_{train}. We can instead, distribute X_{train} and modify $loss\_coef$ to compute g_{node} directly. This would allow us to forget about C_{train} entirely. The computational performance will not be significantly affected.

Question S1

See figures 4, 5, and 6 for the plots. We observe that when the λ is small, time to convergence increases. We also observe that sgd and adam converge faster than lbfgs, this is because sgd and adam dropped dramatically within the first couple of passes. We would like to mention that when we generated the graphs for this section, we did not set a tolerance but instead killed the optimizers manually for adam and sgd. This is why adam took so long to "converge" for the first two λ's.

For the SGD optimizer, we used basic momentum and a decaying learning rate. We set a momentum parameter $\beta = 0.9$ For λ values of $1e-2, 1e-4, 1e-6$, we used a learning rate of $1e-2$ for all. The decay rate was 0.5.

For the adam optimizer, we used the parameters $\beta_1 = 0.9, \beta_2 = 0.999$ for all three levels of λ and also we used a decay rate of 0.5. For λ values of $1e-2, 1e-4, 1e-6$ we used a learning rate of $1e-2, 1e-1, 1$ respectively.

Question S2

We started with the goals of achieving the fastest convergence so we tried to tune for larger learning rates without being so large that training would lead to instant divergence. This strategy worked well, but in the beginning we only tried a constant learning rate which lead to issues as we approached the minimum. We then switched to using a decaying learning rate and basic momentum for sgd, which worked better.

Question S3

See figure 7 for the plot. We observe adam and sgd converged faster and they achieved much lower errors with fewer number of passes. There is a strong correlation between a drop in objective function value and word error.

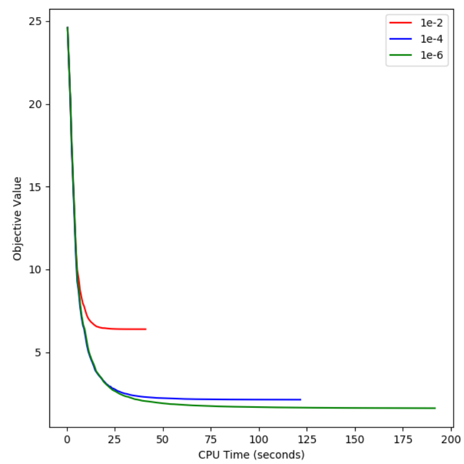

Figure 1. Determining the effect of lambda on Convergence Time

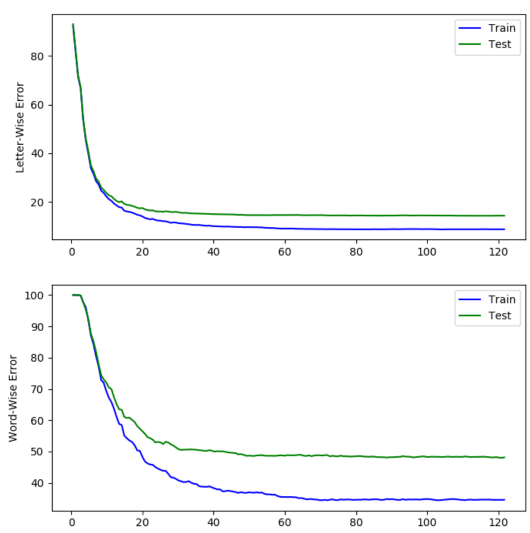

Figure 2. Plot of Letter and Word Error

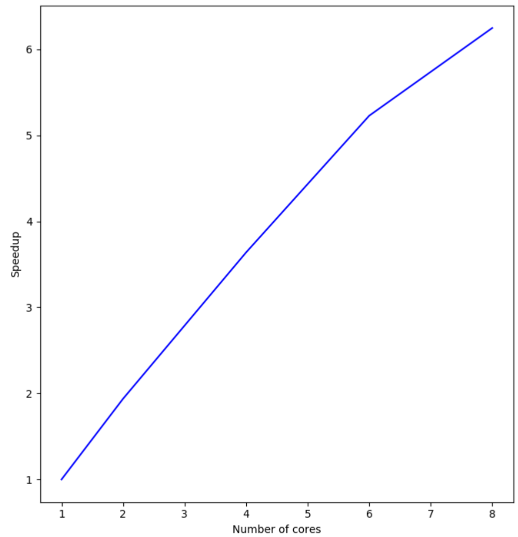

Figure 3. Plot of Letter and Word Error

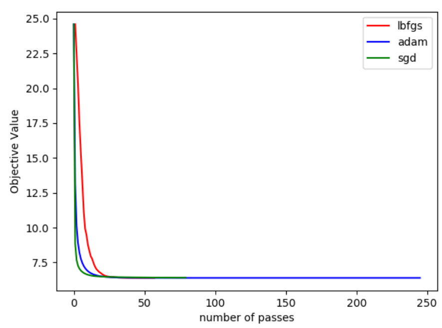

Figure 4. Objective function vs Effective Number of Passes (lambda 1e-2)

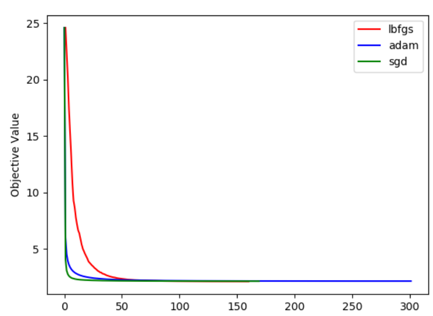

Figure 5. Objective function vs Effective Number of Passes (lambda 1e-4)

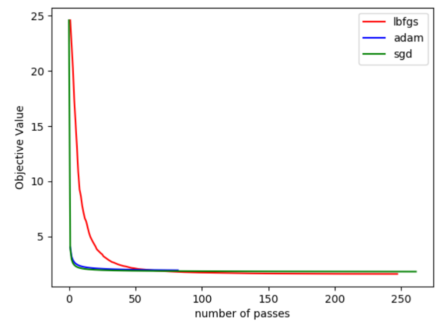

Figure 6. Objective function vs Effective Number of Passes (lambda 1e-6)

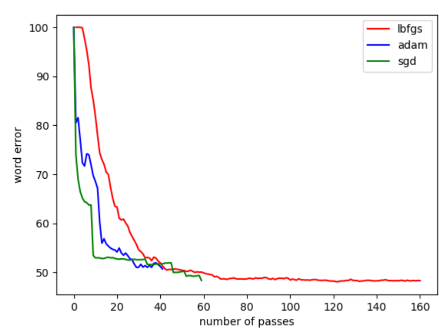

Figure 7. Objective function vs Effective Number of Passes (lambda 1e-6)