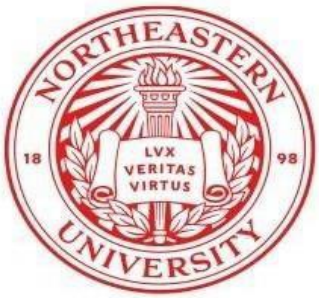


Northeastern University

CSYE 7374 - PARALLEL MACHINE LEARNING AND AI

SUMMER 2020

HOMEWORK 3

SRISHTI ASHOK MISHRA

001305178

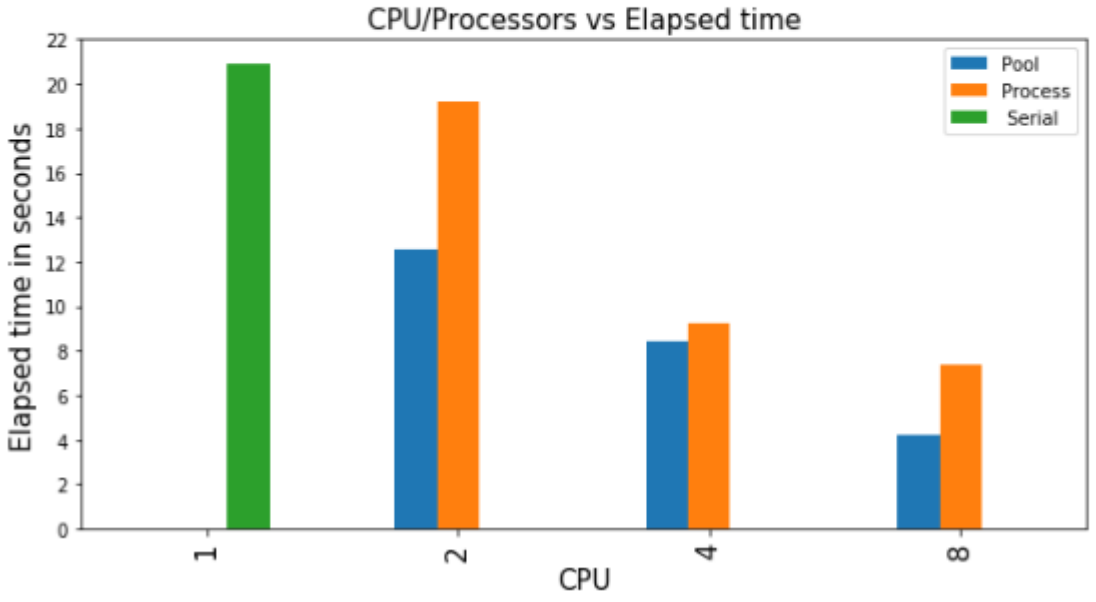
Part 1:

1. Parallel tuning C parameter in SVM:

```
[mishra.sr@d0080 homework3]$ python pool.py
Best tuning param 0.002154.
Serial runs in 20.929 seconds.
Best tuning param 0.002154.
2 CPU Pool runs in 12.571 seconds.
Best tuning param 0.002154.
4 CPU Pool runs in 8.407 seconds.
Best tuning param 0.002154.
8 CPU Pool runs in 4.224 seconds.
```

```
[mishra.sr@login-01 ~]$ srun -p short -N 1 -n 1 -c 4 --cpus-per-task 4 --pty --export=ALL --mem=10Gb --time=08:00:00 /bin/bash
srun: job 12559727 queued and waiting for resources
srun: job 12559727 has been allocated resources
[mishra.sr@c0187 ~]$ module load python/3.8.1
[mishra.sr@c0187 ~]$ cd csye7374-mishra.sr/
[mishra.sr@c0187 csye7374-mishra.sr]$ cd homework3
[mishra.sr@c0187 homework3]$ ls
code1.py  code.py    kfold-serial.py  newcode.py     newQ2.py       part2.py  process.py    simple.py       train.csv
code2.py  Hw3-Q2.py  newcode1.py      newprocess.py  optdigits.txt  pool.py   Q1process.py  svm_process.py  trial.py
[mishra.sr@c0187 homework3]$ python svm_process.py
Process runs in 9.262820959 seconds.
```

```
1  data.plot('CPU',['Pool','Process',' Serial'], kind="bar", figsize=(10,5))
2  plt.xlabel('CPU', fontsize=15)
3  plt.ylabel('Elapsed time in seconds', fontsize=15)
4  plt.title("CPU/Processors vs Elapsed time", fontsize=15)
5  plt.xticks(fontsize=15)
6  plt.yticks(np.arange(0, 23, 2),fontsize=10)
7  plt.show()
8  plt.tight_layout()
9
```



2. Tuning Learning Rate and the Number of Trees in XGBoost:

Thread 8:

```
[mishra.sr@d0004 homework3]$ python newQ2.py
Number of Threads: 8
Best: -0.001063 using {'learning_rate': 0.1, 'n_estimators': 500, 'subsample': 1.0}
-1.584778 (0.000005) with: {'learning_rate': 0.0001, 'n_estimators': 100, 'subsample': 1.0}
-1.560750 (0.000009) with: {'learning_rate': 0.0001, 'n_estimators': 200, 'subsample': 1.0}
-1.537325 (0.000014) with: {'learning_rate': 0.0001, 'n_estimators': 300, 'subsample': 1.0}
-1.514476 (0.000018) with: {'learning_rate': 0.0001, 'n_estimators': 400, 'subsample': 1.0}
-1.492179 (0.000023) with: {'learning_rate': 0.0001, 'n_estimators': 500, 'subsample': 1.0}
-1.388035 (0.000044) with: {'learning_rate': 0.001, 'n_estimators': 100, 'subsample': 1.0}
-1.210626 (0.000081) with: {'learning_rate': 0.001, 'n_estimators': 200, 'subsample': 1.0}
-1.064064 (0.000114) with: {'learning_rate': 0.001, 'n_estimators': 300, 'subsample': 1.0}
-0.940464 (0.000144) with: {'learning_rate': 0.001, 'n_estimators': 400, 'subsample': 1.0}
-0.834693 (0.000171) with: {'learning_rate': 0.001, 'n_estimators': 500, 'subsample': 1.0}
-0.475048 (0.000288) with: {'learning_rate': 0.01, 'n_estimators': 100, 'subsample': 1.0}
-0.167306 (0.000472) with: {'learning_rate': 0.01, 'n_estimators': 200, 'subsample': 1.0}
-0.061058 (0.000635) with: {'learning_rate': 0.01, 'n_estimators': 300, 'subsample': 1.0}
-0.022792 (0.000791) with: {'learning_rate': 0.01, 'n_estimators': 400, 'subsample': 1.0}
-0.008871 (0.000943) with: {'learning_rate': 0.01, 'n_estimators': 500, 'subsample': 1.0}
-0.001104 (0.001479) with: {'learning_rate': 0.1, 'n_estimators': 100, 'subsample': 1.0}
-0.001069 (0.001494) with: {'learning_rate': 0.1, 'n_estimators': 200, 'subsample': 1.0}
-0.001066 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 300, 'subsample': 1.0}
-0.001064 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 400, 'subsample': 1.0}
-0.001063 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 500, 'subsample': 1.0}
Elapsed Duration: 389.5449516773224
```

Thread 6:

```
Number of Threads: 6
Best: -0.001063 using {'learning_rate': 0.1, 'n_estimators': 500, 'subsample': 1.0}
-1.584778 (0.000005) with: {'learning_rate': 0.0001, 'n_estimators': 100, 'subsample': 1.0}
-1.560750 (0.000009) with: {'learning_rate': 0.0001, 'n_estimators': 200, 'subsample': 1.0}
-1.537325 (0.000014) with: {'learning_rate': 0.0001, 'n_estimators': 300, 'subsample': 1.0}
-1.514476 (0.000018) with: {'learning_rate': 0.0001, 'n_estimators': 400, 'subsample': 1.0}
-1.492179 (0.000023) with: {'learning_rate': 0.0001, 'n_estimators': 500, 'subsample': 1.0}
-1.388035 (0.000044) with: {'learning_rate': 0.001, 'n_estimators': 100, 'subsample': 1.0}
-1.210626 (0.000081) with: {'learning_rate': 0.001, 'n_estimators': 200, 'subsample': 1.0}
-1.064064 (0.000114) with: {'learning_rate': 0.001, 'n_estimators': 300, 'subsample': 1.0}
-0.940464 (0.000144) with: {'learning_rate': 0.001, 'n_estimators': 400, 'subsample': 1.0}
-0.834693 (0.000171) with: {'learning_rate': 0.001, 'n_estimators': 500, 'subsample': 1.0}
-0.475048 (0.000288) with: {'learning_rate': 0.01, 'n_estimators': 100, 'subsample': 1.0}
-0.167306 (0.000472) with: {'learning_rate': 0.01, 'n_estimators': 200, 'subsample': 1.0}
-0.061058 (0.000635) with: {'learning_rate': 0.01, 'n_estimators': 300, 'subsample': 1.0}
-0.022792 (0.000791) with: {'learning_rate': 0.01, 'n_estimators': 400, 'subsample': 1.0}
-0.008871 (0.000943) with: {'learning_rate': 0.01, 'n_estimators': 500, 'subsample': 1.0}
-0.001104 (0.001479) with: {'learning_rate': 0.1, 'n_estimators': 100, 'subsample': 1.0}
-0.001069 (0.001494) with: {'learning_rate': 0.1, 'n_estimators': 200, 'subsample': 1.0}
-0.001066 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 300, 'subsample': 1.0}
-0.001064 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 400, 'subsample': 1.0}
-0.001063 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 500, 'subsample': 1.0}
Elapsed Duration: 413.5086364746094
```

Thread 4:

```
Number of Threads: 4
Best: -0.001063 using {'learning_rate': 0.1, 'n_estimators': 500, 'subsample': 1.0}
-1.584778 (0.000005) with: {'learning_rate': 0.0001, 'n_estimators': 100, 'subsample': 1.0}
-1.560750 (0.000009) with: {'learning_rate': 0.0001, 'n_estimators': 200, 'subsample': 1.0}
-1.537325 (0.000014) with: {'learning_rate': 0.0001, 'n_estimators': 300, 'subsample': 1.0}
-1.514476 (0.000018) with: {'learning_rate': 0.0001, 'n_estimators': 400, 'subsample': 1.0}
-1.492179 (0.000023) with: {'learning_rate': 0.0001, 'n_estimators': 500, 'subsample': 1.0}
-1.388035 (0.000044) with: {'learning_rate': 0.001, 'n_estimators': 100, 'subsample': 1.0}
-1.210626 (0.000081) with: {'learning_rate': 0.001, 'n_estimators': 200, 'subsample': 1.0}
-1.064064 (0.000114) with: {'learning_rate': 0.001, 'n_estimators': 300, 'subsample': 1.0}
-0.940464 (0.000144) with: {'learning_rate': 0.001, 'n_estimators': 400, 'subsample': 1.0}
-0.834693 (0.000171) with: {'learning_rate': 0.001, 'n_estimators': 500, 'subsample': 1.0}
-0.475048 (0.000288) with: {'learning_rate': 0.01, 'n_estimators': 100, 'subsample': 1.0}
-0.167306 (0.000472) with: {'learning_rate': 0.01, 'n_estimators': 200, 'subsample': 1.0}
-0.061058 (0.000635) with: {'learning_rate': 0.01, 'n_estimators': 300, 'subsample': 1.0}
-0.022792 (0.000791) with: {'learning_rate': 0.01, 'n_estimators': 400, 'subsample': 1.0}
-0.008871 (0.000943) with: {'learning_rate': 0.01, 'n_estimators': 500, 'subsample': 1.0}
-0.001104 (0.001479) with: {'learning_rate': 0.1, 'n_estimators': 100, 'subsample': 1.0}
-0.001069 (0.001494) with: {'learning_rate': 0.1, 'n_estimators': 200, 'subsample': 1.0}
-0.001066 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 300, 'subsample': 1.0}
-0.001064 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 400, 'subsample': 1.0}
-0.001063 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 500, 'subsample': 1.0}
Elapsed Duration: 423.41237330436707
```

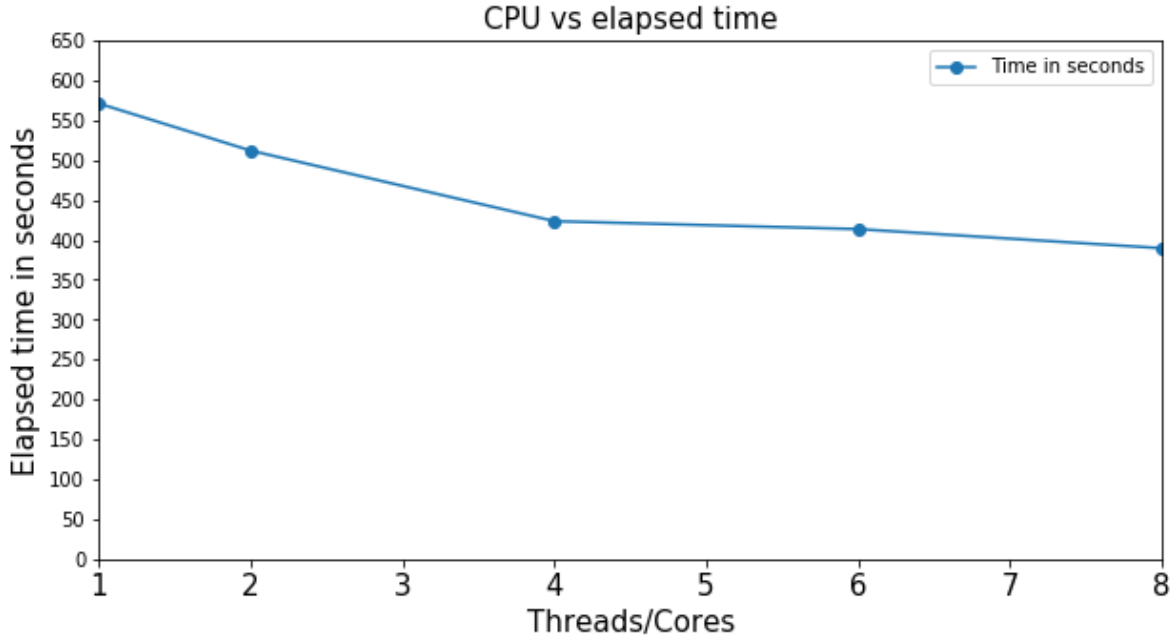
Thread 2:

```
Number of Threads: 2
Best: -0.001063 using {'learning_rate': 0.1, 'n_estimators': 500, 'subsample': 1.0}
-1.584778 (0.000005) with: {'learning_rate': 0.0001, 'n_estimators': 100, 'subsample': 1.0}
-1.560750 (0.000009) with: {'learning_rate': 0.0001, 'n_estimators': 200, 'subsample': 1.0}
-1.537325 (0.000014) with: {'learning_rate': 0.0001, 'n_estimators': 300, 'subsample': 1.0}
-1.514476 (0.000018) with: {'learning_rate': 0.0001, 'n_estimators': 400, 'subsample': 1.0}
-1.492179 (0.000023) with: {'learning_rate': 0.0001, 'n_estimators': 500, 'subsample': 1.0}
-1.388035 (0.000044) with: {'learning_rate': 0.001, 'n_estimators': 100, 'subsample': 1.0}
-1.210626 (0.000081) with: {'learning_rate': 0.001, 'n_estimators': 200, 'subsample': 1.0}
-1.064064 (0.000114) with: {'learning_rate': 0.001, 'n_estimators': 300, 'subsample': 1.0}
-0.940464 (0.000144) with: {'learning_rate': 0.001, 'n_estimators': 400, 'subsample': 1.0}
-0.834693 (0.000171) with: {'learning_rate': 0.001, 'n_estimators': 500, 'subsample': 1.0}
-0.475048 (0.000288) with: {'learning_rate': 0.01, 'n_estimators': 100, 'subsample': 1.0}
-0.167306 (0.000472) with: {'learning_rate': 0.01, 'n_estimators': 200, 'subsample': 1.0}
-0.061058 (0.000635) with: {'learning_rate': 0.01, 'n_estimators': 300, 'subsample': 1.0}
-0.022792 (0.000791) with: {'learning_rate': 0.01, 'n_estimators': 400, 'subsample': 1.0}
-0.008871 (0.000943) with: {'learning_rate': 0.01, 'n_estimators': 500, 'subsample': 1.0}
-0.001104 (0.001479) with: {'learning_rate': 0.1, 'n_estimators': 100, 'subsample': 1.0}
-0.001069 (0.001494) with: {'learning_rate': 0.1, 'n_estimators': 200, 'subsample': 1.0}
-0.001066 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 300, 'subsample': 1.0}
-0.001064 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 400, 'subsample': 1.0}
-0.001063 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 500, 'subsample': 1.0}
Elapsed Duration: 511.87480115890503
```

Thread 1

```
Number of Threads: 1
Best: -0.001063 using {'learning_rate': 0.1, 'n_estimators': 500, 'subsample': 1.0}
-1.584778 (0.000005) with: {'learning_rate': 0.0001, 'n_estimators': 100, 'subsample': 1.0}
-1.560750 (0.000009) with: {'learning_rate': 0.0001, 'n_estimators': 200, 'subsample': 1.0}
-1.537325 (0.000014) with: {'learning_rate': 0.0001, 'n_estimators': 300, 'subsample': 1.0}
-1.514476 (0.000018) with: {'learning_rate': 0.0001, 'n_estimators': 400, 'subsample': 1.0}
-1.492179 (0.000023) with: {'learning_rate': 0.0001, 'n_estimators': 500, 'subsample': 1.0}
-1.388035 (0.000044) with: {'learning_rate': 0.001, 'n_estimators': 100, 'subsample': 1.0}
-1.210626 (0.000081) with: {'learning_rate': 0.001, 'n_estimators': 200, 'subsample': 1.0}
-1.064064 (0.000114) with: {'learning_rate': 0.001, 'n_estimators': 300, 'subsample': 1.0}
-0.940464 (0.000144) with: {'learning_rate': 0.001, 'n_estimators': 400, 'subsample': 1.0}
-0.834693 (0.000171) with: {'learning_rate': 0.001, 'n_estimators': 500, 'subsample': 1.0}
-0.475048 (0.000288) with: {'learning_rate': 0.01, 'n_estimators': 100, 'subsample': 1.0}
-0.167306 (0.000472) with: {'learning_rate': 0.01, 'n_estimators': 200, 'subsample': 1.0}
-0.061058 (0.000635) with: {'learning_rate': 0.01, 'n_estimators': 300, 'subsample': 1.0}
-0.022792 (0.000791) with: {'learning_rate': 0.01, 'n_estimators': 400, 'subsample': 1.0}
-0.008871 (0.000943) with: {'learning_rate': 0.01, 'n_estimators': 500, 'subsample': 1.0}
-0.001104 (0.001479) with: {'learning_rate': 0.1, 'n_estimators': 100, 'subsample': 1.0}
-0.001069 (0.001494) with: {'learning_rate': 0.1, 'n_estimators': 200, 'subsample': 1.0}
-0.001066 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 300, 'subsample': 1.0}
-0.001064 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 400, 'subsample': 1.0}
-0.001063 (0.001495) with: {'learning_rate': 0.1, 'n_estimators': 500, 'subsample': 1.0}
Elapsed Duration: 570.969288110733
```

```
data.plot('Threads',['Time in seconds'], kind="line", figsize=(10,5), marker ='o')
plt.xlabel('Threads/Cores', fontsize=15)
plt.ylabel('Elapsed time in seconds', fontsize=15)
plt.title("CPU vs elapsed time", fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(np.arange(0, 700, 50),fontsize=10)
plt.show()
plt.tight_layout()
```




```
import matplotlib.pyplot as plt
data.plot('Trees',['0.1','0.01','0.001','0.0001'], kind="line", figsize=(10,5), marker='o',markerfacecolor='black')
plt.xlabel('Trees', fontsize=15)
plt.ylabel('log loss in seconds', fontsize=15)
plt.title("Log Loss vs Trees", fontsize=15)
plt.xlim([0, 600])
plt.ylim([-2, 0.5])
plt.yticks(np.arange(-2, 0.3, 0.25))
plt.xticks(fontsize=15)
plt.show()
plt.tight_layout()
```