Please submit a single document for this assignment on Canvas by the beginning of class on **Friday, Oct. 14th**.

Implement mini-batch gradient descent with momentum:

- Download or cut-and-paste `ols.py` from Project 2 in the DL@DU Github repo to your computer. Please make sure to also get the data for Project 2 from here (be sure to click Raw).
- The program `ols.py` uses mini-batch gradient descent to train a linear model to best fit (in the least-squares sense) the housing data.

  As students noted in class, the mini-batches are chosen *with replacement*, which might not be what you want. Feel free to modify the mini-batching scheme in `ols.py`. You may wish to have a look here: excursion into sampling theory.
- Next, implement momentum in your `ols.py`. You may refer to Simmons' solution code for Project 1 (which is called `proj1_ols.py` and is posted on the homepage of our Canvas site for this course).
- Recall that, for linear models, the optimal learning rate and momentum can be computed. In the screenshot below, the optimal learning parameters were used to train a linear model on the housing data. The well-trained linear model predicts about 86 percent of the variation in the data (shown in the graph as `validation`).
- Please make sure that your final code for this assignment successfully trains a linear model with performance similar to that pictured below. Then upload a screenshot of the output of your code training with those optimal parameters.

```
centering
normalizing
total number of examples: 2264
batchsize:2264 learning rate:0.000355 momentum:0.899
training on cpu:0 (data is on cpu:0); validating on cpu:0
epoch 1/1000; loss 0.575534
epoch 2/1000; loss 0.573994
epoch 3/1000; loss 0.571084
epoch 4/1000; loss 0.566969
epoch 5/1000; loss 0.561809
epoch 6/1000; loss 0.555753
epoch 7/1000; loss 0.54894
...
epoch 993/1000; loss 0.138017
epoch 994/1000; loss 0.138015
epoch 995/1000; loss 0.138014
epoch 996/1000; loss 0.138012
epoch 997/1000; loss 0.13801
epoch 998/1000; loss 0.138009
epoch 999/1000; loss 0.138007
epoch 1000/1000; loss 0.138005
trained in 2 min 18.3 sec
```