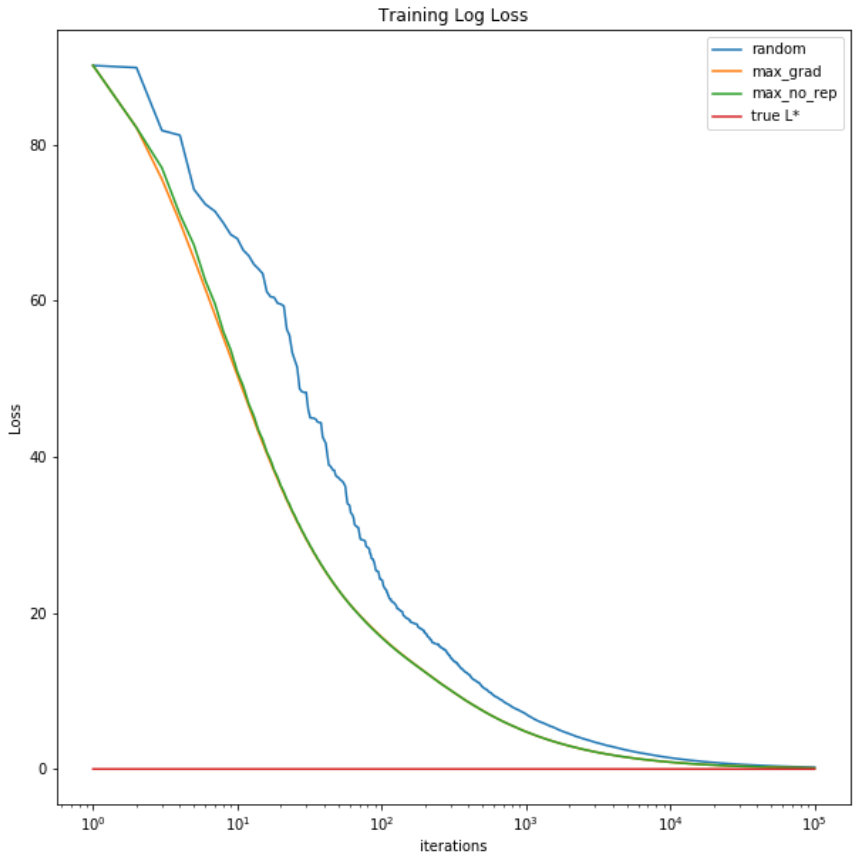
Overview:

The approach I look to use for in my coordinate descent method is what I call “strongest gradient wins”. In my implementation I initialize the weights to zeros and then calculate the gradient of the loss function. Then my algorithm compares the magnitudes of the gradients and selects the element of w that has the largest magnitude. Additionally, my algorithm updates b independently of w, allowing it to adjust to the loss function as the weights are updated. A permutation of this method, called “strongest without repetition” follows a similar approach, but requires that no element of w can be updated two iterations in a row. This method can work with any cost function that is differentiable. The bias term can also be omitted if desired.

Experimental Results:

The algorithm was run over 10,000 iterations with weights and biases initialized to 0 and a learning rate of .02 that was held constant over the duration of the experiment.



It is clear that selecting the dimension of maximum gradient value improves the convergence time over a random selection of dimension. Restricting the options of dimension so that the algorithm does not

step along the same dimension two iterations yielded slight improvements early in the training phase, but made little difference over the total training process. It should be noted that both strongest gradient methods yielded a relatively sparse matrix that had 5 out of the 13 weight elements equal to zero indicating that the nonzero elements held stronger correlation.

Convergence:

This method converges under similar conditions to standard gradient descent. At initialization, a local minimum must be avoided or the learning rate must be high enough to escape the local minimum at initialization. Normally, this is not an issue with gradient descent.

Room for improvement:

There are a few ways that this program can improve. First the learning rate can be modified as we go through epochs of data. Additionally, we can introduce momentum into the gradient update. Instead of simply going in the direction of largest magnitude, we can keep a small chunk of our update for the dimension that we travelled on during the previous update.