Princess Tara Zamani
Adv Opt for Machine Learning
Homework 4

Simulation 1

The objective of this simulation was to apply Frank-Wolfe to solve the leading eigenvector of a matrix.

The leading eigenvector of a matrix Q>0 is given by the constraint problem

$$min_{||x||_2 \leq 1} f(x) \ = - x^T Q x$$

An *n x n* random matrix with all entries given by *N(0,1)* was generated for matrix A.
A matrix Q was generated using $Q = A^T A + \varepsilon I$, where $\varepsilon = 0.75$.

The following algorithm was done iteratively to obtain the leading eigenvalue.

$$x_{k+1} = (1 - \eta) x_k + \frac{\eta\, Q\, x_k}{||Q\, x_k||_2}$$

A python built-in function was used to compute the largest eigenvalue and was used to compare against the algorithm's results to verify findings.

The following results were found using the these conditions: η= 1.75e-2, a randomized x_0 vector, and 3000 iterations.

```
Python Leading Eig Vector
 [[-5.97168691]
 [23.99733709]
 [ 6.09843162]
 [ 9.36544919]
 [ 7.20608523]
 [20.41032491]
 [ 0.561812  ]
 [-4.16843667]
 [ 3.00190146]
 [-1.45504237]]
```

```
Frank-Wolfe Leading Eig Vector
[[-5.97168691]
[23.99733709]
[ 6.09843162]
[ 9.36544919]
[ 7.20608523]
[20.41032491]
[ 0.561812  ]
[-4.16843667]
[ 3.00190146]
[-1.45504237]]
```

```
Direct Comparison of Equality
 [[ True]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]]
```

Both the python and Frank-Wolfe method returned the same results, verifying that the Frank-Wolfe algorithm was properly implemented.

Simulation 2

The objective of this simulation was to apply projected gradient descent to solve the constrained least-mean-square problem defined as:

$$min_{||x||_\infty \leq 1} f(x) = \frac{1}{2}\|Ax - b\|_2^2$$, where x $\in$ R^n , A $\in$ $R^{m\times n}$, b $\in$ R^m.

The projection operator is as follows:

$$P_{\{x:||x||_\infty \leq 1\}}(z) := argmin_{\{x:||x||_\infty \leq 1\}} \|x - z\|_2$$

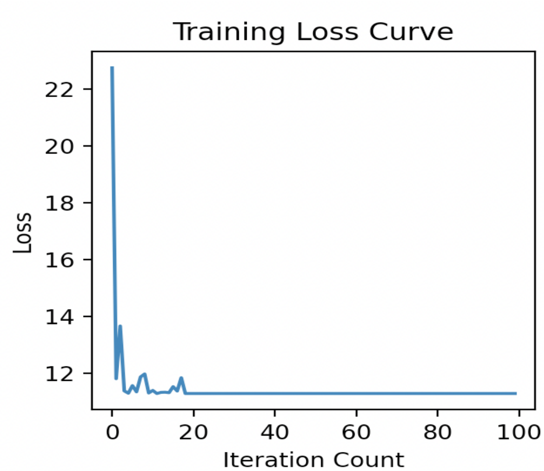
$$\Leftrightarrow \sum_{i=1}^{n} argmin_{\{x_i: |x_i| \leq 1\}} (x_i - z_i)^2$$

The gradient update was applied using

$$x_{k+1} = P_{\{x:||x||_\infty \leq 1\}}(x_k - \eta \nabla f(x_k)).$$

The gradient for the LMS is ∇ f(x) =A^T(Ax−b). The initialization of x_0 was generated randomly using the numpy.random.randn function.

Using n = 100, m = 10, η= 1e-3, and an iteration count of 100, the training loss curve of the implementation is shown below.



The training loss curve follows an expected trajectory. There is a bit of fluctuation in the way that the curve converges, but it does decrease as iterations move forward. I would make the assumption that the fluctuation has to do with the projections onto the constrained set.

In addition, the x_k values were checked on each iteration to verify that they are in the constrained set. Once all iterations were complete, a quick check was done to verify that all values were within the constraint. The terminal result for that evaluation is printed below.

```
All x_k in constraint? :  True
```