CS 260B Homework 3

Hanna Co

Collaborators: Isha Gonugunta

Due: May 11, 2022

1 Problem 1

We can write $v = \sum_i \alpha_i v_i$, where v_i are the columns of V. We can then write Xv as follows:

$$Xv = X\sum_i \alpha_i v_i$$

$$= U\Sigma V^T \sum_i \alpha_i v_i$$

$$= \sum_i \alpha_i v_i u_i \sigma_i v_i^T$$

$\langle v_i^T, v_i \rangle$ is just $\|v_i\|^2$, and since v_i is a unit vector, $\|v_i\|^2 = 1$, giving us

$$Xv = \sum_i \alpha_i u_i \sigma_i$$

We know that $\|Xv\|^2 = \langle Xv, Xv \rangle$,

$$\|Xv\|^2 = \langle \sum_i \alpha_i u_i \sigma_i, \sum_i \alpha_i u_i \sigma_i \rangle$$

$$\|Xv\|^2 = \sum_i (\alpha_i \sigma_i)^2 \langle u_i, u_i \rangle$$

Since u_i are unit vectors, we can simplify further

$$\|Xv\|^2 = \sum_i \alpha_i^2 \sigma_i^2$$

We rewrite this as

$$\|Xv\|^2 = \sigma_1^2\alpha_1^2 + \sigma_2^2\alpha_2^2 + ... + \sigma_n^2\alpha_n^2$$

and since $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$, we can say

$$\|Xv\|^2 \leq \sigma_1^2(\alpha_1^2 + \alpha_2^2 + ... + \alpha_n^2)$$

Additionally, knowing that v is orthonormal, $\sum_i \alpha_i = 1$,

$$\|Xv\|^2 \leq \sigma_1^2$$

Finally giving us

$$\|Xv\| \leq \sigma_1$$

as desired.

2 Problem 2

In lecture, we showed that the span of the first two right singular vectors gives the best-fit subspace for $k = 2$. We now want to generalize this to any k, through induction. We assume this is true for $k = n$, and will prove it for $k = n + 1$.

Knowing that this holds for $k = n$, we know that $S = span\{v_1, v_2, .., v_n\}$ maximizes $Var(S; x)$. That is, $Var(S; x) = \|x \cdot v_1\|^2 + \|x \cdot v_2\|^2 + ... + \|x \cdot v_n\|^2$ and $Var(S; x) \geq Var(S^*; x)$, where $S^* = span\{w_1, w_2, ..., w_n, w_{n+1}\}$, $\{w_1, w_2, ..., w_n, w_{n+1}\}$ some orthonormal basis of S^*.

Now for $k = n + 1$, let's take $S' = span\{v_1, v_2, ..., v_n, v_{n+1}\}$ and $S^{*'} = span\{w_1, w_2, ..., w_n, w_{n+1}\}$.

By definition, $v_{n+1} = argmax_{\|v\|=1} \|x \cdot v\|$
Thus,

$$Var(S'; x) = \|x \cdot v_1\|^2 + \|x \cdot v_2\|^2 + ... + \|x \cdot v_n\|^2 + \|x \cdot v_{n+1}\|^2$$

$$Var(S^{*'}; x) = \|x \cdot w_1\|^2 + \|x \cdot w_2\|^2 + ... + \|x \cdot w_n\|^2 + \|x \cdot w_{n+1}\|^2$$

We can rewrite this as:

$$Var(S'; x) = Var(S; x) + \|x \cdot v_{n+1}\|^2$$

$$Var(S^{*'}; x) = Var(S^*; x) + \|x \cdot w_{n+1}\|^2$$

We know that $Var(S; x) \geq Var(S^*; x)$, and by definition, $\|x \cdot v_{n+1}\| \geq \|x \cdot w_{n+1}\|$, which gives us

$$Var(S'; x) \geq Var(S^{*'}; x)$$

We've proved this for some arbitrary $k = n$ and $k = n + 1$, thus we can generalize this, proving that the span of the first k right singular vectors gives the best-fit subspace.

3 Problem 3

To compute the smallest singular vector with power iteration, we need to perform a shift on Y. We know

$$Y = \sigma_1 \cdot v_1 \cdot v_1^T + \sigma_2 \cdot v_2 \cdot v_2^T + ... + \sigma_n \cdot v_n \cdot v_n^T$$

and

$$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$$

Power iteration will give us v_1, since σ_1 is the largest, so if we want the smallest singular vector (v_n), we need to make σ_n the largest singular value.

We have $Y = V\Sigma^2 V^T$, where Σ is a diagonal matrix with $\sigma_1, \sigma_2, ..., \sigma_n$. We can take Σ and shift it, making each $\sigma_i = \frac{\sigma_i}{\sigma_i - \alpha}$, where $\alpha < \sigma_n$. This shift transforms σ_n into the largest singular value. Power iteration will then find v_n, the smallest singular vector.

4 Problem 4a

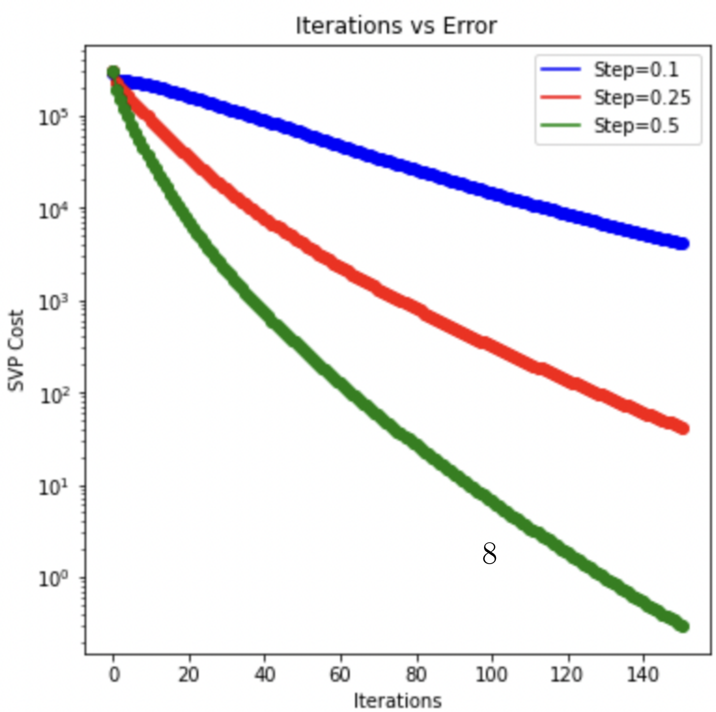
The gradient for the loss with respect for matrix outputs a matrix, where each element is the derivative of L with respect to Y. That is, for every element of O such that $O[i][j] == 1$, the output of $L(Y)$ at element (i,j) is $-2(X_{ij} - Y_{ij})$. In short,

$$\nabla L(Y) = \sum_{(i,j)\in O} -2(X_{ij} - Y_{ij})$$

5 Problem 4b

In [7]:
```
# loss function
def cost(X,Y,O):
    return np.linalg.norm(np.multiply(X-Y, O),'fro')**2

# gradient of loss function
def loss_gradient(X,Y,O):
    return np.multiply(-2*(X-Y),O)

# parameters and generating O and X
n = 1000
d = 500
p = 0.1
R = np.random.rand(n,d)
O = np.zeros((n,d))
O[R < p] = 1
# print(O)

k = 5
U = np.random.normal(0,1,(n,k))
V = np.random.normal(0,1,(d,k))
X = np.matmul(U,np.transpose(V))
# print(X)
```

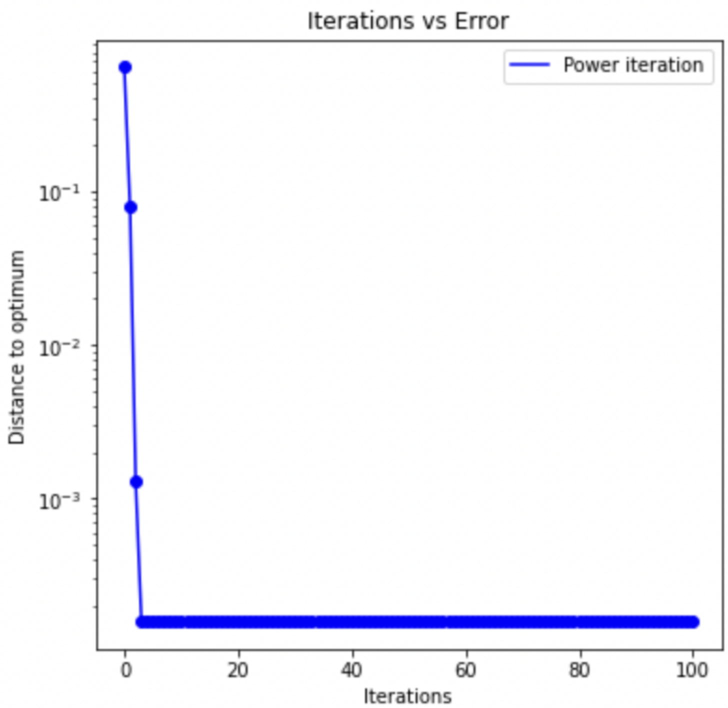
In [8]:
```
# function for singular value projection
# compute and plot the losses for X0, X1,...,XT
# use output of svd function to put together XT
# O is the matrix that tells us what values of Y we know, and Y is our approximation

def svp(step, X, T):
    X_0 = np.random.normal(0,1,(n,d))
    approximations = [X_0]
    error = [cost(X,X_0,O)]
    for t in range(T):
        Y_t = approximations[t]-(step*loss_gradient(X, approximations[t], O))
        u,s,v = scipy.sparse.linalg.svds(Y_t, k = 5)
        X_t = u @ np.diag(s) @ v
        approximations.append(X_t)
        error.append(cost(X,X_t,O))
    return [approximations, error]
```



6 Problem 5a

Iterations	Built-In	Power Iteration
10	1.0004s	0.3957s
20	1.0004s	0.4955s
30	1.0004s	0.5814s
40	1.0004s	0.7139s
50	1.0004s	0.8492s
60	1.0004s	1.0208s
70	1.0004s	1.391s
80	1.0004s	1.5173s
90	1.0004s	1.7037s
100	1.0004s	1.8656s

7 Problem 5b


Iterations vs Error
Power iteration
Distance to optimum
10^{-1}
10^{-2}
10^{-3}
0
20
40
60
80
100
Iterations