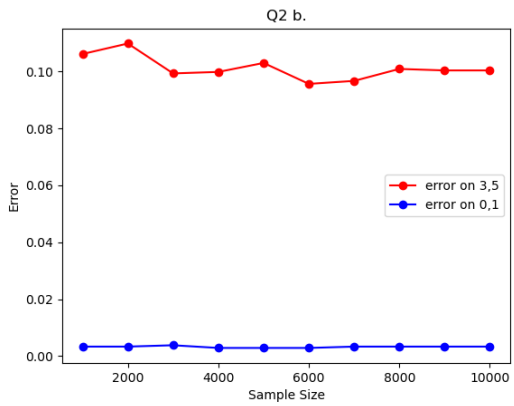
Assignment 3: Introduction to Big Data

Yotam Lifschytz 209579077; Pan Eyal 208722058

1) Added to ZIP file.

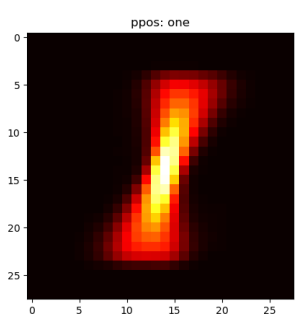
2)

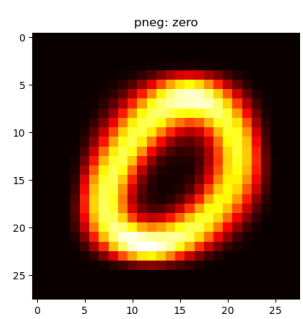
a.



b. As we can see from the plot, the algorithm was able to differentiate between 0 and 1 quite good already from 1000 sample points with error (~0.004).
The error on 3 and 5 is bigger, (~0.1). We can see that it is noisier then the received error graph on 0 and 1.

c.





Every pixel in each heatmaps represents respectively the probability it will lit if the label was one, and if the label was 0. Therefore, the images show the 'average' of all the ones and zeros images. We can notice that the 1 image has high 'heat' on the middle, but on the edge, it gets smudged as the 1-digit might skew, 0-digit image is more stable as it is more symmetric.

d. We received the following result:

```
percent of the 0_1 set that their label changed from -1 to 1: 0.0
percent of the 0_1 set that their label changed from 1 to -1: 0.0
percent of the 3_5 set that their label changed from -1 to 1: 0.008937960042060988
percent of the 3_5 set that their label changed from 1 to -1: 0.0
```

We can explain those results as:

$$pos(x) = \log(allpos) + \sum_{i:x(i)=1.} \log\big(ppos(i)\big) + \sum_{i:x(i)=-1.} \log\big(1 - pneg(i)\big)$$

$$neg(x) = \log(1 - allpos) + \sum_{i:x(i)=1.} \log\big(pneg(i)\big) + \sum_{i:x(i)=-1.} \log\big(1 - ppos(i)\big)$$

$$h_{bayes}(x) = Sign\big(pos(x) - neg(x)\big)$$

$$= Sign\left(\log\left(\frac{allpos}{1 - allpos}\right) + \sum_{i:x(i)=1.} \log\left(\frac{ppos(i)}{pneg(i)}\right) + \sum_{i:x(i)=-1.} \log\left(\frac{1 - pneg(i)}{1 - ppos(i)}\right)\right)$$

Therefore, the only difference of $h_{bayes}(x)$ will occur if the sign will flip. Because only the $allpos$ parameter changes, it means that $\left|\log\left(\frac{allpos}{1-allpos}\right)\right|$ will become larger or smaller then $\left|\sum_{i:x(i)=1.} \log\left(\frac{ppos(i)}{pneg(i)}\right) + \sum_{i:x(i)=-1.} \log\left(\frac{1-pneg(i)}{1-ppos(i)}\right)\right|$.

$allpos$ parameter value as calculated by the code resulted around 0.5.

Thus, $\log_e(\frac{0.5}{1-0.5}) = \log_e(1) = 0$. Now, $\log_e\left(\frac{0.75}{1-0.75}\right) = \log_e(3) \approx 1.0986$

This amount can be neglect compared to:

$$\left|\sum_{i:x(i)=1.} \log\left(\frac{ppos(i)}{pneg(i)}\right) + \sum_{i:x(i)=-1.} \log\left(\frac{1 - pneg(i)}{1 - ppos(i)}\right)\right|$$

As in each sum we will go roughly around half of the $d = 784$ features.

The more 'certain' the algorithm is, the larger of these sums absolute value get.

Therefore, we can see that for the 0-1 set (which achieved very low error) no changes in labels had occurred, while on the less accurate 3-5 results, only a minor change (0.0089%) had occurred.

3) The hypothesis class of homogenous linear predictors is defined as:

$$H_L^d = \{h_w(x) \mid w \in \mathbb{R}^d\}$$

Where we have:

$$h_w(x) = sign(\langle w, x \rangle)$$

We will use the "vector" version of $NN's$ as seen in class, where we assume an ordered V where $o_1 = x(1) \in V, \dots, o_d = x(d) \in V, \dots * hidden\ layers * \dots, o_{|V|} = o_{out} \in V$. (BTW this is all going to be indexed horrendously since we were told in the forum to not use a "layered" version of a NN unless we show equivalence to the "vector" version we saw in class, so sorry in advance).

a. We define a neural network with a d sized input layer where the input is x. We have no "hidden layers", just an output layer of size 1. The output is, by definition:

$$\psi(o_{out}) = sign\left(\sum_{\substack{i=1 \\ i:(i,d)\in E}}^{d} w_{i,out} o_i\right)$$

Let us define a vector w of dimension d where:

$$w(i) = \begin{cases} w_{i,out} & (i, out) \in E \\ 0 & else \end{cases}$$

Thus:

$$\psi(o_{out}) = sign\left(\sum_{i:(i,d)\in E}^{d} w_{i,out} o_i\right) = sign\left(\sum_{i=1}^{d} w(i)x(i)\right) = sign(\langle w, x \rangle)$$

We denote the hypothesis class this NN represents as H_{NN}^*.

So, as seen above:

$$H_{NN}^* = \{sign(\langle w, x \rangle) \mid w \in \mathbb{R}^d\} = \{h_w(x) \mid w \in \mathbb{R}^d\} = H_L^d.$$

b. Say we have $d = 3$, and 1 hidden layer of size 3. Thus:

$$\psi(o_{out}) = sign(o_{out}) = sign\left(\sum_{\substack{i=d+1 \\ i:(i,out)\in E}}^{d+1+L} w_{i,out} o_i\right) =$$

$$sign\left(\sum_{\substack{i=4 \\ i:(i,out)\in E}}^{6} w_{i,out}\cdot\sigma\left(\sum_{\substack{j=1 \\ j:(j,i)\in E}}^{3} w_{j,i}\cdot o_j\right)\right)$$

Let us denote w_i:

$$w_i(j)=\begin{cases} w_{j,i} & (j,i)\in E\ ,j\le i \\ 0 & else \end{cases}$$

Thus:

$$\psi(o_{out})=sign\left(\sum_{\substack{i=4 \\ i:(i,out)\in E}}^{6} w_{i,out}\cdot sign\left(\sum_{j=1}^{3} w_i(j)\cdot x(j)\right)\right)=$$

$$sign\left(\sum_{\substack{i=4 \\ i:(i,out)\in E}}^{6} w_{i,out}\cdot sign(\langle w_i,x\rangle)\right)=$$

$$sign\left(\sum_{\substack{i=4 \\ i:(i,out)\in E}}^{6} w_{i,out}\cdot h^3_{w_i}(x)\right)$$

Let us denote the hypothesis class this NN represents as H^3_{NN}. First, we shall show that $H^3_L\subseteq H^3_{NN}$:

Let there be some $w\in\mathbb{R}^d\Rightarrow h^3_w(x)\in H^3_L$. Let's say we have a NN with the same architecture as above. If we set one of the output weights (those that appear in the term above) for some $i=k$ index as $w_{i,out}=1$, and the others as 0, we have:

$$\psi(o_{out})=sign\left(h^3_{w_k}(x)\right)=sign(sign(\langle w_k,x\rangle))=sign(\langle w_k,x\rangle)=h^3_{w_k}(x)$$

Thus, if we set the chosen weights as $w_k=w$ then we have:

$$h^3_w(x)=h^3_{w_k}(x)\in H^3_{NN}\Rightarrow H^3_L\subseteq H^3_{NN}.$$

Now we must prove $H^3_L\subset H^3_{NN}$:

Let us set all the output weights uniformly as $w_{i,out}=1$.

$$\psi(o_{out}) = sign\left(\sum_{\substack{i=4 \\ i:(i,out)\in E}}^{6} w_{i,out} \cdot h^3_{w_i}(x)\right) = sign\left(\sum_{\substack{i=4 \\ i:(i,out)\in E}}^{6} sign(\langle w_i, x\rangle)\right)$$

We now set w_i's s.t $sign(\langle w_i, x\rangle) = x(i-3)$. We change indexes $j(i) = i - 3$, and we now have:

$$\psi(o_{out}) = sign\left(\sum_{\substack{j=1 \\ j(i):(i,out)\in E}}^{3} sign(x(j))\right)$$

This is a function that sums the signs of the coordinates of x, and outputs 1 iff the result is > 0, i.e., iff there are more positive coordinates in x. Otherwise, it returns -1. Let's assume by contradiction that there exists a $w \in \mathbb{R}^3$ s.t $h^3_w(x) = 1$ iff there are more positive coordinates then negative, otherwise $h^3_w(x) = -1$. We have:

$$h^3_w(x) = sign(\langle w, x\rangle) = sign\big(w(1)x(1) + w(2)x(2) + w(3)x(3)\big)$$

For $x_1 = \{0,0,1\}$ we have:

$$h^3_w(x) = sign(w(3))) = 1$$
$$\Rightarrow w(3) > 0$$

And this is true also for $w(1), w(2)$.

For $x_2 = \{1,1,-a\}$ where $a \in \mathbb{R}^+$ we have (more positive coordinates):

$$h^3_w(x) = sign(w(1) + w(2) - a \cdot w(3)) = 1$$
$$\Rightarrow w(1) + w(2) > a \cdot w(3)$$

$a \in \mathbb{R}^+$ is arbitrary, so we must have this for all $a \in \mathbb{R}^+$

$$\Rightarrow w(3) = 0$$

In contradiction with the fact that $w(3) > 0$.

So, there cannot be a $w \in \mathbb{R}^3$ s.t. $h^3_w(x)$ acts as stated. Contradiction.

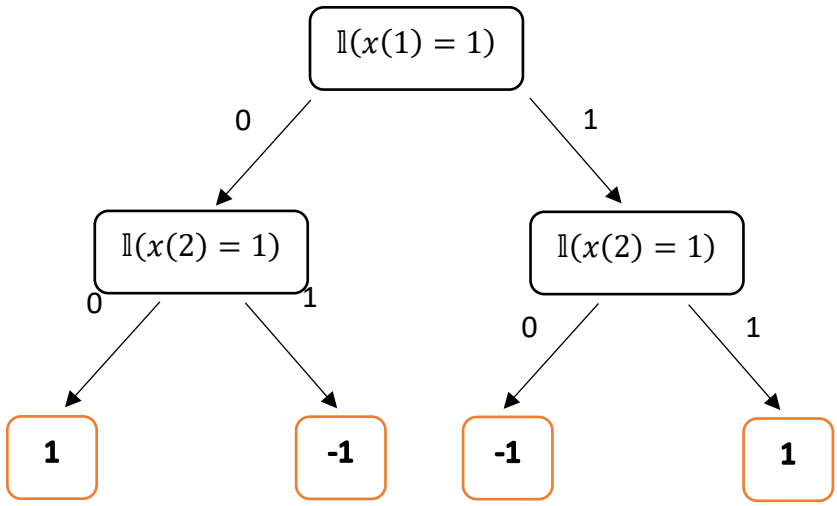
So:

$$\exists f \; s.t \; f \in H^3_{NN}, f \notin H^3_L$$
$$\Rightarrow H^3_L \subset H^3_{NN}$$

4) We assume we are restricted to using test functions of the form

$$\mathbb{I}(x(i) = 1)$$

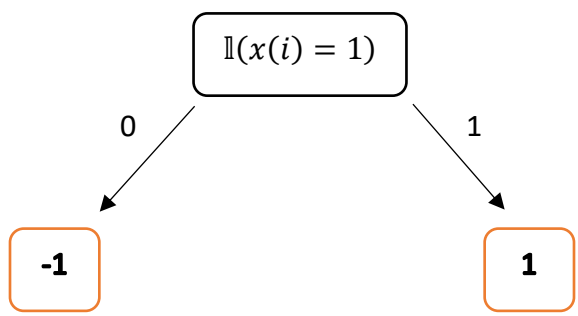
a. First, creating a tree of the form:



We can easily see that we return 1 iff $x(1) = x(2)$, otherwise we return -1. So, we have a tree of depth 2 that perfectly classifies the sample.

Can we do the same for a tree of smaller depth (depth of 1)?

Obviously not. If for some $1 \leq i \leq 3$:



Obviously, this will trivially not work – all coordinates are independent, thus $x(3)$ is totally irrelevant, and also, we can have $x(1) = 1$ for example and have this test on our 1 node, with $x(2) = 1$ or $x(2) = -1$ and no way of knowing if $x(2) = x(1)$ without conducting a test on $x(2)$ (this is the definition of independence, basically). So, we will have a completely random answer in the above example. Thus, we conclude that depth 1 does not work and we need a depth of at least 2.

b. First, we will observe that if every possible example appears the same number of times, we can w.l.o.g look at the case of the sample where all examples appear once. In this case $q = \frac{1}{2}$ as half of the examples hold $x_i(1) = x_i(2)$ and the other half does not.

Let's see how the greedy algorithm will work using the

$Gain(S, i) \coloneqq err_{before}(S) - err_{after}(S, i)$ function,

where $err_{after}(S, i) = p_i^0 * err(q_i^0) + (1 - p_i^0) * err(q_i^1)$

that has been defined in class:

Iteration 1:

$$err_{before} = err(q) = err\left(\frac{1}{2}\right) = \frac{1}{2}$$

Now, for each $i \in \{1,2,3\}$: $err_{after}(S, i) = \frac{1}{2} * \frac{1}{2} + \frac{1}{2} * \frac{1}{2} = \frac{1}{2}$

And therefore, for each $i \in \{1,2,3\}$, $Gain(S, i) = \frac{1}{2} - \frac{1}{2} = 0$

Let assume the returned $j = argmax_{i \in \{1,2,3\}}\big(Gain(S, i)\big) = 3$

Iteration 2: (In the left node where $x_i(3) = 0$)

$$err_{before} = \frac{1}{2}$$

$$err_{after}(S, 1) = err_{after}(S, 2) = \frac{1}{2} * \frac{1}{2} + \frac{1}{2} * \frac{1}{2} = \frac{1}{2}$$

And therefore, for each $i \in \{1,2\}$, $Gain(S, i) = \frac{1}{2} - \frac{1}{2} = 0$

Let assume the returned $j = argmax_{i \in \{1,2\}}\big(Gain(S, i)\big) = 2$

Iteration 3: (In the right node where $x_i(3) = 1$)

$$err_{before} = \frac{1}{2}$$

$$err_{after}(S, 1) = err_{after}(S, 2) = \frac{1}{2} * \frac{1}{2} + \frac{1}{2} * \frac{1}{2} = \frac{1}{2}$$

And therefore, for each $i \in \{1,2\}$, $Gain(S, i) = \frac{1}{2} - \frac{1}{2} = 0$

Let assume the returned $j = argmax_{i \in \{1,2\}}\big(Gain(S, i)\big) = 2$

We received a tree with depth of 2 that each of its leaves has an error of $\frac{1}{2}$.

Therefore, the error of the decision tree that is received from this $ID3$ run, achieved an error of $\frac{1}{2}$.

5) Our objective function defined as: $\lambda\|w\|_1 + \sum_{i=1}^{m}(\langle w, x_i\rangle - y_i)^2$. Let us decompose it to linear and quadratic parts:

$$\lambda\|w\|_1 + \sum_{i=1}^{m}(\langle w, x_i\rangle - y_i)^2 = \lambda\|w\|_1 + \sum_{i=1}^{m}(\langle w, x_i\rangle^2 - 2y_i\langle w, x_i\rangle - y_i^2)$$

$$= \sum_{i=1}^{m}(\langle w, x_i\rangle^2) + \lambda\|w\|_1 - \sum_{i=1}^{m}(2y_i\langle w, x_i\rangle + y_i^2) =$$

$$= \sum_{i=1}^{m}(\langle w, x_i\rangle^2) + \lambda\sum_{j=1}^{d}|w(j)| - \sum_{i=1}^{m}(2y_i\langle w, x_i\rangle + y_i^2)$$

y_i^2 does not depend on w and therefore can be neglected on optimization.

Quadratic programming: $minimize_{z\in\mathbb{R}^{2d}}\left(\frac{1}{2}z^T Hz + \langle u, z\rangle\right)$ s.t. $Az \geq v$

Let us define $\forall j \in \{1, \dots, d\}: \xi_j$. And now, $z = (w(1), \dots, w(d), \xi_1, \dots, \xi_d)$

<u>Linear part</u>: $-2\sum_{i=1}^{m}(y_i\langle w, x_i\rangle) + \lambda\sum_{j=1}^{d}|w(j)|$

For $j \in \{1, \dots, d\}$: Let $u(j) = -2\sum_{i=1}^{m}(y_i x_i(j))$

And for $j \in \{d+1, \dots, 2d\}$: Let $u(j) = 0$

So: $\boldsymbol{u} = \left(\underbrace{\boldsymbol{-2\sum_{i=1}^{m}(y_i x_i(1)), \dots, -2y_i\sum_{i=1}^{m}(x_i(d))}}_{\boldsymbol{d}}, \underbrace{\boldsymbol{\lambda, \dots, \lambda}}_{\boldsymbol{d}}\right)$

In this way: $\langle u, z\rangle = -2\sum_{i=1}^{m}(y_i\langle w, x_i\rangle) + \lambda\sum_{j=1}^{d}\xi_i$

We will need the following restrictions to preserve the absolute value:

$\forall j \in \{1, \dots, d\}$: $\quad w(j) \leq \xi_j$ and $-w(j) \leq \xi_j$

Therefore:

For $i \in \{1, \dots, d\}$: Let $A_{i,j} = \begin{cases} 1, & j = i \\ -1, & j = d + i \\ 0, & else \end{cases}$

For $i \in \{d+1, \dots, 2d\}$: Let $A_{i,j} = \begin{cases} 1, & j = i \ or \ j = i + d \\ 0, & else \end{cases}$

So:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I_d} & \boldsymbol{-I_d} \\ \boldsymbol{I_d} & \boldsymbol{I_d} \end{bmatrix}$$

And $\boldsymbol{v} = \left(\underbrace{\boldsymbol{0, \dots, 0}}_{\boldsymbol{2d}}\right)$

<u>Quadratic part</u>: $\sum_{i=1}^{m}(\langle w, x_i\rangle^2)$

Let us define:

For $i \in \{1, \dots, d\}$: Let $H^*_{i,j} = x_j(i)$

For $i \in \{d+1, \dots, 2d\}$: Let $H^*_{i,j} = 0$

$H = 2H^*$, and now, if we will look on the $[1{:}\,d, 1{:}\,d]$ matrix block we will receive:

$$\frac{1}{2} w^T H_{[1:d,1:d]} w = w^T X^T X w = w^T X^T (w^T X^T)^T =$$

$$= [w(1), \dots, w(d)] \begin{bmatrix} x_1(1), & \dots, x_m(1) \\ \dots, & \dots, \dots \\ x_1(d), & \dots, x_m(d) \end{bmatrix} \left([w(1), \dots, w(d)] \begin{bmatrix} x_1(1), & \dots, x_m(1) \\ \dots, & \dots, \dots \\ x_1(d), & \dots, x_m(d) \end{bmatrix} \right)^T =$$

$$= [\langle w, x_1\rangle, \dots, \langle w, x_m\rangle] \begin{bmatrix} \langle w, x_1\rangle \\ \dots \\ \langle w, x_m\rangle \end{bmatrix} = \sum_{i=1}^{m} (\langle w, x_i\rangle^2)$$

And all other elements for $i > d, j > d$ will be zeros.

So:

$$\boldsymbol{H_{(2d)\times(m+d)} = 2 * \begin{bmatrix} x_1(1) & \dots & x_m(1) & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_1(d) & \dots & x_m(d) & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix}}$$

6)

a. Because there is a unique solution $w \in \mathbb{R}^d$ for $w = (XX^T)^+ Xy$, when X rows are the given examples from training sample S, we can deduce that XX^T is invertible. If it wasn't, then multiple solutions could have been found.

We know that invertible matrices are full ranked.

$$XX^T \in M_{d\times d} \rightarrow Rank(XX^T) = d$$

We also know that $Rank(A) = Rank(AA^T)$ for any matrix A.

Therefore, $Rank(X) = Rank(XX^T) = d$

b. Let $\widehat{w}$ is the optimal solution to the linear regression problem on S,

The set of optimal solutions to the linear regression problem on S' is:

$$OPT_{\widehat{w}} = \{\, [\widehat{w}(1), \dots, \widehat{w}(d), T] \mid T \in \mathbb{R} \,\}$$

Proof:

Because every $x_i'(d+1) = 0$, for every $w' \in \mathbb{R}^{d+1}$:

$$\langle w', x_i' \rangle - y_i = \langle [w'(0), \dots, w'(d)], x_i \rangle - y_i$$

Let us mark the last equation as $(*)$

Let there be $\widehat{w}' \in OPT_{\widehat{w}}$

From $(*)$:

$$\sum_{i=1}^{m} (\langle \widehat{w}', x_i' \rangle - y_i)^2 = \sum_{i=1}^{m} (\langle \widehat{w}, x_i \rangle - y_i)^2$$

Now, because $\widehat{w}$ is the optimal solution to the linear regression problem on S, for every $w' \in \mathbb{R}^{d+1}$:

$$\sum_{i=1}^{m} (\langle \widehat{w}, x_i \rangle - y_i)^2 \leq \sum_{i=1}^{m} (\langle [w'(0), \dots, w'(d)], x_i \rangle - y_i)^2$$

And now again from $(*)$:

$$\sum_{i=1}^{m} (\langle [w'(0), \dots, w'(d)], x_i \rangle - y_i)^2 = \sum_{i=1}^{m} (\langle w', x_i' \rangle - y_i)^2$$

Overall, we received that: $\sum_{i=1}^{m} (\langle \widehat{w}', x_i' \rangle - y_i)^2 \leq \sum_{i=1}^{m} (\langle w', x_i' \rangle - y_i)^2$ for every $\widehat{w}' \in OPT_{\widehat{w}}$ and every $w' \in \mathbb{R}^{d+1}$. Therefore: $OPT_{\widehat{w}}$ is the optimal solution.

7) (All relevant python code will be attached in the zip file)

a. For $x_1 = (1, -2,5,4), x_2 = (3,2,1, -5), x_3 = (-10,1, -4,6)$

We will build: $X = \begin{bmatrix} 1 & -2 & 5 & 4 \\ 3 & 2 & 1 & -5 \\ -10 & 1 & -4 & 6 \end{bmatrix}$

The python code will calculate $A = X^T X$,

and then summarize the $(d - k) = (4 - 2) = 2$ smallest eigenvalues that returned from $numpy.linalg.eigvals()$ function.

The distortion that received from the algorithm is: 4.151633772242231

b. We will use the largest k eigenvalues and correspondence eigen vectors that we found in (a.) of matrix A and stack them on top of each other in descending order from largest eigen vector to smallest. Then we will normalize them (each row of the matrix) to receive U^T.

The result that received from the algorithm is:

$$U^T = \begin{bmatrix} 0.77243642 & -0.00944118 & 0.30530838 & -0.55681203 \\ -0.18467286 & 0.37083788 & -0.6614683 & -0.62516789 \end{bmatrix}$$

c. The restored X matrix will be calculated by the algorithm in the following way:

$$restored\_x = (U * U^T * X^T)^T$$

The restored X that received from the algorithm is:

$$\begin{bmatrix} 1.31364626 & -2.49821189 & 4.48223153 & 4.15965253 \\ 3.67197174 & 0.93260544 & -0.10929356 & -4.65795229 \\ -9.70324553 & 0.5286199 & -4.48988343 & 6.15105425 \end{bmatrix}$$

Meaning that:

$$restored\_x_1 = [1.31364626, -2.49821189, 4.48223153, 4.15965253]$$

$$restored\_x_2 = [3.67197174, 0.93260544, -0.10929356, -4.65795229]$$

$$restored\_x_3 = [-9.70324553, 0.5286199, -4.48988343, 6.15105425]$$

After using the function: $\sum_{i=1}^{m(=3)} \|x_i - UU^T x_i\|^2$ we will receive that the distortion is: 4.1516337722422545. It is similar to the answer we got in (a.), that is because: $\sum_{i=1}^{m(=3)} \|x_i - UU^T x_i\|^2 = \{sum\ of\ (d - k) = 2\ minimal\ eigen\ values\}$, as we have seen in class.

8)

a. Single linkage clustering does not satisfy the scale invariance axiom.

<u>Proof</u>:

We will show it with a counter example:

For $S = \{0,1\}$, $\rho(x_1, x_2) = |x_1 - x_2|$ and $r = 2$, the single linkage algorithm will choose to merge cluster $\{0\}$ and cluster $\{1\}$ as $|1 - 0| < r = 2$ and therefore $F(S, \rho) = (\{0,1\})$.

But, for $\alpha\rho$, where $\alpha = 10$, we will receive that the new distance between cluster $\{0\}$ and cluster $\{1\}$ is: $10 * |1 - 0| = 10 \nless r = 2$ and therefore, no merging will occur and $F(S, \alpha\rho) = (\{0\}, \{1\})$.

Because $F(S, \rho) \neq F(S, \alpha\rho)$ the invariance axiom does not hold.

■

b. Single linkage clustering does satisfy the richness axiom.

<u>Proof</u>:

Let $C = (c_1, \dots, c_k)$ be a valid partition of S.

We will define $\rho(x_1, x_2) = \begin{cases} 0 \ , & x_1 = x_2 \\ \frac{1}{2}r \ , & x_1, x_2 \in c_i \ for\ some\ i \in \{1, \dots, k\} \\ 2r \ , & else \end{cases}$

Firstly, we will show that $\rho(x_1, x_2)$ holds the symmetric, positive and triangle inequality axioms.

Let there be $x_1, x_2 \in S$:

Symmetric:

- If $x_1 = x_2$, then $\rho(x_1, x_2) = 0 = \rho(x_2, x_1)$
- If x_1 and x_2 are both in cluster c_i, then $\rho(x_1, x_2) = \frac{1}{2}r = \rho(x_2, x_1)$
- Else, x_1 and x_2 are not in the same cluster,
  and then $\rho(x_1, x_2) = 2r = \rho(x_2, x_1)$

Positive:

- $0 < \frac{1}{2}r < 2r$, and therefore: $\rho(x_1, x_2) > 0$

Triangle inequality:

Let there be $x_3 \in S$

- If $x_1 = x_3$, then $\rho(x_1, x_3) = 0 \leq \rho(x_1, x_2) + \rho(x_1, x_3)$, as ρ is positive.
- If x_1 and x_3 are both in cluster c_i, then:
  - w.l.o.g, if $x_2 = x_1$:

    $\rho(x_1, x_3) = \frac{1}{2}r \leq \rho(x_1, x_2) + \rho(x_2, x_3) = 0 + \frac{1}{2}r = \frac{1}{2}r$
  - If $x_2 \in c_i$ then

    $\rho(x_1, x_3) = \frac{1}{2}r \leq \rho(x_1, x_2) + \rho(x_2, x_3) = \frac{1}{2}r + \frac{1}{2}r = r$
  - If $x_2 \notin c_i$ then

    $\rho(x_1, x_3) = \frac{1}{2}r \leq \rho(x_1, x_2) + \rho(x_2, x_3) = 2r + 2r = 4r$
- Else, x_1 and x_2 are not in the same cluster, then:
  - w.l.o.g, if $x_2 = x_1$:

    $\rho(x_1, x_3) = 2r \leq \rho(x_1, x_2) + \rho(x_2, x_3) = 0 + 2r = 2r$
  - w.l.o.g, if x_1 and x_2 are in the same cluster,

    $\rho(x_1, x_3) = 2r \leq \rho(x_1, x_2) + \rho(x_2, x_3) = \frac{1}{2}r + 2r = \frac{3}{2}r$
  - If all x_1, x_2, x_3 are not in the same cluster,

    $\rho(x_1, x_3) = 2r \leq \rho(x_1, x_2) + \rho(x_2, x_3) = 2r + 2r = 4r$

Now, we will show that the algorithm output will be C.

As we seen in class, for graph $G_r = (V, E)$, where $V = S$ and $E = \{x_1, x_2 | \rho(x_1, x_2) < r\}$ the returned clusters from Single-linkage method will be the connected components of G_r.

Therefore, we need to show that for every $x_1, x_2 \in c_i$: $\rho(x_1, x_2) < r$ and that for every $x_1 \in c_i, x_2 \in c_j$ where $i \neq j$: $\rho(x_1, x_2) \nless r$.

Let there be $x_1, x_2 \in c_i$, thus, by the definition of $\rho(x_1, x_2)$ as declared above, $\rho(x_1, x_2) = \frac{1}{2}r < r$. Therefore $(x_1, x_2) \in E$ and x_1, x_2 will be classified together.

Let there be $x_1 \in c_i, x_2 \in c_j$ where $i \neq j$. thus, by the definition of $\rho(x_1, x_2)$ as declared above, $\rho(x_1, x_2) = 2r \nless r$. Therefore $(x_1, x_2) \notin E$ and x_1, x_2 will not be classified together.

■