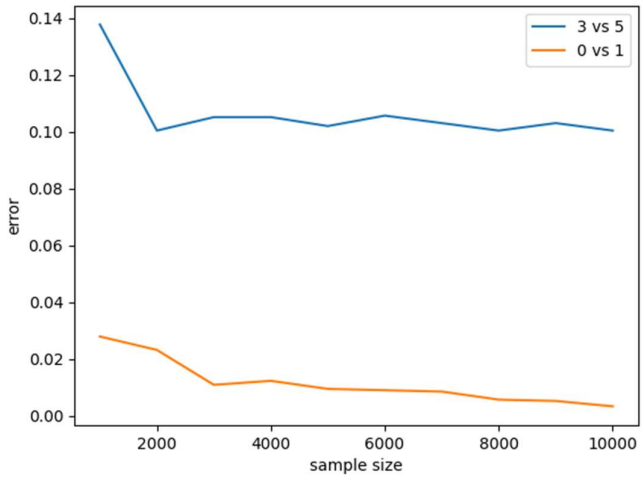
Assignment 3

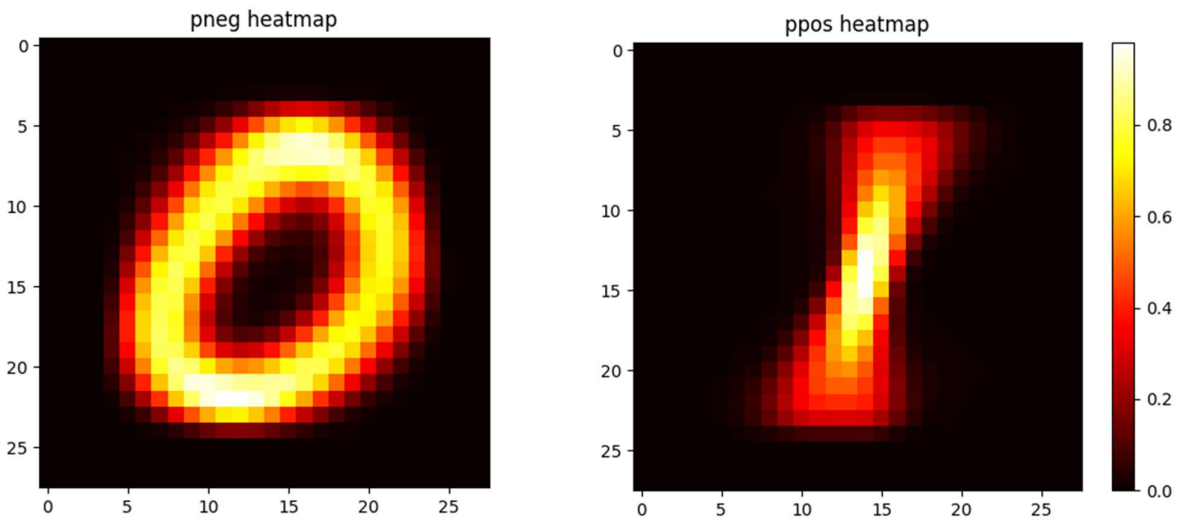
Question 2:

a)



b) The algorithm had much better success on the 0 vs 1 problem then 3 vs 5 problem, the error was almost 10 times smaller on the first. We can see some trending in the plot, as the sample size gets bigger the error at 3 vs 5 problem does not change a lot, while at 0 vs 1 it is decreasing although the difference is not very significant.

c)



We can see that the heatmap of ppos is similar to 1 digit, and the heatmap of pneg is similar to 0 digit.

Looking at different areas of the heatmap:

At the <u>center</u> of the images, we can see that ppos is hot (~0.8). Since $ppos(i) = \mathbb{P}[X(i)|Y = 1]$, and the label 1 represents the digit 1, it makes sense that coordinates that are used to draw 1 will be brighter than others. At pneg image the center is colder, meaning these coordinated are rarely used when drawing 0. Reminding the prediction function: $\log\left(\frac{allpos}{1-allpos}\right) + \sum_{x(i)=1} \log(\frac{ppos(i)}{pneg(i)}) - \sum_{x(i)=-1} \log(\frac{1-pne\ (i)}{1-ppo\ (i)})$.

By this formula we can see that coordinates from the center gets high $\frac{ppos(i)}{pneg(i)}$ and low $\frac{1-pneg(i)}{1-ppos(i)}$ which implies the prediction to be positive.

The coordinates at the edges are cold in both images, since there are rarely used when drawing these digits, meaning they got a small value and they don't help us for the prediction.

d) First notice that the sample was chosen uniformly random making allpos ~0.5, setting allpos=0.75 increased it most of the time. Reminding the formula for prediction:

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

Since the change increased $\frac{allpos}{1-allpos}$ we expect that it will yield more percentage of changes from label -1 to 1 and no opposite changes.

Results for 0 vs 1:

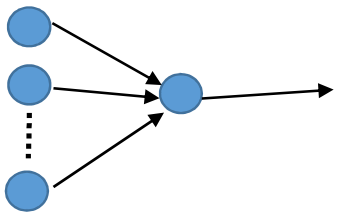
Changes from -1 to 1: 0%
Changes from 1 to -1: 0%
Here the change made a minor impact. Most of the runs it didn't made any changes at the prediction at all, some runs it caused 0.1% change only from -1 to 1 (as expected). It can be explained by the fact that the heatmap of this digits are completely different, the hot areas of 1 are cold areas of 0 and vice versa. It means that 1 and 0 are different enough from each other such that even change of the $\log\left(\frac{allpos}{1-allpos}\right)$ does not cause wrong predictions.

Results for 3 vs 5:

Changes from -1 to 1: 1.4%
Changes from 1 to -1: 0%
Here the change made much more impact, the change from -1 to 1 was always between 1.2% to 1.5%. Again, none of the runs caused change from 1 to -1. The difference from the 0 vs 1 is probably due to some similarities between 3 and 5 digit, meaning the heatmap has shared hot areas. At these cases the sum within the sign function was probably close to zero and increasing $\log\left(\frac{allpos}{1-allpos}\right)$ made some images that were less than zero to be positive.

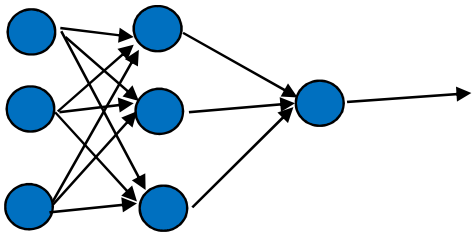
Question 3:

a) Take a fully connected network with no hidden layers. So it has d input neurons and a single output neuron. We will prove that the hypothesis class it represents is equivalent to the class of homogeneous linear predictors. Denote the network function by $f_w^{G,\sigma}$ where G is the graph of the network and σ is the sign function, and w are the weights on the edges.



The architecture of the network implies that for $x \in \mathcal{X}$, $f_w^{G,\sigma}(x) = \sum_{i=1}^{d} w_i x_i$. So the prediction for x is $sign\left(\sum_{i=1}^{d} w_i x_i\right) = sign(< w, x >)$ which means that the network prediction behaves like a linear predictor $w \in \mathbb{R}^d$. On the other side, for every linear predictor $w \in \mathbb{R}^d$ the take the networks with weights w. So $sign$ of $f_w^{G,\sigma}$ is in the hypothesis function of the network and it behaves the same as linear predictor. So we saw that for any linear predictor there exist an equivalent $f_w^{G,\sigma}$ and vice versa, which means the hypothesis classes are equivalent.

b) Take a fully connected network with one hidden layer with d neurons. For example for $d = 3$:



We will prove that the hypothesis class of linear predictor included but not equivalent to the hypothesis class of such network.

For $w \in \mathbb{R}^3$ let h_w be a linear predictor. Define $f_W \in \mathcal{H}_{G,\sigma}$ such that for every neuron at the second layer $o_{3+i} = sign\left(\sum_{j=1}^{3} x_j w_j\right) = sign(< w, x >)$ where $1 \leq i \leq 3$ meaning all weights of input neuron o_i are set to w_i, and for the output layer $o_7 = \sum_{i=1}^{d} o_{3+i}$ meaning we set all the weights from the hidden layer to the output to equal 1. So f_W acts as follows:

$$f_W(x) = sign\left(\sum_{i=1}^{3} o_{3+i}\right) = sign(\sum_{i=1}^{3} sign(< w, x >)) = sign\big(3 * sign(< w, x >)\big) =$$

$$= sign(< w, x >)$$

The calculation shows that $f_W \equiv h_w$ which means that $h_w \in \mathcal{H}_{G,\sigma}$, hence the hypothesis class of the network includes the hypothesis class of homogeneous linear predictors.

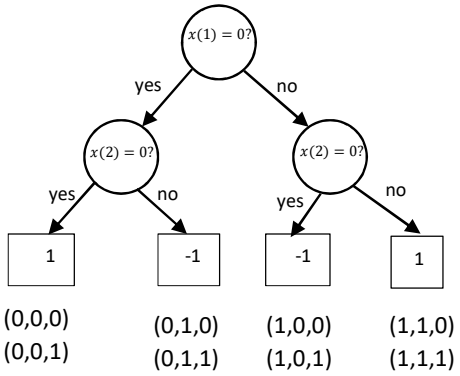
We will now show an example of $f_W \in \mathcal{H}_{G,\sigma}$ that is not equivalent to any linear predictor. Set the weights such that $o_{3+i} = sign(x_i)$ meaning each neuron is connected to weight 1 to the neuron o_{3+i} and the rest are 0. Also, set the weights of the edges connecting the hidden layer to the output layer to be 1. So we have: $f_W(x) = sign(\sum_{i=1}^{3} sign(x_i))$. This is a function which is positive iff there are more positive coordinates than negative coordinates.

Assume towards contradiction that there is a linear predictor h_w such that $h_w \equiv f_W$.
Then for $x = (1,1,-1)$, $h_w(x) = sign(w_1 + w_2 - w_3) = 1$ so $w_1 + w_2 > w_3$.
For $x = (1,-3,1)$, $w_1 + w_3 > 3w_2$ together with the first inequality: $w_1 > w_2$.
For $x = (-3,1,1)$, $w_2 + w_3 > 3w_1$ together with the first inequality: $w_2 > w_1$, which is a contradiction. Hence, there is no w such that $h_w \equiv f_W$.

So the hypothesis class of the network is not equivalent to linear predictors.

Question 4:

a) First we will construct a tree of depth 2 that perfectly classify this sample set:



This tree has depth 2 and it is classifying an example from the sample set as 1 iff $x(1) = x(2)$

Moreover, we will show that there is no tree of depth 1 that perfectly classify this sample. A tree of depth 1 asks the value of one feature of x_i and then it must decide its label. If it asks for the first feature then it must ask also the second in order to determine if $x(1) = x(2)$. So it must at least ask for two features in order to determine the label, hence the depth is greater than 1.

b) We will show that a greedy algorithm which uses one of the $Gain$ function we saw at class can get error of 50% after getting depth 2.

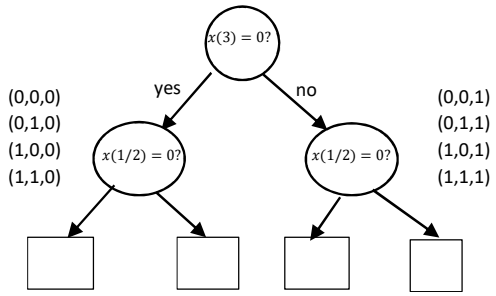
In the example we will use "Improvement in the sample error" $Gain$ function. Notice that in this case, $err(q) = \frac{1}{2}$. Notice that because all values of X are observed same number of times, then: $q_i^0 = q_i^1 = p_i^0 = \frac{1}{2}$. So from here we can calculate $Gain$ for $i = 1,2,3$:

$$Gain(S,1) = err_{Before}(S) - err_{After}(S,1) = err(q) - p_1^0 err(q_1^0) - (1 - p_1^0) err(q_1^1) = \frac{1}{2} - \frac{1}{4} - \frac{1}{4} = 0$$

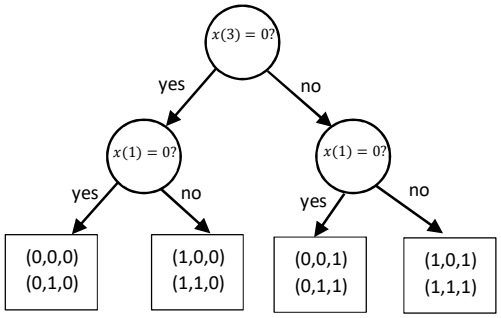
$$Gain(S,2) = err_{Before}(S) - err_{After}(S,2) = err(q) - p_2^0 err(q_2^0) - (1 - p_2^0) err(q_2^1) = \frac{1}{2} - \frac{1}{4} - \frac{1}{4} = 0$$

$$Gain(S,3) = err_{Before}(S) - err_{After}(S,3) = err(q) - p_3^0 err(q_3^0) - (1 - p_3^0) err(q_3^1) = \frac{1}{2} - \frac{1}{4} - \frac{1}{4} = 0$$

We can see that $Gain(S,i) = 0$ so a greedy algorithm does not prefer to choose some index more than another. If for instance in this case the algorithm choose an index randomly, it has a chance to choose $i = 3$. From here the tree will look as follows:



The tree can ask in the next level about $x(1)$ or $x(2)$. We will show how the tree looks for the next level when choosing to ask "$x(1) = 0$?", the other cases are symmetric and will yield the same result.



We can see that on each node of depth 2 there are exactly two type of elements from X, and each one has different label. Hence, no matter what label we choose to put on the leaves we will always get 50% error on the sample.

We showed that for a that $Gain$ function, if the algorithm will first ask about the 3'rd feature then the algorithm will have 50% error on depth 2.

Question 5:

Define the minimization problem:

$$Minimize_{w\in\mathbb{R}^d, s\in\mathbb{R}^d} \left||X^T w - y|\right|^2 + \lambda \sum_{i=0}^{d} s_i$$
$$s.t \quad s_i \geq w_i$$
$$s_i \geq -w_i$$

The constrains ensures that $s_i \geq |w_i|$ so in the optimal solution we will get $s_i = |w_i|$, meaning it is equivalent to the LASSO objective.
Notice that:
$$\left||X^T w - y|\right|^2 = (X^T w - y)^T (X^T w - y) = (w^T X - y^T)(X^T w - y) = w^T X X^T w - 2y^T X^T w + y^T y$$

So w which minimizes $\left||X^T w - y|\right|^2$ will also minimize $w^T X X^T w - 2y^T X^T w$. Hence we can rewrite our objective:

$$Minimize_{w\in\mathbb{R}^d, s\in\mathbb{R}^d} w^T X X^T w - 2y^T X^T w + \lambda \sum_{i=0}^{d} s_i$$
$$s.t \quad s_i \geq w_i$$
$$s_i \geq -w_i$$

From here we can define a quadratic program:

$H \in \mathbb{R}^{2d\times 2d}$ matrix of size $2d \times 2d$ defined as $H = \begin{pmatrix} 2XX^T & 0 \\ 0 & 0 \end{pmatrix}$.
$u \in \mathbb{R}^{2d}$ is a vector which its first d are the same as the vector $2(Xy)^T \in \mathbb{R}^d$, and the last d elements are λ, $u = (-2\sum_{i=1}^m X_{1,i}\, y_i, \dots, -2\sum_{i=1}^m X_{d,i}\, y_i, \lambda, \dots, \lambda)$.
The constraints can rewritten as $s_i - w_i \geq 0$ and $s_i + w_i \geq 0$ and from here denote $I_d \in \mathbb{R}^{d\times d}$ to be the identity matrix of size d. So $A = \begin{pmatrix} -I_d & I_d \\ I_d & I_d \end{pmatrix}$ and $v = (0, \dots, 0) \in \mathbb{R}^{2d}$.

If we think of the vector $z \in \mathbb{R}^{2d}$ as $z = (w_1, \dots, w_d, s_1, \dots, s_d)$ than the quadratic program:

$$Minimize_{z\in\mathbb{R}^{2d}} \frac{1}{2} z^T H z + < u, z >$$
$$s.t \quad Az \geq v$$

Will be the solution to the minimization problem stated above.

Question 6

a) In class we saw that the solution w to the problem of linear regression with squared loss must satisfy: $XX^T w = Xy$. Since there is a unique solution to the system, the matrix XX^T is invertible, hence the rank of $XX^T \in \mathbb{R}^{d\times d}$ is d. From here we can use the fact that $rank(XX^T) \leq rank(X)$ (this is true to every two matrixes) and deduce that $d \leq rank(X)$. But X has d rows meaning its rank can't be greater then d. So we can deduce that $rank(X) = d$.

b) After adding the extra coordinate X' will be at the form: $X' = \begin{pmatrix} | & | & \cdots & | \\ x_1 & x_2 & \cdots & x_m \\ | & | & \cdots & | \\ 0 & 0 & \cdots & 0 \end{pmatrix}$ so we know that $X'X'^T = \begin{pmatrix} XX^T & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{(d+1)\times(d+1)}$. Moreover, $X'y = \begin{pmatrix} Xy \\ 0 \end{pmatrix}$. Denote $w \in \mathbb{R}^d$ to be the unique solution of the original minimization problem. Notice that since the last element of $x_i{}'$ is zero, then for all $c \in \mathbb{R}$ if we denote $w'^T = (w, c)$ we know that:

$$X'X^T w' = \begin{pmatrix} XX^T & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} w \\ c \end{pmatrix} = \begin{pmatrix} Xy \\ 0 \end{pmatrix} = X'y$$

So every w' as above is a solution. On the other hand, if some $\widetilde{w} \in \mathbb{R}^{d+1}$ is a solution, then by the second equality above, its first d coordinates must be the same as w. Putting it together we deduce

that w' is an optimal solution to the minimization problem iff w is a solution to the original problem. The set of optimal solution is defined as follows:

$$\{w' \in \mathbb{R}^{d+1} \big| w' = (w, c) where\ w \in \mathbb{R}^d\ satisfy\ XX^T w = Xy\ and\ c \in \mathbb{R}\}$$

Question 7:

a) The distortion rate for the sample is: 4.151633772242231

b) Calculating U^T was taking two eigenvectors of $X^T X$ which correspond to the two largest eigenvalues and put them as rows of the matrix. The result we got:

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

The matrix is indeed orthonormal since all rows has norm 1 and are orthogonal to each other.

c) The reconstructed matrix X after multiplying by U^T and U is:

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

Now we can calculate the distortion rate by evaluating the objective: $\sum_{i=1}^{4} ||x_i^T - UU^T x_i^T||_2^2$.

Vectorizing the expression we evaluate: $np.linalg.norm(X.T - U\ @U.T@X.T) ** 2$
The result we got: 4.1516337722422545.

The result is similar to the one we got at section a (up to numerical error 14 digits after point). Not surprisingly since we proved in class that the destruction rate is the minimum value of the objective.

Question 8:

a) The algorithm **does not** satisfy the scale-inverse axiom.

Assume $\rho(x_1, x_2) = \big||x_1 - x_2|\big|_2$ and sample $S = (x_1, x_2)$, such that $\rho(x_1, x_2) = r > 0$. An algorithm $F(S, \rho)$ that has cluster distance stopping condition r does not satisfy the scale-inverse axiom. Since $\rho(x_1, x_2) = r$ then $F(S, \rho)$ will stop with the partition $C = \{\{x_1, x_2\}\}$ meaning it will unify both elements from the sample set.

Now take $\alpha = 2$, and run the algorithm $F(S, \alpha\rho)$. Now $\alpha\rho(x_1, x_2) = 2\big||x_1 - x_2|\big|_2 = 2r$, so when we run $F(S, \alpha\rho)$ we will get the partition $C = \{\{x_1\}, \{x_2\}\}$ meaning it separates the elements. So we got $F(S, \alpha\rho) \neq F(S, \rho)$ which means F does not satisfy the scale-inverse axiom.

b) The algorithm **does** satisfy the richness axiom.

Given finite sample $S \subset \mathcal{X}$ and $C = (C_1, \dots, C_k)$ partition of S define a function $\rho: \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ such that: $\rho(x, x') = \begin{cases} 0 & if\ x = x' \\ r & if\ x, x' \in C_i\ and\ x \neq x' \\ 2r & else \end{cases}$

First, we need to prove that ρ is indeed a metric:

Symmetric: ρ is symmetric by definition.

Positive: Since $r > 0$ we can see that $\rho(x, x') \geq 0$. Moreover, by definition $\rho(x, x') = 0 \; iff \; x = x'$.

Triangle inequality: Take x_1, x_2. For the case where $x_1 = x_2$ since ρ is positive then for any x_3 we get: $\rho(x_1, x_2) = 0 \leq \rho(x_1, x_3) + \rho(x_3, x_2)$. For the case where $x_1, x_2 \in C_i$ for some $1 \leq i \leq k$ and $x_1 \neq x_2$, if also $x_3 \in C_i$ then $\rho(x_1, x_3) + \rho(x_3, x_2) \geq r = \rho(x_1, x_2)$. If $x_3 \notin C_i$ then: $\rho(x_1, x_3) + \rho(x_3, x_2) = 2r + 2r > r = \rho(x_1, x_2)$. For the last case where $\rho(x_1, x_2) = 2r$, if x_3 shares the same C_i with x_1 or x_2 then: $\rho(x_1, x_3) + \rho(x_3, x_2) \geq 2r = \rho(x_1, x_2)$. If x_3 does not share the same C_i with x_1 or x_2 then $\rho(x_1, x_3) + \rho(x_3, x_2) = 2r + 2r > 2r = \rho(x_1, x_2)$. So for any case the triangle inequality holds.

We can conclude that ρ is indeed a metric.

Now we will prove that $F(S, \rho) = C$.

Let $C' = (C_1', \dots, C_n')$ be the output of $F(S, \rho)$. Notice that since F uses single-linking, together with cluster distance stopping condition which ensures $\rho(C_i', C_j') = 2r$ for $i \neq j$, we can deduce that the partition does not separate two elements x_1, x_2 that are in the same cluster in the target partition. Meaning that if $x_1, x_2 \in C_i$ then $x_1, x_2 \in C_j{}'$.

In addition, we want to show that if $x_1, x_2 \in C_j{}'$ finish in the same cluster, then $x_1, x_2 \in C_i$ for some $1 \leq i \leq k$. Assume towards contradiction that this is not the case. Let x_1, x_2 be the first two elements of S such that $\rho(x_1, x_2) = 2r$ and F chooses to unify their clusters. So $x_1 \in A$ and $x_2 \in B$, since it is the first time the algorithm unifies elements with $\rho(x_1, x_2) = 2r$, it must satisfy that $\rho(A, B) = 2r$ (otherwise it is not the first time it unifies such elements). F is single-linkage algorithm so it means that A and B are closest clusters, which is a contradiction to the stopping condition of F.

We can conclude that $x_1, x_2 \in C_i \; iff \; x_1, x_2 \in C_j{}'$. So $n = k$ and $C = C'$ which is exactly what we wanted to prove.