MLHW3

yz5946 Sky Zhou

May 2024

1 Problem 1

a

First, note that for $K_G(I_i, I_i) = 1$ since the Euclidean distance of a vector to itself is 0. Thus $e^0 = 1$.

In addition, the definition of the Euclidean distance ensures that $K_G(I_i, I_j) = K_G(I_j, I_i)$. Therefore, we only need to compute $K_G(I_1, I_2)$, $K_G(I_1, I_3)$, $K_G(I_1, I_4)$, $K_G(I_2, I_3)$, $K_G(I_2, I_4)$, $K_G(I_3, I_4)$.

And the Euclidean distance is the sum of the number of different digits between two matrices.

$$K_G(I_1, I_2) = e^{-7} \tag{1}$$

$$K_G(I_1, I_3) = e^{-8} \tag{2}$$

$$K_G(I_1, I_4) = e^{-5} \tag{3}$$

$$K_G(I_2, I_3) = e^{-5} \tag{4}$$

$$K_G(I_2, I_4) = e^{-3} \tag{5}$$

$$K_G(I_3, I_4) = e^{-11} \tag{6}$$

The final Gaussian kernel matrix should be:

$$K_G = \begin{bmatrix} 1 & e^{-7} & e^{-8} & e^{-5} \\ e^{-7} & 1 & e^{-5} & e^{-3} \\ e^{-8} & e^{-5} & 1 & e^{-11} \\ e^{-5} & e^{-3} & e^{-11} & 1 \end{bmatrix}$$

b

To classify I_3 and I_4, we just need to compare the entries $(1,3), (2,3)$ and $(1,4), (2,4)$ in K_G.

We have $K_G(1,3) = e^{-8} < K_G(2,3) = e^{-5}$

$K_G(1,4) = e^{-5} < K_G(2,4) = e^{-3}$

Therefore, I_2 is the most similar to both I_3 and I_4 according to the Gaussian kernel similarity.

In this case, I expect a kernel classifier to correctly classify I_4 as 1 but they will incorrectly classify I_3 since I_3 should be 0 instead of 1.

c

We first separately compute $K_G^{right-right}, K_G^{left-left}, K_G^{right-left}, K_G^{left-right}$ and sum them up to get K_{shift}

For simplicity I omit the calculation process and display the result matrices directly.

$$K_G^{right-right} = \begin{bmatrix} 1 & e^{-7} & e^{-8} & e^{-5} \\ e^{-7} & 1 & e^{-5} & e^{-3} \\ e^{-8} & e^{-5} & 1 & e^{-11} \\ e^{-5} & e^{-3} & e^{-11} & 1 \end{bmatrix}$$

$$K_G^{left-left} = \begin{bmatrix} 1 & e^{-7} & e^{-5} & e^{-5} \\ e^{-7} & 1 & e^{-2} & e^{-3} \\ e^{-5} & e^{-2} & 1 & e^{-8} \\ e^{-5} & e^{-3} & e^{-8} & 1 \end{bmatrix}$$

$$K_G^{right-left} = \begin{bmatrix} 1 & e^{-5} & e^{-7} & e^{-7} \\ e^{-5} & 1 & e^{-8} & 1 \\ 1 & e^{-7} & 1 & e^{-5} \\ e^{-11} & e^{-6} & e^{-8} & 1 \end{bmatrix}$$

$$K_G^{left-right} = (K_G^{right-left})^T \begin{bmatrix} 1 & e^{-5} & 1 & e^{-11} \\ e^{-5} & 1 & e^{-7} & e^{-6} \\ e^{-7} & e^{-8} & 1 & e^{-8} \\ e^{-7} & 1 & e^{-5} & 1 \end{bmatrix}$$

Therefore

$$K_{shift} = \begin{bmatrix} 4 & 2*(e^{-5}+e^{-7}) & 1+e^{-8}+e^{-5}+e^{-7} & 2*e^{-5}+e^{-7}+e^{-11} \\ 2*(e^{-5}+e^{-7}) & 4 & e^{-2}+e^{-8}+e^{-5}+e^{-7} & 1+2*e^{-3}+e^{-6} \\ 1+e^{-8}+e^{-5}+e^{-7} & e^{-2}+e^{-8}+e^{-5}+e^{-7} & 4 & 2*e^{-8}+e^{-5}+e^{-11} \\ 2*e^{-5}+e^{-7}+e^{-11} & 1+2*e^{-3}+e^{-6} & 2*e^{-8}+e^{-5}+e^{-11} & 4 \end{bmatrix}$$

For ease of comparision to classify, we calculate out the numerical values:

$$K_{shift} = \begin{bmatrix} 4 & 0.01529 & 1.00798 & 0.01440 \\ 0.01529 & 4 & 0.14332 & 1.10205 \\ 1.00798 & 0.14332 & 4 & 0.00742 \\ 0.01440 & 1.10205 & 0.00742 & 4 \end{bmatrix}$$

d

Use the "left-right shift" kernel, I_3 is most similar to I_1 and I_4 is most similar to I_2. A kernel classifier would correctly classify I_3 as 0 and I_4 as 1 now.

e

Given K_G is SPD, we have that $\forall x_i, x_j$ in I_1, I_2, I_3, I_4 where x_i, x_j are concatenation of row vectors, $\exists$ a transformation function ϕ, such that $K_G(I_i, I_j) = \phi(x_i)^T \phi(x_j)$ where $\phi(x_i), \phi(x_j)$ can be also deemed as a transformation that applies separately to the column vectors of I_i, I_j. Since I_i^{left}, I_i^{right} are linearly dependent to $I_i \forall i \in \{1, 4\}$, so the same transformation function ϕ is also applicable to I_i^{left}, I_i^{right}, making $K_G^{right-right}, K_G^{left-left}, K_G^{right-left}, K_G^{left-right}$ all SPD. And since K_{shift} is the linear combination of these 4 SPDs, K_{shift} itself is also a SPD.

2

a

To solve for biases and weights, we first need to generate a list of coordinates for the data points for fitting.

Dataset 1: (0,8), (1,6), (2,4), (3,2), (4,0), (5,2),(6,4),(7,2),(8,0),(9,2),(10,4),(11,6),(12,8)

We can observe a 4-segment linear relationship. Therefore we can activate 4 different neurons sequentially according to different x values.

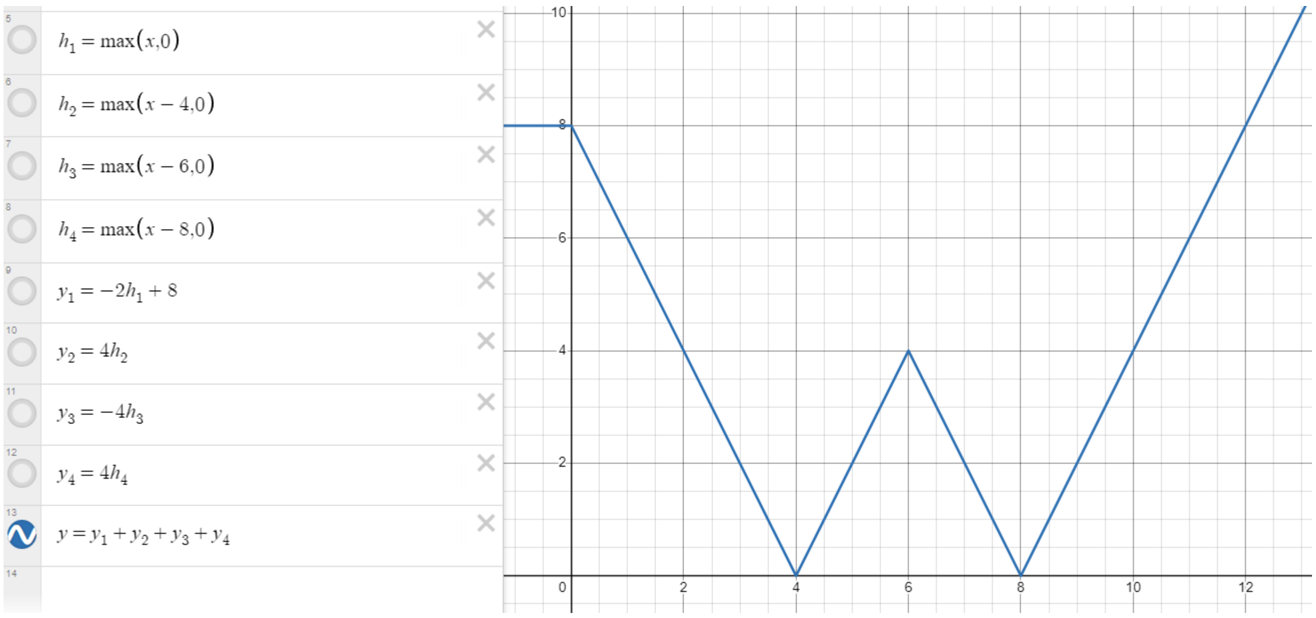
When $0 \leq x < 4$, let's activate neuron z_1. To let z_1 fit the data, we need $z_1 = max(x, 0)$ so that the value x will be passed to z_1 completely. In this case, we have $W_{(H,1)} = 1, b_{(H,1)} = 0$. To fit the data points, we need $W_{(O,1))} = -2, b_O = -8$. In this case the first line segment is fit.

Similarly, when $4 \leq x < 6$, z_1, z_2 will both be activated. The actual calculations should be similar so I will omit the details here. We need $z_2 = max(x-4, 0)$, so $W_{(H,2)} = 1, b_{(H,2)} = -4$. To fit the data points, we need $W_{(O,2))} = 4$ given $W_{(O,1))} = -2, b_O = -8$.

when $6 \leq x < 8$, z_1, z_2, z_3 will all be activated. We need $z_3 = max(x-6, 0)$, so $W_{(H,2)} = 1, b_{(H,2)} = -6$. To fit the data points, we need $W_{(O,3))} = -4$.

when $8 \leq x$, z_1, z_2, z_3, z_4 will all be activated. We need $z_4 = max(x-8, 0)$, so $W_{(H,2)} = 1, b_{(H,2)} = -8$. To fit the data points, we need $W_{(O,4))} = 4$.

The graph below shows the result of the fit:



$h_1 = \max(x,0)$

$h_2 = \max(x-4,0)$

$h_3 = \max(x-6,0)$

$h_4 = \max(x-8,0)$

$y_1 = -2h_1 + 8$

$y_2 = 4h_2$

$y_3 = -4h_3$

$y_4 = 4h_4$

$y = y_1 + y_2 + y_3 + y_4$

Dataset 2: (0,0), (1,1), (2,2), (3,3), (4,2), (5,1),(6,1.5),(7,2),(8,2.5),(9,3),(10,3.5),(11,4),(12,4.5)

For this dataset, only 3 segements are observed. Therefore we only need to activate 3 of the 4 neurons.

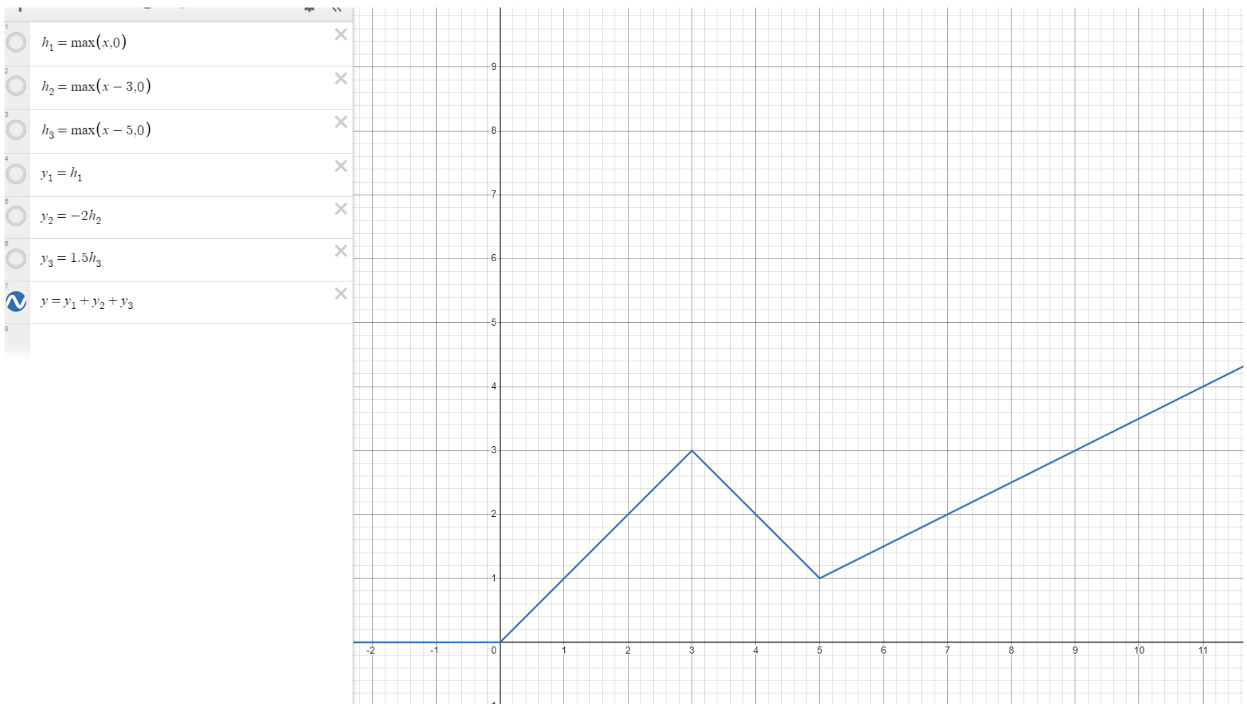
When $0 \le x < 3$, let's activate neuron z_1. To let z_1 fit the data, we need $z_1 = max(x, 0)$. In this case, we have $W_{(H,1)} = 1, b_{(H,1)} = 0$. To fit the data points, we need $W_{(O,1))} = 1, b_O = 0$. The first line segment is fit.

when $3 \le x < 5$, z_1, z_2 will both be activated. We need $z_2 = max(x - 3, 0)$, so $W_{(H,2)} = 1, b_{(H,2)} = -3$. To fit the data points, we need $W_{(O,2))} = -2$ given $W_{(O,1))} = 1, b_O = 0$.

when $5 \le x$, z_1, z_2, z_3 will all be activated. We need $z_3 = max(x - 5, 0)$, so $W_{(H,3)} = 1, b_{(H,3)} = -5$. To fit the data points, we need $W_{(O,3))} = 1.5$.

z_4 will remain 0, so $W_{(H,4)} = 0, b_{(H,4)} = 0$

The graph below shows the result of the fit:



$h_1 = \max(x, 0)$

$h_2 = \max(x - 3, 0)$

$h_3 = \max(x - 5, 0)$

$y_1 = h_1$

$y_2 = -2h_2$

$y_3 = 1.5h_3$

$y = y_1 + y_2 + y_3$

b

We apply a simple chain rule to compute $\nabla \boldsymbol{L}(\theta)$

$$\nabla \boldsymbol{L}(\theta) = \sum_{i=1}^{n} 2(y_i - f(x_i, \theta)) * \frac{\partial}{\partial\theta]}(y_i - f(x_i, \theta)) \tag{1}$$

$$= \sum_{i=1}^{n} 2(y_i - f(x_i, \theta)) * (-\nabla f(x_i, \theta)) \tag{2}$$

c

According to the given data, we have a set of activation functions: $z_1 = max(-x + 1, 0), z_2 = max(x + 1, 0), z_3 = max(x - 1, 0), z_4 = max(-x + 1, 0)$

Therefore, when $x = 2$, only z_2, z_3 will be activated. Plug in $x = 2$, we have $z_2 = 3, z_3 = 1$.

We also have $W_{(O,2)} = -1, W_{(O,3)} = -1, b_O = 1,\ f(x, \theta_0) = W_{(O,2)} * z_2 + W_{(O,3)} * z_3 + b_O = -3 - 1 + 1 = -3$

d

Normally we need an actual output to measure the loss in order to compute gradients, i.e. the degree of contribution of each parameter to the loss. However, it is not provided here. Therefore I will use two different approaches, one is use the method in the slides, the other is to keep the actual output as an unknown. Since only z_2, z_3 are activated, we can omit the gradient for z_1, z_4

d.1

First set up the gradient for last layer:

$$\frac{\partial y}{\partial W_{(O,2)}} = z_2 \cdot \frac{\partial z_2}{z} = z_2 = 3 \tag{1}$$
$$\frac{\partial y}{\partial W_{(O,3)}} == z_3 \cdot \frac{\partial z_3}{z} = z_3 = 1 \tag{2}$$
$$\frac{\partial y}{\partial b_O} = 1 \tag{3}$$

Therefore, the gradient matrix for last layer is:

$$\nabla_{last} = \begin{bmatrix} 0 & 3 & 1 & 0 & 1 \end{bmatrix} \tag{4}$$

Then set up the gradient for first layer:

$$\frac{\partial y}{\partial W_{H,2}} = \frac{\partial y}{\partial \bar{z_2}} \cdot \frac{\partial \bar{z_2}}{\partial W_{H,2}} \tag{1}$$
$$\frac{\partial y}{\partial W_{H,3}} = \frac{\partial y}{\partial \bar{z_3}} \cdot \frac{\partial \bar{z_3}}{\partial W_{H,3}} \tag{2}$$
$$\frac{\partial y}{\partial b_{H,2}} = \frac{\partial y}{\partial \bar{z_2}} \cdot \frac{\partial \bar{z_2}}{\partial b_{(H,2)}} \tag{3}$$
$$\frac{\partial y}{\partial b_{H,3}} = \frac{\partial y}{\partial \bar{z_3}} \cdot \frac{\partial \bar{z_3}}{\partial b_{(H,3)}} \tag{4}$$

Compute the partial derivatives, we have:

$$\frac{\partial y}{\partial W_{(H,2)}} = W_{(O,2)} \cdot \frac{\partial max(0, z_2)}{\partial z_2} \cdot x = -2 \tag{1}$$
$$\frac{\partial y}{\partial W_{(H,3)}} = W_{(O,3)} \cdot \frac{\partial max(0, z_3)}{\partial z_3} \cdot x = -2 \tag{2}$$
$$\frac{\partial y}{\partial b_{H,2}} = W_{(O,2)} \cdot 1 = -1 \tag{3}$$
$$\frac{\partial y}{\partial b_{H,3}} = W_{(O,3)} \cdot 1 = -1 \tag{4}$$

Therefore, the gradient matrix for the first layer is:

$$\nabla_{first} = \begin{bmatrix} 0 & -2 & -2 & 0 \\ 0 & -1 & -1 & 0 \end{bmatrix} \tag{5}$$

d.2

Gradient for output layer:

$$\frac{\partial L}{\partial y} = 2(y_{actual} - y_{output}) = 2(y - (-3)) = 2y + 6 \tag{1}$$
$$\frac{\partial L}{\partial W_{(O,2)}} = \frac{\partial L}{\partial y} * z_2 = (2y + 6) * 3 \tag{2}$$
$$\frac{\partial L}{\partial W_{(O,3)}} = \frac{\partial L}{\partial y} * z_3 = (2y + 6) * 1 \tag{3}$$
$$\frac{\partial L}{\partial b_O} = \frac{\partial L}{\partial y} * 1 = 2y + 6 \tag{4}$$

Gradient for hidden layer:

$$\frac{\partial L}{\partial z_2} = \frac{\partial L}{\partial y} * W_{(O,2)} = (2y+6) * -1 \tag{1}$$
$$\frac{\partial L}{\partial z_3} = \frac{\partial L}{\partial y} * W_{(O,3)} = (2y+6) * -1 \tag{2}$$
$$\frac{\partial z_2}{\partial W_{(H,2)}} = \frac{\partial max(0,x)}{\partial x} \cdot \frac{\partial z_2}{\partial W_{(H,2)}} = x = 2 \tag{3}$$
$$\frac{\partial z_3}{\partial W_{(H,3)}} = \frac{\partial max(0,x)}{\partial x} \cdot \frac{\partial z_3}{\partial W_{(H,3)}} = x = 2 \tag{4}$$

Multiply:

$$\frac{\partial L}{\partial W_{(H,2)}} = \frac{\partial L}{\partial z_2} \cdot \frac{\partial z_2}{\partial W_{(H,2)}} = (-2y-6) * 2 \tag{1}$$
$$\frac{\partial L}{\partial W_{(H,3)}} = \frac{\partial L}{\partial z_3} \cdot \frac{\partial z_3}{\partial W_{(H,3)}} = (-2y-6) * 2 \tag{2}$$
$$\frac{\partial L}{\partial b_{(H,2)}} = \frac{\partial L}{\partial z_2} \cdot \frac{\partial z_2}{\partial W_{(H,2)}} = 2y+6 \tag{3}$$
$$\frac{\partial L}{\partial b_{(H,3)}} = \frac{\partial L}{\partial z_3} \cdot \frac{\partial z_3}{\partial W_{(H,3)}} = 2y+6 \tag{4}$$

3

a

For each entry (i,j) in X^TX, the cooresponding value should be $< x_i, x_j >$. Therefore, from the given sqaured Eculidean distance matrix D, $D_{ij} = ||x_i||_2^2 + ||x_j||_2^2 - 2 < x_i, x_j >$ If we assume that $x_1 = 0$, then $||x_i||_2^2 = ||x_i - x_1||_2^2 = D_{i1}$. Same for x_j. Therefore, rearrange $D_{ij} = ||x_i||_2^2 + ||x_j||_2^2 - 2 < x_i, x_j >$, we have $(X^TX)_{ij} =< x_i, x_j >= \frac{D_{i1}+D_{j1}-D_{ij}}{2}$. In this approach, we can reconstruct every entry in X^TX

b

Suppose $X = U\Sigma V^T$ where U, V^T are the left and right singular matrix of X with respect to the eigenvalue matrix Σ. In the same manner, $X^TX = U(\Sigma)^2V^T$ where Σ^2 is the square of the eigenvalue matrix Σ. Therefore, we can simply take the square root of the computed eigenvalue matrix to derive $U\Sigma V^T$. To find the k loading vectors, we just take first k columns of U, denote as U_k; first k squares of Σ, denote as Σ_k; first k rows of V^T, denote as V_k^T. Then, we have the desired k loading vectors $Z = XV_k^T = U_k\Sigma_k$ Thus, without knowing any information of X, we can use the SVD information derived from X^TX to find the k loading vectors of X.

c

When $k = d$, we use all the eigenvalues as the loading vectors without reducing the original dimension. In this case, we have $z_i = Vx_i$. Rewrite $||z_i - z_j||_2^2$, we have

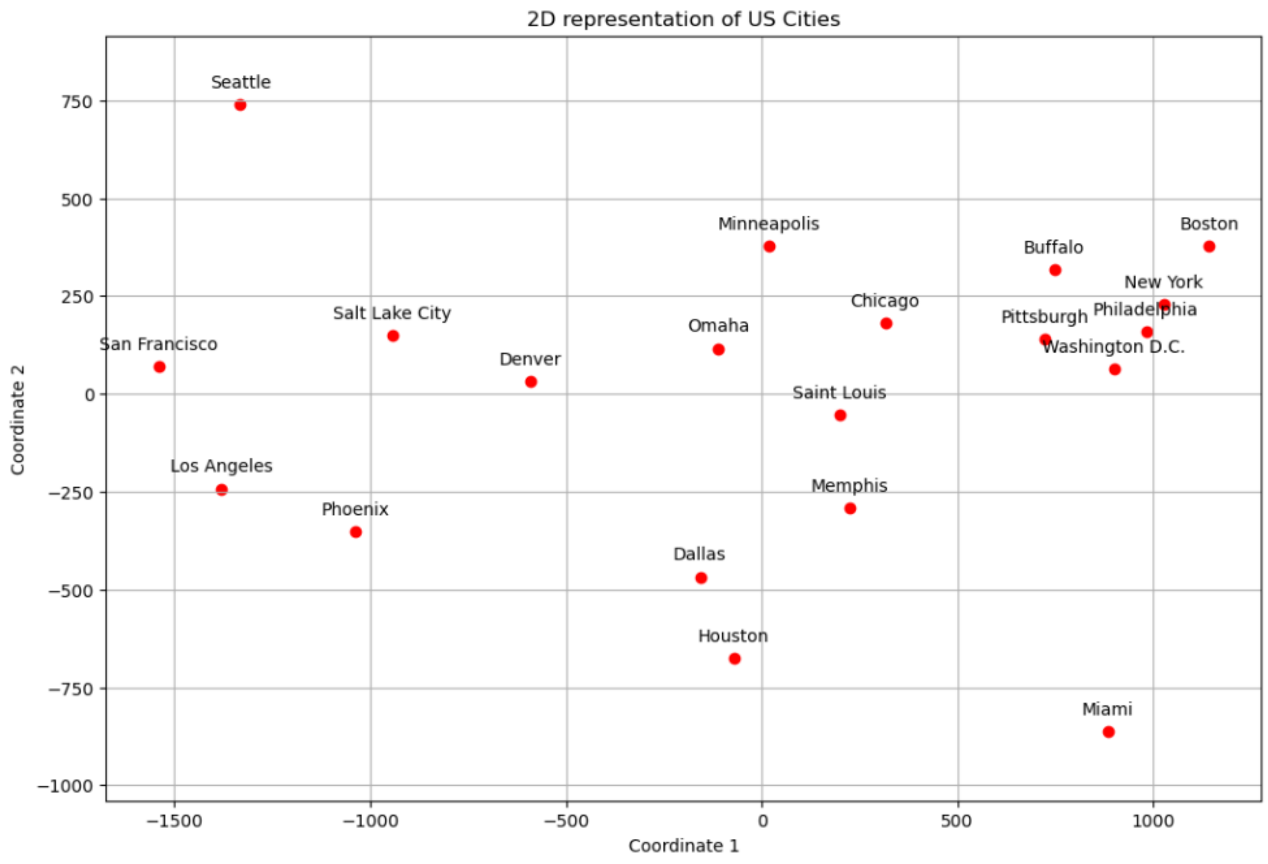
$$\begin{aligned}
||z_i - z_j||_2^2 &= ||V(x_i - x_j)||_2^2 && (1)\\
&= (V(x_i - x_j))^T(V(x_i - x_j)) && (2)\\
&= (x_i - x_j)^T V^T V(x_i - x_j) && (3)
\end{aligned}$$

Since V is orthogonal

$$\begin{aligned}
&= (x_i - x_j)^T I(x_i - x_j) && (4)\\
&= ||x_i - x_j||_2^2 && (5)
\end{aligned}$$

d

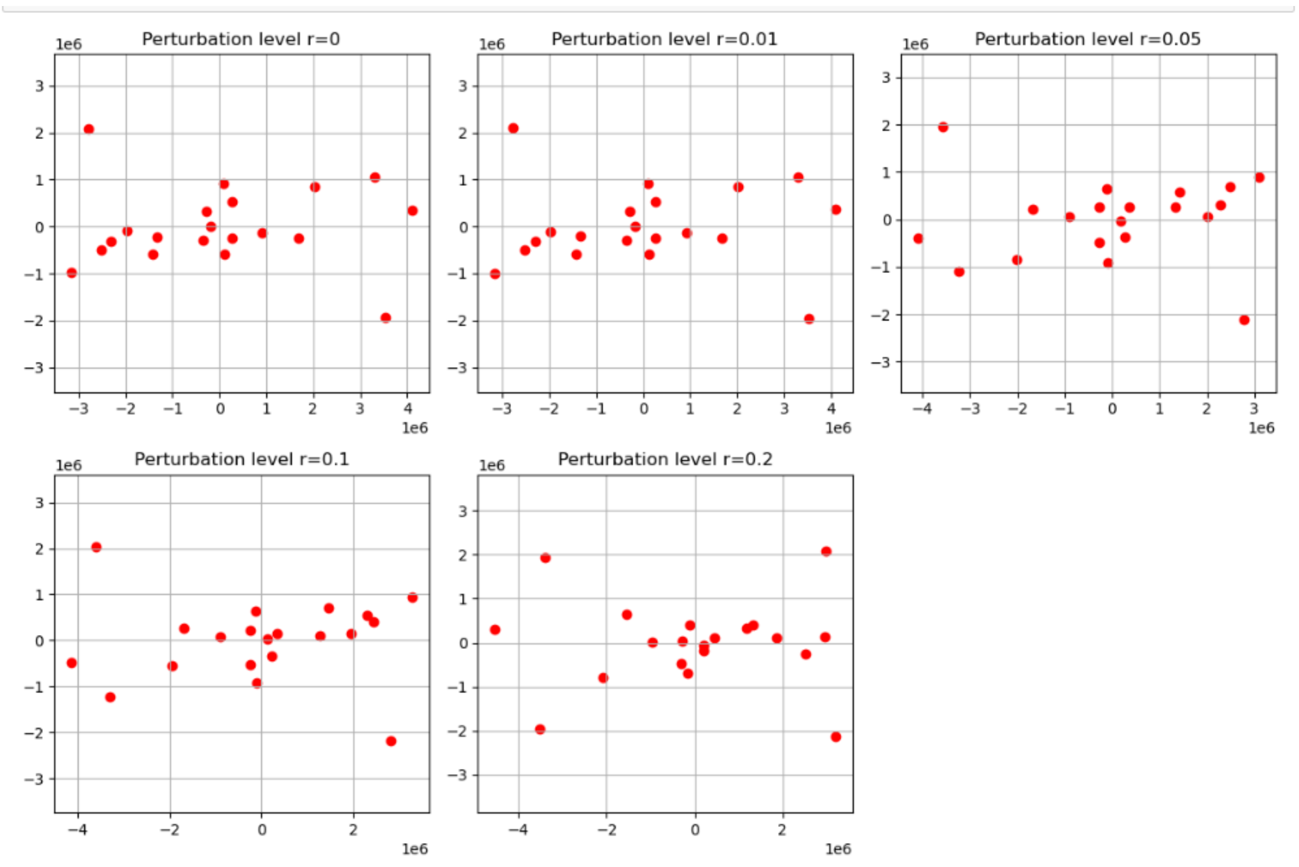
This is the result of my plot:



The code is attached to a separate notebook file.

e

From the following graph, we can see that when perturb level goes higher than 0.1, the original relative position is dramatically changes and loses representativeness.


Perturbation level r=0
Perturbation level r=0.01
Perturbation level r=0.05
Perturbation level r=0.1
Perturbation level r=0.2
1e6