ML Homework 4 Solutions

Ahuva Bechhofer

April 24, 2023

1 From distances to embeddings

1.1 i

i) let $f = \sum_{i,j} \left(\|x_i - x_j\| - D_{ij}\right)^2$ $\quad \frac{\partial}{\partial x}\|x-a\|_2 = \frac{x-a}{\|x-a\|_2}$ Consider terms where xi is fixed, we can separate the summation into two summations one over i and one over j. The contents of the summations are identical therefore was can compute the derivative for one of them and then we will have 2 times our final answer by symmetry.

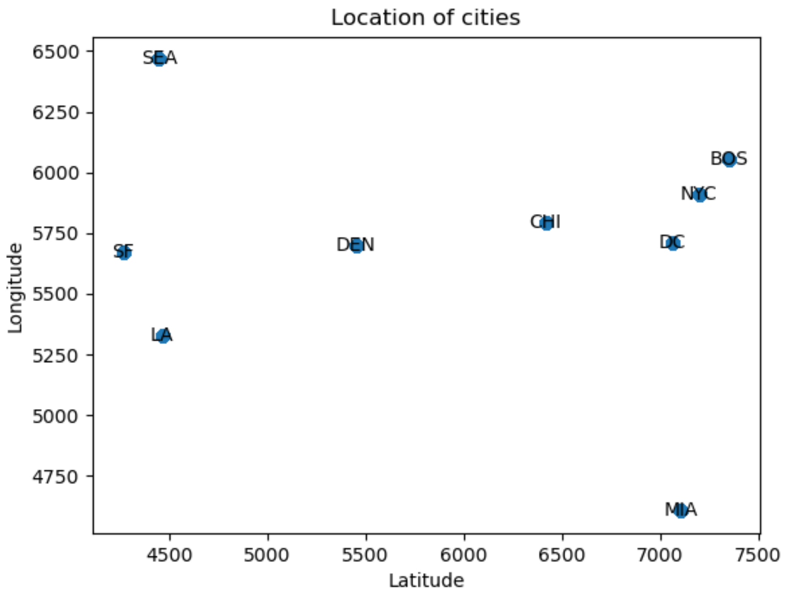
$$\frac{df}{dx_i} = \sum_j 2\left(\|x_i - x_j\| - D_{ij}\right) \cdot \frac{x_i - x_j}{\|x_i - x_j\|} \rightarrow \frac{\|x_i - x_j\|\left(x_i - x_j\right) - D_{ij}\left(x_i - x_j\right)}{\|x_i - x_j\|}$$

Pull the 2 out of the summation

$$\frac{d}{dx_i} = 4\sum_j \left(x_i - x_j - \frac{D_{ij}\left(x_i - x_j\right)}{\|x_i - x_j\|}\right)$$

1.2 ii-code

1.3 iii



The distances between the cities are pretty accurate, and the locations plotted on the map are relatively close to where the cities are on the actual map in this run of the code. In other runs I got some rotations of this map but the distance between the cities is always preserved because the only known data is the relative distance between the cities.

2 Kernelized SVM vs Neural Networks: Theory and Empirics

2.1 a

Here we are minimizing in terms of the slope which has a regularization effect. This means as we minimize the flatter the slope gets, and a flatter slope which still fits all the points, is less susceptible to overfitting.

2.2 b

Adding the slack variables allows us to access points which are beyond our boundary. ξ_i is expanding the boundary for points that high up above the line, and γ_i is expanding the boundary for points that are low below the line.

2.3 c

$S_i\{w, b, \xi_i, \gamma_i\}, \lambda_i\{\alpha_i, \beta_i, \delta_i, \sigma_i\}$

$$L\left(S_i, \lambda_i\right) = \frac{1}{2}w^2 + c\sum_{i=1}^{N}\left(\xi_i + \gamma_i\right) + \sum_{i=1}^{N}\alpha_i\left(y_i - \left(wx_i + b\right) - \epsilon + \xi_i\right)$$

$$+ \sum_{i=1}^{N}\beta_i\left(\left(wx_i + b\right) - y_i - \epsilon + \gamma_i\right)$$

$$+ \sum_{i=1}^{N}\delta_i\left(-\xi_i\right)$$

$$+ \sum_{i=1}^{N}\sigma_i\left(-\gamma_i\right)$$

$$d* = \max_{\lambda_i \geq 0}\min_{S_i} L(S_i, \lambda_i)$$

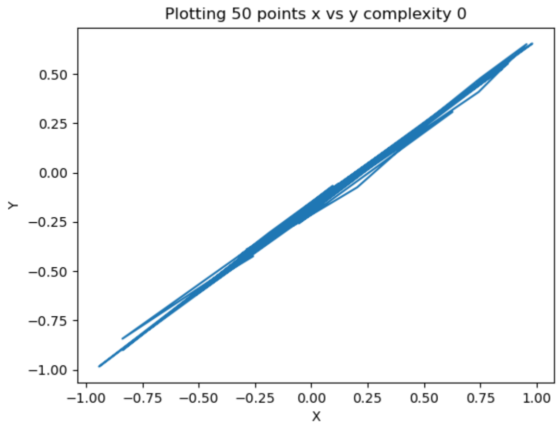
$$\frac{d}{dw}L\left(S_i, \lambda_i\right) = w + \sum_{i=1}^{N} -\alpha_i x_i + \sum_{i=1}^{N}\beta_i x_i = 0$$

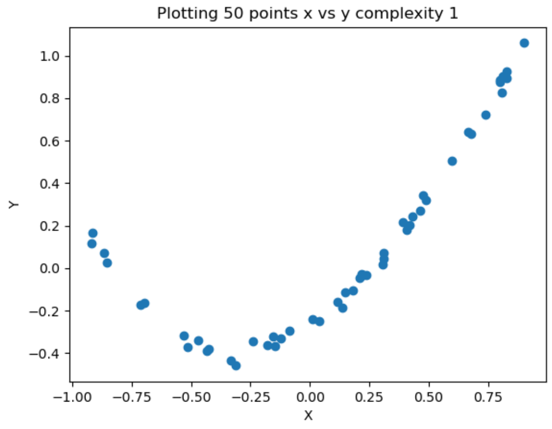
$$w = -\sum_{i=1}^{N} -\alpha_i x_i - \sum_{i=1}^{N}\beta_i x_i = \sum_{i=1}^{N}\left(\alpha_i - \beta_i\right)x_i$$

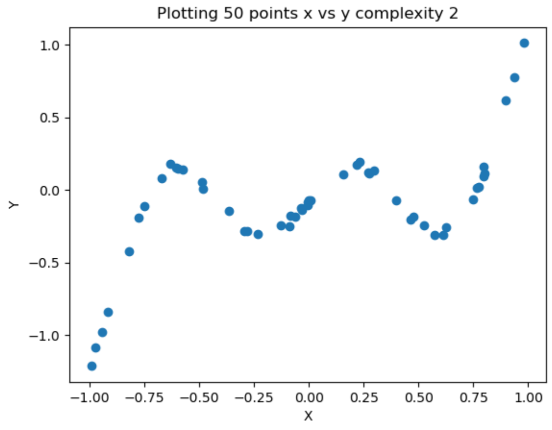
$$f(x) = wx + b$$

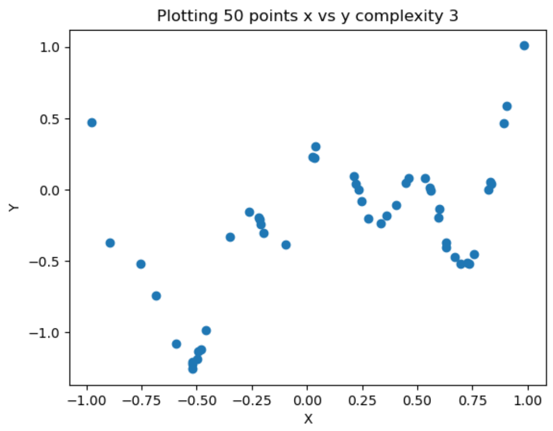
$$f(x) = \sum_{i=1}^{N}\left(\alpha_i - \beta_i\right)x_i x + b$$

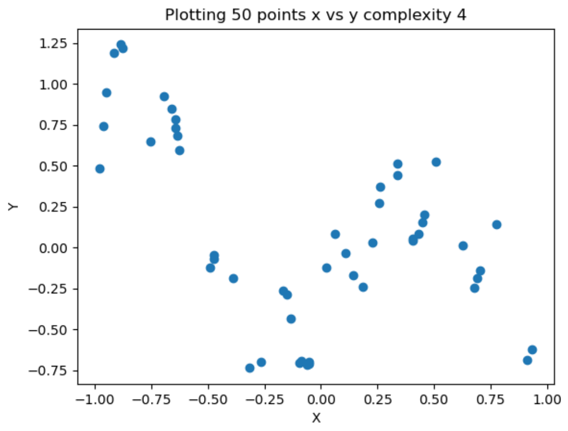
2.4 i

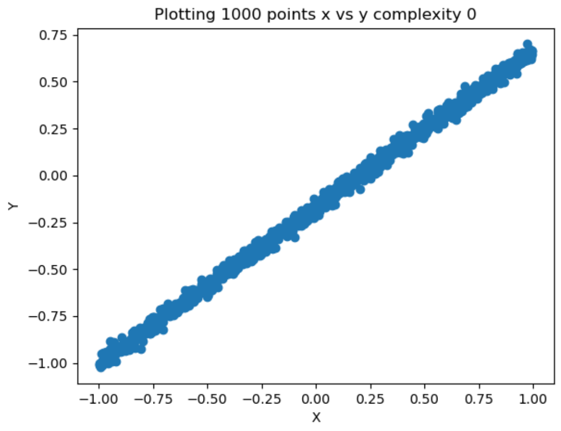

Plotting 50 points x vs y complexity 0
Y
X

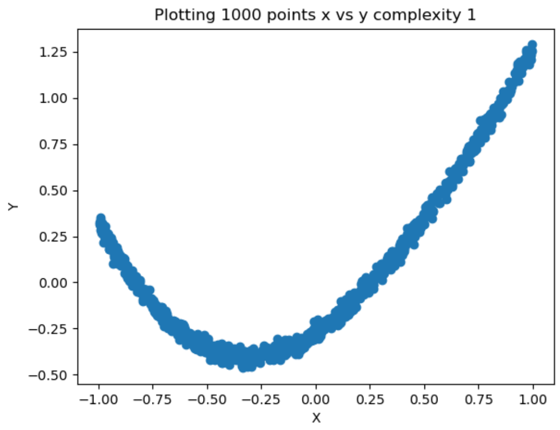

Plotting 50 points x vs y complexity 1
Y
X

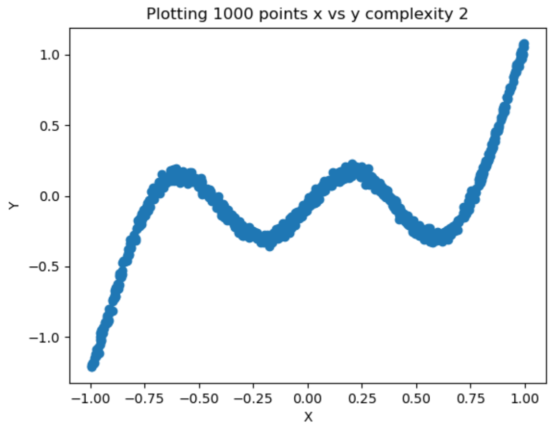

Plotting 50 points x vs y complexity 2
Y
X

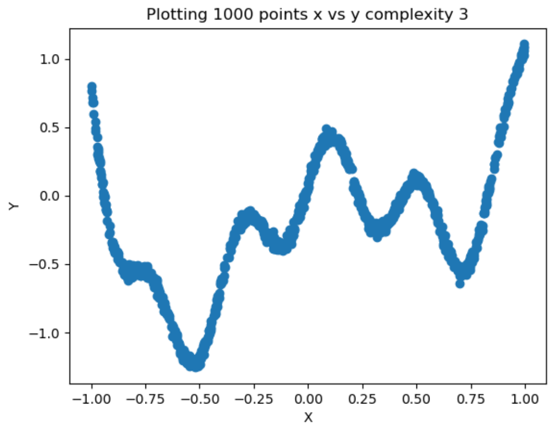

Plotting 50 points x vs y complexity 3
Y
X
1.0
0.5
0.0
−0.5
−1.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

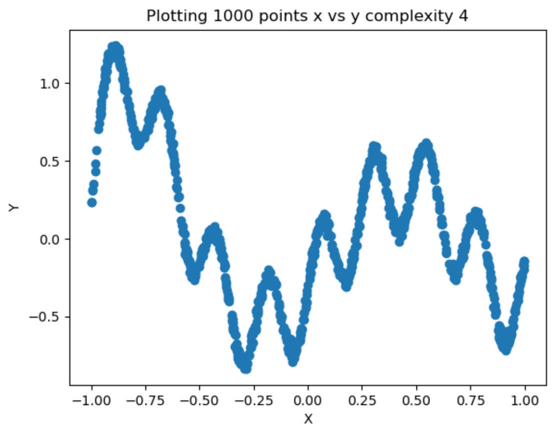

Plotting 50 points x vs y complexity 4
Y
X
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Plotting 1000 points x vs y complexity 0
Y
X
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Plotting 1000 points x vs y complexity 1
Y
X
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Plotting 1000 points x vs y complexity 2
Y
X
1.0
0.5
0.0
−0.5
−1.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

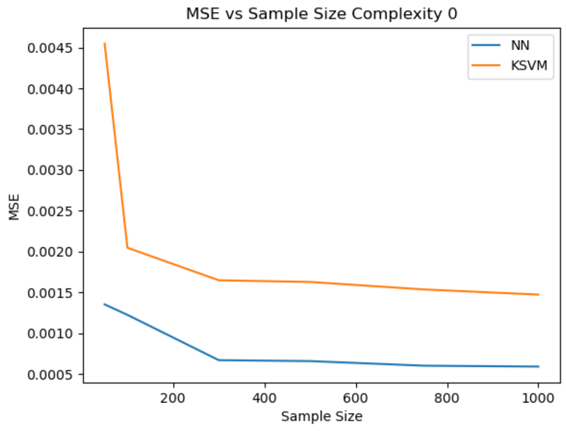

Plotting 1000 points x vs y complexity 3
Y
X
1.0
0.5
0.0
−0.5
−1.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

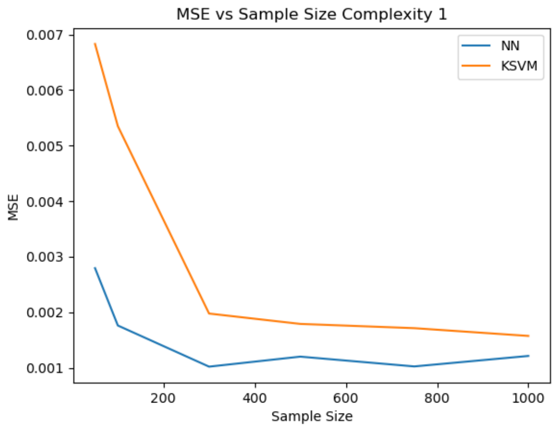


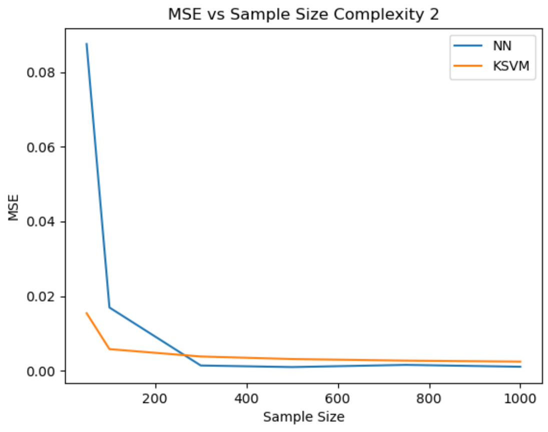
2.5 j-code

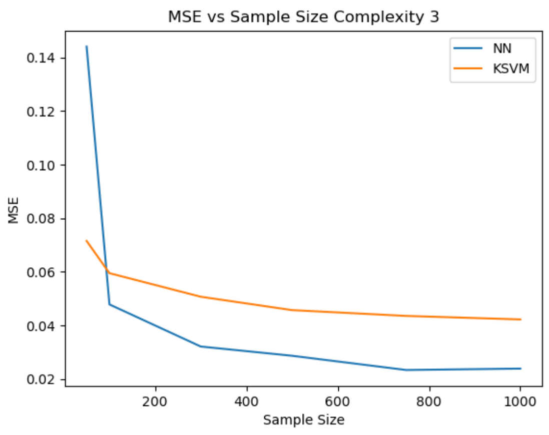
citation: https://pytorch.org/tutorials/beginner/blitz/cifar10$_t utorial.html$
$https : //pytorch.org/tutorials/beginner/basics/data_t utorial.html$

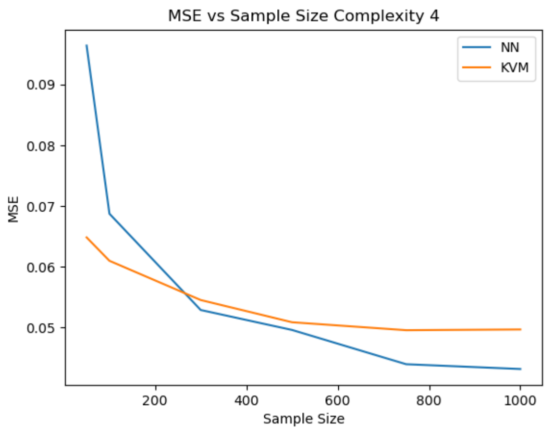
2.6 k


MSE vs Sample Size Complexity 0
NN
KSVM
MSE
0.0045
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
200
400
600
800
1000
Sample Size


MSE vs Sample Size Complexity 1
NN
KSVM
MSE
0.007
0.006
0.005
0.004
0.003
0.002
0.001
200
400
600
800
1000
Sample Size


MSE vs Sample Size Complexity 2
NN
KSVM
MSE
0.08
0.06
0.04
0.02
0.00
200
400
600
800
1000
Sample Size





2.7 l

From the graphs we can see that a) as the training size increases the MSE decreases, b) the NN is always preforming better than the KSVM therfore it is a better model and C) the KSVM is getting better as the complexity of the function increases. This is most likely due to the fact the KSVM works better for more complex functions as it can now apply the complex kernelization functions which it could not on the less complex data. And overall as the complexity increases the MSE is also increasing as the functions are getting more complex and harder to predict.