ML HW4

Maitar Asher

04/23/2023

1 From distances to embeddings

(i)

We need to account for all the different values i,j can take when taking the derivative.

$$\sum_{i,j}(||\vec{x}_i-\vec{x}_j||-D_{ij})^2 = \sum_{i=j}(||\vec{x}_i-\vec{x}_j||-D_{ij})^2+\sum_{i\neq j,i<j}(||\vec{x}_i-\vec{x}_j||-D_{ij})^2+\sum_{i\neq j,i>j}(||\vec{x}_i-\vec{x}_j||-D_{ij})^2$$

$$= 2\sum_{i\neq j}(||\vec{x}_i-\vec{x}_j||-D_{ij})^2$$

$$\frac{\partial\left(2\sum_{i,j\forall i\neq j}(||\vec{x}_i-\vec{x}_j||-D_{ij})^2\right)}{\partial(\vec{x}_i)} = \frac{\partial\left(2\sum_{i,j\forall i\neq j}((\vec{x}_i-\vec{x}_j)\cdot(\vec{x}_i-\vec{x}_j))^{\frac{1}{2}}-D_{ij})^2\right)}{\partial(\vec{x}_i)}$$

$$= 4\sum_{i,j}(||\vec{x}_i-\vec{x}_j||-D_{ij})\frac{1}{2}((\vec{x}_i-\vec{x}_j)\cdot(\vec{x}_i-\vec{x}_j))^{-\frac{1}{2}}2(\vec{x}_i-\vec{x}_j)$$

$$= \frac{4\sum_{i,j}(||\vec{x}_i-\vec{x}_j||-D_{ij})(\vec{x}_i-\vec{x}_j)}{((\vec{x}_i-\vec{x}_j)\cdot(\vec{x}_i-\vec{x}_j))^{0.5}}$$

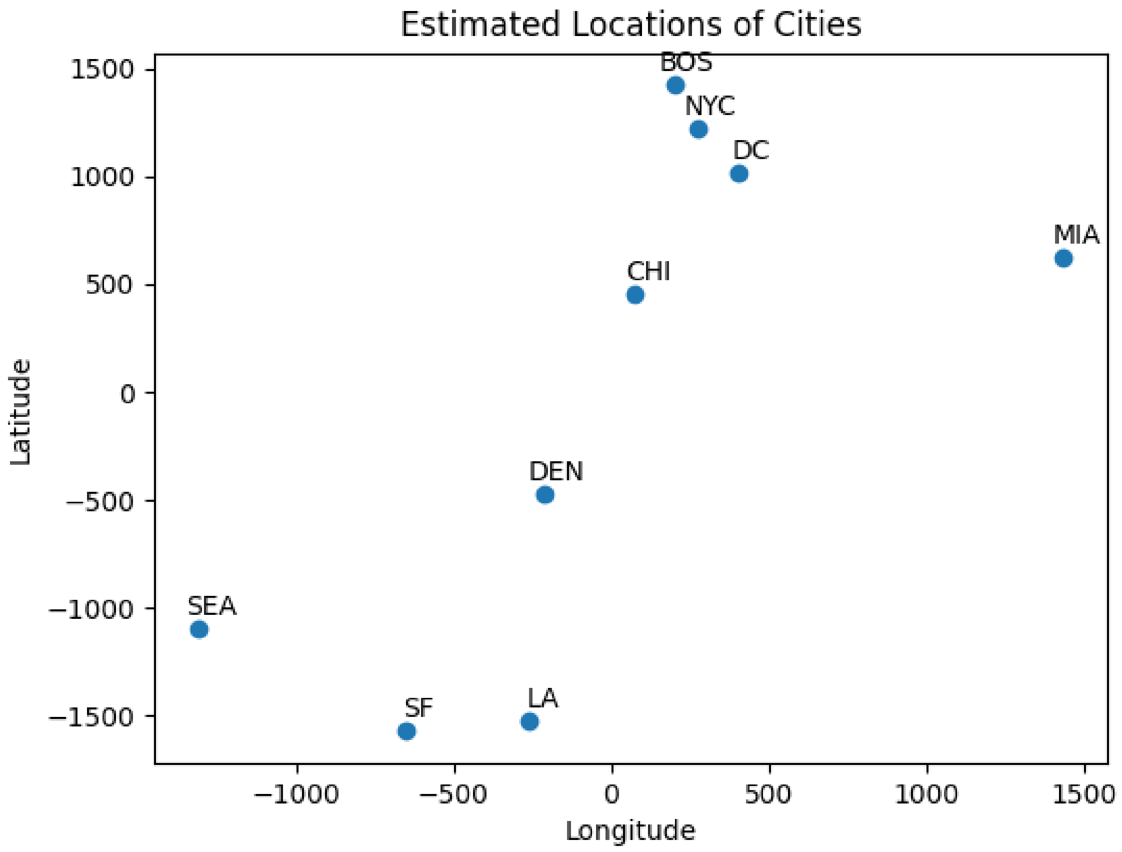
$$= \frac{4\sum_{i,j}(||\vec{x}_i-\vec{x}_j||-D_{ij})(\vec{x}_i-\vec{x}_j)}{||\vec{x}_i-\vec{x}_j||}$$

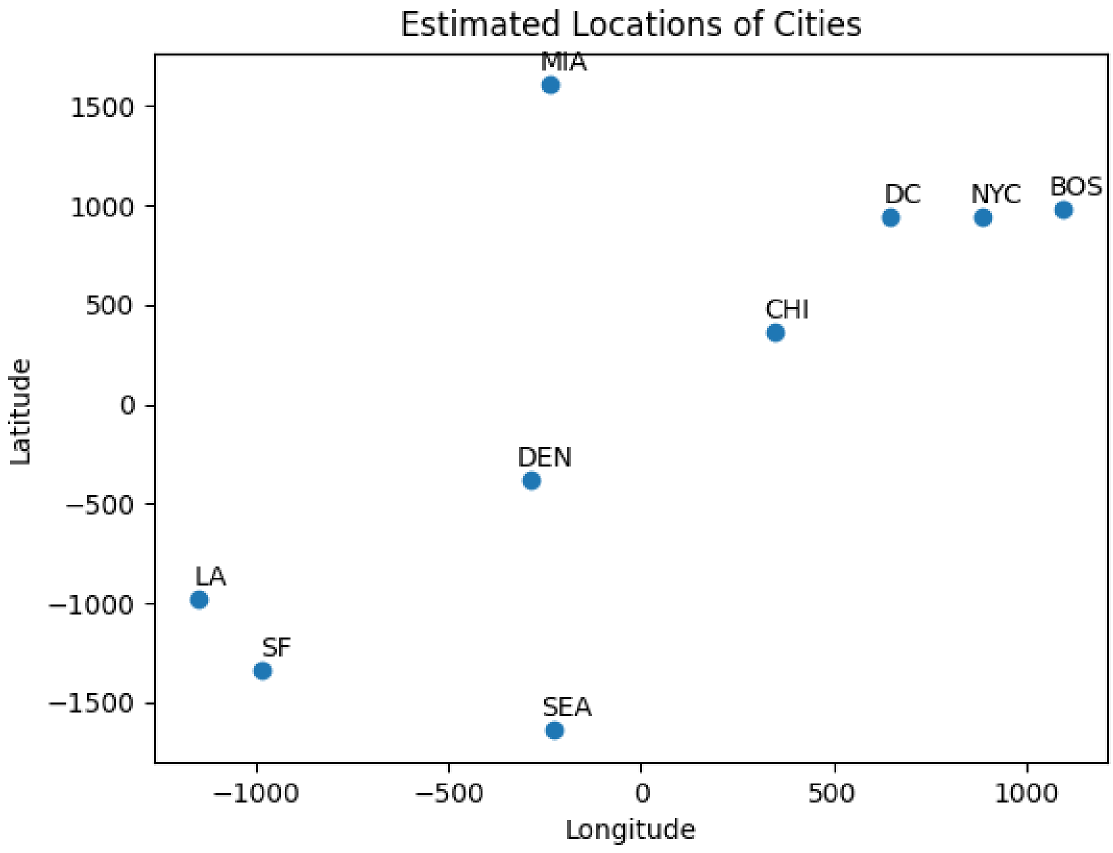
(ii)

Submitted via coursework

(iii)

When looking at the estimated locations, it appears that the relative distances between the cities are accurate and consistent with their actual geographical locations!




Estimated Locations of Cities
MIA
DC
NYC
BOS
CHI
DEN
LA
SF
SEA
1500
1000
500
0
−500
−1000
−1500
Latitude
−1000
−500
0
500
1000
Longitude

2 Kernelized SVM vs Neural Networks: Theory and Empirics

(a)

Part of the constraints in this optimization problem (ignoring the epsilon part) is essentially the sum of the least squares.

$$\forall i \in [n] : |y_i - (\vec{w} \cdot \vec{x_i} + b)| \leq \epsilon$$

Same as:

$$\forall i \in [n] : (y_i - (\vec{w} \cdot \vec{x_i} + b))^2 \leq \epsilon^2$$

If we were to put it in Lagrange we get:

$$L(\vec{w}, b, \vec{\lambda}) = \frac{1}{2}||\vec{w}||^2 + \sum_{i=1}^{n} \lambda_i (y_i - (\vec{w} \cdot \vec{x_i} + b))^2 - \epsilon^2$$

As we can see, this optimization problem is very similar to ridge regression. Our objective function is similar to the constraints in the ridge regression problem, whereas here we don't learn λ for $||\vec{w}||^2$ and it is constant ($\frac{1}{2}$). And the constraints in this problem are similar to the objective function in the ridge regression. This helps us see that our objective function is essentially acting as a regularizer parameter that ensures that no single feature or a few sets of features ($\vec{w}$,b) are going to contribute too much toward a particular problem. So the objective function helps prevents over-fitting and reduce complexity (when a model is too complex the hyperparameters can be too big, which leads to over-fitting).

(b)

The role of ξ_i and γ_i is to introduce flexibility in the constraints and address scenarios where points may be misclassified. Two slacks are used to cover both directions, i.e., when a point is either above or below the hyperplane. This approach allows for the consideration of the absolute difference between the predicted class and the true label.

(c)

$$L(\vec{w}, b, \vec{\xi}, \vec{\gamma}, \vec{\alpha}, \vec{\beta}) = \frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{n}(\xi_i + \gamma_i)$$

$$+ \sum_{i=1}^{n} (\alpha_i (y_i - (\vec{w} \cdot \vec{x_i} + b) - \epsilon - \xi_i)) + \sum_{i=1}^{n} (\beta_i (\vec{w} \cdot \vec{x_i} + b - y_i - \epsilon - \gamma_i))$$

Note:

* $\vec{\alpha}, \vec{\beta}$ are the Lagrange variables

** $\vec{w}, \vec{\xi}, \vec{\gamma}, \vec{\alpha}, \vec{\beta}$ are all vectors and the notation of γ_i for instance is the i's element in $\vec{\gamma}$

$$\frac{L(\vec{w}, b, \vec{\xi}, \vec{\gamma}, \vec{\alpha}, \vec{\beta})}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^{n}(\alpha_i \vec{x}_i) + \sum_{i=1}^{n}(\beta_i \vec{x}_i) = 0$$

$$\vec{w} = \sum_{i=1}^{n}(\alpha_i - \beta_i)\vec{x}_i$$

$$\frac{L(\vec{w}, b, \vec{\xi}, \vec{\gamma}, \vec{\alpha}, \vec{\beta})}{\partial b} = -\sum_{i=1}^{n}\alpha_i + \sum_{i=1}^{n}\beta_i = 0$$

So b can take any value.

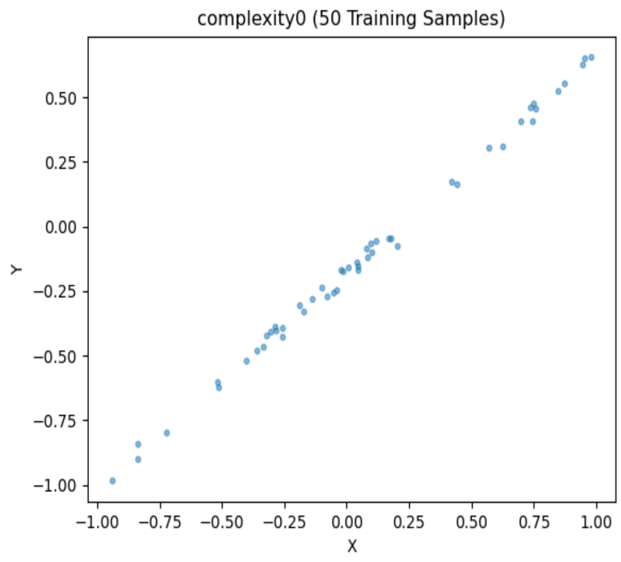
We are given that we aim to learn a predictor of the form $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$

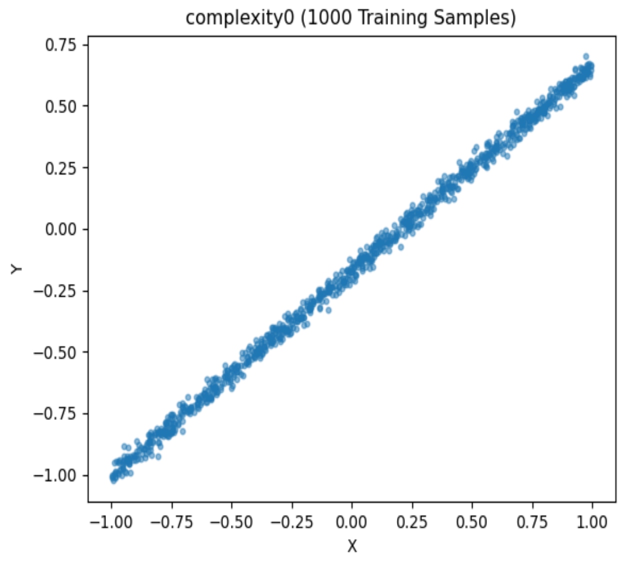
Plugging $\vec{w}$ in (it's a convex problem so this must be the optimal value) we get:

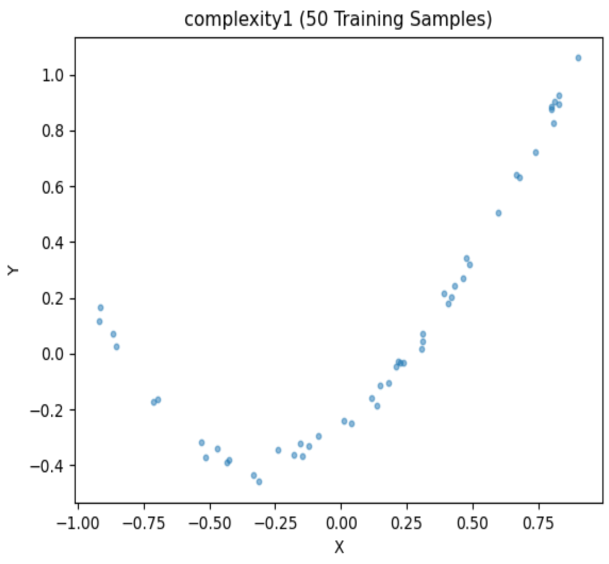
$$f(\vec{x}) = (\sum_{i=1}^{n}(\alpha_i - \beta_i)\vec{x}_i) \cdot \vec{x} + b$$

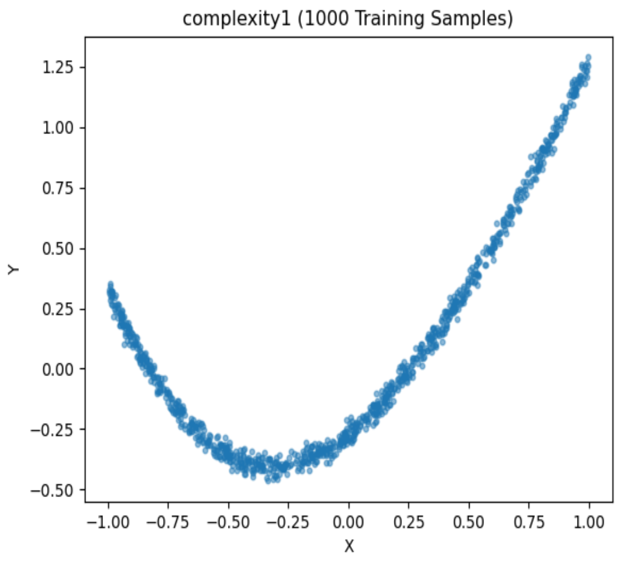
Comparing Empirical Performance of KSVMs and NNs:

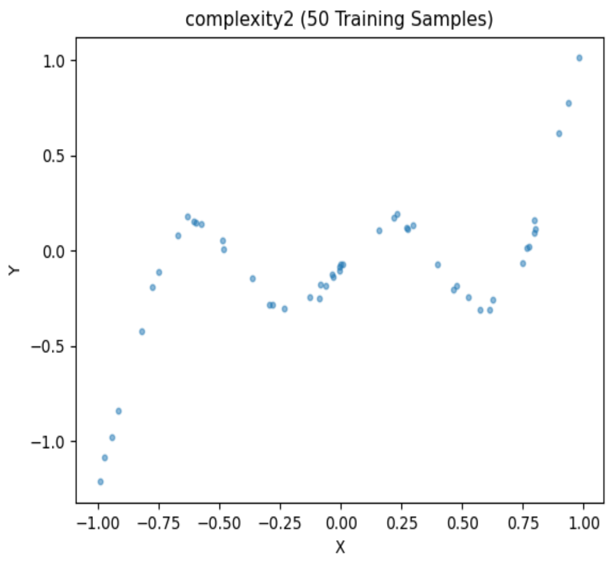
(i)

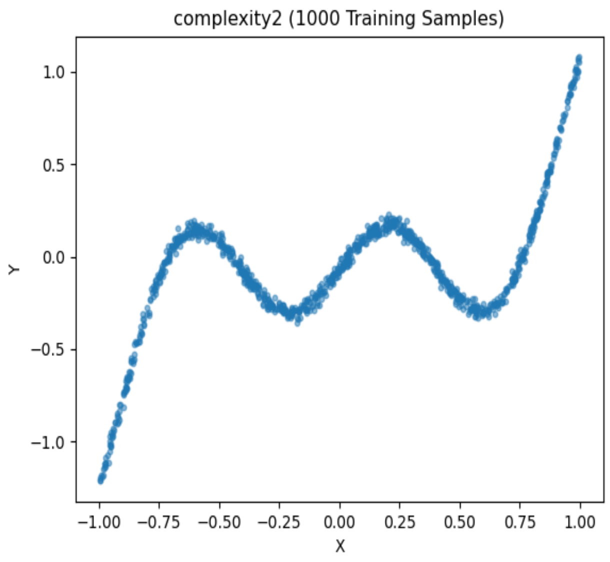


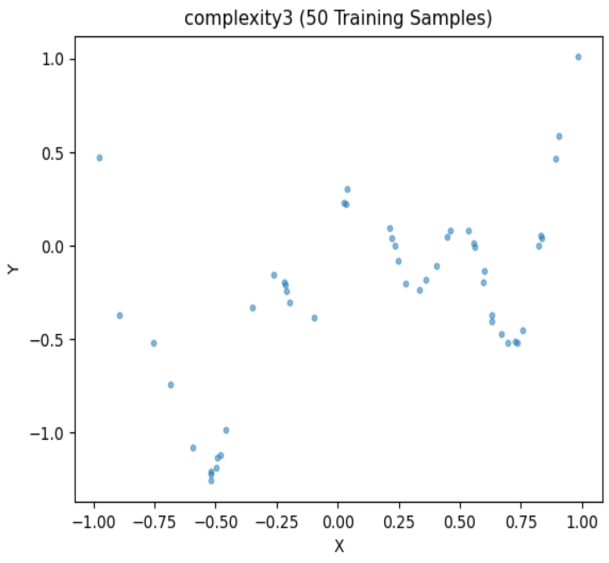


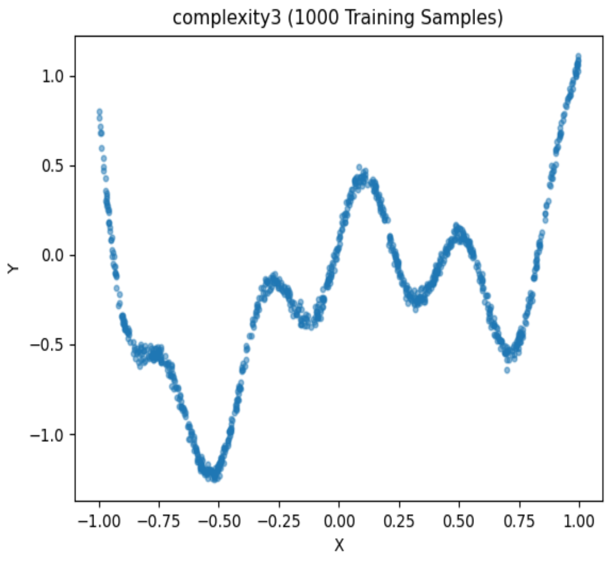

complexity1 (50 Training Samples)
Y
X
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75

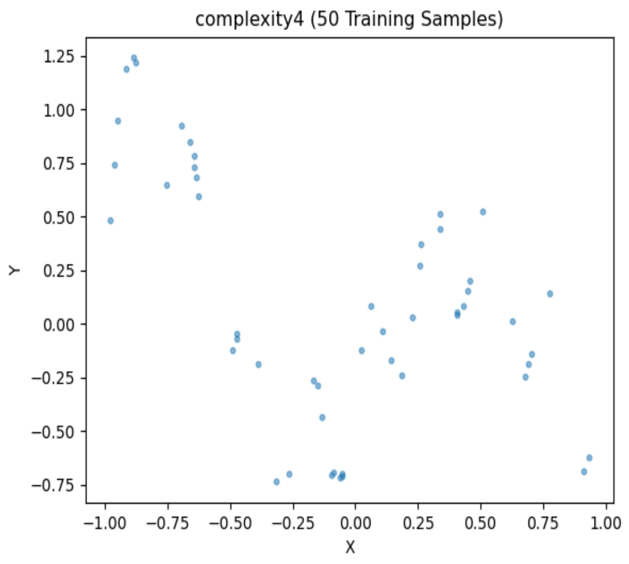

complexity1 (1000 Training Samples)
Y
X
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

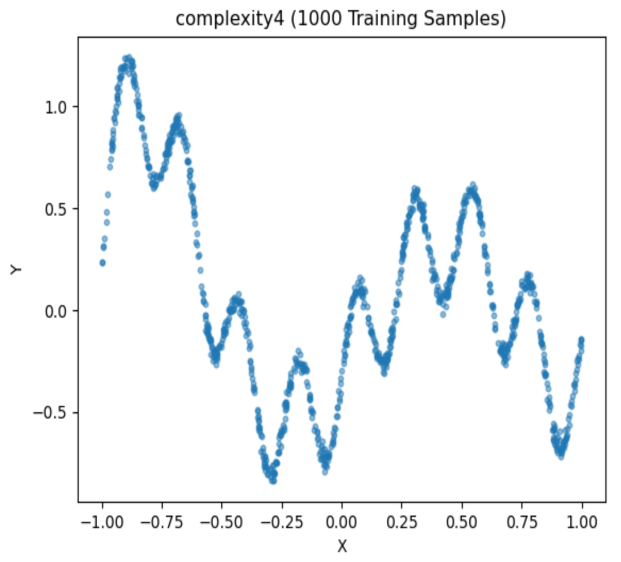

complexity2 (50 Training Samples)
Y
X
1.0
0.5
0.0
−0.5
−1.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


complexity2 (1000 Training Samples)
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


complexity3 (50 Training Samples)
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


complexity3 (1000 Training Samples)
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

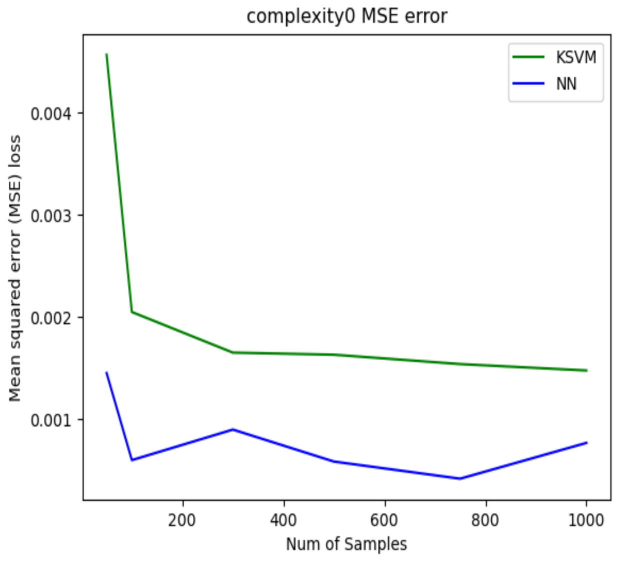

complexity4 (50 Training Samples)
X
Y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25

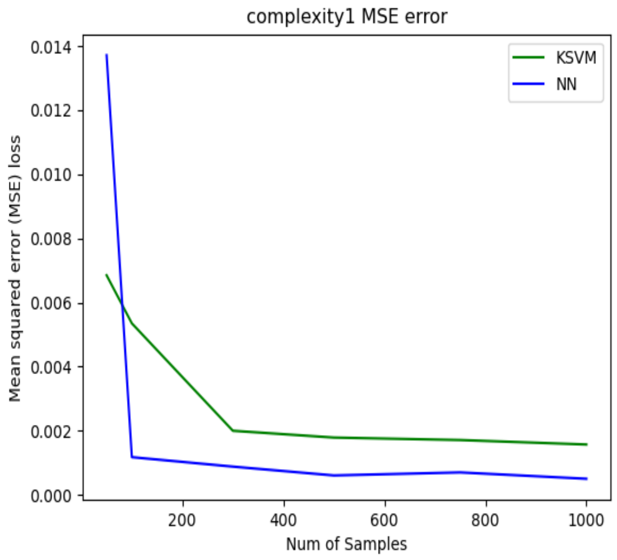

complexity4 (1000 Training Samples)
X
Y
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
−0.5
0.0
0.5
1.0

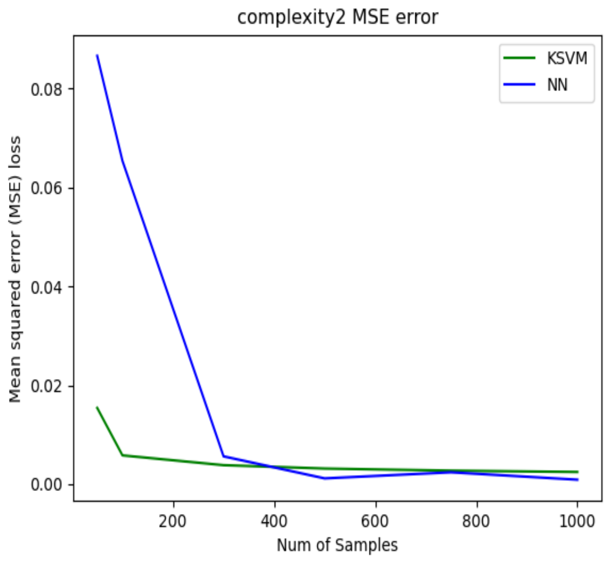
(j)

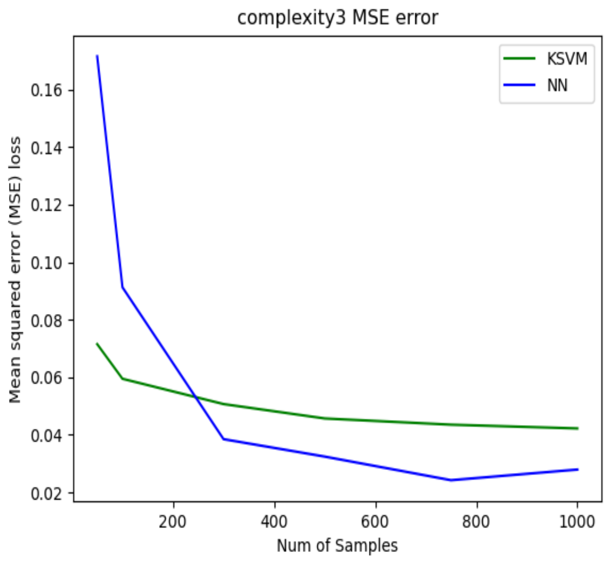
Submitted via coursework

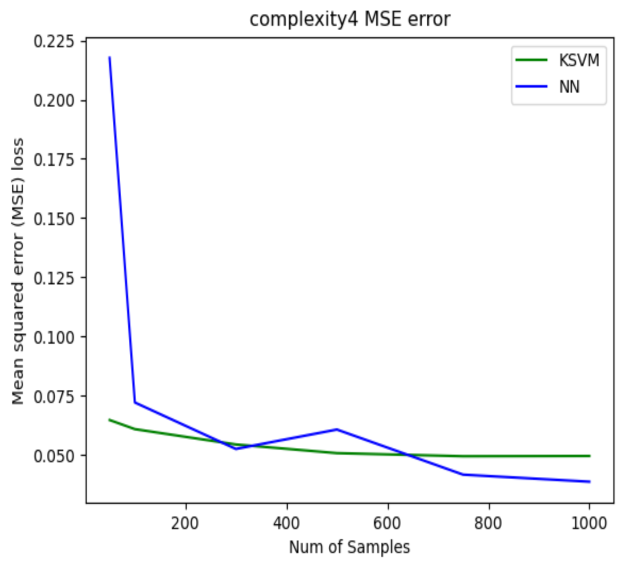
(k)






complexity2 MSE error
KSVM
NN
Mean squared error (MSE) loss
0.08
0.06
0.04
0.02
0.00
200
400
600
800
1000
Num of Samples


complexity3 MSE error
KSVM
NN
Mean squared error (MSE) loss
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
200
400
600
800
1000
Num of Samples



(1)

From the results, we can conclude that in general, the Neural Network model performed better than the Kernelized SVM model. The MSE rates for NNs were almost consistently lower than those of SVMs, indicating better accuracy. For both models, as the function complexity increased, the prediction quality decreased(higher error rates). This is because the models found it harder to capture the underlying patterns in the data when the functions became more complex. However, we can observe that the NN model needed less number of samples in order to learn the data, get a stable MSE rate and yield good predictions. As the sample size increased (more training data), both models were able to learn the data and the disparity between their error rates diminished.