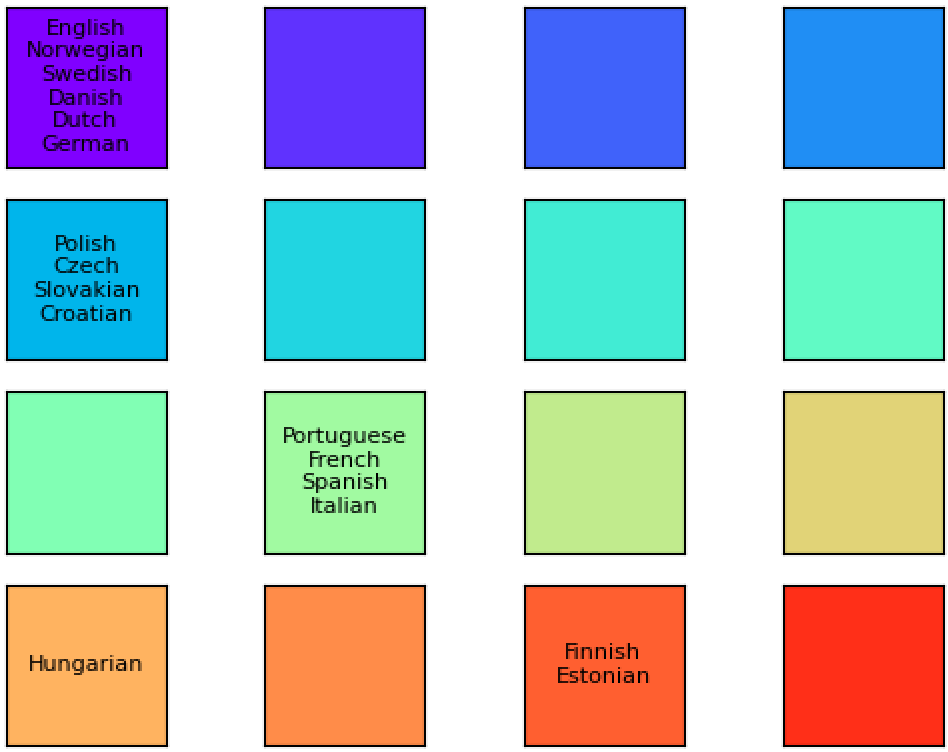
Predictive learning HW1

P1

Follow the instructions and use SOM, I get this result.



According the result, I marked the countries to anaylsis

We can find, the nearer countries would be classift to same class. It prove our experience is success. And each would be correspond to a country in Middle Ages.

In Iberic peninsula(伊比利半島) and Italy(義大利半島) , there is Portuguese Republic(葡萄牙王國) and Reino de España(西班牙王國) with green.

In North and Middle Europe, there are Kingdom Of France(法蘭西王國) and Kingdom of England(英格蘭王國) with purple.

In Middle Europe, there is Imperium Romanum(羅馬帝國) with blue.

In Hungary, there is Magyar Királyság(匈牙利王國) with orange.

P2

loss function : $L = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic}\log(p_{ic})$

y_{ic} : 0 or 1 if i is belong to c

p_{ic} : probability of i is belong to c

Output layer : $p_{ic} = f\left(\sum_i w_{ic}\, x_i + b_c\right)$

w_{ic}: weight from last layer neuron i to c class

x_i : Output from last layer neuron i

b_c: bias on class

$f(x)$: activation function

hidden layer : $h_j^{(L)} = f\left(\sum_i w_{ij} x_i + b_j\right)$

$h_j^{(L)}$: hidden neuron j on L layer

w_{ij} : weight from $L-1$ layer neuron i to L layer neuron j

x_i : Output from $L-1$ layer neuron i

b_j : Bias L layer neuron j

$f(x)$: activation function

Update Output layer w :

$$\frac{\partial L}{\partial w_{ic}} = \frac{\partial L}{\partial p_{ic}} \cdot \frac{\partial p_{ic}}{\partial z} \cdot \frac{\partial z}{\partial w_{ic}}$$

$$= \frac{y_{ic}}{p_{ic}} \cdot \frac{\partial p_{ic}}{\partial z} \cdot h_i^{(L)}$$

L : loss function

p_{ic} : prediction probability i on c

z : activation function

w_{ic} : weight on last layer to c

$h_i^{(L)}$: Output from last layer neuron i

$$\boxed{w_{ic} = w_{ic} - \eta \frac{\partial L}{\partial w_{ic}}}$$

η : learning rate

Update hidden layer w :

$$\frac{\partial L}{\partial w_{ij}^{(L-1)}} = \frac{\partial L}{\partial h_j^{(L)}} \cdot \frac{\partial h_j^{(L)}}{\partial z} \cdot \frac{\partial z}{\partial w_{ij}^{(L-1)}}$$

$w_{ij}^{(L-1)}$: weight from L-1 layer i neuron to L layer j neuron

$h_j^{(L)}$: the output from neuron j on layer L

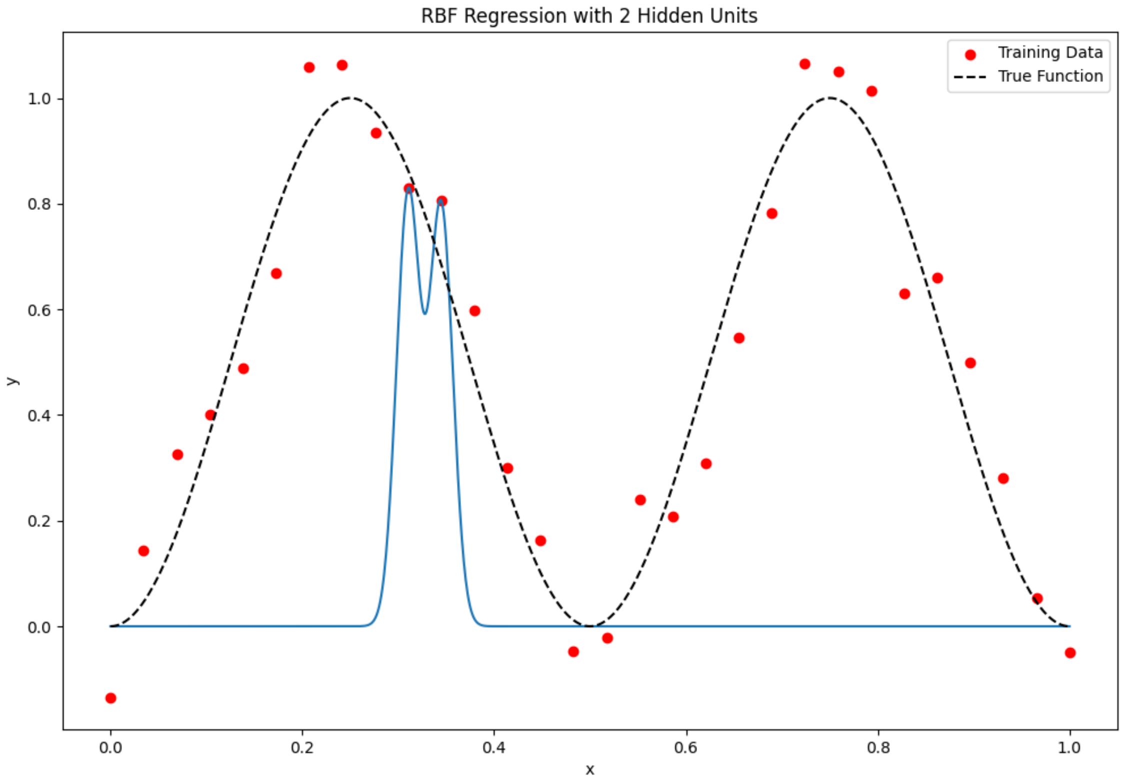
z : activation function

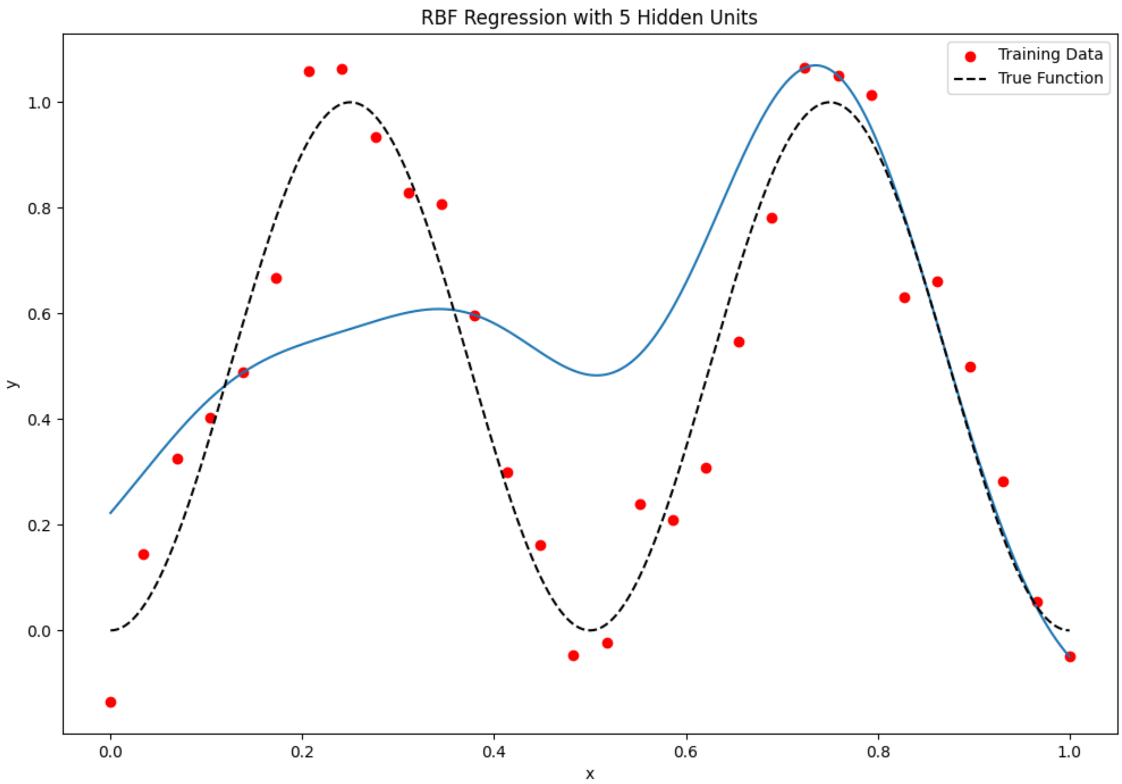
$$\boxed{w_{ij}^{(L-1)} = w_{ij}^{(L-1)} - \eta \frac{\partial L}{\partial w_{ij}^{(L-1)}}}$$

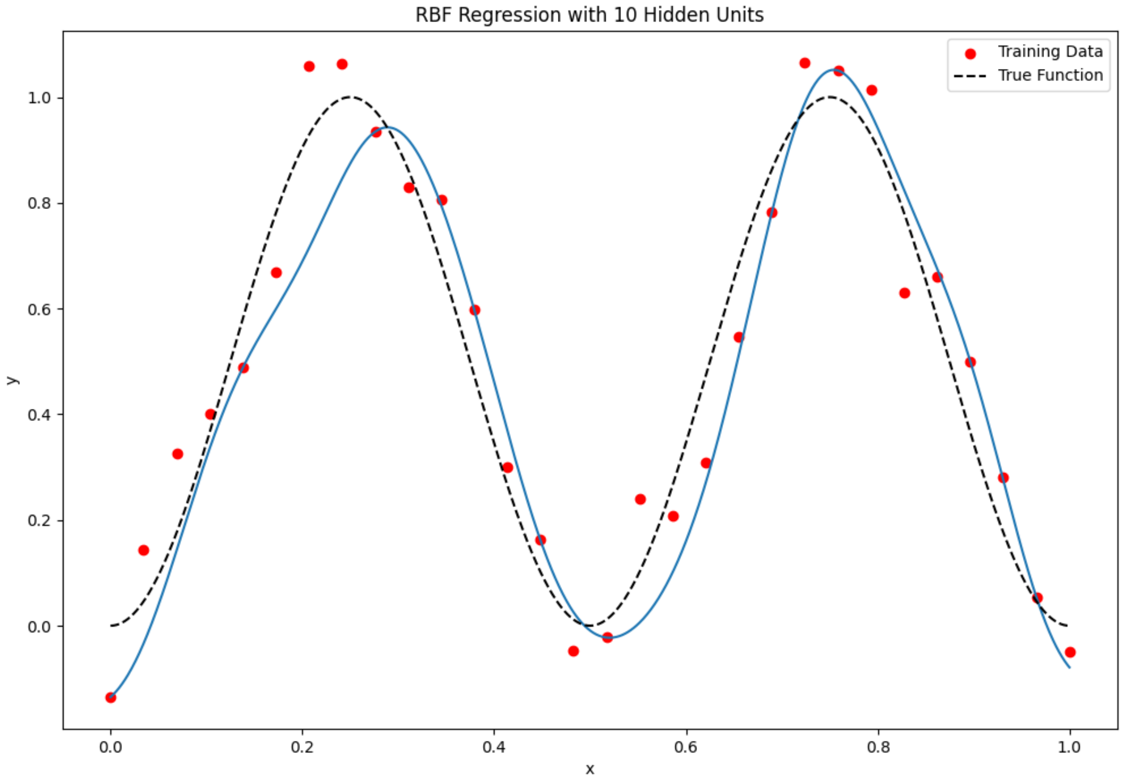
η : learning rate

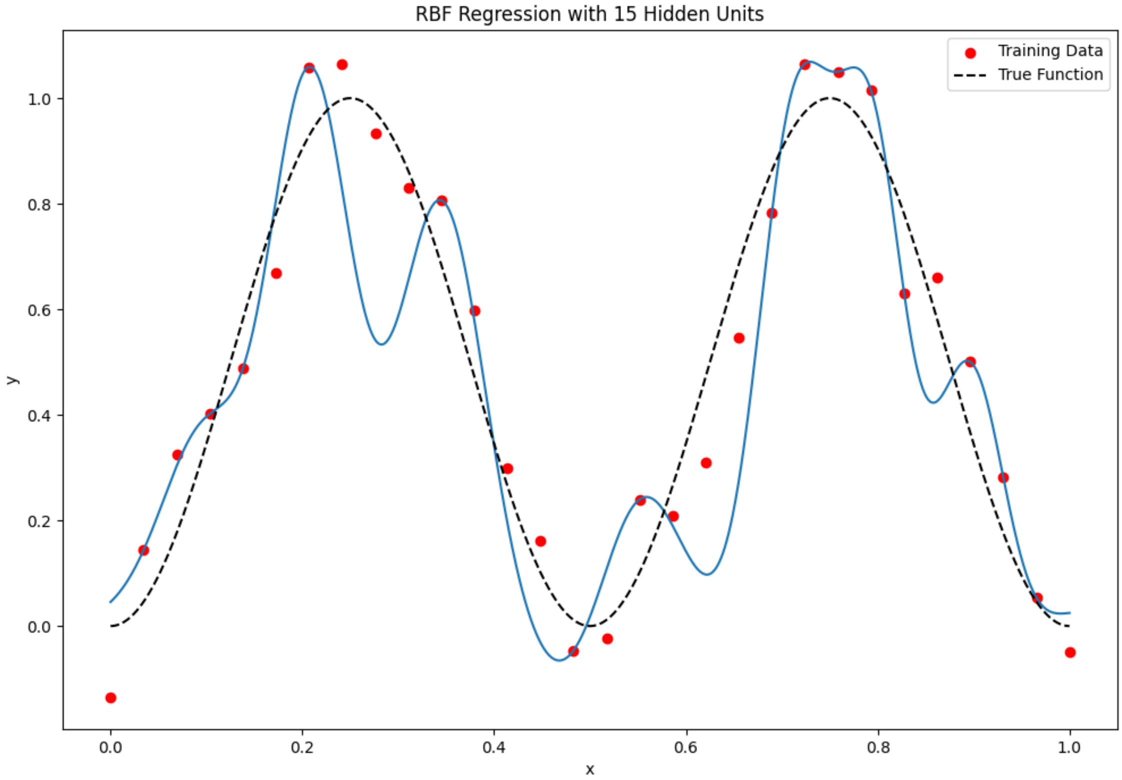
P3

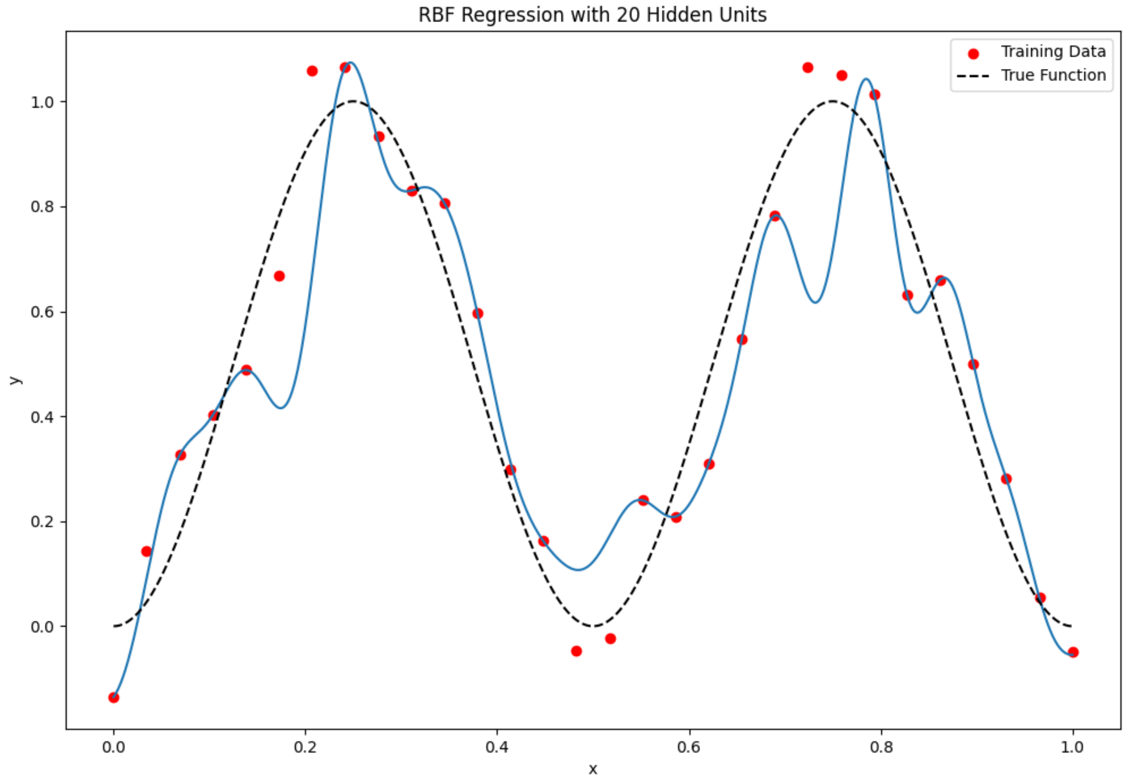
Follow the instruction, I get these result:

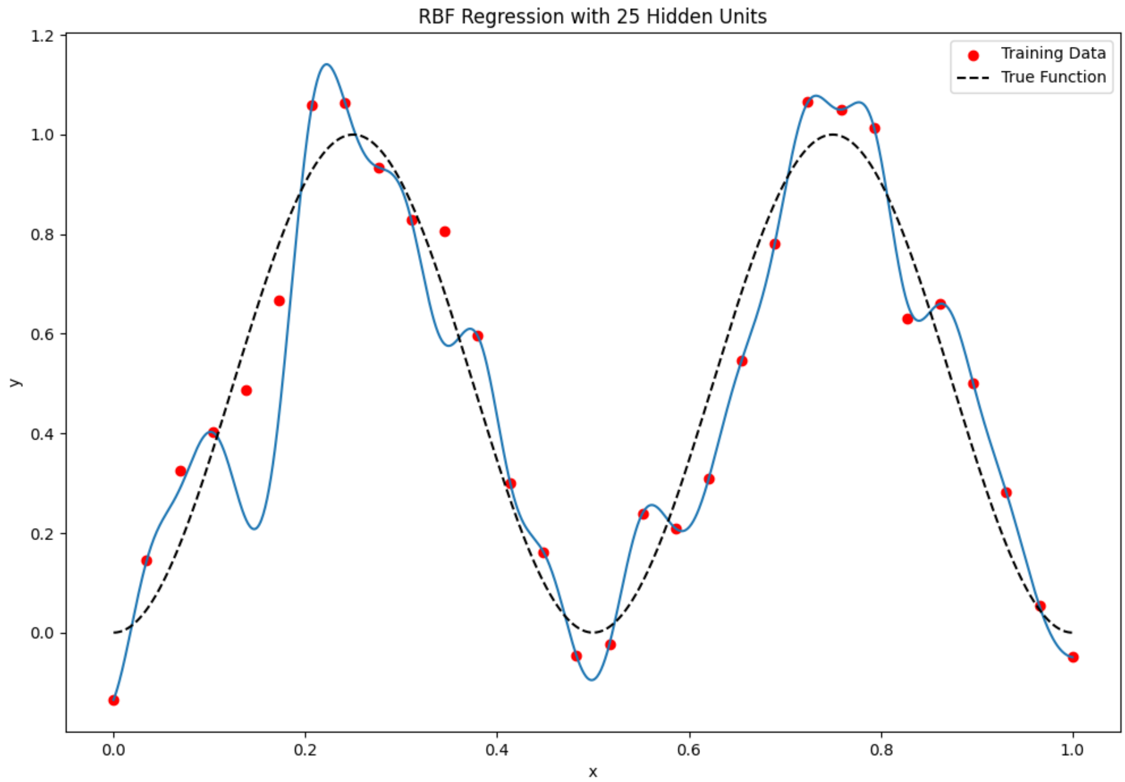




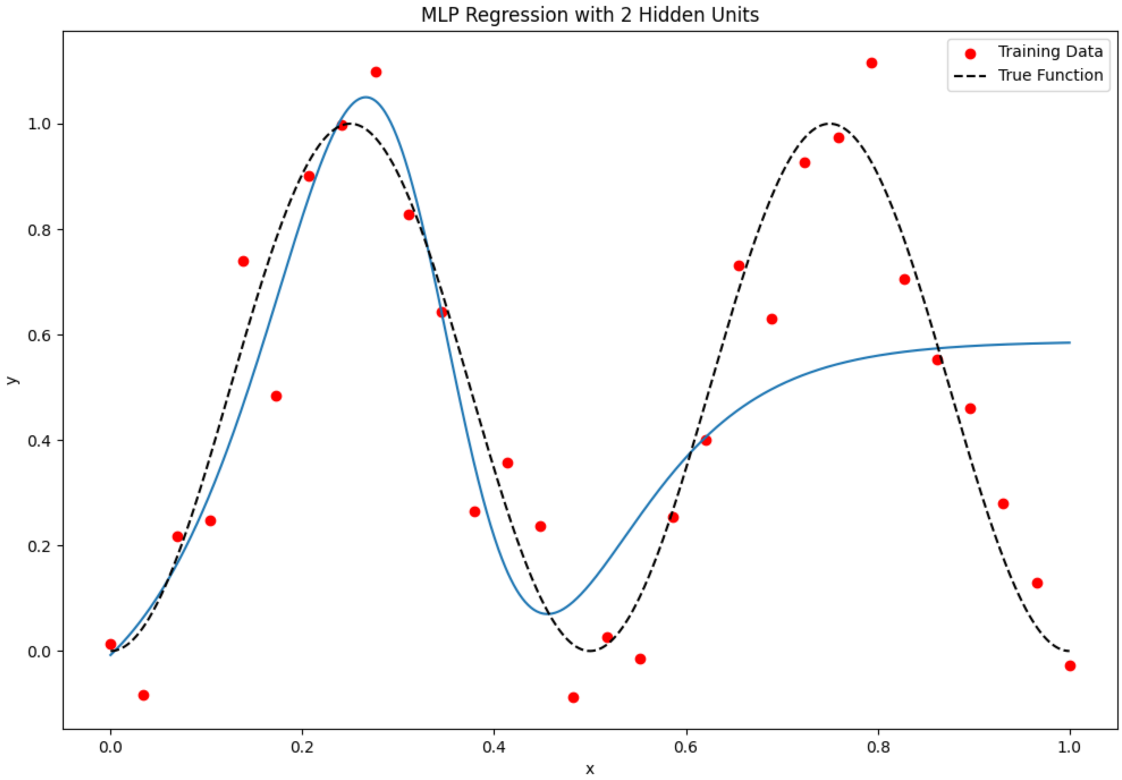

RBF Regression with 10 Hidden Units
Training Data
True Function
y
x
0.0
0.2
0.4
0.6
0.8
1.0

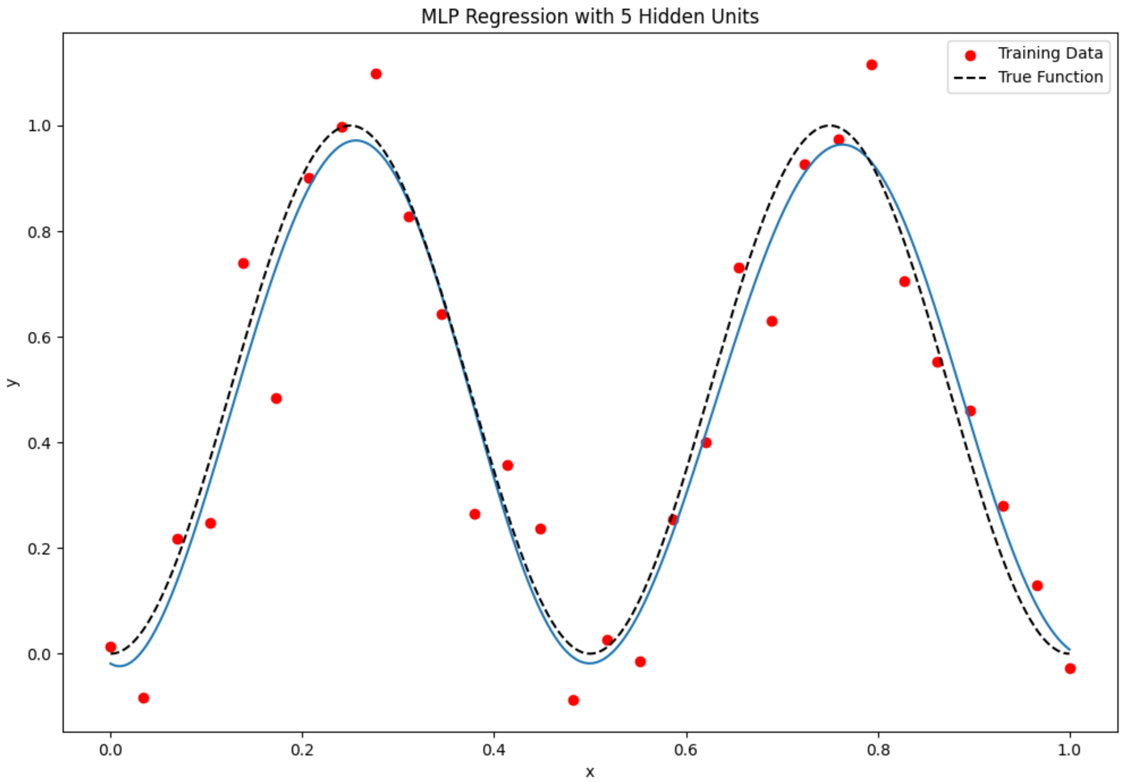

RBF Regression with 15 Hidden Units
Training Data
True Function
y
x
0.0
0.2
0.4
0.6
0.8
1.0

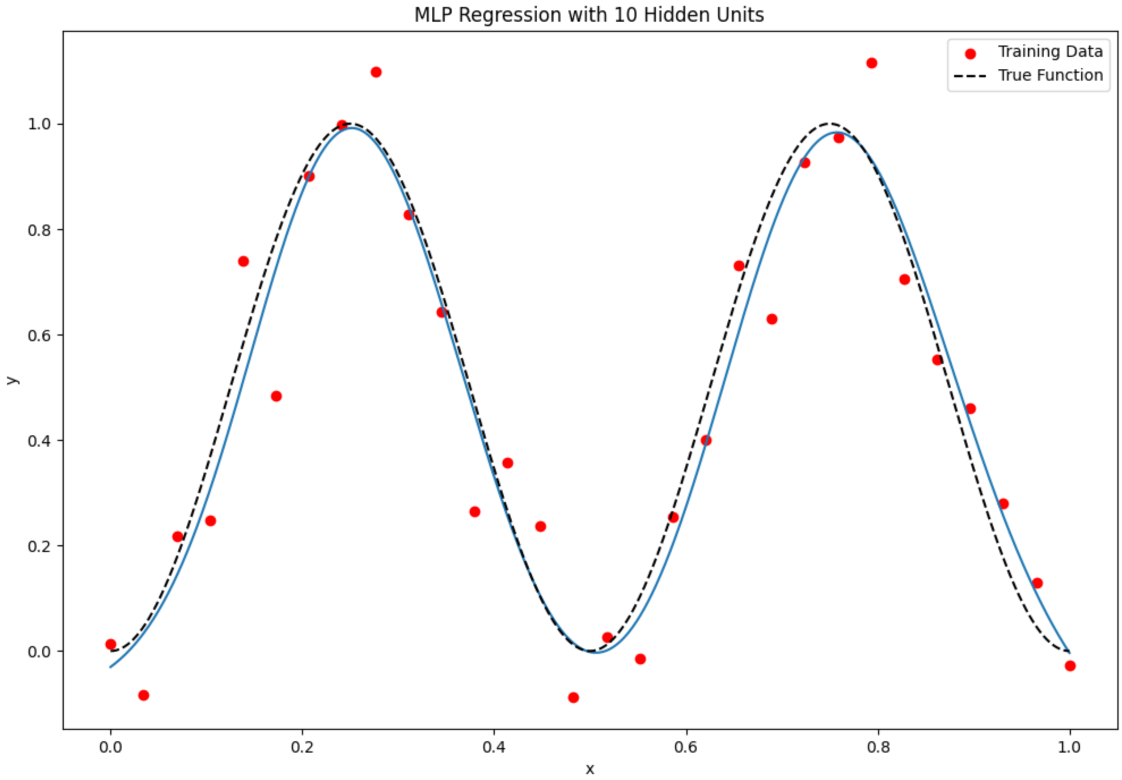


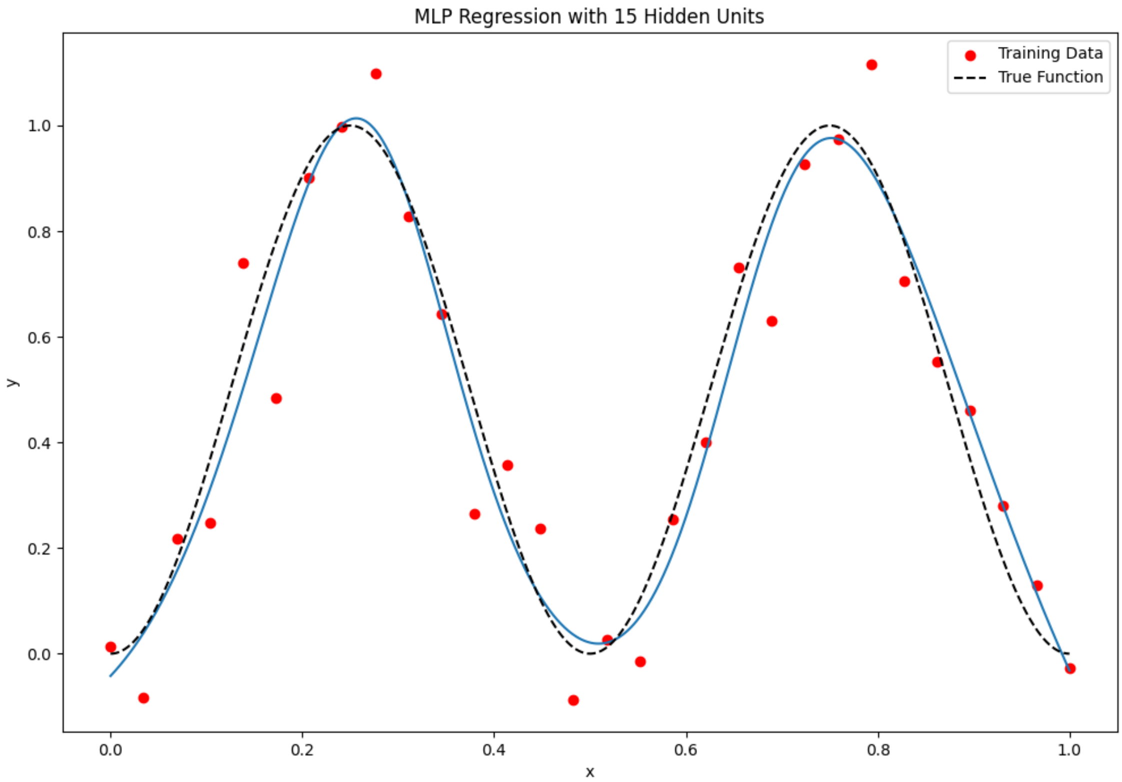


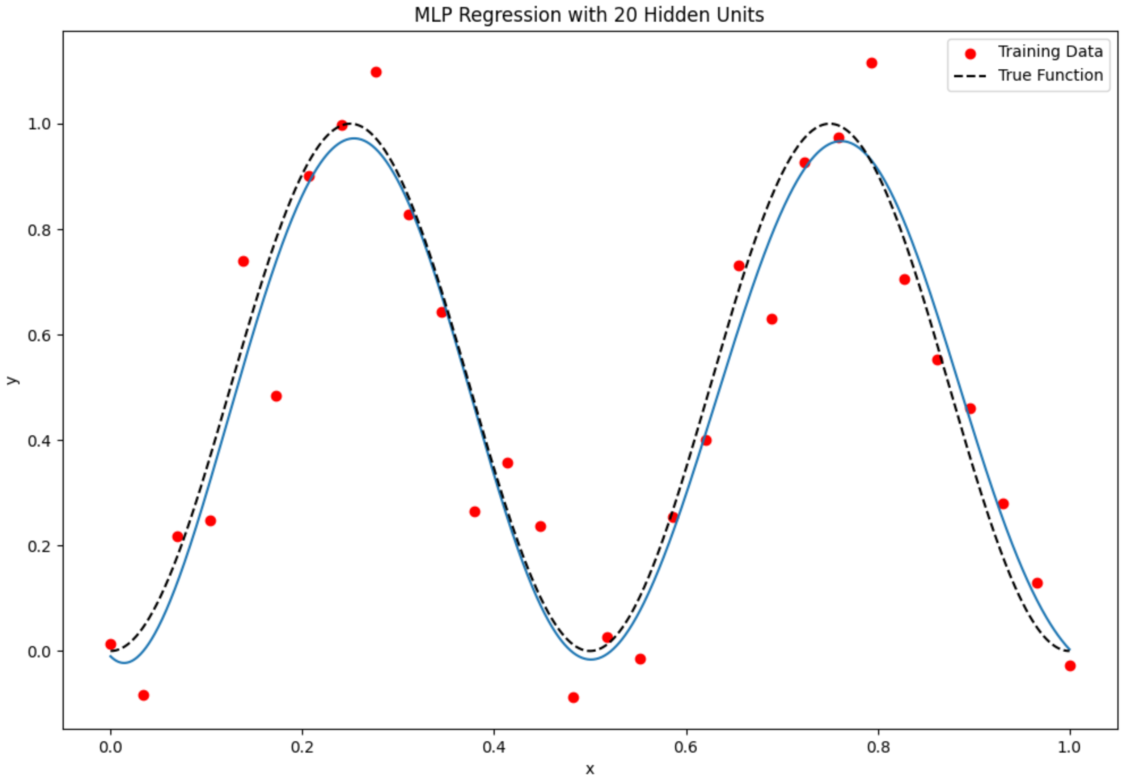
Experience with six pictures with RBF, we can find with 2 and 5 neurons, it looks like underfitting. and with 10 neurons, it seem almost fit target function. With 15, 20, 25 neurons, it looks overfitting.

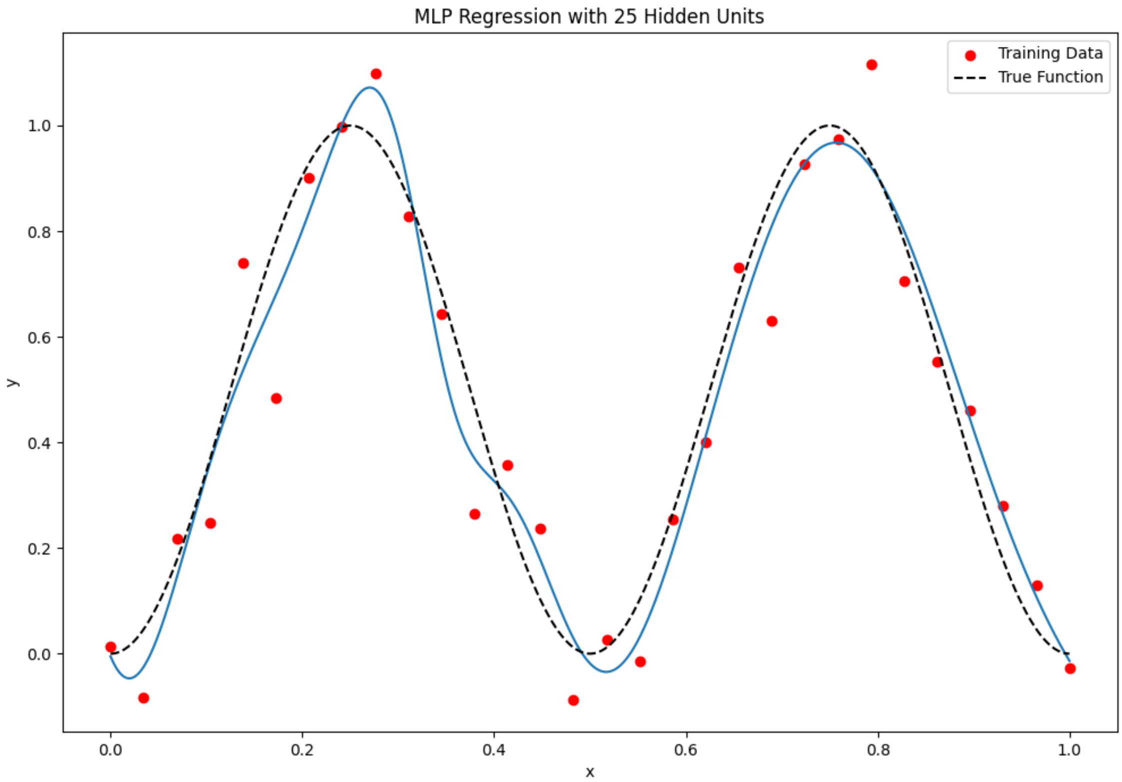

MLP Regression with 2 Hidden Units
Training Data
True Function
y
x
0.0
0.2
0.4
0.6
0.8
1.0


MLP Regression with 5 Hidden Units
Training Data
True Function
y
x
0.0
0.2
0.4
0.6
0.8
1.0


MLP Regression with 10 Hidden Units
Training Data
True Function
1.0
0.8
0.6
0.4
0.2
0.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x


MLP Regression with 15 Hidden Units
Training Data
True Function
1.0
0.8
0.6
0.4
0.2
0.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x





Experience with six pictures with MLP, we can find with 2 neurons, it looks like underfitting. and with 5, 10, 15, 20 neurons, it seem almost fit target function. With 25 neurons, it looks overfitting.

In this experience, we can find that when the neuron less, it will result in underfitting. However, when model become large, it would result in the model overfitting. So, we can use model complexity control to restict our model to prevent overfitting.