Neural Network and Deep Learning

Radial Basis Function (RBF)

YANIKA KONGSOROT

Outline

- RBF Neural Network
- RBF Neural Network - Learning

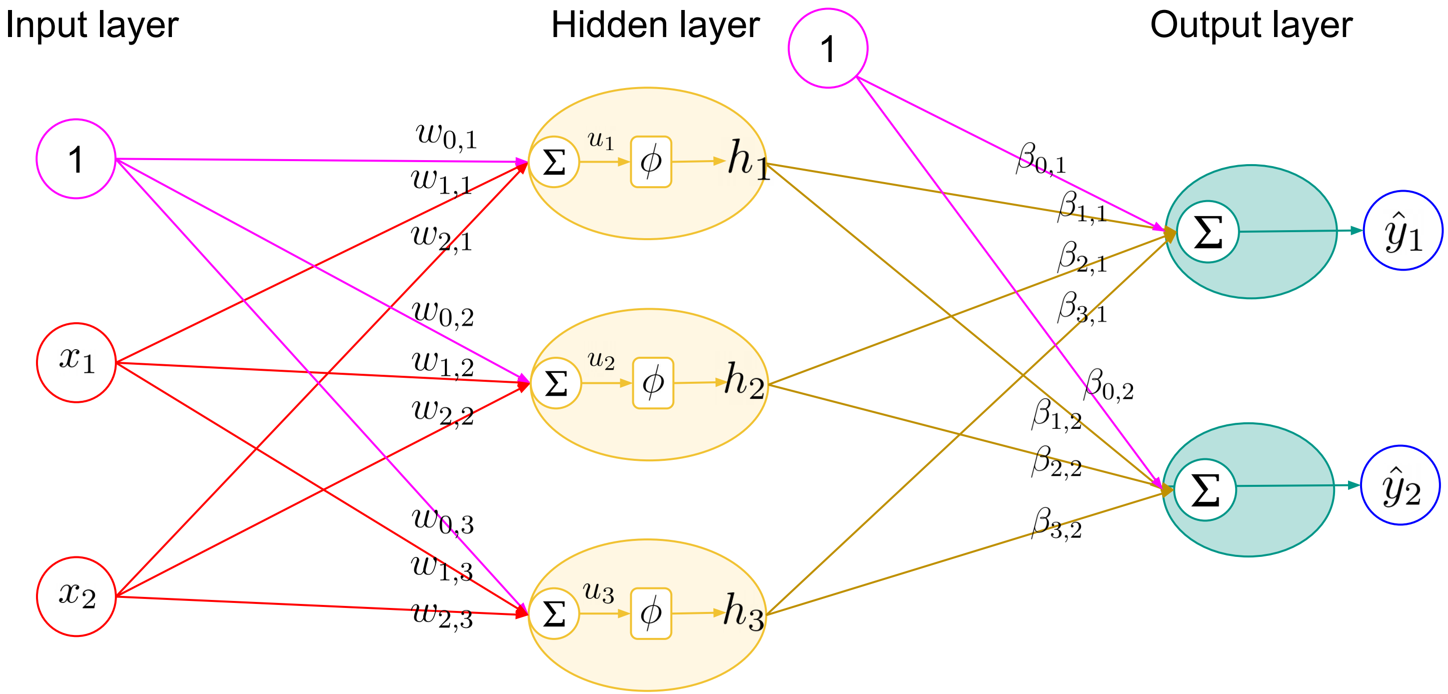
Radial Basis Function (RBF) Neural Network

RBF Neural Networks

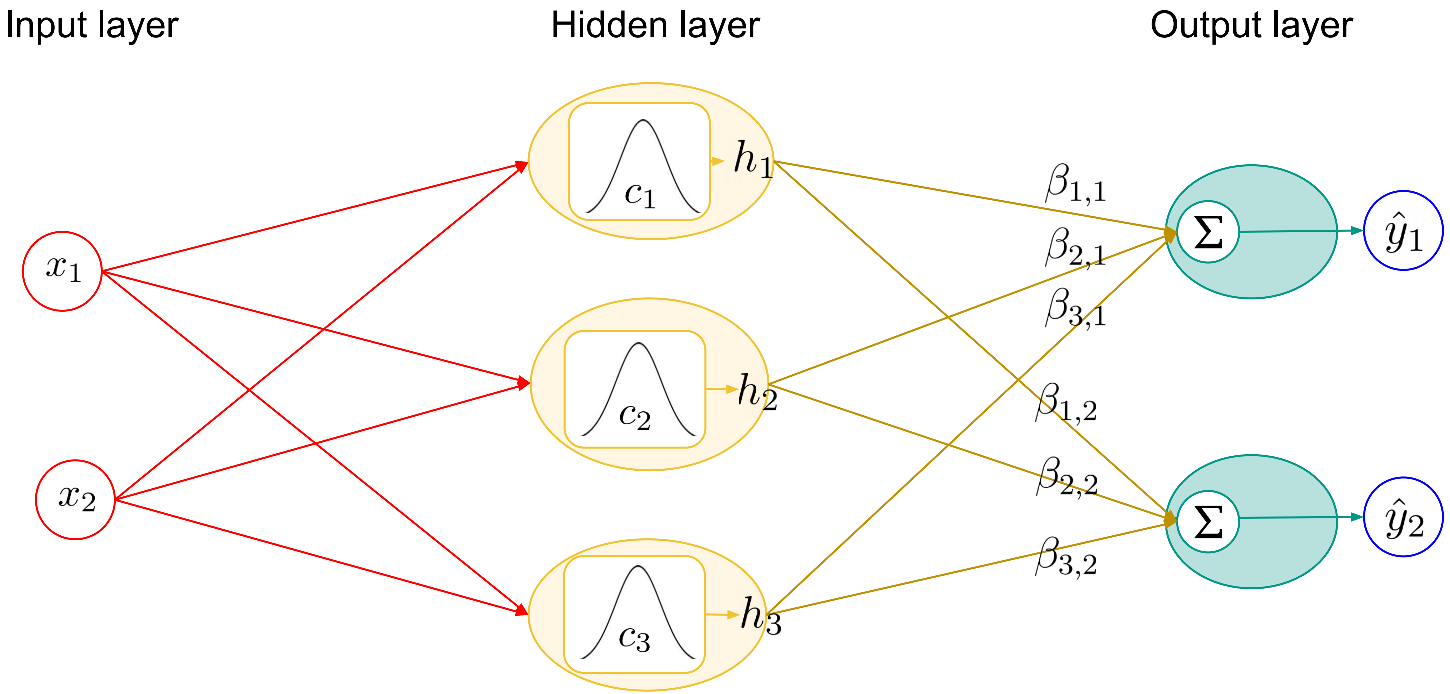
- RBF network is the single hidden layer feedforward neural network which consists of an input layer, a single hidden layer, and an output layer.

- There are no input weights on the lines from the input nodes to the hidden nodes.
- The **Radial-Basis Function** is used as the activation function in hidden layer.
- Each hidden node stores a “prototype” vector which also often called “**center**” vector because it is the value at the center of the bell shaped radial basis function.
- The leaning method for tuning the **output weights** can be:
  - LMS, Gradient descent
  - Generalized pseudo inverse

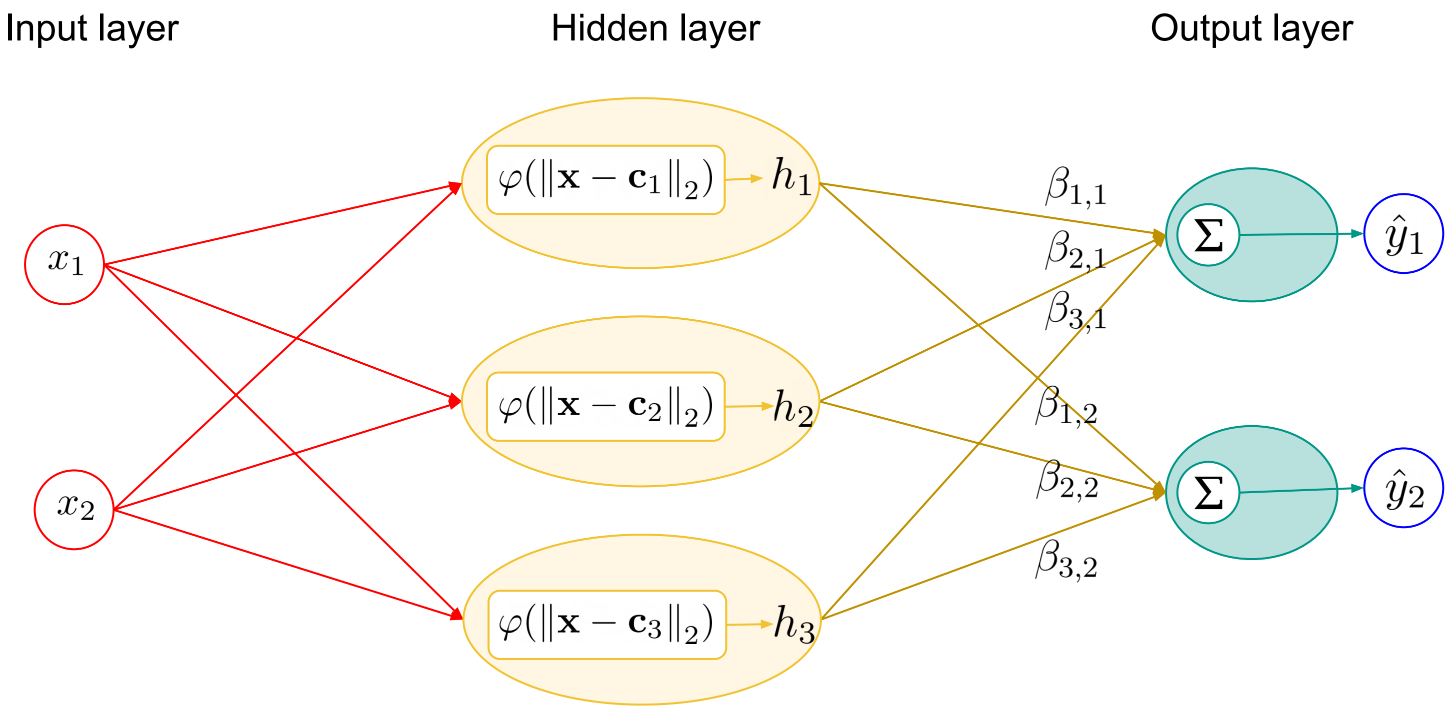
Single hidden layer feedforward neural network



RBF neural network



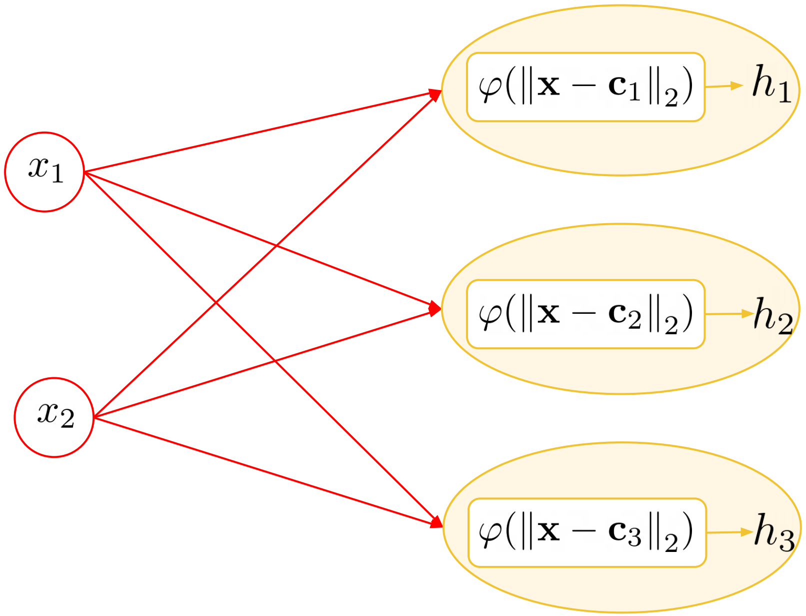
RBF neural network

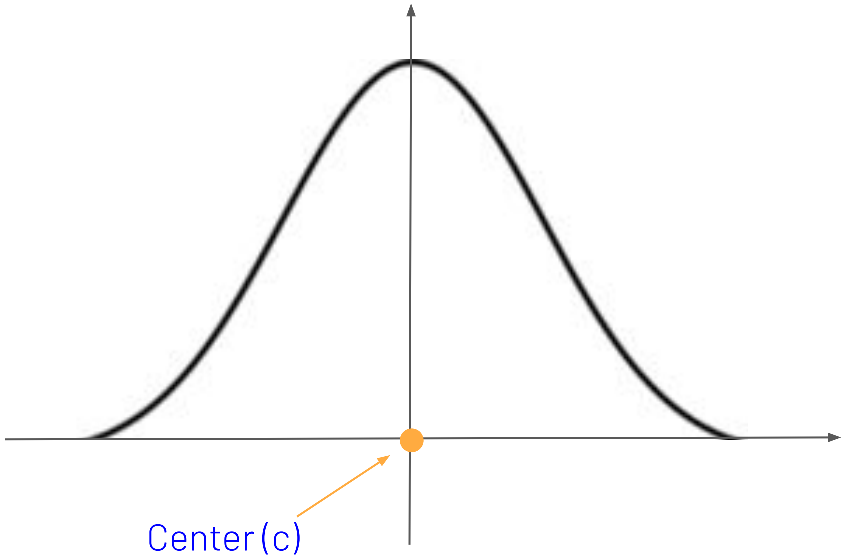


RBF Activation Function

RBF Activation Function

- The **Radial-Basis Function** is used as the activation function



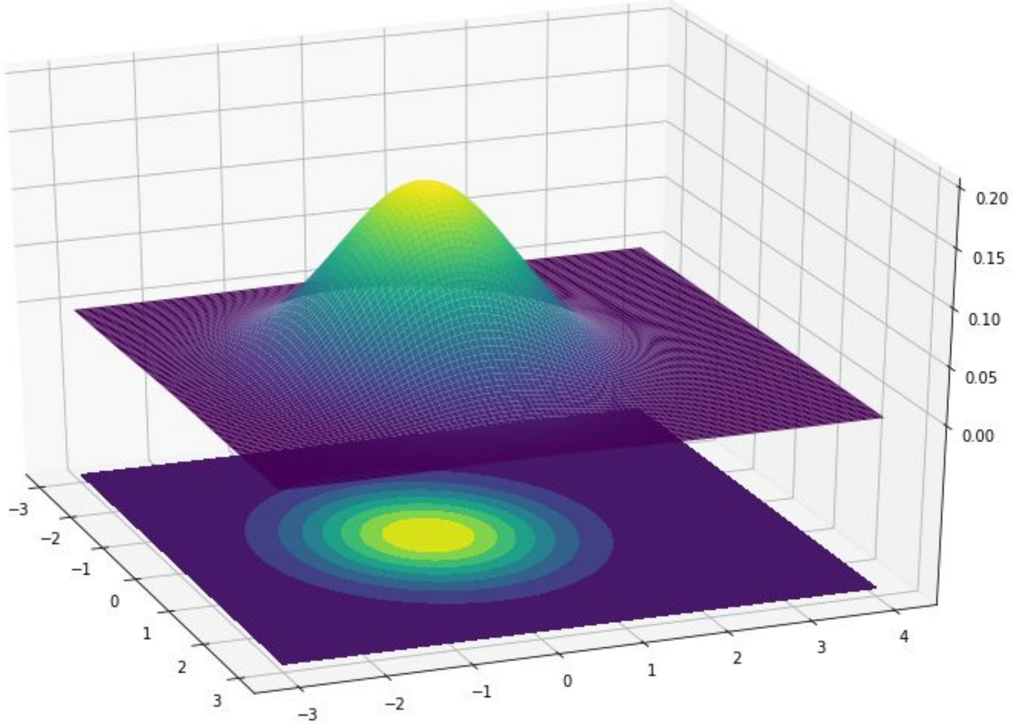


Gaussian Function

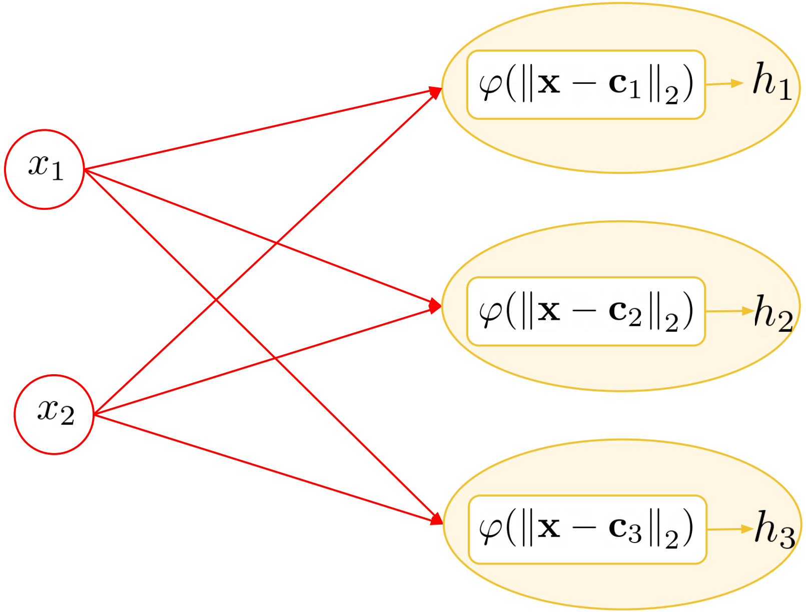
- The **Gaussian Functions** are generally used for Radial Basis Function.

Given : $r = \|\mathbf{x} - \mathbf{c}\|_2$

$$\varphi(r) = exp(-\frac{r^2}{2\sigma^2})$$



Gaussian RBF



Gaussian functions:

$$\varphi(r) = exp(-\frac{r^2}{2\sigma^2})$$

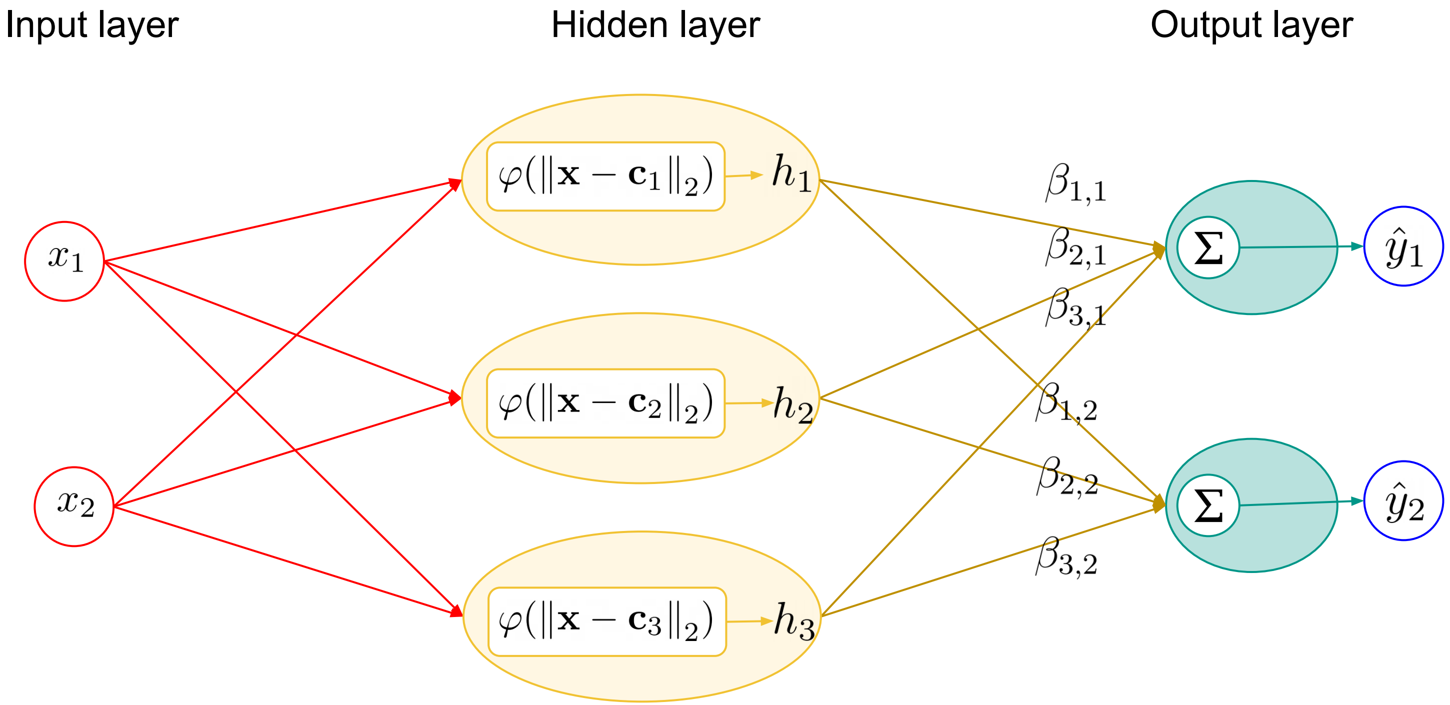
where $r = \|\mathbf{x}-\mathbf{c}\|_2$

Gaussian RBF:

$$\varphi(\|\mathbf{x}-\mathbf{c}\|_2) = exp(-\frac{\|\mathbf{x}-\mathbf{c}\|_2^2}{\sigma^2})$$

RBF Neural Network ***Learning***

RBF neural network



RBF Neural Network Learning

1. **Hidden layer learning**
   - Learn to **find the center** vectors of hidden nodes
   - Compute the output of hidden layer
2. **Output weights learning**
   - Any learning algorithms can be used such as
     - *LMS,*
     - *Generalized pseudo inverse,*
     - *etc.*

Hidden layer learning

Selecting the centers vectors

- The number of center vectors relate to the number of hidden nodes.
- There are many possible learning strategies that can be used to select the center vectors of RBFN
  a. The center vectors are randomly selected from training set
  b. Using the "K-means" clustering algorithm to set center vectors

Output weights learning

Compute the weight between hidden layer and output layer

- To adjust the output weights of RBF network, any algorithm can be used such as
  a. The *gradient descent* (similar to MLP neural network learning algorithm)

$$\begin{aligned}\beta_{l,m} &\leftarrow \beta_{l,m} - \eta \frac{\partial J_i}{\partial \beta_{l,m}} \\ &\leftarrow \beta_{l,m} + \eta \left(y_{i,m} - \hat{y}_{i,m}\right) h_{i,l}\end{aligned}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_{1,1} & \cdots & \beta_{1,M} \\ \vdots & \ddots & \vdots \\ \beta_{L,1} & \cdots & \beta_{L,M} \end{bmatrix}$$

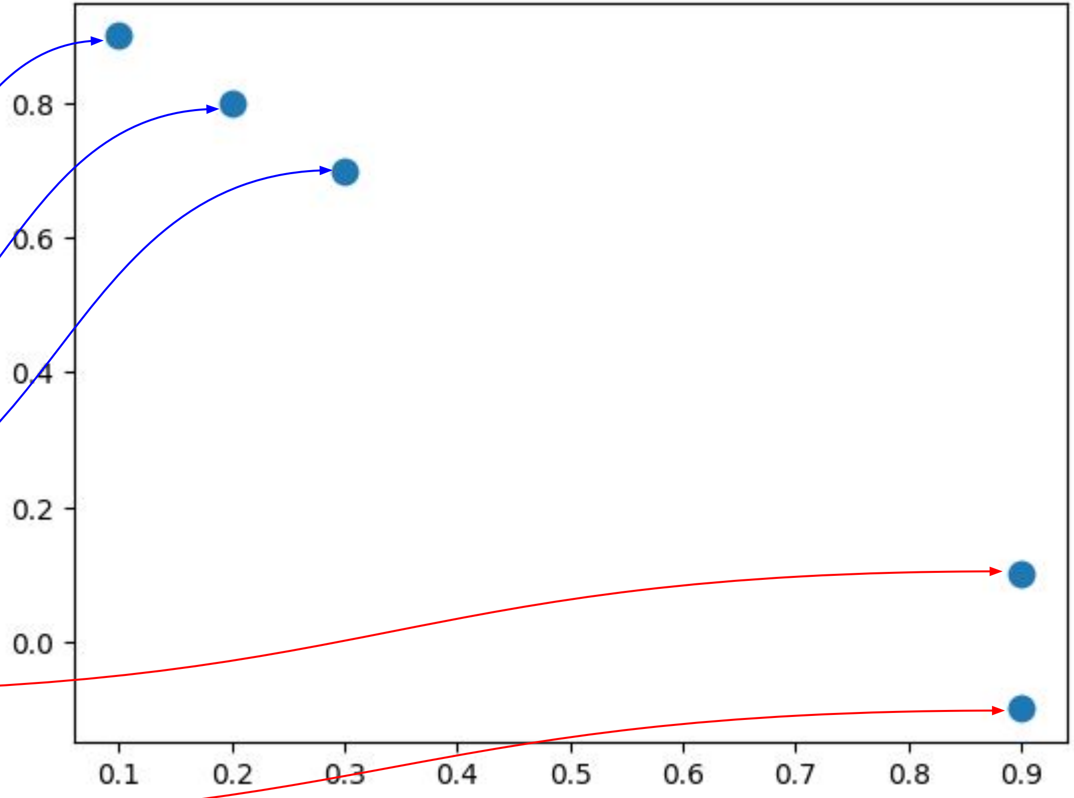
  b. The generalized pseudo inverse

$$\boldsymbol{\beta} = (\mathbf{H}^\top \mathbf{H})^{-1} \mathbf{H}^\top \mathbf{Y}$$

RBF Training Example

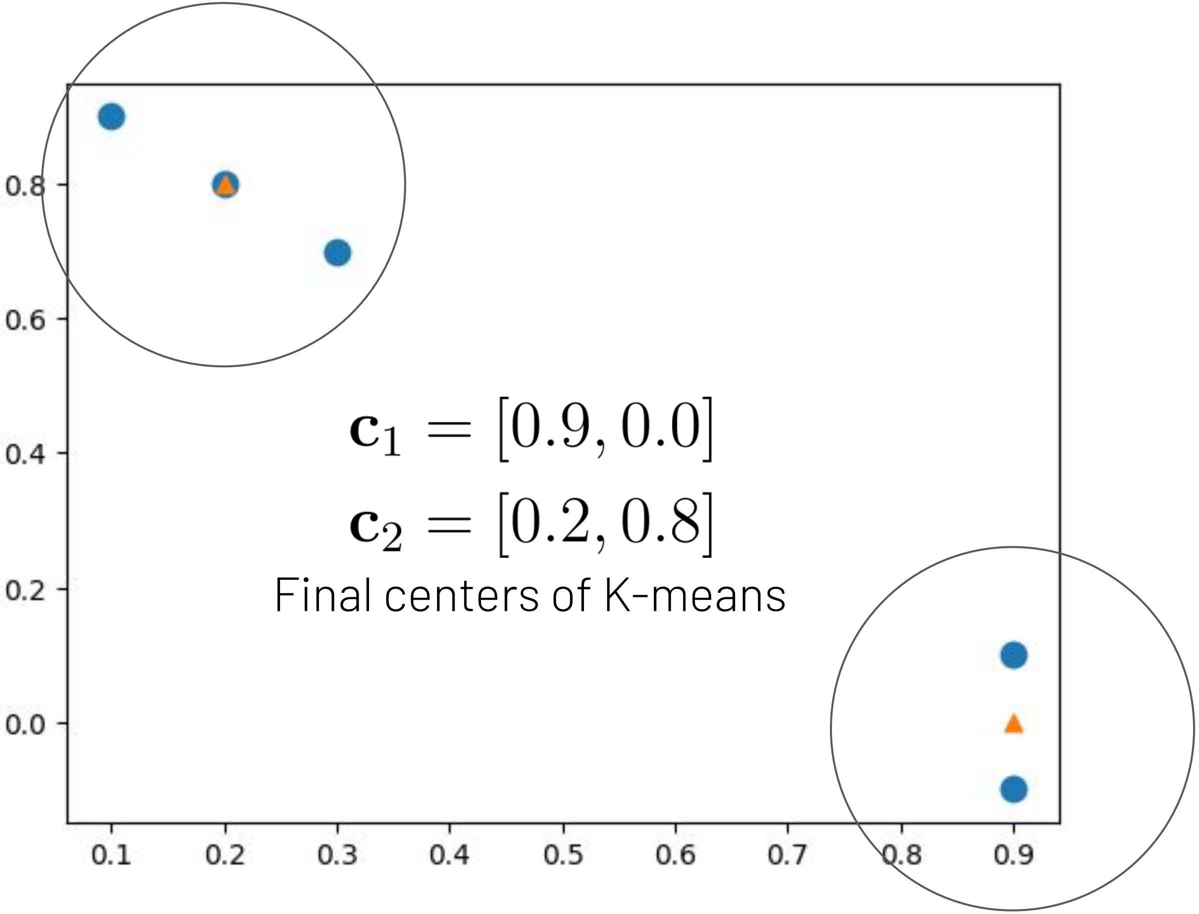
Example

x_1	x_2	y
0.1	0.9	-1
0.2	0.8	-1
0.3	0.7	-1
0.9	0.1	1
0.9	-0.1	1



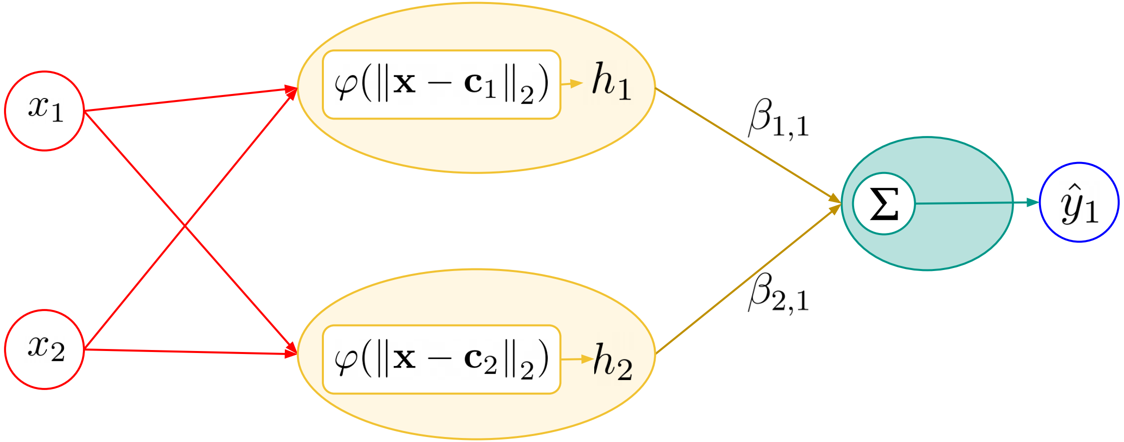
Example

x_1	x_2	y
0.1	0.9	-1
0.2	0.8	-1
0.3	0.7	-1
0.9	0.1	1
0.9	-0.1	1



Example

x_1	x_2	y
0.1	0.9	-1
0.2	0.8	-1
0.3	0.7	-1
0.9	0.1	1
0.9	-0.1	1



- Fixed centers:

$$\mathbf{c}_1 = [0.9, 0.0]$$

$$\mathbf{c}_2 = [0.2, 0.8]$$

- Fixed the spread (sigma) of RBF to 1

Example

x_1	x_2	y
0.1	0.9	-1
0.2	0.8	-1
0.3	0.7	-1
0.9	0.1	1
0.9	-0.1	1

- Compute the hidden layer output

$$\mathbf{H} = \varphi(\|\mathbf{x} - \mathbf{c}\|_2) = exp(-\frac{\|\mathbf{x} - \mathbf{c}\|_2^2}{\sigma^2})$$

$$\begin{aligned} h_1 &= exp(-\frac{\|\mathbf{x}_1 - \mathbf{c}_1\|^2}{2(1)^2}) \\ &= exp(-\frac{\|\mathbf{x}_1 - \mathbf{c}_1\|^2}{2(1)^2}) \\ &= exp(-\frac{(\sqrt{(0.1-0.9)^2 + (0.9-0)^2})^2}{2(1)^2}) \\ &= exp(-\frac{1.2041595^2}{2(1)^2}) \\ &= 0.485649 \end{aligned}$$

$$\mathbf{H} = \begin{bmatrix} \boxed{0.48432457} & 0.99004983 \\ 0.56836015 & 1. \\ 0.65376979 & 0.99004983 \\ 0.99501248 & 0.61262639 \\ 0.99501248 & 0.52204578 \end{bmatrix}$$

Example

x_1	x_2	y
0.1	0.9	-1
0.2	0.8	-1
0.3	0.7	-1
0.9	0.1	1
0.9	-0.1	1

- ➢ Output weights learning: pseudo inverse

$$\boldsymbol{\beta} = (\mathbf{H}^\top \mathbf{H})^{-1} \mathbf{H}^\top \mathbf{Y}$$

$$\beta = \begin{bmatrix} 2.28967078 \\ -2.30355664 \end{bmatrix}$$

Example

x_1	x_2	y
0.1	0.9	-1
0.2	0.8	-1
0.3	0.7	-1
0.9	0.1	1
0.9	-0.1	1

- Prediction

$$\mathbf{H}\beta = \begin{bmatrix} -1.17169205 \\ -1.00219902 \\ -0.78371829 \\ 0.8670314 \\ 1.07568899 \end{bmatrix}$$

Workshop