CSC4007 Advanced Machine Learning

Working example for RBF features

Example: RBF features

- Assume there are three data points as

$$X = \begin{bmatrix} 2 & 3 \\ 4 & 5 \\ 0 & 1 \end{bmatrix} \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix}$$

- Assume there are a set of two centers

$$c_1 = [1, 0] \qquad c_2 = [1, 1]$$

Receives 2 features

$$\phi(x) = [1, \phi_1(x), \phi_2(x)]$$

$$\phi_1(x) = e^{\frac{-\|x - c_1\|}{2\sigma^2}} \qquad \phi_2(x) = e^{\frac{-\|x - c_2\|}{2\sigma^2}}$$

- So the data matrix used in computing optimum parameters is

- The **optimal β is the same**

$$\hat{\beta}^{ls} = (X^\top X)^{-1} X^\top Y \quad \text{but with} \quad X = \begin{pmatrix} \phi(x_1)^\top \\ \vdots \\ \phi(x_n)^\top \end{pmatrix} \in \mathbb{R}^{n \times k}$$

$$X = \begin{bmatrix} 1 & e^{\frac{-\|x_1 - c_1\|}{2\sigma^2}} & e^{\frac{-\|x_1 - c_2\|}{2\sigma^2}} \\ 1 & e^{\frac{-\|x_2 - c_1\|}{2\sigma^2}} & e^{\frac{-\|x_2 - c_2\|}{2\sigma^2}} \\ 1 & e^{\frac{-\|x_3 - c_1\|}{2\sigma^2}} & e^{\frac{-\|x_3 - c_2\|}{2\sigma^2}} \end{bmatrix} \begin{matrix} \phi(x_1) \\ \phi(x_2) \\ \phi(x_3) \end{matrix}$$

Assignment: RBF Features

- Task 2.1: “Take all training instances to assign as centers” means create a set of centers

$$c_i = x_i \quad \text{For all input instance } x_i$$

Create a matrix of all centers as

$$C = \begin{bmatrix} c_1^T \\ c_2^T \\ \vdots \\ c_n^T \end{bmatrix} = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix}$$

n is the number of training instances

So we will receive n features as $\phi_i(x) = e^{\frac{-\|x - x_i\|}{2\sigma^2}}$

$$\phi(x) = [1, \phi_1(x), \phi_2(x), \ldots, \phi_n(x)]$$

Assignment: RBF Features

- Snippet code to compute RBF features for given training input data X, and a matrix of centers C.

```
def RBF_Features(X,C,bandwith):
    PHI = np.ones(X.shape[0]) # prepend ones
    for i in range(C.shape[0]): # loop over each center c_i
        distance   = X - C[i,:]   # (difference from all x to center c_i)
        distance_2  = np.sum(np.multiply(distance,distance),axis=1) # square the difference, and use sum to compute the squared magnitude /length of this difference vector
        PHI_i = np.exp(-distance_2/(2*bandwith)) # feature i correspond to center c_i
        PHI   = np.column_stack([PHI, PHI_i]) # stack all feature phi  i

    return PHI
```

Return a matrix of all feature fectors

$$X = \begin{pmatrix} \phi(x_1)^\top \\ \vdots \\ \phi(x_n)^\top \end{pmatrix} \in \mathbb{R}^{n \times k}$$