Training One Neural Layer

Deep Learning (ENGG*6600*07)

School of Engineering,
University of Guelph, ON, Canada

Course Instructor: Benyamin Ghojogh
Fall 2023

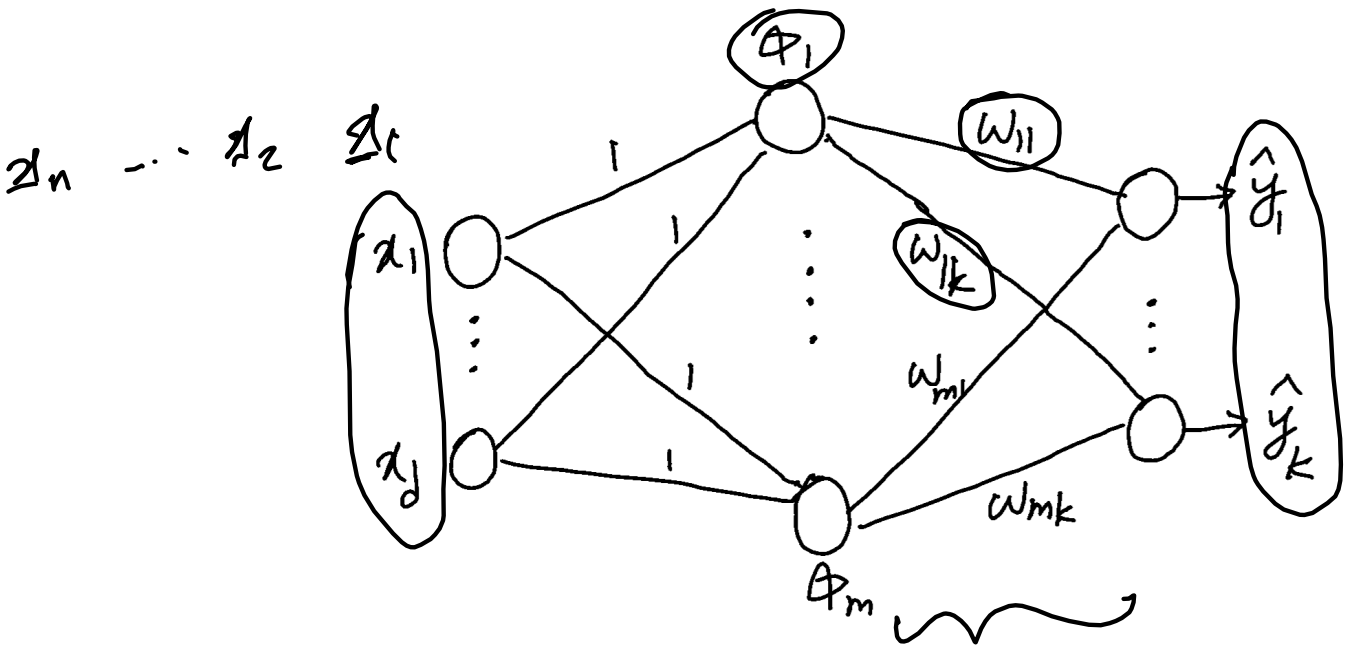
Introduction

- In the lecture of "Training one neuron", we were introduced to the neural networks with only one layer and only one neuron.
- In this lecture, we are introduced to the neural networks with only one [learnable] layer but possibly multiple neurons.
- Two of these networks are **Radial Basis Function (RBF)** network and **Self-Organizing Map (SOM)**.

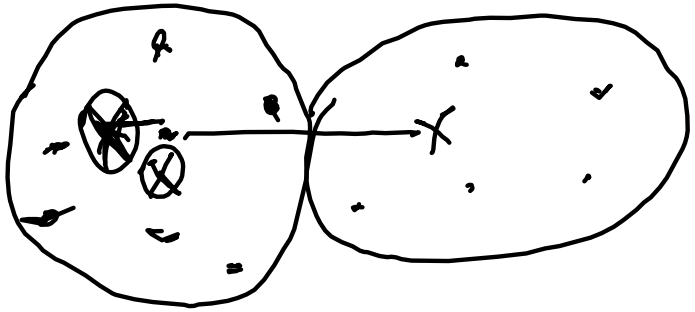
Radial Basis Function Network

Radial Basis Function Network

- A **Radial Basis Function (RBF)** network was first proposed in 1988 [1, 2].
- It has two layers but the first layer has fixed weights equal to one. The second layer has learnable weights.
- The first layer connects the data $\boldsymbol{x} \in \mathbb{R}^d$ to m basis kernel functions $\{\phi_i(\boldsymbol{x})\}_{i=1}^m$.
- The second layer connects the basis functions $\{\phi_i(\boldsymbol{x})\}_{i=1}^m$ to the output neurons $\{\widehat{y}_j\}_{j=1}^k$.
- The weight w_{ij} denotes the weight connecting $\phi_i(\boldsymbol{x})$ to $\widehat{y}_j$.



Radial Basis Function Network



- The basis functions can be various kernel functions such as:

Gaussian distribution: $\phi_i(\boldsymbol{x}) = e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{\mu}_i\|_2^2}{2\sigma_i^2}}$, (1)

Logistic (sigmoid) function: $\phi_i(\boldsymbol{x}) = \frac{1}{1 + e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{\mu}_i\|_2^2}{2\sigma_i^2}}}$, (2)

where $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and $\sigma_i^2 \in \mathbb{R}$ are the mean and variance for $\phi_i(\boldsymbol{x})$.

- At the first step, the means $\{\boldsymbol{\mu}_i\}_{i=1}^m$ are found by applying a clustering method, such as K-means, on the training data with m clusters. The variances of clusters determine the variances $\{\sigma_i\}_{i=1}^m$.

$$\sigma_i^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu_i)^2$$

Radial Basis Function Network

- RBF networks can be considered as an **additive model**. An additive model, first proposed in [3], maps data to a space with several basis functions and then tries to learn a weighted average of those bases.
- In RBF, the output is obtained as:

$$\widehat{y}_j = w_{1j}\,\phi_1(\boldsymbol{x}) + \cdots + w_{mj}\,\phi_m(\boldsymbol{x}) = \sum_{i=1}^{m} w_{ij}\,\phi_i(\boldsymbol{x}). \tag{3}$$

- Consider n data points together in a matrix $\boldsymbol{X} \in \mathbb{R}^{d \times n}$. In matrix form:

$$\widehat{\boldsymbol{Y}} = \boldsymbol{W}^\top \boldsymbol{\Phi}, \tag{4}$$

where:

$$\mathbb{R}^{k \times n} \ni \widehat{\boldsymbol{Y}} = \begin{bmatrix} \widehat{y}_{11} & \cdots & \widehat{y}_{1n} \\ \vdots & \ddots & \vdots \\ \widehat{y}_{k1} & \cdots & \widehat{y}_{kn} \end{bmatrix}, \quad \mathbb{R}^{m \times k} \ni \boldsymbol{W} = \begin{bmatrix} w_{11} & \cdots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mk} \end{bmatrix},$$

$$\mathbb{R}^{m \times n} \ni \boldsymbol{\Phi} = \begin{bmatrix} \phi_1(x_1) & \cdots & \phi_1(x_n) \\ \vdots & \ddots & \vdots \\ \phi_m(x_1) & \cdots & \phi_m(x_n) \end{bmatrix}.$$

Radial Basis Function Network

- Least squares error minimization between the label of data $\boldsymbol{Y} \in \mathbb{R}^{k \times n}$ and the output of network $\widehat{\boldsymbol{Y}} \in \mathbb{R}^{k \times n}$:

$$\underset{\boldsymbol{W}}{\text{minimize}} \quad \|\boldsymbol{Y} - \widehat{\boldsymbol{Y}}\|_F^2 = \|\boldsymbol{Y} - \boldsymbol{W}^\top \boldsymbol{\Phi}\|_F^2. \tag{5}$$

- Simplification of the cost function:

$$\begin{aligned} \|\boldsymbol{Y} - \boldsymbol{W}^\top \boldsymbol{\Phi}\|_F^2 &= \operatorname{tr}\big((\boldsymbol{Y} - \boldsymbol{W}^\top \boldsymbol{\Phi})^\top (\boldsymbol{Y} - \boldsymbol{W}^\top \boldsymbol{\Phi})\big) = \operatorname{tr}\big((\boldsymbol{Y}^\top - \boldsymbol{\Phi}^\top \boldsymbol{W})(\boldsymbol{Y} - \boldsymbol{W}^\top \boldsymbol{\Phi})\big) \\ &= \operatorname{tr}(\boldsymbol{Y}^\top \boldsymbol{Y} - \boldsymbol{Y}^\top \boldsymbol{W}^\top \boldsymbol{\Phi} - \boldsymbol{\Phi}^\top \boldsymbol{W} \boldsymbol{Y} + \boldsymbol{\Phi}^\top \boldsymbol{W} \boldsymbol{W}^\top \boldsymbol{\Phi}) \\ &\overset{(a)}{=} \operatorname{tr}(\boldsymbol{Y}^\top \boldsymbol{Y}) - \operatorname{tr}(\boldsymbol{W}^\top \boldsymbol{\Phi} \boldsymbol{Y}^\top) - \operatorname{tr}(\boldsymbol{W} \boldsymbol{Y} \boldsymbol{\Phi}^\top) + \operatorname{tr}(\boldsymbol{W}^\top \boldsymbol{\Phi} \boldsymbol{\Phi}^\top \boldsymbol{W}), \end{aligned}$$

  where (a) is because of the linearity and cyclic property of the trace operator.

- Solving this optimization problem:

$$\begin{aligned} &\frac{\partial}{\partial \boldsymbol{W}} \|\boldsymbol{Y} - \boldsymbol{W}^\top \boldsymbol{\Phi}\|_F^2 = -\boldsymbol{\Phi} \boldsymbol{Y}^\top - \boldsymbol{\Phi} \boldsymbol{Y}^\top + 2 \boldsymbol{\Phi} \boldsymbol{\Phi}^\top \boldsymbol{W} \overset{\text{set}}{=} \boldsymbol{0} \Longrightarrow \boldsymbol{\Phi} \boldsymbol{\Phi}^\top \boldsymbol{W} = \boldsymbol{\Phi} \boldsymbol{Y}^\top \\ &\Longrightarrow \boldsymbol{W} = (\boldsymbol{\Phi} \boldsymbol{\Phi}^\top)^{-1} \boldsymbol{\Phi} \boldsymbol{Y}^\top. \end{aligned} \tag{6}$$

$$tr(ABC) = tr(CAB) = tr(B\,CA) = tr(ABC)$$

$$tr(W^T(\Phi Y^T)) = tr(W^T A)$$

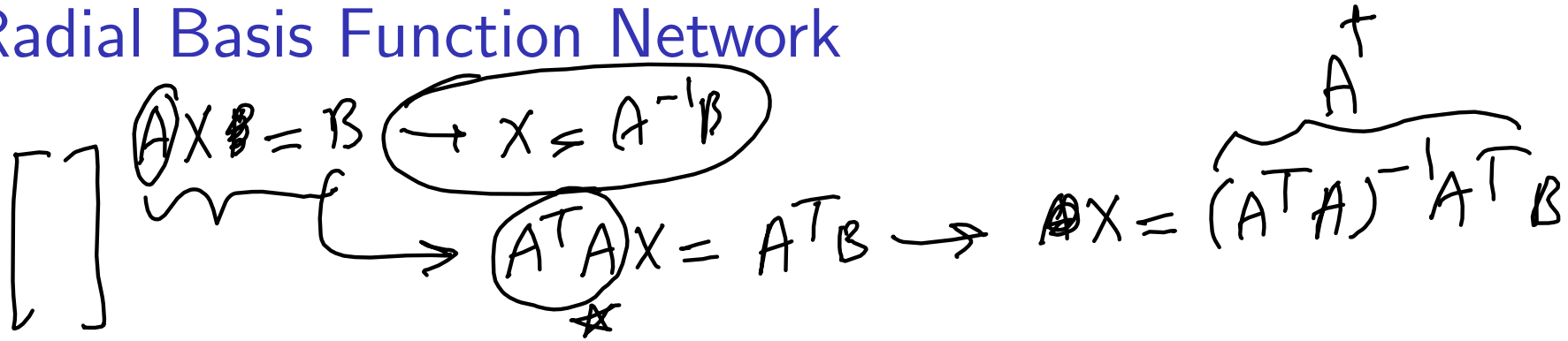
5×3 $\quad A \quad 3\times 5$

$$\frac{\partial\, tr(W^T \Phi)}{\partial W} \quad (3\times 5)$$

$$= \begin{cases} A \to 3\times 5 \ \checkmark \\ A^T \quad \times \end{cases}$$

$$\begin{cases} \Phi Y^T \\ Y^T \Phi \quad \times \\ \Phi^T Y \quad \times \\ \vdots \end{cases}$$

Radial Basis Function Network



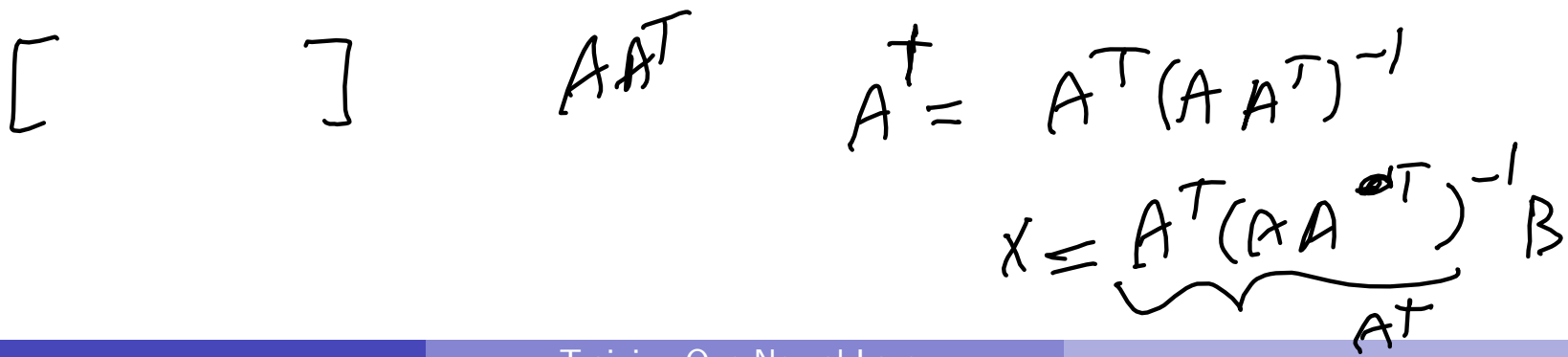
- If $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ is a singular matrix, the pseudo-inverse can be used:

$$\boldsymbol{W} = (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top)^\dagger \boldsymbol{\Phi}\boldsymbol{Y}^\top, \tag{7}$$

  where † denotes the pseudo-inverse of matrix.

- The output of the RBF network, for either the training or test data, is:

$$\widehat{\boldsymbol{Y}} = \boldsymbol{W}^\top \boldsymbol{\Phi} = ((\boldsymbol{\Phi}\boldsymbol{\Phi}^\top)^{-1}\boldsymbol{\Phi}\boldsymbol{Y}^\top)^\top \boldsymbol{\Phi} = \boldsymbol{Y}\boldsymbol{\Phi}^\top (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top)^{-\top}\boldsymbol{\Phi} = \boldsymbol{Y}\boldsymbol{\Phi}^\top (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top)^{-1}\boldsymbol{\Phi}. \tag{8}$$



$$W = (\Phi\Phi^T)^{-1}\Phi Y^T$$

$$\approx (\Phi\Phi^T + \lambda I)^{-1}\Phi Y^T$$

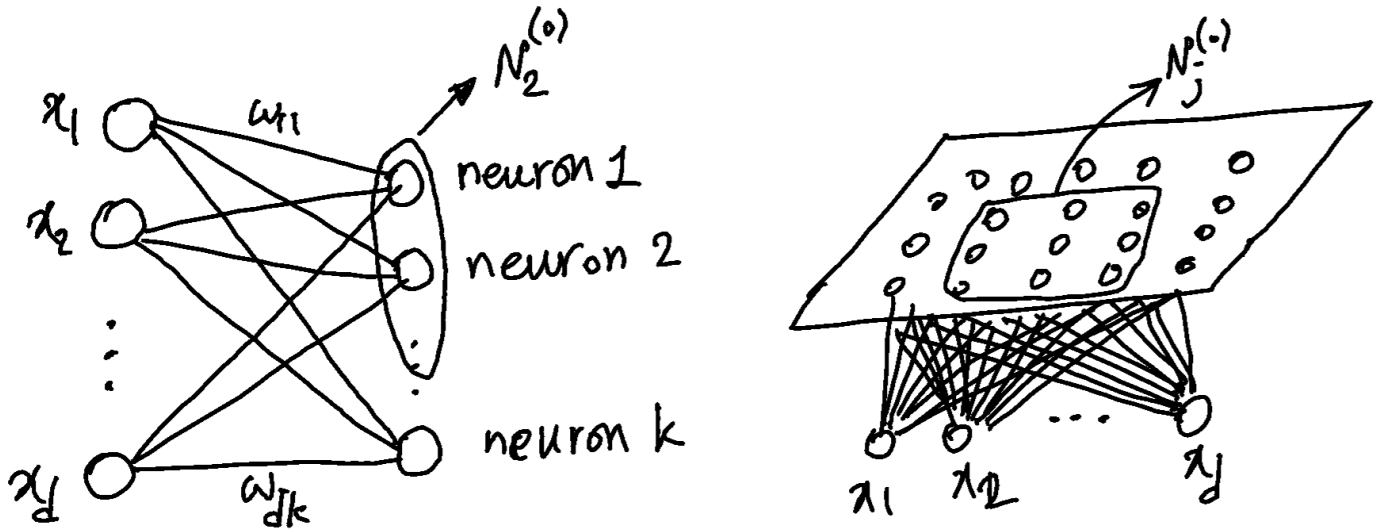
$$(AB)^T = B^T A^T$$

$$(AB)^{-1} = B^{-1}A^{-1}$$

Self-Organizing Map

Self-Organizing Map

- Self-Organizing Map (SOM) is a neural network with one layer. It is used for unsupervised clustering, where the name "self-organizing" comes from.
- It was proposed by Teuvo Kohonen in 1982 [4]; therefore, it is also called the Kohonen network [5].
- It is one layer connecting $\boldsymbol{x} = [x_1, \ldots, x_d]^\top \in \mathbb{R}^d$ to k neurons. Let w_{ij} denote the weight connecting x_i to the j-th neuron. Each neuron represents a cluster. SOM trains the weights to cluster the input data $\boldsymbol{x}$ to one of the k clusters.
- The neurons can be put in 1D or 2D structure.
- Every neuron has a neighborhood around it in the 1D or 2D structure of neurons. This neighborhood is decreased gradually during the training phase. Let $\mathcal{N}_j^{(\tau)}$ denote the neighborhood of the j-th neuron at iteration τ.



Self-Organizing Map

- Step 1 of training: Initialize all weights to small random values. Set all neighborhoods $\{\mathcal{N}_j^{(0)}\}_{j=1}^k$ to half of the neuron structure grid. Set the initial learning rate $\eta^{(0)}$ to a number in range $(0, 1]$.
- Step 2 of training: Select one the input data points, $\boldsymbol{x} = [x_1, \ldots, x_d]^\top$, and feed it to the network. Select the wining neuron z by:

$$z := \arg\min_j \sum_{i=1}^{d} \|x_i - w_{ij}\|_2. \tag{9}$$

- Step 3 of training: Update the weights of the neurons in the neighborhood of the winning neuron z:

$$w_{ij}^{(\tau+1)} := \begin{cases} w_{ij}^{(\tau)} + \eta^{(\tau)}(x_i - w_{ij}^{(\tau)}) & \text{if } j \in \mathcal{N}_z^{(\tau)} \\ w_{ij}^{(\tau)} & \text{Otherwise,} \end{cases} \tag{10}$$

  for all $i \in \{1, \ldots, d\}$.
- Step 4 of training: Decrease the learning rate and the neighborhoods:

$$\eta^{(\tau+1)} := \eta^{(0)}(1 - \frac{\tau}{t}), \tag{11}$$

$$\mathcal{N}_j^{(\tau+1)} := \mathcal{N}_j^{(\tau+1)}/2, \quad \forall j \in \{1, \ldots, k\}, \tag{12}$$

  where t denotes the total number of training iterations.
- Step 5 of training: Increase τ and go to step 2.

Acknowledgment

- Some slides of this slide deck were inspired by teachings of Prof. Ali Ghodsi (at University of Waterloo, Department of Statistics), Prof. Fakhri Karray (at University of Waterloo, Department of Electrical and Computer Engineering), and Prof. Saeed Bagheri Shouraki (at Sharif University of Technology, Department of Electrical Engineering).

References

[1] D. S. Broomhead and D. Lowe, "Radial basis functions, multi-variable functional interpolation and adaptive networks," tech. rep., Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[2] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks, complex systems, vol. 2," *Complex Systems*, vol. 2, pp. 321–355, 1988.

[3] J. H. Friedman and W. Stuetzle, "Projection pursuit regression," *Journal of the American statistical Association*, vol. 76, no. 376, pp. 817–823, 1981.

[4] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[5] T. Kohonen and T. Honkela, "Kohonen network," *Scholarpedia*, vol. 2, no. 1, p. 1568, 2007.