Multilayer Perceptron For Regression To Predict The House Prices

Huynh Huu Dinh[1]

[1] *Institute of Statistics & Department of Applied Mathematics, National Chung Hsing University*

April 19, 2019

1 Forward Propagation

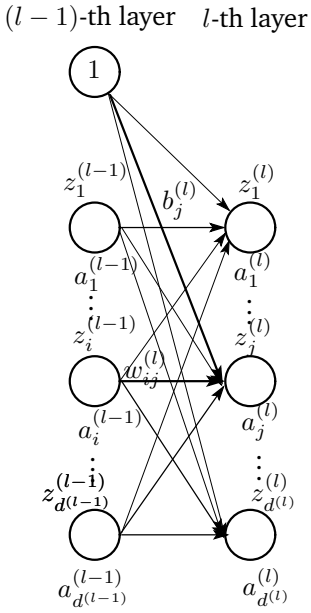

Figure 1: *Multilayer Perceptron Model*

Convention:

- L is number of layers (except input layer).
- $\mathbf{z}^{(l)}$ is input vector of l-th layer.
- $z_i^{(l)}$ is input element of i-th node in l-th layer.
- f is activation function for l-th layer.
- $\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)})$ is output vector of l-th layer.
- $a_i^{(l)}$ is output element of i-th node in l-th layer.
- $d^{(l)}$ is number of nodes in l-th layer (except bias).
- $\mathbf{W}^{(l)} = (w_{ij}^{(k)}) \in \mathbb{R}^{d^{(l-1)} \times d^{(l)}}$ is a weight matrix for $l = 1, \ldots, L$ (input layer is 0-th layer), where $w_{ij}^{(l)}$ is a element which connects the i-th node of $(l-1)$-th layer to j-th node of l-th layer.
- $\mathbf{w}_j^{(l)}$ is j-th column of matrix $\mathbf{W}^{(l)}$.
- $\mathbf{w}_{j:}^{(l)}$ is j-th row of matrix $\mathbf{W}^{(l)}$.
- $\mathbf{b}^{(l)} \in \mathbb{R}^{d^{(l)}}$ is a bias vector of l-th layer.
- $\mathbf{W}$ is a set of all weights of neural network model.
- $\mathbf{b}$ is a set of all biases of neural network model.

Forward Propagation:
For j-th node in l-th layer, we implement two steps:

- Computing $z_j^{(l)} = \mathbf{w}_j^{(l)T} \mathbf{a}^{(l-1)} + b_j^{(l)}$.
- Using activation function $a_j^{(l)} = f(z_j^{(l)})$.

Consequently, for l-th layer, we have

- $\mathbf{z}^{(l)} = \mathbf{W}^{(l)T} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$.
- $\mathbf{a}^{(l)} = f\left(\mathbf{z}^{(l)}\right)$.

Hence, we obtain the algorithm of Forward Propagation Vectorization:

- $\mathbf{a}^{(0)} = \mathbf{x}$- input vector.
- $\mathbf{z}^{(l)} = \mathbf{W}^{(l)T} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$ for $l = 1, \ldots, L$.
- $\mathbf{a}^{(l)} = f\left(\mathbf{z}^{(l)}\right)$ for $l = 1, \ldots, L$.
- $\hat{\mathbf{y}} = \mathbf{a}^{(L)} = f(\mathbf{z}^{(L)})$.

The scheme of Forward Propagation Vectorization is

$$\mathbf{a}^{(0)} \to \mathbf{z}^{(1)} \xrightarrow{f} \mathbf{a}^{(1)} \to \cdots \to \mathbf{z}^{(L)} \xrightarrow{f} \mathbf{a}^{(L)} = \hat{\mathbf{y}}$$

Finally, we obtain the algorithm of Batch (or Mini-Batch) Forward Propagation:

- $\mathbf{A}^{(0)} = \mathbf{X} \in \mathbb{R}^{d^{(0)} \times N}$, where $d^{(0)}$ is dimension of data (number of feature variables) and N is number of data points.
- $\mathbf{Z}^{(l)} = \mathbf{W}^{(l)T} \mathbf{A}^{(l-1)} + \mathbf{B}^{(l)}$ for $l = 1, \ldots, L$, where $\mathbf{B}^{(l)}$ is bias matrix which all columns identify with bias vector $\mathbf{b}^{(l)}$.
- $\mathbf{A}^{(l)} = f\left(\mathbf{Z}^{(l)}\right)$ for $l = 1, \ldots, L$.

- $\hat{\mathbf{Y}} = \mathbf{A}^{(L)} = f(\mathbf{Z}^{(L)})$.

The scheme of Batch (or Mini-Batch) Forward Propagation is

$$\mathbf{A}^{(0)} \to \mathbf{Z}^{(1)} \xrightarrow{f} \mathbf{A}^{(1)} \to \cdots \to \mathbf{Z}^{(L)} \xrightarrow{f} \mathbf{A}^{(L)} = \hat{\mathbf{Y}}$$

2 Backward Propagation

The cost function of model is

$$J(\mathbf{W}, \mathbf{b}) = \sum_{n=1}^{N} \|\mathbf{y}_n - \hat{\mathbf{y}}_n\|^2,$$

where N is number of pairs $(\mathbf{x}, \mathbf{y})$. In order to determine $\mathbf{W}, \mathbf{b}$ such that the cost function is minimize, we will implement the following steps:

- For $l = L$, we obtain
  + Find the partial derivative of J with respect to the element $w_{ij}^{(L)}$ of matrix $\mathbf{W}^{(L)}$

    $$\frac{\partial J}{\partial w_{ij}^{(L)}} = \frac{\partial J}{\partial z_j^{(L)}} \frac{\partial z_j^{(L)}}{w_{ij}^{(L)}} = e_j^{(L)} a_i^{(L-1)},$$

    where $e_j^{(L)} = \frac{\partial J}{\partial z_j^{(L)}}$.
  + Find the partial derivative of J with respect to the element $b_j^{(L)}$ of bias vector $\mathbf{b}^{(L)}$

    $$\frac{\partial J}{\partial b_j^{(L)}} = \frac{\partial J}{\partial z_j^{(L)}} \frac{\partial z_j^{(L)}}{\partial b_j^{(L)}} = e_j^{(L)}.$$

- For $1 \leq l < L$, by induction, we obtain
  + $\frac{\partial J}{\partial w_{ij}^{(l)}} = \frac{\partial J}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{w_{ij}^{(L)}} = e_j^{(l)} a_i^{(l-1)}$,
  + $\frac{\partial J}{\partial b_j^{(l)}} = \frac{\partial J}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial b_j^{(l)}} = e_j^{(l)}$.

  where

  $$\begin{aligned} e_j^{(l)} &= \frac{\partial J}{\partial z_j^{(l)}} = \frac{\partial J}{\partial a_j^{(l)}} f'\left(z_j^{(l)}\right) \\ &= \left(\sum_{k=1}^{d^{(l+1)}} \frac{\partial J}{\partial z_k^{(l+1)}} \frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}}\right) f'\left(z_j^{(l)}\right) \\ &= \left(\sum_{k=1}^{d^{(l+1)}} e_k^{(l+1)} w_{jk}^{(l+1)}\right) f'\left(z_j^{(l)}\right) \\ &= \left(\mathbf{w}_{j:}^{(l+1)} \mathbf{e}^{(l+1)}\right) f'\left(z_j^{(l)}\right) \end{aligned}$$

  with $\mathbf{e}^{(l+1)} = (e_1^{(l+1)}, e_2^{(l+1)}, \ldots, e_{d^{(l+1)}}^{(l+1)})^T$.

Consequently, we obtain the algorithm of Backward Propagation Vectorization:

- Implementing the Forward Propagation Vectorization.
- For output layer, compute
  + $\mathbf{e}^{(L)} = \nabla_{\mathbf{z}^{(L)}} J \in \mathbb{R}^{d^{(L)}}$,
  + $\nabla_{\mathbf{W}^{(L)}} J = \mathbf{a}^{(L-1)} \mathbf{e}^{(L)T} \in \mathbb{R}^{d^{(L-1)} \times d^{(L)}}$,
  + $\nabla_{\mathbf{b}^{(L)}} J = \mathbf{e}^{(L)}$.
- For $l = L-1, L-2, \ldots, 1$, compute

  $$\mathbf{e}^{(l)} = \left(\mathbf{W}^{(l+1)} \mathbf{e}^{(l+1)}\right) \odot f'\left(\mathbf{z}^{(l)}\right)$$

  where the operator $\odot$ is Hadamard product.
- Updating the partial derivative for weight matrices and bias vector:
  + $\nabla_{\mathbf{W}^{(l)}} J = \mathbf{a}^{(l-1)} \mathbf{e}^{(l)T} \in \mathbb{R}^{d^{(l-1)} \times d^{(l)}}$,
  + $\nabla_{\mathbf{b}^{(l)}} J = \mathbf{e}^{(l)}$.

Finally, we obtain the algorithm of Batch (or Mini-Batch) Forward Propagation:

- Implementing the Batch (or Mini-Batch) Forward Propagation.
- For output layer, compute
  + $\mathbf{E}^{(L)} = \nabla_{\mathbf{Z}^{(L)}} J$,
  + $\nabla_{\mathbf{W}^{(L)}} J = \mathbf{A}^{(L-1)} \mathbf{E}^{(L)T}$,
  + $\nabla_{\mathbf{B}^{(L)}} J = \left(\underbrace{\mathbf{e}^{(L)}, \ldots, \mathbf{e}^{(L)}}_{N \text{ columns}}\right)$.
- For $l = L-1, L-2, \ldots, 1$, compute

  $$\mathbf{E}^{(l)} = \left(\mathbf{W}^{(l+1)} \mathbf{E}^{(l+1)}\right) \odot f'\left(\mathbf{Z}^{(l)}\right)$$

  where the operator $\odot$ is Hadamard product.
- Updating the partial derivative for weight matrices and bias matrices:
  + $\nabla_{\mathbf{W}^{(l)}} J = \mathbf{A}^{(l-1)} \mathbf{E}^{(l)T}$,
  + $\nabla_{\mathbf{B}^{(l)}} J = \left(\underbrace{\mathbf{e}^{(l)}, \ldots, \mathbf{e}^{(l)}}_{N \text{ columns}}\right)$.

Using the gradient descent method to update all elements $w_{ij}^{(l)}$, we have

$$w_{ij}^{(l)} := w_{ij}^{(l)} - \eta \frac{\partial J}{\partial w_{ij}^{(l)}},$$

where η is learning rate. Similarly, we will update all element $b_j^{(l)}$ by formula

$$b_j^{(l)} := b_j^{(l)} - \eta \frac{\partial J}{\partial b_j^{(l)}}$$

3 Boston Housing Dataset

This dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Mass. The dataset is small in size with only 506 cases. There are 14 attributes in each case of the dataset. They are

- CRIM: per capita crime rate by town
- ZN: proportion of residential land zoned for lots over $25,000$ sq.ft.
- INDUS: proportion of non-retail business acres per town.
- CHAS: Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX: nitric oxides concentration (parts per 10 million)
- RM: average number of rooms per dwelling
- AGE: proportion of owner-occupied units built prior to 1940
- DIS: weighted distances to five Boston employment centres
- RAD: index of accessibility to radial highways
- TAX: full-value property-tax rate per $\$10,000$
- PTRATIO: pupil-teacher ratio by town
- B: $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town
- LSTAT: $\%$ lower status of the population
- MEDV: Median value of owner-occupied homes in $\$1000's$

Our goal is to use multilayer perceptron model to predict values of MEDV.