Back Propagation Basics

Gerome Yoo

Department of Statistics, Sungkyunkwan University

2018

INDEX

Single Unit Back Propagation

Artificial Neuron(=Single Unit)

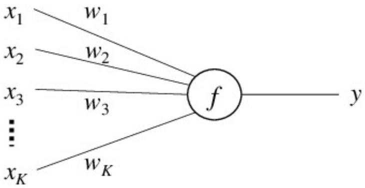

Figure: An artificial neuron

- $\{x_1, \cdots, x_K\}$: input values
- $\{w_1, \cdots, w_K\}$: weights
- y : scalar output
- f : link function (aka activation/decision/transfer function)

Basic Mechanism of Artificial Neuron

The way unit works :

$$y = f(u)$$

where u is a scalar number,
which is the net input(or "new input") of neuron.

How u is defined

$$u = \sum_{i=0}^{K} w_i x_i = \mathbf{w}^T \mathbf{x}$$

Note : here we ignore the bias term in u.
To include a bias term, one can simply add an input dimension (e.g., x_0) that is constant 1.

LINK FUNCTION

- Different Link functions result in distinct neuron behaviors.

Unit step function(Heaviside step function)

$$f(u) = \begin{cases} 1 & \textit{if } u > 0 \\ 0 & \textit{otherwise} \end{cases}$$

- *Perceptron* is a neuron with Unist step function as link.
- Perceptron algorithm is Learning algorithm for perceptron.

Update Equation for Perceptron

$$w^{(new)} := w^{(old)} - \eta \cdot (y - t) \cdot x$$

- where t is label(gold standard), η is learning rate ($\eta > 0$).
- Perceptron is a linear classifier.
  $\Longleftrightarrow$ Limited Description Capacity.

Link Function

Logistic Function(Sigmoid Function)

$$\sigma(u) = \frac{1}{1 + e^{-u}}$$

Properties of Logistic Function

- The output y is always between 0 and 1.
- Unlike Heaviside, $\sigma(u)$ is smooth and differentiable.
- $\sigma(-u) = 1 - \sigma(u)$
- $\frac{d\sigma(u)}{du} = \sigma(u)\sigma(-u)$

Link Function

Proof for $\sigma(-u) = 1 - \sigma(u)$

$$\sigma(-u) = \frac{1}{1+e^{u}} = \frac{e^{-u}}{e^{-u}+1} = \frac{1+e^{-u}}{1+e^{-u}} - \frac{1}{1+e^{-u}}$$
$$= 1 - \frac{1}{1+e^{-u}} = 1 - \sigma(u)$$

Proof for $\frac{d\sigma(u)}{du} = \sigma(u)\sigma(-u)$

$$\frac{d\sigma(u)}{du} = \frac{d}{du}\left(\frac{1}{1+e^{-u}}\right) = \frac{d}{du}(1+e^{-u})^{-1} = -(1+e^{-u})^{-2}(-e^{-u})$$
$$= (1+e^{-u})^{-1}(1+e^{-u})^{-1}(e^{-u}) = \frac{1}{1+e^{-u}} \cdot \frac{e^{-u}}{1+e^{-u}}$$
$$= \frac{1}{1+e^{-u}} \cdot \frac{1}{1+e^{u}} = \sigma(u)\sigma(-u)$$

Update Equation

Learning Algorithm = Stochastic Gradient Model

- Define error function.
  Note that, Error = Cost = Loss = Objective

$$E = \frac{1}{2}(t-y)^2$$

- We take derivative of E with regard to w_i

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial w_i}$$
$$= (y-t) \cdot y(1-y) \cdot x_i$$

- Applying Stochastic Gradient Descent will be :

$$w^{(new)} := w^{(old)} - \eta \cdot (y-t) \cdot y(1-y) \cdot x_i$$

Update Equation

Proof for $\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial w_i} = (y - t) \cdot y(1 - y) \cdot x_i$

- $\frac{\partial E}{\partial y} = \frac{1}{2} \cdot 2(t - y) \cdot (-1) = (y - t)$
- $\frac{\partial y}{\partial u} = \frac{\partial \sigma(u)}{\partial u} = \sigma(u)\sigma(-u) = y(1 - y)$
  Since $y = f(u) = \sigma(u)$, $\frac{d\sigma(u)}{du} = \sigma(u)\sigma(-u)$
  and $\sigma(-u) = 1 - \sigma(u)$
- $\frac{\partial u}{\partial w_i} = x_i$ since $u = \mathbf{w}^T \mathbf{x}$

Multi-Layer Network Back Propagation

Multi-layer neural network

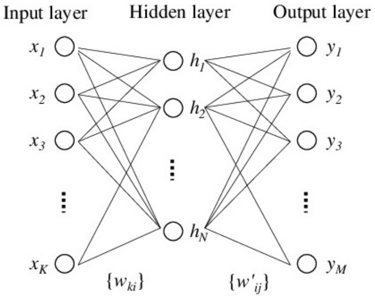

Figure: Multi-layer neural network with one hidden layer

MULTI-LAYER NETWORK BACK PROPAGATION

- $\{x_k\} = \{x_1, \cdots, x_K\}$: input layer
- $\{h_i\} = \{h_1, \cdots, h_N\}$: hidden layer
- $\{y_j\} = \{y_1, \cdots, y_M\}$: output layer
- u_i : net input of hidden layer
- u_j : net input of output layer
- w_{ki} : weights between the input and hidden layer
- w'_{ij} : weights between the hidden and ouput layer

- Logistic function is used as Link function.

Update Equation

Unit h_i in hidden layer

$$h_i = \sigma(u_i) = \sigma\left(\sum_{k=1}^{K} w_{ki} x_k\right)$$

where u_i : net input of hidden layer.

Unit y_j in output layer

$$y_j = \sigma(u'_j) = \sigma\left(\sum_{i=1}^{N} w'_{ij} h_i\right)$$

where u_j : net input of output layer.

Update Equation

Squared sum error function

$$E(\mathbf{x},\mathbf{t},\mathbf{W},\mathbf{W}') = \frac{1}{2}\sum_{j=1}^{M}(y_j - t_j)^2$$

- $\mathbf{W} = w_{ki}$: a K $\times$ N weight matrix (input-hidden)
- $\mathbf{W}' = w'_{ij}$: a N $\times$ M weight matrix (hidden-output)
- $\mathbf{t} = \{\ t_1, \cdots t_M\ \}$: a M-dimension vector
  = gold-standard labels of output

UPDATE EQUATION

Derivation Process

- To obtain update equation for w_{ki} and w_{ij},
  take derivative of E regard to weights respectively.
- Derivation start from right-most layer(output)
  and move left to make derivation straight forward.
- For each layer, computation split into 3 steps.
  $\rightarrow$ Derivative of error regard to output, net input, weight

Output Layer

1st Step

- Take derivative of error w.r.t. output.

$$\frac{\partial E}{\partial y_j} = y_j - t_j$$

2nd Step

- Take derivative of error w.r.t. net input of output layer.

$$\frac{\partial E}{\partial u'_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial u'_j} = (y_j - t_j) \cdot y_j(1 - y_j) := EI'_j$$

Output Layer

3rd Step

- Take derivative of error w.r.t. weight between hidden-output layer.

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u'_j} \cdot \frac{\partial u'_j}{\partial w'_{ij}} = EI'_j \cdot h_i$$

Output Layer

Update Equation for weights between hidden-output layer

$$
\begin{aligned}
{w'_{ij}}^{(new)} &= {w'_{ij}}^{(old)} - \eta \cdot \frac{\partial E}{\partial w'_{ij}} \\
&= {w'_{ij}}^{(old)} - \eta \cdot EI'_j \cdot h_i
\end{aligned}
$$

- where $\eta > 0$ is the learning rate.

Hidden Layer

1st Step

- Take derivative of error w.r.t. hidden.

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{M} \frac{\partial E}{\partial u'_j} \cdot \frac{\partial u'_j}{\partial h_i} = \sum_{j=1}^{M} EI'_j \cdot w'_{ij}$$

2nd Step

- Take derivative of error w.r.t. net input of hidden layer.

$$\frac{\partial E}{\partial u_i} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial u_i} = \sum_{j=1}^{M} EI'_j \cdot w'_{ij} \cdot h_i(1 - h_i) := EI_i$$

Hidden Layer

3rd Step

- Take derivative of error w.r.t. weight between input-hidden layer.

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial u_i} \cdot \frac{\partial u_i}{\partial w_{ki}} = EI_i \cdot x_k$$

Hidden Layer

Update Equation for weights between hidden-output layer

$$w_{ki}^{(new)} = w_{ki}^{(old)} - \eta \cdot \frac{\partial E}{\partial w_{ki}}$$
$$= w_{ki}^{(old)} - \eta \cdot EI_i \cdot x_k$$

- where $\eta > 0$ is the learning rate.

Back Propagation

- From above, we can see intermediate results EI'_j, when computing the derivatives for one layer can be reused for the previous one.
- Imagine, there were another layer prior to input layer. Then, EI'_j can also be reused for derivation efficiency.
- For single unit,

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial w_i} = (y - t) \cdot y(1 - y) \cdot x_i$$

Back Propagation

- For hidden-output layer,

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u'_j} \cdot \frac{\partial u'_j}{\partial w'_{ij}} = EI'_j \cdot h_i$$

- For input-hidden layer,

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial u_i} \cdot \frac{\partial u_i}{\partial w_{ki}} = EI_i \cdot x_k = \sum_{j=1}^{M} EI'_j \cdot w'_{ij} \cdot h_i(1 - h_i) \cdot x_k$$

- $\sum_{j=1}^{M} EI'_j \cdot w'_{ij}$ is like "error" of the hidden layer unit h_i.
- We may interpret this term "back-propagated" from next layer. This propagation may go back further.

REFERENCE

- word2vec Parameter Learning Explained
- 신경망 첫걸음 by 타리크 라시드