Feed Forward Neural Network

Masahiro Ogawa

January 20, 2016

1 Notation

- $\boldsymbol{x}$: vector
- $\boldsymbol{A}$: matrix
- ${}^t\boldsymbol{A}$: transpose of $\boldsymbol{A}$
- $f.(\boldsymbol{A}) = \begin{pmatrix} f(a_{1,1}) & f(a_{1,2}) & \cdots & f(a_{1,n}) \\ f(a_{2,1}) & f(a_{2,2}) & \cdots & f(a_{2,n}) \\ \vdots & \vdots & \ddots & \vdots \\ f(a_{m,1}) & f(a_{m,2}) & \cdots & f(a_{m,n}) \end{pmatrix}$
- $(\boldsymbol{A}. * \boldsymbol{B})_{i,j} = a_{i,j} b_{i,j}$
- $\mathbf{1}_{m\times n} = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}$: $m \times n$ dim
- $\tilde{\boldsymbol{y}} = \begin{pmatrix} 1 \\ \boldsymbol{y} \end{pmatrix}$: homogeneous vector
- n : data number
- N : total number of data
- l : layer number
- L : output layer number. thus, the total number of layer is $L + 1$
- k : learning iteration number
- K : final number of the learning iteration
- J : cost
- $f(x)$: activation function

- $\boldsymbol{X}_0 = (\boldsymbol{x}_{0,1}, \ldots, \boldsymbol{x}_{0,N}) = \begin{pmatrix} x_{0,1,1} & \cdots & x_{0,1,N} \\ \vdots & \ddots & \vdots \\ x_{0,d_0,1} & \cdots & x_{0,d_0,N} \end{pmatrix} : d_0 \times N$ dim : input vectors
- $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_N) = \begin{pmatrix} b_{1,1} & \cdots & b_{1,N} \\ \vdots & \ddots & \vdots \\ b_{d_L,1} & \cdots & b_{d_L,N} \end{pmatrix} : d_L \times N$ dim : instruction signals
- $\tilde{\boldsymbol{W}}_l(k) = (\tilde{\boldsymbol{w}}_{l,1}(k), \ldots, \tilde{\boldsymbol{w}}_{l,d_{l+1}}(k)) = \begin{pmatrix} \tilde{w}_{l,0,1}(k) & \cdots & \tilde{w}_{l,0,d_{l+1}}(k) \\ \vdots & \ddots & \vdots \\ \tilde{w}_{l,d_l,1}(k) & \cdots & \tilde{w}_{l,d_l,d_{l+1}}(k) \end{pmatrix}$: weights : $(d_l + 1) \times d_{l+1}$ dim

2 Input

parameters

- L : output layer number.
- $\{d_l\}_{l=0}^{L}$: neuron numbers in each layer l
- $\{\tilde{\boldsymbol{W}}_l(0)\}_{l=0}^{L} \mid \tilde{\boldsymbol{W}}_l(0) = (\tilde{\boldsymbol{w}}_{l,1}(0), \ldots, \tilde{\boldsymbol{w}}_{l,d_{l+1}}(0)) : (d_l + 1) \times d_{l+1}$ dim : initial weights
- ρ : learning rate parameter
- T_e : error threshold
- T_K : threshold of the maximum number of iteration

data

- $\boldsymbol{X}_0 = (\boldsymbol{x}_{0,1}, \ldots, \boldsymbol{x}_{0,N}) : d_0 \times N$ dim : input vectors
- $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_N) : d_L \times N$ dim : instruction signals

3 Output

- $\{\tilde{\boldsymbol{W}}_l(K)\}_{l=0}^{L} \mid \tilde{\boldsymbol{W}}_l(K) = (\tilde{\boldsymbol{w}}_{l,1}(K), \ldots, \tilde{\boldsymbol{w}}_{l,d_{l+1}}(K)) : (d_l + 1) \times d_{l+1}$ dim : final (after Kth iteration) learned weights
- $\boldsymbol{Y}_L = (\boldsymbol{y}_{L,1}, \ldots, \boldsymbol{y}_{L,N}) : dL \times N$ dim : output vectors
- $J = \sum_{n=1}^{N} J_n$: cost, final error

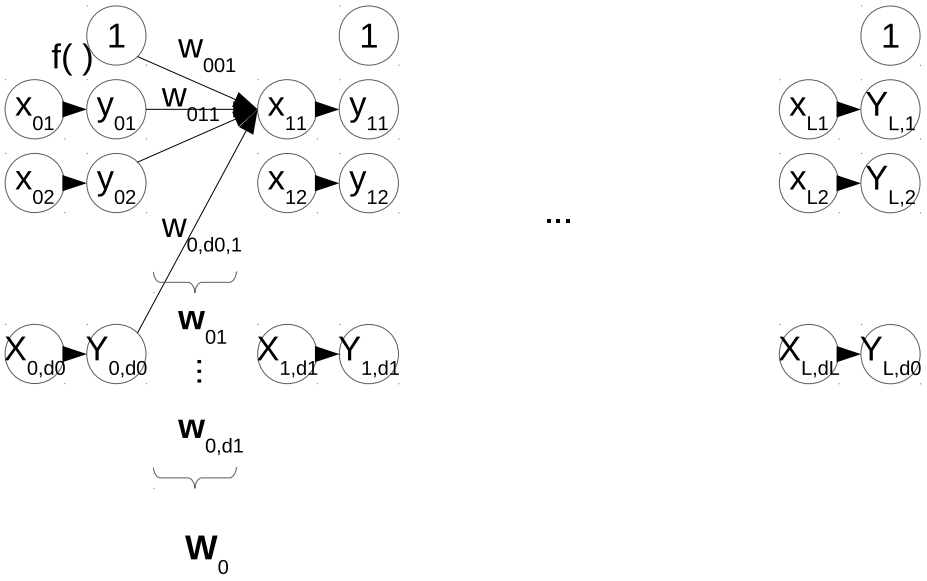

Figure 1: Neural network architecture

4 Algorithm(Iterative version, Stochastic Gradient descent)

4.1 Iteration

Iterate feed forward and back propagation algorithm for each input datum until the error become smaller than the threshold or the iteration number exceeds the maximum iteration number. That is,

$$J < T_e, \text{ or } k < T_K. \tag{1}$$

The iteration means update of the weights as,

$$\tilde{w}_{lij}(k+1) = \tilde{w}_{lij}(k) - \rho \frac{\partial J_n}{\partial \tilde{w}_{lij}(k)} \tag{2}$$

$$\iff \tilde{\boldsymbol{W}}_l(k+1) = \tilde{\boldsymbol{W}}_l(k) - \rho \frac{\partial J_n}{\partial \tilde{\boldsymbol{W}}_l(k)} \tag{3}$$

From this equation, once we could compute $\frac{\partial J_n}{\partial \tilde{\boldsymbol{W}}_l(k)}$, we can update the weights. Therefore we concentrate on computing $\frac{\partial J_n}{\partial \tilde{\boldsymbol{W}}_l(k)}$ below.

4.2 Feed Forward

The below are true for all data and all iteration. So we omit the data number n and iteration number k as e.g. $\boldsymbol{x}_{l,n} = \boldsymbol{x}_l, \tilde{\boldsymbol{W}}_l(k) = \tilde{\boldsymbol{W}}_l$.

$$\boldsymbol{x}_l = {}^t\tilde{\boldsymbol{W}}_{l-1}\tilde{\boldsymbol{y}}_{l-1} \tag{4}$$

$$\boldsymbol{y}_l = f.(\boldsymbol{x}_l) \tag{5}$$

$$\tag{6}$$

4.2.1 activation function for hidden layers

$$f(x) = \begin{cases} (1+e^{-x})^{-1} : \text{logistic function} \\ \tanh(x) \end{cases} \tag{7}$$

4.2.2 activation function for an output layer

To fit output value as desired, we have to choose appropriate activation function for an output layer.

$$f(x) = \begin{cases} x & : \text{identity function for regression} \\ (1+e^{-x})^{-1} & : \text{logistic function for [0,1] output binary classification} \\ \frac{\exp(x)}{\sum_j \exp(x_{Lj})} & : \text{soft max function for multiclass classification} \\ \tanh(x) & : \text{for [-1,1] output} \end{cases} \tag{8}$$

Note that we divide binary and multiclass case, because if we use soft max function for binary class classification, $f(x)$ always become 1.

4.3 Back Propagation

In this section, we compute $\frac{\partial J_n}{\partial \tilde{\boldsymbol{W}}_{l-1}(k)}$ instead of $\frac{\partial J_n}{\partial \tilde{\boldsymbol{W}}_l(k)}$ for just index convenience. And we omit n as $J_n = J$ in this section.

$$
\begin{aligned}
\frac{\partial J}{\partial \tilde{w}_{l-1,ij}} &= \frac{\partial J}{\partial x_{lj}} \frac{\partial x_{lj}}{\partial \tilde{w}_{l-1,ij}} && (9)\\
&= \epsilon_{lj} \frac{\partial (\sum_m \tilde{w}_{l-1,mj} \tilde{y}_{l-1,m})}{\partial \tilde{w}_{l-1,ij}} \quad | \ \epsilon_{lj} := \frac{\partial J}{\partial x_{lj}} && (10)\\
&= \epsilon_{lj} \tilde{y}_{l-1,i} && (11)\\
\iff \frac{\partial J}{\partial \tilde{\boldsymbol{W}}_{l-1}} &= \tilde{\boldsymbol{y}}_{l-1}{}^t \boldsymbol{\epsilon_l} && (12)\\
\Rightarrow \tilde{\boldsymbol{W}}_{l-1}(k+1) &= \tilde{\boldsymbol{W}}_{l-1}(k) - \rho \tilde{\boldsymbol{y}}_{l-1}{}^t \boldsymbol{\epsilon}_l && (13)\\
\epsilon_{lj} &= \frac{\partial J}{\partial x_{lj}} && (14)\\
&= \frac{\partial J}{\partial y_{lj}} \frac{\partial y_{lj}}{\partial x_{lj}} && (15)\\
&= \frac{\partial J}{\partial y_{lj}} \frac{\partial f(x_{lj})}{\partial x_{lj}} && (16)\\
& && (17)
\end{aligned}
$$

4.3.1 back propagation for hidden layers

$$
\begin{aligned}
\epsilon_{lj} &= \frac{\partial J}{\partial y_{lj}} \frac{\partial f(x_{lj})}{\partial x_{lj}} && (18)\\
&= \sum_h \frac{\partial J}{\partial x_{l+1,h}} \frac{\partial x_{l+1,h}}{\partial y_{lj}} f'(x_{lj}) && (19)\\
&= \sum_h \epsilon_{l+1,h} \frac{(\sum_m w_{lmh} y_{lm} + w_{l0h})}{\partial y_{lj}} f'(x_{lj}) && (20)\\
&= \sum_h \epsilon_{l+1,h} w_{ljh} f'(x_{lj}) && (21)\\
\Rightarrow \boldsymbol{\epsilon}_l &= (\boldsymbol{W}_l \boldsymbol{\epsilon}_{l+1}) .* f'.(\boldsymbol{x}_l) && (22)\\
&= \begin{cases} (\boldsymbol{W}_l \boldsymbol{\epsilon}_{l+1}) .* (\boldsymbol{y}_l - \boldsymbol{y}_l .* \boldsymbol{y}_l) & \text{if } f(x) = (1+e^{-x})^{-1} \\ (\boldsymbol{W}_l \boldsymbol{\epsilon}_{l+1}) .* (\boldsymbol{1}_{dl} - \boldsymbol{y}_l .* \boldsymbol{y}_l) & \text{if } f(x) = \tanh(x) \end{cases} && (23)
\end{aligned}
$$

4.3.2 back propagation for an output layer

To compute back propagation for an output layer, we need cost function form. Depend on that, we have to choose appropriate cost function for each problem as below.

$$
J = \begin{cases} \frac{1}{2} \| \boldsymbol{y}_L - \boldsymbol{b} \|^2 & \text{regression} \\ -(b \ln y_L + (1-b) \ln(1-y_L)) & \text{binary classification} \\ -{}^t \boldsymbol{b} \ln \boldsymbol{y}_L & \text{multiclass classification} \end{cases} \quad (24)
$$

Using cost(24) and activation function(8), ϵ_L can be computed as below. Note that $f(x_h)$ is related to x_j, so we have to think about the term $\frac{\partial f(x_{Lh})}{\partial x_{Lj}}$.

$$\begin{aligned}
\epsilon_{Lj} &= \sum_h \frac{\partial J}{\partial y_{Lh}} \frac{\partial f(x_{Lh})}{\partial x_{Lj}} && (25)\\
&= \begin{cases} \sum_h (\frac{\partial (1/2\|\boldsymbol{y}_L - \boldsymbol{b}\|^2)}{\partial y_{Lh}}) \frac{\partial x_{Lh}}{\partial x_{Lj}} & \text{regression} \\ -(\frac{b}{y_L} - \frac{1-b}{1-y_L}) \frac{\partial}{\partial x_L} (1+e^{-x_L})^{-1} & \text{binary classification} \\ -\sum_h \frac{b_h}{y_{L,h}} \frac{\partial}{\partial x_{Lj}} (\frac{\exp(x_{Lh})}{\sum_i \exp(x_{Li})}) & \text{multiclass classification} \end{cases} && (26)\\
&= \begin{cases} \sum_h (y_{Lh} - b_h)\delta_{h,j} & \\ -(\frac{b}{y_L} - \frac{1-b}{1-y_L})(y_L(1-y_L)) & \\ -\sum_h \frac{b_h}{y_{L,h}} (\delta_{h,j} y_{Lj} - y_{Lh} y_{Lj}) & \end{cases} && (27)\\
&= y_{Lj} - b_j && (28)
\end{aligned}$$

We use the softmax property $\sum_h b_h = 1$ in the last equation of multi-class classification case.

In both case the result becomes $y - b$, so there is no gradient diminishing problem. That's why we use these combination of the cost and final layers activation function.

4.3.3 final result

Finally, we get below results.

$$\begin{aligned}
\boldsymbol{\epsilon}_l &= \begin{cases} \boldsymbol{y}_L - \boldsymbol{b} & \text{if } l = L \\ (\boldsymbol{W}_l \boldsymbol{\epsilon}_{l+1}) .* f'.(\boldsymbol{x}_l) & \text{otherwise} \end{cases} && (29)\\
&= \begin{cases} \boldsymbol{y}_L - \boldsymbol{b} & \\ (\boldsymbol{W}_l \boldsymbol{\epsilon}_{l+1}) .* (\mathbf{1}_{dl} - \boldsymbol{y}_l .* \boldsymbol{y}_l) & \text{if } f(x) = \tanh(x) \end{cases} && (30)\\
\tilde{\boldsymbol{W}}_{l-1}(k+1) &= \tilde{\boldsymbol{W}}_{l-1}(k) - \rho\ \tilde{\boldsymbol{y}}_{l-1}{}^t \boldsymbol{\epsilon}_l && (31)
\end{aligned}$$

5 Algorithm(Matrix version)

5.1 Iteration

Iterate feed forward algorithm for all input data and back propagation for all data until the error become smaller than the threshold or the iteration number exceeds the maximum iteration number. That is,

$$J < T_e, \text{ or } k < T_K. \tag{32}$$

5.2 Feed Forward

The below are true for all iteration. So we omit the iteration number as e.g. $\tilde{\boldsymbol{W}}_l(i) = \tilde{\boldsymbol{W}}_l$.

$$\boldsymbol{X}_l = {}^t\tilde{\boldsymbol{W}}_{l-1}\tilde{\boldsymbol{Y}}_{l-1} \tag{33}$$

$$\boldsymbol{Y}_l = f.(\boldsymbol{X}_l) \tag{34}$$

$f(x)$ is defined by (7) and (8).

5.3 Back Propagation

$$J = \begin{cases} \frac{1}{2}\mathrm{Tr}({}^t(\boldsymbol{Y}_L - \boldsymbol{B})(\boldsymbol{Y}_L - \boldsymbol{B})) & \text{regression} \\ -\mathrm{Tr}\{\begin{pmatrix} {}^t\boldsymbol{B} \\ \boldsymbol{1}_{1\times N} - {}^t\boldsymbol{B} \end{pmatrix} \ln.(\boldsymbol{Y}_L \quad \boldsymbol{1}_{N\times 1} - \boldsymbol{Y}_L)\} & \text{binary classification} \\ -\mathrm{Tr}({}^t\boldsymbol{B} \ln.(\boldsymbol{Y}_L)) & \text{multiclass classification} \end{cases} \tag{35}$$

$$\boldsymbol{E}_l = (\boldsymbol{\epsilon}_{l,1}, \cdots, \boldsymbol{\epsilon}_{l,N}) = \begin{cases} \boldsymbol{Y}_L - \boldsymbol{B} & \text{if } l = L \\ (\boldsymbol{W}_l\boldsymbol{E}_{l+1}).*f'.(\boldsymbol{X}_l) & \text{otherwise} \end{cases} \tag{36}$$

$$= \begin{cases} \boldsymbol{Y}_L - \boldsymbol{B} \\ (\boldsymbol{W}_l\boldsymbol{E}_{l+1}).*(\boldsymbol{1}_{dl\times N} - \boldsymbol{Y}_l.*\boldsymbol{Y}_l) & \text{if } f(x) = \tanh(x) \end{cases} \tag{37}$$

$$\tilde{\boldsymbol{W}}_{l-1}(k+1) = \tilde{\boldsymbol{W}}_{l-1}(k) - \rho\sum_{n=1}^{N}\tilde{\boldsymbol{y}}_{l-1,n}{}^t\boldsymbol{\epsilon}_{l,n} \tag{38}$$

$$= \tilde{\boldsymbol{W}}_{l-1}(k) - \rho\,\tilde{\boldsymbol{Y}}_{l-1}{}^t\boldsymbol{E}_l \tag{39}$$