Neural Network Derivatives

Table of Contents

1 Preface

This is my personal collection of neural network derivatives. I started this document during the machine learning course at Universidade de São Paulo, when I created the algorithms from scratch.

Derivatives are the cornerstone of neural networks, they're essential part of backpropagation. The algorithm doesn't converge if the any small detail is missing or misplaced. It requires time, patience and practice to master the skills of calculating derivatives.

Well, someone might ask: “why should I bother learning derivatives and building neural networks from scratch as there're plenty of libraries available?”. I would say for a couple of good reasons. While off the shelve libraries usually are optimized for speed, understanding how they work internally is essential when setting up the parameters and selecting the architecture, and becomes crucial for debuging or optimizing the model for speed or handling large datasets.

Besides this, the frequency that commercial libraries are updated is too long after new models are published. The fast development of machine learning algorithms in recent years leaves commercial libraries well behind the scientific research and the academia. Being able to deploy new models soon after a paper is released gives enormous advantage.

I use this document to implement network architectures from scientific publications, and as a reference for my codes. I tested all derivatives in my code, which are available in my GitHub account. If you find any bug or typo, please submit a pull request. I hope you enjoy this document as much as I did writing it.

August, 2020

Victor Ivamoto

Sao Paulo, Brazil

2 Common Derivatives

Sigmoid

$$f(x)=\frac{1}{1+e^{-x}}\rightarrow\frac{d\ f(x)}{dx}=f(x)(1-f(x)) \tag{2.1}$$

Softmax

$$f(x)=\frac{e^{x_i}}{\sum_{j=1}^{N}e^{x_j}}\rightarrow\frac{d\ f(x)}{dx}=f(x_i)(\delta_{ij}-f(x_j)),\delta_{ik}=\begin{cases}1 & if\ i=j\\ 0 & if\ i\neq j\end{cases} \tag{2.2}$$

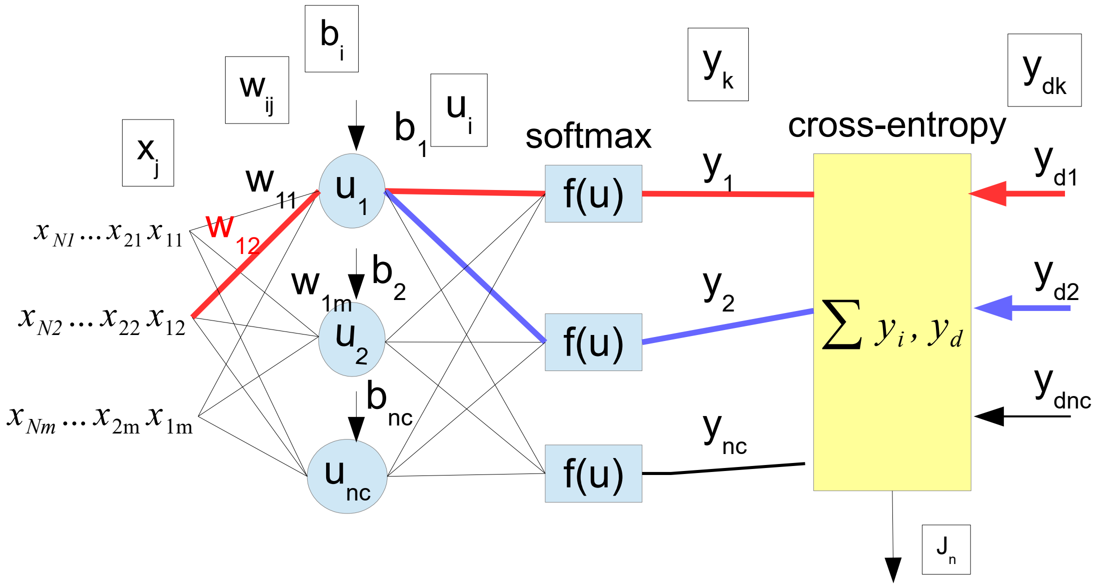
Hyperbolic tangent

$$f(x)=\tanh(x)\rightarrow\frac{d\ f(x)}{dx}=1-\tanh^2(x) \tag{2.3}$$

Common Functions	**Function**	**Derivative**
Constant	c	0
Line	x	1
	ax	a
Square	x^2	2x
Square Root	$\sqrt{x}$	$\frac{1}{2\sqrt{x}}$
Exponential	e^x	e^x
	a^x	$\ln(a)a^x$
Logarithms	$\ln(x)$	$\frac{1}{x}$
	$\log_a(x)$	$\frac{1}{x\ln(a)}$

Rules	**Function**	**Derivative**
Multiplication by constant	$c\,f(x)$	$c\,f'(x)$
Power Rule	x^n	$n\,x^{n-1}$
Sum Rule	$f(x)+g(x)$	$f'(x)+g'(x)$
Difference Rule	$f(x)-g(x)$	$f'(x)-g'(x)$
Product Rule	$f'(x)+g'(x)$	$f'(x)g(x)+f(x)g'(x)$
Quotient Rule	$\frac{f(x)}{g(x)}$	$\frac{f'(x)g(x)-f(x)g'(x)}{g^2(x)}$
Reciprocal Rule	$\frac{1}{f(x)}$	$\frac{-f'(x)}{f^2(x)}$
Chain Rule		$\frac{dy}{dx}=\frac{dy}{du}\frac{du}{dx}$

3 Perceptron



Dimensions	**Definitions**
X: N x m	W_{ij} : connects u_i to x_j
w: nc x m	nc: number of classes
u: nc x 1	m: number of attributes
b: nc x 1	N: number of instances
y: N x nc	
y_d: N x nc	

Output equations

$$u_i=\sum_{j=1}^{m} x_j w_{ij}+b_1 \tag{3.1}$$

$$y_k=f(u)=f(u_{1,}\dots,u_{nc})=\frac{e^{u_k}}{\sum_{t=1}^{nc} e^{u_t}} \tag{3.2}$$

$$J(n)=-\sum_{k=1}^{nc} y_{d_k}\log y_k \tag{3.3}$$

$$J_T=\frac{1}{N}\sum_{n=1}^{N} J(n) \tag{3.4}$$

Back propagation

$$\frac{\partial J}{\partial w_{ij}}=\frac{\partial J}{\partial y_k}\frac{\partial y_k}{\partial u_i}\frac{\partial u_k}{\partial w_{ij}} \tag{3.5}$$

$$\frac{\partial J}{\partial y_k}=-\sum_{k=1}^{nc}\frac{y_{dk}}{y_k} \tag{3.6}$$

For $i=k$

$$\frac{\partial y_k}{\partial u_i}=\frac{e^{u_k}\sum_{t=1}^{nc} e^{u_t}-e^{u_k}e^{u_i}}{\left(\sum_{t=1}^{nc} e^{u_t}\right)^2}=\frac{e^{u_k}\sum_{t=1}^{nc} e^{u_t}}{\left(\sum_{t=1}^{nc} e^{u_t}\right)^2}-\frac{e^{u_k}e^{u_i}}{\left(\sum_{t=1}^{nc} e^{u_t}\right)^2}=y_k-y_k y_i=y_k(1-y_i) \tag{3.7}$$

For $i\neq k$

$$\frac{\partial y_k}{\partial u_i}=\frac{0-e^{u_k}e^{u_i}}{\left(\sum_{t=1}^{nc} e^{u_t}\right)^2}=\frac{-e^{u_k}}{\left(\sum_{t=1}^{nc} e^{u_t}\right)}\cdot\frac{e^{u_i}}{\left(\sum_{t=1}^{nc} e^{u_t}\right)}=-y_k y_i \tag{3.8}$$

Combining (3.7) and (3.8):

$$\frac{\partial y_k}{\partial u_i}=y_k(\delta_{ik}-y_i),\delta_{ik}=\begin{cases}1 & if\ i=k\\ 0 & if\ i\neq k\end{cases} \tag{3.9}$$

$$\frac{\partial u_i}{\partial w_{ij}}=x_j \tag{3.10}$$

$$\frac{\partial u_i}{\partial b_i}=1 \tag{3.11}$$

Combining (3.6), (3.9) and (3.10) into (3.5):

$$\frac{\partial J}{\partial w_{ij}}=-\sum_{k=1}^{nc}\frac{y_{dk}}{y_k}y_k(\delta_{ik}-y_i)x_j=\sum_{k=1}^{nc}y_{dk}(y_i-\delta_{ik})x_j=\sum_{k=1}^{nc}y_{dk}y_ix_j-\sum_{k=1}^{nc}y_{dk}\delta_{ik}x_j \tag{3.12}$$

Since $\sum_{k=1}^{nc}y_{dk}=1$ and replacing δ_{ik} from equation 9 into 12:

$$\frac{\partial J}{\partial w_{ij}}=y_ix_j-y_{di}x_j=(y_i-y_{di})x_j \tag{3.13}$$

Finaly:

$$\frac{\partial J_T}{\partial w_{ij}}=\frac{1}{N}\sum_{n=1}^{N}\frac{\partial J_n}{\partial w_{ij}} \tag{3.14}$$

$$\boxed{\frac{\partial J_T}{\partial w_{ij}}=\frac{1}{N}\sum_{n=1}^{N}(y_i-y_{di})x_j} \tag{3.15}$$

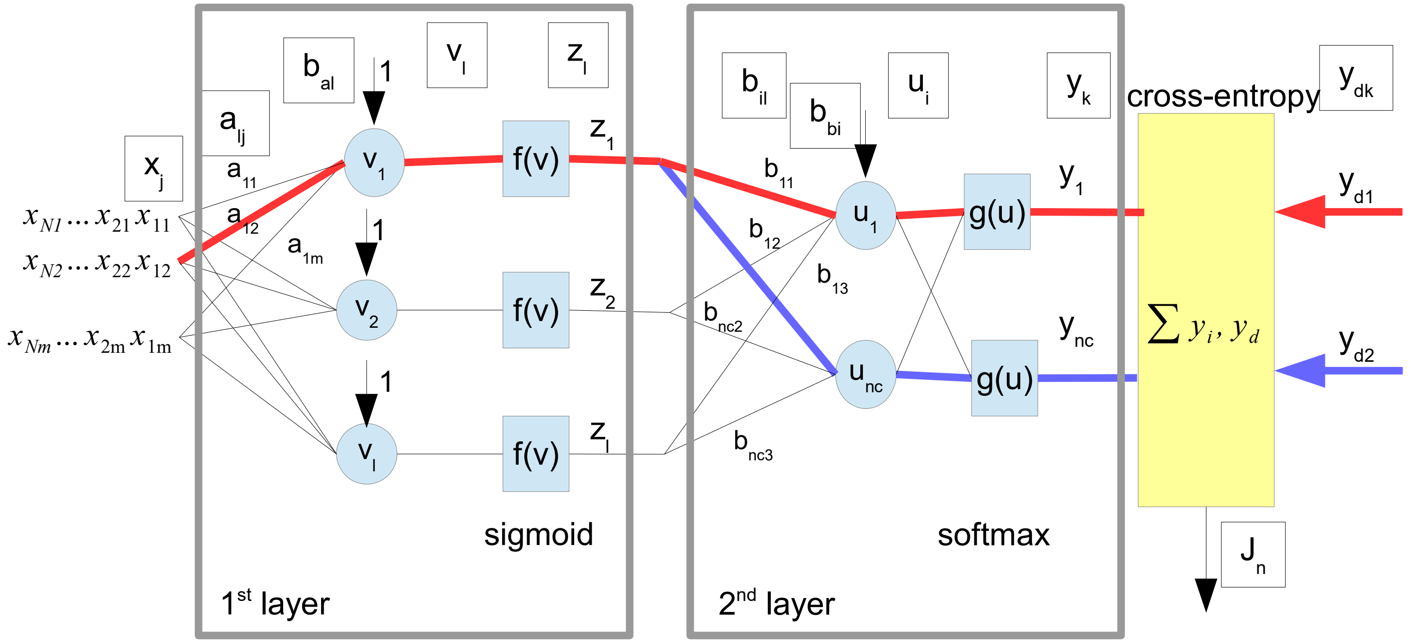
Similarly, we compute the derivative of *J* w.r.t. bias B:

$$\frac{\partial J}{\partial b_i}=-\sum_{k=1}^{nc}\frac{y_{dk}}{y_k}y_k(\delta_{ik}-y_i)=y_i-y_{di} \tag{3.16}$$

$$\frac{\partial J_T}{\partial b_i}=\frac{1}{N}\sum_{n=1}^{N}\frac{\partial J_n}{\partial b_i} \tag{3.17}$$

$$\boxed{\frac{\partial J_T}{\partial b_i}=\frac{1}{N}\sum_{n=1}^{N}(y_i-y_{di})} \tag{3.18}$$

4 Multi Layer Perceptron



Definitions	**Dimensions**
a_{lj} : connects v_l to x_j nc: number of classes m: number of attributes N: number of instances L: number of 1st layer neurons	X: N x m a: L x m b_a: L x 1 v: L x 1 z: L x 1 b: nc x L b_b: nc x 1 u: nc x 1 y: N x nc y_d: N x nc

Output equations

$$v_l = \sum_{j=1}^{m} x_j a_{lj} + b_{al} \tag{4.1}$$

$$z_l = f(v) = \frac{1}{1+e^{-v_l}} \tag{4.2}$$

$$u_i = \sum_{l=1}^{L} z_l b_{il} + b_{bl} \tag{4.3}$$

$$y_k = f(u) = f(u_{1,}\dots, u_{nc}) = \frac{e^{u_k}}{\sum_{t=1}^{nc} e^{u_t}} \tag{4.4}$$

$$J(n) = -\sum_{k=1}^{nc} y_{d_k} \log y_k \tag{4.5}$$

$$J_T = \frac{1}{N} \sum_{n=1}^{N} J(n) \tag{4.6}$$

2nd layer derivatives

$$\frac{\partial J(n)}{\partial b_{il}} = \frac{\partial J(n)}{\partial y_k} \frac{\partial y_k}{\partial u_i} \frac{\partial u_i}{\partial b_{il}} \tag{4.7}$$

$$\frac{\partial J(n)}{\partial y_k} = -\sum_{k=1}^{nc} \frac{y_{d_k}}{y_k} \tag{4.8}$$

$$\frac{\partial y_k}{\partial u_i}=\frac{e^{u_k}\sum_{t=1}^{nc}e^{u_t}-e^{u_k}e^{u_i}}{\left(\sum_{t=1}^{nc}e^{u_t}\right)^2}=y_k-y_k y_i \tag{4.9}$$

$$\frac{\partial y_k}{\partial u_i}=\sum_{k=1}^{nc}y_k(\delta_{ik}-y_i),\,\delta_{ik}=\begin{cases}1 & if\ i=k\\ 0 & if\ i\neq k\end{cases} \tag{4.10}$$

$$\frac{\partial u_i}{\partial b_{il}}=z_l \tag{4.11}$$

Combining (4.8), (4.10) and (4.11) into (4.7):

$$\frac{\partial J(n)}{\partial b_{il}}=-\sum_{k=1}^{nc}\frac{y_{d_k}}{y_k}y_k(\delta_{ik}-y_i)z_l=\sum_{k=1}^{nc}y_{d_k}(y_i-\delta_{ik})z_l \tag{4.12}$$

$$\frac{\partial J(n)}{\partial b_{il}}=\sum_{k=1}^{nc}y_{d_k}y_i z_l-\sum_{k=1}^{nc}y_{d_k}\delta_{ik}z_l \tag{4.13}$$

Since y_d is one-hot encoded, $\sum_{k=1}^{nc}y_{d_k}=1$. Also, $i=k$ if $i=k$ and $\delta_{ik}=0$ otherwise.

$$\frac{\partial J(n)}{\partial b_{il}}=y_i z_l-y_{d_i}z_l=(y_i-y_{d_i})z_l \tag{4.14}$$

Derivative of J w.r.t. weights:

$$\boxed{\frac{\partial J_T}{\partial b_{il}}=\frac{1}{N}\sum_{n=1}^{N}(y_i-y_{d_i})z_l} \tag{4.15}$$

Derivative of J w.r.t. bias:

$$\boxed{\frac{\partial J_T}{\partial b_{bi}}=\frac{1}{N}\sum_{n=1}^{N}(y_i-y_{d_i})} \tag{4.16}$$

1st layer derivatives

$$\frac{\partial J(n)}{\partial a_{lj}}=\frac{\partial J(n)}{\partial y_k}\frac{\partial y_k}{\partial u_i}\frac{\partial u_i}{\partial v_l}\frac{\partial v_l}{\partial a_{lj}} \tag{4.17}$$

$$\frac{\partial u_i}{\partial v_l}=b_{il}z_l(1-z_l) \tag{4.18}$$

$$\frac{\partial v_l}{a_{lj}}=x_j \tag{4.19}$$

Combining (4.8), (4.10), (4.18) and (4.19) into (4.17)

$$\frac{\partial J(n)}{\partial a_{lj}}=-\sum_{k=1}^{nc}\frac{y_{d_k}}{y_k}y_k(\delta_{ik}-y_i)b_{il}z_l(1-z_l)x_j \tag{4.20}$$

$$\frac{\partial J(n)}{\partial a_{lj}}=\sum_{k=1}^{nc}y_{d_k}(y_i-\delta_{ik})b_{il}z_l(1-z_l)x_j=\sum_{k=1}^{nc}y_{d_k}y_ib_{il}z_l(1-z_l)x_j-\sum_{k=1}^{nc}y_{d_k}\delta_{ik}b_{il}z_l(1-z_l)x_j \tag{4.21}$$

If $i=k$

$$\frac{\partial J(n)}{\partial a_{lj}}=\sum_{k=1}^{nc}y_{d_k}y_ib_{il}z_l(1-z_l)x_j-y_{d_i}b_{il}z_l(1-z_l)x_j=(y_i-y_{d_i})[b_{il}z_l(1-z_l)x_j] \tag{4.22}$$

$$\frac{\partial J(n)}{\partial a_{lj}}=(y_i-y_{d_i})[b_{il}z_l(1-z_l)x_j] \tag{4.23}$$

Derivative w.r.t. weights:

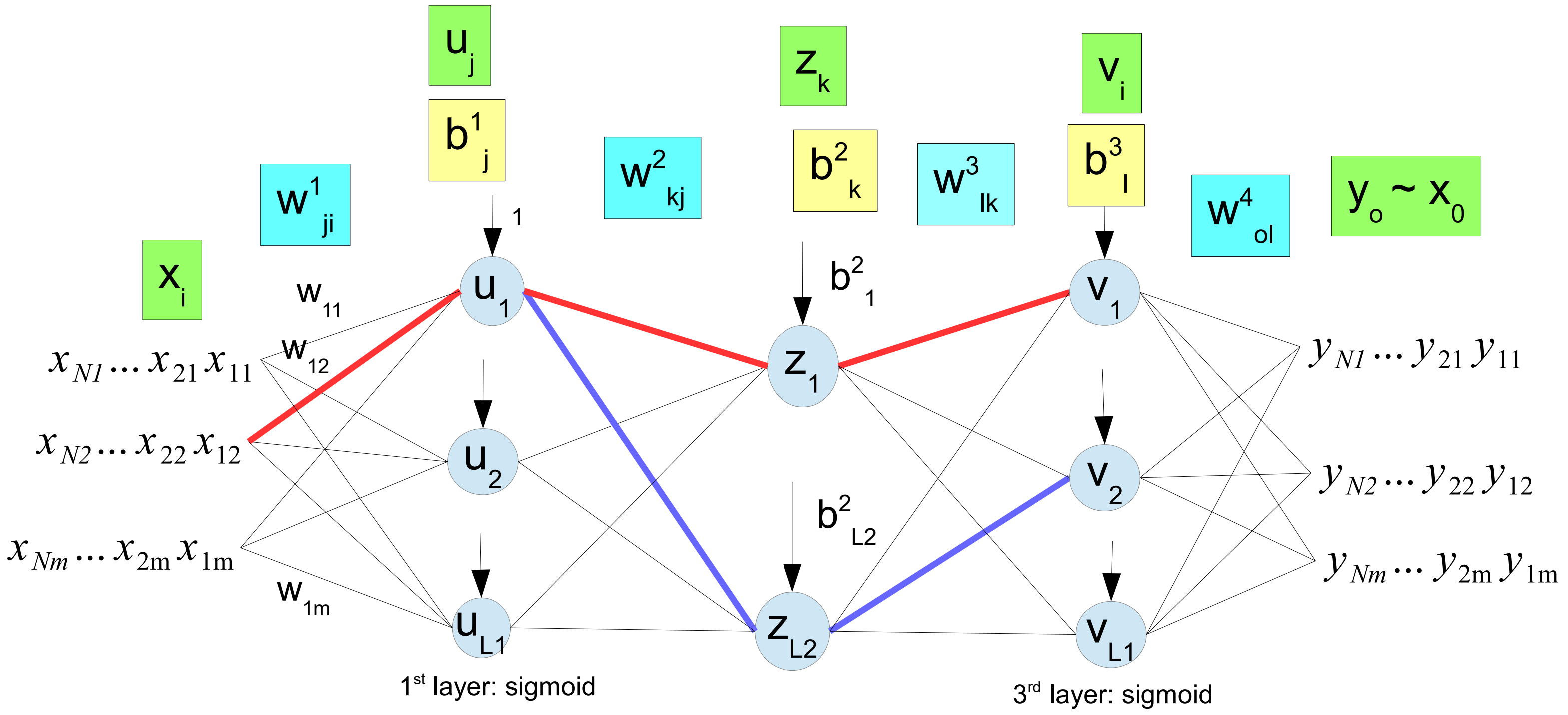
$$\boxed{\frac{\partial J_T}{\partial a_{lj}}=\frac{1}{N}\sum_{n=1}^{N}(y_i-y_{d_i})[b_{il}z_l(1-z_l)x_j]} \tag{4.24}$$

$$\frac{\partial v_l}{b_{al}}=1 \tag{4.25}$$

Derivative w.r.t. bias:

$$\boxed{\frac{\partial J_T}{\partial b_{al}}=\frac{1}{N}\sum_{n=1}^{N}(y_i-y_{d_i})[b_{il}z_l(1-z_l)]} \tag{4.26}$$

5 Autoencoder



Definitions:	**Dimensions:**
L1: 1st and 3rd layer neurons L2: 2nd layer neurons m: number of inputs N: number of instances 1st and 3rd layers activation function: sigmoid 2nd and 4th layers activation function: none Cost function: MSE	X (N x m) W^1 (m x L1) W^2 (L1 x L2) W^3 (L2 x L1) W^4 (L1 x m) b^1 (L1) b^2 (L2) b^3 (L1) b^4 (m) U (N x L1) Z (N x L2) V (N x L1) Y (N x m)

Output equations

$$Uin_j = \sum_{i=1}^{m} X_i w_{ji}^1 + b_j^1 \tag{5.1}$$

$$U_j = \frac{1}{1+e^{-Uin_j}} \tag{5.2}$$

$$Zin_k = \sum_{j=1}^{L1} U_j w_{kj}^2 + b_k^2 \tag{5.3}$$

$$Vin_l = \sum_{k=1}^{L2} Z_k w_{lk}^3 + b_l^3 \tag{5.4}$$

$$V_l = \frac{1}{1+e^{-Vin_l}} \tag{5.5}$$

$$Y_o = \sum_{l=1}^{L1} V_l w_{ol}^4 + b_o^4 \tag{5.6}$$

$$J = \frac{(Y_o - X_o)^2}{2} \tag{5.7}$$

$$J_T = \frac{1}{N}\sum_{n=1}^{N} J(n) \tag{5.8}$$

4th layer derivatives

$$\boxed{\frac{\partial J_T}{\partial w_{ol}^4} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial J}{\partial y_o}\frac{\partial y_o}{\partial w_{ol}^4} = \frac{1}{N}\sum_{n=1}^{N}(y_o - y_o)v_l} \tag{5.9}$$

$$\boxed{\frac{\partial J_T}{\partial b_o^4} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial J}{\partial y_o}\frac{\partial y_o}{\partial b_o^4} = \frac{1}{N}\sum_{n=1}^{N}(y_o - y_o)} \tag{5.10}$$

3rd layer derivatives

$$\frac{\partial J_T}{\partial w_{lk}^3} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial J}{\partial y}\frac{\partial y}{\partial v_l}\frac{\partial v_l}{\partial w_{lk}^3} \tag{5.11}$$

$$\frac{\partial J}{\partial y} = \sum_{o=1}^{m}(y_o - x_o) \tag{5.12}$$

$$\frac{\partial y_o}{\partial v_l} = w_{ol}^4 v_l (1 - v_l) \tag{5.13}$$

$$\frac{\partial v_l}{\partial w_{lk}^3} = z_k \tag{5.14}$$

$$\boxed{\frac{\partial J_T}{\partial w_{lk}^3} = \frac{1}{N}\sum_{n=1}^{N}\sum_{o=1}^{m}(y_o - x_o) w_{ol}^4 v_l (1 - v_l) z_k} \tag{5.15}$$

$$\boxed{\frac{\partial J_T}{\partial b_l^3} = \frac{1}{N}\sum_{n=1}^{N}\sum_{o=1}^{m}(y_o - x_o) w_{ol}^4 v_l (1 - v_l)} \tag{5.16}$$

2nd layer derivatives

$$\frac{\partial J_T}{\partial w_{kj}^2} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial J}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial z_k}\frac{\partial z_k}{\partial w_{kj}^2} \tag{5.17}$$

$$\frac{\partial y}{\partial v}=\sum_{l=1}^{L1} w_{ol}^4 v_l(1-v_l) \tag{5.18}$$

$$\frac{\partial v}{\partial z_k}=w_{lk}^3 \tag{5.19}$$

$$\frac{\partial z_k}{\partial w_{kj}^2}=u_j \tag{5.20}$$

$$\boxed{\frac{\partial J_T}{\partial w_{kj}^2}=\frac{1}{N}\sum_{n=1}^{N}\sum_{o=1}^{m}\sum_{l=1}^{L1}(y_o-x_o)w_{ol}^4 v_l(1-v_l)w_{lk}^3 u_j} \tag{5.21}$$

$$\frac{\partial J_T}{\partial b_k^2}=\frac{1}{N}\sum_{n=1}^{N}\frac{\partial J}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial z_k}\frac{\partial z_k}{\partial b_k^2} \tag{5.22}$$

$$\boxed{\frac{\partial J_T}{\partial b_k^2}=\frac{1}{N}\sum_{n=1}^{N}\sum_{o=1}^{m}\sum_{l=1}^{L1}(y_o-x_o)w_{ol}^4 v_l(1-v_l)w_{lk}^3} \tag{5.23}$$

1st layer derivatives

$$\frac{\partial J_T}{\partial w_{ji}^1}=\frac{1}{N}\sum_{n=1}^{N}\frac{\partial J}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial z}\frac{\partial z}{\partial u_j}\frac{\partial u_j}{\partial w_{ji}^1} \tag{5.24}$$

$$\frac{\partial y}{\partial v}=\sum_{l=1}^{L1} w_{ol}^4 v_l(1-v_l) \tag{5.25}$$

$$\frac{\partial v}{\partial z}=\sum_{k=1}^{L2} w_{lk}^3 \tag{5.26}$$

$$\frac{\partial z}{\partial u_j}=w_{kj}^2 u_j(1-u_j) \tag{5.27}$$

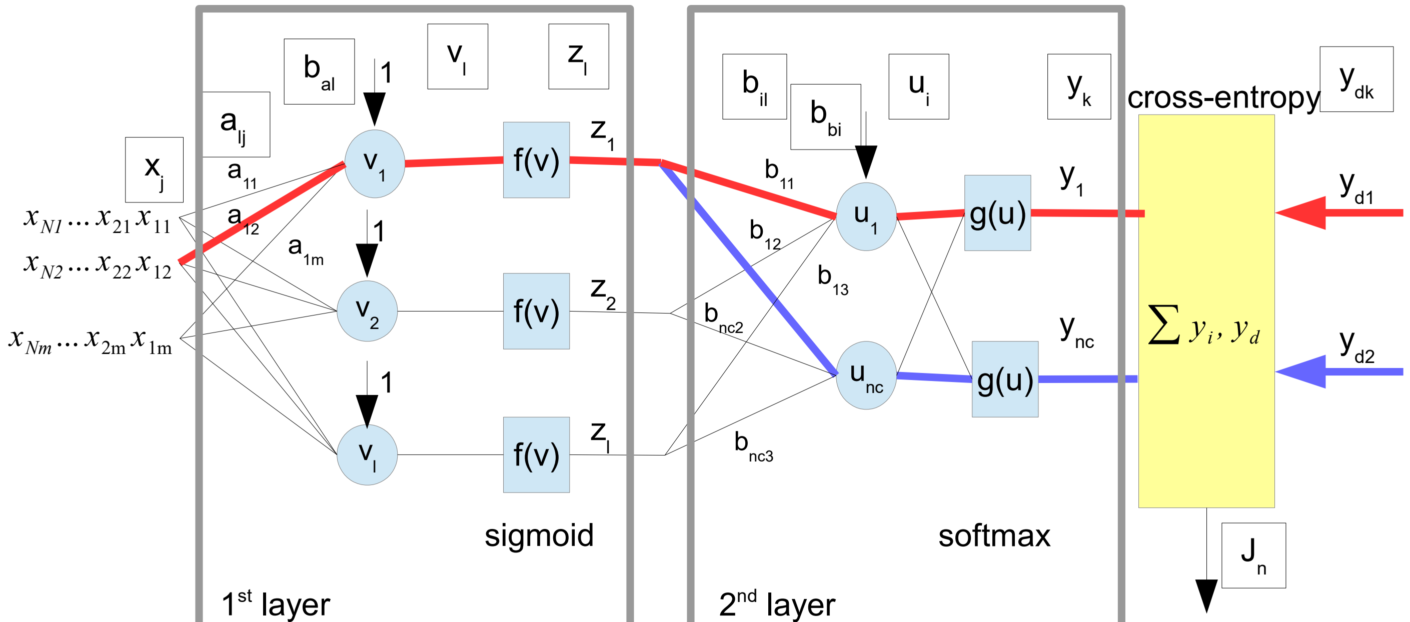
$$\frac{\partial u_j}{\partial w_{ji}^1}=x_i \tag{5.28}$$

$$\boxed{\frac{\partial J_T}{\partial w_{ji}^1}=\frac{1}{N}\sum_{n=1}^{N}\sum_{o=1}^{m}\sum_{l=1}^{L1}\sum_{k=1}^{L2}(y_o-x_o)w_{ol}^4 v_l(1-v_l)w_{lk}^3 w_{kj}^2 u_j(1-u_j)x_i} \tag{5.29}$$

$$\frac{\partial J_T}{\partial b_j^1} = \frac{1}{N}\sum_{n=1}^{N} \frac{\partial J}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial z}\frac{\partial z}{\partial u_j}\frac{\partial u_j}{\partial b_j^1} \tag{5.30}$$

$$\boxed{\frac{\partial J_T}{\partial b_j^1} = \frac{1}{N}\sum_{n=1}^{N}\sum_{o=1}^{m}\sum_{l=1}^{L1}\sum_{k=1}^{L2}(y_o - x_o)\, w_{ol}^4\, v_l(1-v_l)\, w_{lk}^3\, w_{kj}^2\, u_j(1-u_j)} \tag{5.31}$$

6 Ensemble learning via negative correlation



Definitions:	**Dimensions:**
L1: 1st and 3rd layer neurons L2: 2nd layer neurons m: number of inputs N: number of instances 1st and 3rd layers activation function: sigmoid 2nd and 4th layers activation function: none Cost function: MSE	X (N x m) W^1 (m x L1) W^2 (L1 x L2) W^3 (L2 x L1) W^4 (L1 x m) b^1 (L1) b^2 (L2) b^3 (L1) b^4 (m) U (N x L1) Z (N x L2) V (N x L1) Y (N x m)

Output equations

$$v_l=\sum_{j=1}^{m} x_j a_{lj}+b_{al} \tag{6.1}$$

$$z_l=f(v)=\frac{1}{1+e^{-v_l}} \tag{6.2}$$

$$u_i=\sum_{l=1}^{L} z_l b_{il}+b_{bl} \tag{6.3}$$

$$y_k=f(u)=f(u_{1,\dots,}u_{nc})=\frac{e^{u_k}}{\sum_{t=1}^{nc} e^{u_t}} \tag{6.4}$$

Negative correlation learning equations

Dataset:

$$D=\{(x(1),d(1)),\dots,(x(N),d(N))\} \tag{6.5}$$

Where $x\in\Re^p$, *d* is a scalar, and *N* is the size of the training set. The assumption that the output *d* is a scalar has been made merely to simplify exposition of ideas without loss of generality.

Error function for network *i* (Equation 2):

$$E_i=\frac{1}{N}\sum_{n=1}^{N}E_i(n)=\frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}\left(F_i(n)-d(n)\right)^2+\frac{1}{N}\sum_{n=1}^{N}\lambda p_i(n) \tag{6.6}$$

where $E_i(n)$ is the value of the error function of network *i* at presentation of the *n*th training pattern. The parameter $0\leqslant\lambda\leqslant1$ is used to adjust the strength of the penalty.

Correlation penalty function p_i (Equation 3):

$$p_i(n)=\left(F_i(n)-F(n)\right)\sum_{j\neq i}\left(F_j(n)-F(n)\right) \tag{6.7}$$

Partial derivative of $E_i(n)$ with respect to the output of network *i* on the *n*th training pattern (Equation 4):

$$\frac{\partial E_p(n)}{\partial F_p(n)}=(1-\lambda)(F_p(n)-y_d(n))+\lambda(F(n)-y_d(n)) \tag{6.8}$$

$$F_p=y\,,d=y_d \tag{6.9}$$

2nd layer derivatives

$$\frac{\partial E_p(n)}{\partial b_{il}(n)}=\frac{\partial E_p}{\partial F_p}\frac{\partial F_p}{\partial u_i}\frac{\partial u_i}{\partial b_{il}} \tag{6.10}$$

$$\frac{\partial F_p}{\partial u_i}=\frac{\partial y_k}{\partial u_i}=y_k(\delta_{ik}-y_i)\,,\delta_{ik}=\begin{cases}1 & if\ i=k\\ 0 & if\ i\neq k\end{cases} \tag{6.11}$$

$$\frac{\partial u_i}{\partial b_{il}}=z_l \tag{6.12}$$

$$\frac{\partial E_p(n)}{\partial b_{il}(n)}=[(1-\lambda)(F_p(n)-y_d(n))+\lambda(F(n)-y_d(n))]y_k(\delta_{ik}-y_i)z_l \tag{6.13}$$

$If\ i=k\,,\delta_{ii}=1:$

$$\boxed{\frac{\partial E_p(n)}{\partial b_{il}(n)}=[(1-\lambda)(F_p(n)-y_d(n))+\lambda(F(n)-y_d(n))]y_i(1-y_i)z_l} \tag{6.14}$$

$If\ i\neq k\,,\delta_{ik}=0:$

$$\boxed{\frac{\partial E_p(n)}{\partial b_{il}(n)}=-[(1-\lambda)(F_p(n)-y_d(n))+\lambda(F(n)-y_d(n))]y_k y_i z_l} \tag{6.15}$$

1st layer derivatives

$$\frac{\partial E_p(n)}{\partial a_{lj}(n)}=\frac{\partial E_p}{\partial F_p}\frac{\partial F_p}{\partial u_i}\frac{\partial u_i}{\partial v_l}\frac{\partial v_l}{\partial a_{lj}} \tag{6.16}$$

$$\frac{\partial F_p}{\partial u_i}=\frac{\partial y_k}{\partial u_i}=\sum_{k=1}^{nc} y_k(\delta_{ik}-y_i),\ \delta_{ik}=\begin{cases}1 & if\ i=k\\ 0 & if\ i\neq k\end{cases} \tag{6.17}$$

$$\frac{\partial u_i(n)}{\partial v_l(n)}=b_{il}z_l(1-z_l) \tag{6.18}$$

$$\frac{\partial v_l(n)}{\partial a_{lj}(n)}=x_j \tag{6.19}$$

$$\frac{\partial E_p(n)}{\partial a_{lj}(n)}=[(1-\lambda)(F_p(n)-y_d(n))+\lambda(F(n)-y_d(n))]y_k(\delta_{ik}-y_i)b_{il}z_l(1-z_l)x_j \tag{6.20}$$

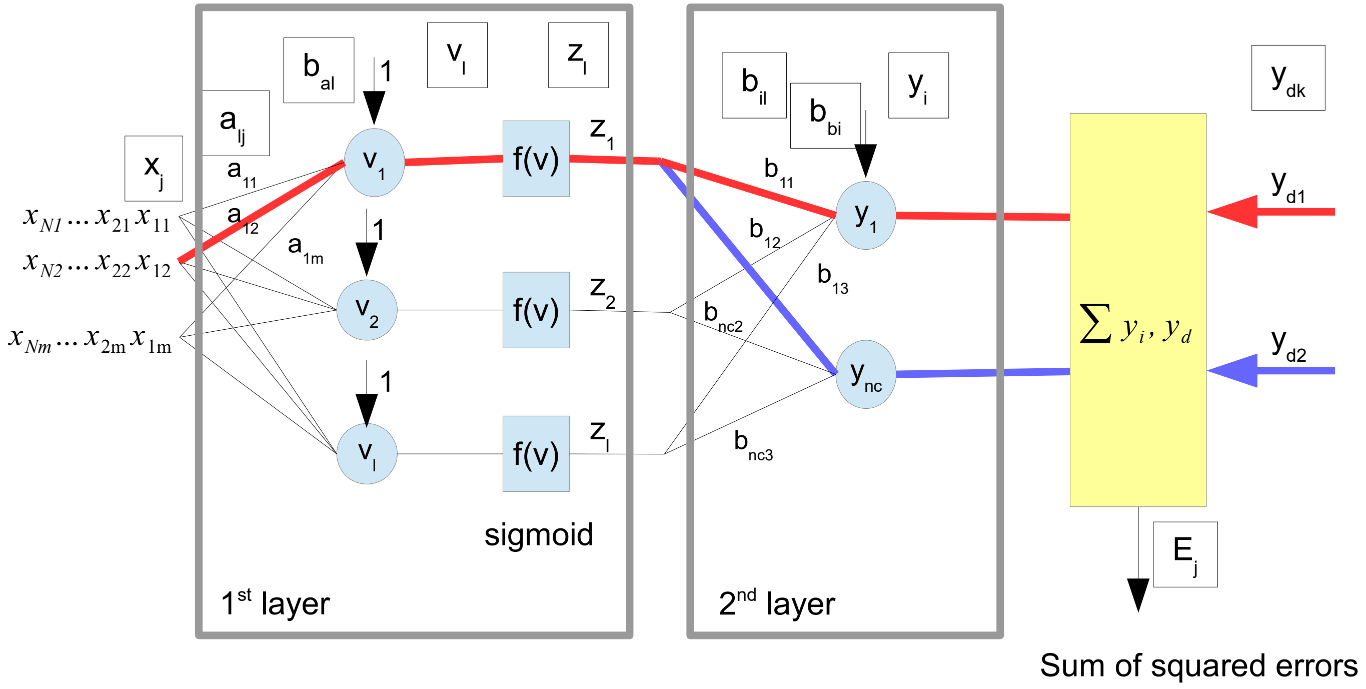
$If\ i=k,\delta_{ii}=1:$

$$\boxed{\frac{\partial E_p(n)}{\partial a_{lj}(n)}=[(1-\lambda)(F_p(n)-y_d(n))+\lambda(F(n)-y_d(n))]y_i(1-y_i)b_{il}z_l(1-z_l)x_j} \tag{6.21}$$

$If\ i\neq k,\delta_{ik}=0:$

$$\boxed{\frac{\partial E_p(n)}{\partial a_{lj}(n)}=-[(1-\lambda)(F_p(n)-y_d(n))+\lambda(F(n)-y_d(n))]y_k y_i b_{il}z_l(1-z_l)x_j} \tag{6.22}$$

7 Ensemble Learning Using Decorrelated Nns



Decorrelation equations

The training set is a set of *N* patterns $\{(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N)\}$ with the pth pattern defined by some unknown relationship:

$$y_p = g(\vec{x}_p) + \varepsilon \qquad (7.1)$$

where *g* is a regression function, and ε is some mean zero additive noise with finite variance σ².
Error function for an individual network *i* (equation 7):

$$E_j = \sum_{p=1}^{N} \left[(y_p - f_j(\vec{x}_p))^2 + \sum_{i=1}^{j-1} \lambda(t) d(i,j) P(\vec{x}_p, y_p, f_i, f_j) \right] \qquad (7.2)$$

Where $f_j(\vec{x})$ is the output of the *j*th network.
Correlation penaly function *P* (equation 8):

$$P(\vec{x}, y, f_i, f_j) = (y - f_i(\vec{x}))(y - f_j(\vec{x})) \qquad (7.3)$$

The indicator function *d* specifies which individual networks are to be decorrelated. For example, to penalize an individual network for being correlated with the previously trained network, the indicator function is (eq. 9):

$$d(i,j) = \begin{cases} 1, & \text{if } i = j-1 \\ 0, & \text{otherwise} \end{cases} \qquad (7.4)$$

To allow alternate networks to be trained independently of one another yet decorrelate pairs of networks, the indicator function can be defined as (equation 10)

$$d(i,j) = \begin{cases} 1, & \text{if } i = j-1 \text{ and } i \text{ is even} \\ 0, & \text{otherwise} \end{cases} \qquad (7.5)$$

The scaling function $\lambda(t)$ is either constant1 or is time dependent.

$$E_j = \sum_{p=1}^{N} \left[(y_p - f_j(\vec{x}_p))^2 + \sum_{i=1}^{j-1} \lambda(t) d(i,j) (y - f_i(\vec{x}))(y - f_j(\vec{x})) \right] \qquad (7.6)$$

Replacing with our variables:

$$y = y_d, f_j(\vec{x}) = y \qquad (7.7)$$

$$E_j = \sum_{p=1}^{N} \left[(y_{d_p} - y_j)^2 + \sum_{i=1}^{j-1} \lambda(t) d(i,j)(y_d - y_i)(y_d - y_j) \right] \qquad (7.8)$$

Output equations

$$v_l = \sum_{j=1}^{m} x_j a_{lj} + b_{al} \qquad (7.9)$$

$$z_l = f(v) = \frac{1}{1+e^{-v_l}} \tag{7.10}$$

$$f_j(\vec{x}) = y_i = \sum_{l=1}^{L} z_l b_{il} + b_{bl} \tag{7.11}$$

2nd layer derivatives

$$\frac{\partial E_j}{\partial b_{il}} = \frac{\partial E_j}{\partial f_j(\vec{x}_p)} \frac{\partial f_j(\vec{x}_p)}{\partial b_{il}} \tag{7.12}$$

$$\frac{\partial E_j}{\partial f_j(\vec{x}_p)} = \sum_{p=1}^{N} \left[-2(y_p - f_j(\vec{x}_p)) f'_j(\vec{x}_p) + \sum_{i=1}^{j-1} \lambda(t) d(i,j)(y - f_i(\vec{x}))(-f'_j(\vec{x})) \right] \tag{7.13}$$

Rearranging (7.13):

$$\frac{\partial E_j}{\partial f_j(\vec{x}_p)} = f'_j(\vec{x}) \sum_{p=1}^{N} \left[2(f_j(\vec{x}_p) - y_p) + \sum_{i=1}^{j-1} \lambda(t) d(i,j)(f_i(\vec{x}) - y) \right] \tag{7.14}$$

$$\frac{\partial f_j(\vec{x}_p)}{\partial b_{il}} = z_l \tag{7.15}$$

Combining (7.14) and (7.15) into (7.12) (Derivative w.r.t. weights):

$$\boxed{\frac{\partial E_j}{\partial b_{il}} = \sum_{p=1}^{N} \left[2(f_j(\vec{x}_p) - y_p) + \sum_{i=1}^{j-1} \lambda(t) d(i,j)(f_i(\vec{x}) - y) \right] z_l} \tag{7.16}$$

Derivative w.r.t. bias:

$$\boxed{\frac{\partial E_j}{\partial b_{il}} = \sum_{p=1}^{N} \left[2(f_j(\vec{x}_p) - y_p) + \sum_{i=1}^{j-1} \lambda(t) d(i,j)(f_i(\vec{x}) - y) \right]} \tag{7.17}$$

1st layer derivatives

$$\frac{\partial E_j}{\partial a_{lj}} = \frac{\partial E_j}{\partial f_j(\vec{x}_p)} \frac{\partial f_j(\vec{x}_p)}{\partial v_l} \frac{\partial v_l}{\partial a_{lj}} \tag{7.18}$$

$$\frac{\partial f_j(\vec{x}_p)}{\partial v_l(n)} = b_{il} z_l (1 - z_l) \tag{7.19}$$

$$\frac{\partial v_l(n)}{\partial a_{lj}(n)} = x_j \tag{7.20}$$

Derivative w.r.t. weights:

$$\boxed{\frac{\partial E_j}{\partial a_{lj}} = \sum_{p=1}^{N} \left[2\left(f_j(\vec{x}_p) - y_p\right) + \sum_{i=1}^{j-1} \lambda(t)\, d(i,j)\left(f_i(\vec{x}) - y\right) \right] b_{il}\, z_l (1 - z_l)\, x_j} \tag{7.21}$$

Derivative w.r.t. bias:

$$\frac{\partial v_l(n)}{\partial b_{il}(n)} = 1 \tag{7.22}$$

$$\boxed{\frac{\partial E_j}{\partial b_{il}} = \sum_{p=1}^{N} \left[2\left(f_j(\vec{x}_p) - y_p\right) + \sum_{i=1}^{j-1} \lambda(t)\, d(i,j)\left(f_i(\vec{x}) - y\right) \right] b_{il}\, z_l (1 - z_l)} \tag{7.23}$$

8 Reference

Math Is Fun website - https://www.mathsisfun.com/calculus/derivatives-rules.html

Eli Bendersky's website The Softmax function and its derivative
https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/

Liu, Yong & Yao, Xin. (2000). Ensemble learning via negative correlation. Neural networks : the official journal of the International Neural Network Society. 12. 1399-1404. 10.1016/S0893-6080(99)00073-8.

Rosen, B. (1996). Ensemble Learning Using Decorrelated Neural Networks. Connect. Sci., 8, 373-384. DOI: 10.1080/095400996116820 https://doi.org/10.1080/095400996116820