Math of AI system

©DuyKy Nguyen, PhD EE @ Unitethc.com 2024-10-22

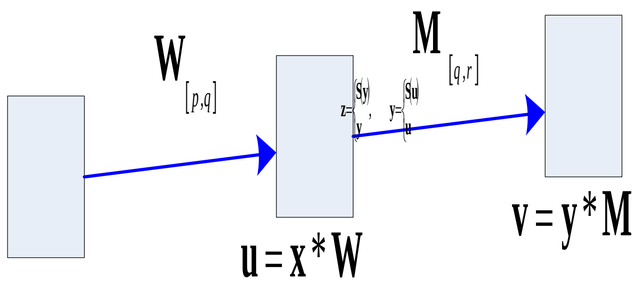
1. Objectives

We have This note presented an evaluation of AI model using Activate function with an mathematic analysis of Activate function may result failure of AI system
To the best of my knowledge it could be the very first ever been carried out
Consider AIsytem with hidden layer so we have 2 weight matrices to be dertmined **W M**

2. Traning proces

At start all weights are initilized randomly and we have NONZERO the system error E betwwen system ouput and reference onesout training updates them to get E=0

The system Error E is a function of multivariable input x **weight m and w** Note that matrix W and M must be converted to vector for function of multi var in form of vector not matrix
During the computation matrix converted to vector for passing into function then inside function vector converted back to matrix to compute the error
Fortunately MatLabfunction is very easyconinient to use for this kind of operation
But we have use a free version Octave instead of few thousand MLab license fee and I did not forget a little donation of $20 due to my limit retiree income after 25 years working with few biz contribution but with few layoff and no pay raise or any kind of promotionat all during 11 years at Symmetricom , probably my boss claimed all mines as his own contributionsprobably I had to pay him back all debts I owed him in my previous life per buddishm belief??!!
I have composed other 3 articles in overview optimal method and Octave MLab programming to run and see the result if so required
Octave programmind note should be read before optimal so easy to try some optimalmethods
As an result oriented my practice is to evaluate first to see if something worth to try on
You may note MLab programming note far more comprehensive than any on www
If you find better one please kindly let me know via my personal email dkn@unitedthc.com I truely aprreciate your kindness.
All these methods rely on gradient[1st derivative for multivariate function] of system ouput error to the gradient to update the initial weights to final weights with systemouput equal reference ones
Currently Activate function is being used in AI system To the best of our our multivariate calculus we have foundZERO derivative hence ZERO gradient of Activate function cause faoilure in training as the ZERO derivative


$W_{[p,q]}$
$M_{[q,r]}$
$z=\begin{cases}S(y)\\y\end{cases}, \quad y=\begin{cases}S(u)\\u\end{cases}$
$u = x * W$
$v = y * M$

Z

R

Y

Q

X

P

$$S(u)=[S(u_1) \quad S(u_2) \quad \cdots], \quad y=\begin{cases}S(u)\\ u\end{cases} \quad z=\begin{cases}S(v)\\ v\end{cases}$$

Input are sum of all incomings

No need to have equal number of inputs and of incommings

$$u_j(x,w)=\sum_{i=1}^{p} x_i w_{ij}$$

$$y(u)=S(u_w)$$

$$u_j=\sum_{i=1}^{p} x_j w_{ij}$$

$$v(y,m)$$

$$v_k=\sum_{j=1}^{q} y_j m_{jk}$$

$$\frac{\partial y_k}{\partial w_{ij}}=m_{jk}\frac{\partial y_j}{\partial w_{ij}}=m_{jk}S'(u_j)x_j$$

$$z=S(v_m)$$

$$\frac{\partial e_k}{\partial w_{ij}}=\frac{\partial z_k}{\partial w_{ij}}=S'(v_k)\frac{\partial v_k}{\partial w_{ij}}$$

2.1. Activate function

Activating Switching differential function

$$S(x)=\frac{1}{1+e^{-x}}$$

$$S+Se^{-x}=1$$

$$S'+S'e^{-x}-e^{-x}S=0$$

$$e^{-x}(S'-S)+S'=0$$

$$e^{-x}(S'-S)+S'=0$$

$$Se^{-x}=1-S$$

$$e^{-x}=\frac{1-S}{S}$$

$$\frac{1-S}{S}(S'-S)+S'=0$$

$$(S'-S)(1-S)+SS'=0$$

$$S'(1-S)-S(1-S)+SS'=0$$

$$S'-S(1-S)=0$$

$$S'=S(1-S)$$

$$S'=S(1-S)$$

$$S'(ax)=aS'(x)$$

Using convention $u(x,w)=u_{xw}$ for simplification We have

$$z=S(v_{ym})$$

$$v=S(u_{xw})$$

$$e_k=z_k-r_k$$

$$E=\frac{1}{2}\sum^{r}(e_k)^2$$

$$\frac{\partial E}{\partial m}=\sum^{r}\left(e_k\frac{\partial e}{\partial m}\right)$$

$$\frac{\partial E}{\partial w}=\sum^{r}\left(e_k\frac{\partial e}{\partial w}\right)$$

$$\frac{\partial e}{\partial m}=\frac{\partial z}{\partial m}=S'(v_{ym})\frac{\partial v_{ym}}{\partial m}$$

$$\frac{\partial e}{\partial w}=\frac{\partial z}{\partial w}=S'(v_{ym})\frac{\partial v_{ym}}{\partial w}$$

$$\frac{\partial v_{ym}}{\partial w}=S'(u_{xw})\frac{\partial u_{xw}}{\partial w}$$

$$\mathbf{u} = \mathbf{x} * \mathbf{W}$$

$$u_j = \sum_{i=1}^{p} x_i w_{ij} \Rightarrow \frac{\partial u_j}{\partial w_{ij}} = x_i$$

$$y_j = u_j$$

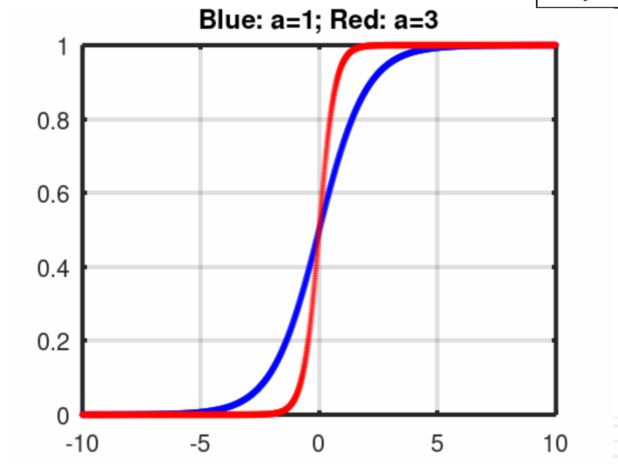
$$y_j = S(u_j) \Rightarrow \frac{\partial y_j}{\partial w_{ij}} = S'(u_j)\frac{\partial u_j}{\partial w_{ij}} = x_i S'(u_j)$$

$$\mathbf{v} = \mathbf{y} * \mathbf{M}$$

$$v_k = \sum_{j=1}^{r} y_j m_{jk} \Rightarrow \begin{cases} \dfrac{\partial v_k}{\partial w_{ij}} = \sum_{j=1}^{r} m_{jk} x_i S'(u_j) \\ \dfrac{\partial v_k}{\partial m_{jk}} = y_j \end{cases}$$

$$z_k = S(v_k)$$

$$\frac{\partial z_k}{\partial w_{ij}} = S'(v_k)\frac{\partial v_k}{\partial w_{ij}}$$



$$S' = S(1 - S)$$

We have $S = 1$ at point far away on the right offrom Activate point so $S' = 0$
We have $S = 0$ at point far away on the left Activate point so $S' = 0$

$$\mathbf{y}\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_q \end{bmatrix} = \mathbf{x}_{[1,p]} * \mathbf{W}_{[p,q]} = \begin{bmatrix} x_1 & x_2 & \cdots & x_p \end{bmatrix} * \begin{bmatrix} \mathbf{W}_{[1,*]} & \mathbf{W}_{[2,*]} & \cdots & \mathbf{W}_{[q,*]} \end{bmatrix}$$

$$y_k(x_i) = \mathbf{x} * \mathbf{W}_{[k,*]} = \sum^{p} x_i w_{ik} \Rightarrow \frac{\partial y_k}{\partial x_i} = w_{ik}$$

$$\mathbf{z} = \mathbf{S}(\mathbf{y})$$

$$\mathbf{e} = \mathbf{z} - \mathbf{r}$$

$$z_k = S(y_k)$$

$$\frac{\partial z_k}{\partial x_i} = S'(y_k)\frac{\partial y_k}{\partial x_i} = w_{ik} S'(y_k)$$

So Act func has no impact on output per change of input only if if $S'(*) = 0$
But in such case any grad-base method cannot be employed as it's impossible to move from starting point to point of solution with ZERO GRAD

Convention

Bold upcase for matrixwith [] for dim and () fot index with row cones before col like natrix of r rows and c col is $\mathbf{M}_{[r,c]}$ row n of matrix is $\mathbf{M}_{(n,*)}$ col n is $\mathbf{M}_{(*,n)}$
Bold lower case for vector like vector **x** of n eles [elements] is **x** ments
Matrix/Vector conformant condition must be required in multiplication $\mathbf{A}_{[m,n]} * \mathbf{B}_{[n,r]}$

There're 2 vector types row as $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$ and colas $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

We prefer row vector as it save space ofin display
The main idea behind bk prp is an divide conquer approch starting from output back to input to update weight to to get system out equal reference out
First in forward prp we start from input to ouput to get output then bck prp to find grad for weight update
Given **p** input and **r** ouput we require **inputweight** $\mathbf{W}_{[p,q]}$ **outputweight** $\mathbf{M}_{[a,r]}$
To ease tracking down matric multiplyLet p=4 q=3, r=2

$$\mathbf{u}_{[1,q]} = \mathbf{x}_{[1,p]} * \mathbf{W}_{[p,q]} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix}$$

$$=$$

$$u_j = \mathbf{x}_{[1,p]} * \mathbf{W}(*,j) col\ j = \sum_{i=1}^{p} x_i w_{ij}\ j \in (1,3) \Rightarrow \frac{\partial u}{\partial w_{ij}} = x_j$$

$$y = S(u)$$

$$\frac{\partial y}{\partial w_{ij}} = S'(u) \frac{\partial u}{\partial w_{ij}} = x_j S'(u)$$

$$\mathbf{v}_{[1,r]} = \mathbf{y}_{[1,q]} * \mathbf{W}_{[q,r]} = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} + \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \\ m_{31} & m_{32} \end{bmatrix}$$

$$v_k = \mathbf{y}_{[1,p]} * \mathbf{M}(*,k) col\ k \in (1,r)$$

$$v_k = \sum_{j}^{p} y_j m_{jk} \Rightarrow \frac{\partial v}{\partial m_{jk}} = y_j = \begin{cases} u \\ S(u) \end{cases}$$

and $u = \mathbf{x} * \mathbf{W}$

$$z = S(v)$$

$$e = z - r$$

$$\frac{\partial e}{\partial m_{jk}} = \frac{\partial z}{\partial m_{jk}} = S'(v) \frac{\partial v}{\partial m_{jk}} y_j S'(v)$$

2.1.1. Delata rule

Bold upcase for matrixwith [] for dim and () fot index with row cones before col like natrix of r rows and c col is $\mathbf{M}_{[r,c]}$ row n of matrix is $\mathbf{M}_{(n,*)}$ col n is $\mathbf{M}_{(*,n)}$
Bold lower case for vector like vector **x** of n eles [elements] is **x** ments
Matrix/Vector conformant condition must be required in multiplication $\mathbf{A}_{[m,n]} * \mathbf{B}_{[n,r]}$

There're 2 vector types row as $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$ and colas $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

We prefer row vector as it save space ofin display
The main idea behind bk prp is an divide conquer approch starting from output back to input to update weight to to get system out equal reference out
First in forward prp we start from input to ouput to get output then bck prp to find grad for weight update
Given **p** input and **r** ouput we require **inputweight** $\mathbf{W}_{[p,q]}$ **outputweight** $\mathbf{M}_{[a,r]}$
To ease tracking down matric multiplyLet p=4 q=3, r=2

$$\mathbf{u}_{[1,q]} = \mathbf{x}_{[1,p]} * \mathbf{W}_{[p,q]} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix}$$

$$u_j = \mathbf{x}_{[1,p]} * \mathbf{W}(*,j) col\ j = \sum_{i=1}^{p} x_i w_{ij}\ j \in (1,3) \Rightarrow \frac{\partial u_j}{\partial w_{ij}} = x_j$$

$$y = S(u)$$

$$\frac{\partial y}{\partial w_{ij}} = S'(u)\frac{\partial u}{\partial w_{ij}} = x_j S'(u)$$

$$v_k = \sum_{j}^{q} y_j m_{jk}$$

$$\frac{\partial v_k}{\partial w_{ij}} = \sum_{j}^{q} m_{jk}\frac{\partial y_j}{\partial w_{ij}} = \sum_{j}^{q} m_{jk} x_j S'(u_j)$$

$$z_k = S(v_k)$$

$$\frac{\partial e_k}{\partial w_{ij}} = \frac{\partial z_k}{\partial w_{ij}} = S'(v_k)\frac{\partial v_k}{\partial w_{ij}}$$

$$\mathbf{v}_{[1,r]} = \mathbf{y}_{[1,q]} * \mathbf{W}_{[q,r]} = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} + \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \\ m_{31} & m_{32} \end{bmatrix}$$

$$v_k = \mathbf{y}_{[1,p]} * \mathbf{M}(*,k) col\ k \in (1,r)$$

$$v_k = \sum_{j}^{p} y_j m_{jk} \Rightarrow \frac{\partial v}{\partial m_{jk}} = y_j = \begin{cases} u \\ S(u) \end{cases}$$

and $u = \mathbf{x} * \mathbf{W}$

$$z = S(v)$$

$$e = z - r$$

$$\frac{\partial e}{\partial m_{jk}} = \frac{\partial z}{\partial m_{jk}} = S'(v)\frac{\partial v}{\partial m_{jk}} y_j S'(v)$$

We have system Err

$$\frac{\partial E}{\partial w} = \sum^{r}\left[(e_k)\frac{\partial e_k}{\partial w}\right] = \sum^{r}\left[(z_k)\frac{\partial z_k}{\partial w}\right]$$

$$E(w,m) = \frac{1}{2}\sum^{r}\left[(e_k)^2\right] \qquad \frac{\partial E}{\partial m} = \sum^{r}\left[(e_k)\frac{\partial e_k}{\partial m}\right] = \sum^{r}\left[(e_k)\frac{\partial z_k}{\partial m}\right]$$

$$e_k = z_k - r_k \qquad \frac{\partial e_k}{\partial w_{ij}} = \frac{\partial z_k}{\partial w_{ij}} = S'(v_k)\frac{\partial v_k}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w} = \sum^{r}\left[(e_k)\frac{\partial e_k}{\partial w}\right] = \sum^{r}\left[(z_k)\frac{\partial z_k}{\partial w}\right]$$

$$\frac{\partial e_k}{\partial m_{jk}} = \frac{\partial z}{\partial m_{jk}} = S'(v) = \frac{\partial v}{\partial m_{jk}} y_j S'(v)$$

$$\frac{\partial E}{\partial m} = \sum^{r}\left[(e_k)\frac{\partial e_k}{\partial m}\right] = \sum^{r}\left[(e_k)\frac{\partial z_k}{\partial m}\right]$$

$$\frac{\partial E}{\partial w} = \sum^{r}\left[(e_k)\frac{\partial e_k}{\partial w}\right] = \sum^{r}\left[(e_k)S'(v_k)\frac{\partial v_k}{\partial w_{ij}}\right]$$

So total differentiation of E is

$$dE = \frac{\partial E}{\partial w}dw + \frac{\partial E}{\partial m}dm \qquad \frac{\partial E}{\partial w} = \frac{\partial v_k}{\partial m_{jk}} y_j S'(v_k)$$

Given a positive learning rate $\lambda > 0$
using update rule below

$$dw = -\lambda\left(\frac{\partial E}{\partial w}\right)$$

$$dm = -\lambda\left(\frac{\partial E}{\partial m}\right)$$

then

$$dE = \frac{\partial E}{\partial w}dw + \frac{\partial E}{\partial m}dm = -\lambda\left[\left(\frac{\partial E}{\partial w}\right)^2 + \left(\frac{\partial E}{\partial m}\right)^2\right] < 0 \text{so to reduce E}$$

$$w_{k+1} = w_k + dw,\ dw = -\lambda\left(\frac{\partial E}{\partial w}\right)$$

$$m_{k+1} = m_k + dmdm = -\lambda\left(\frac{\partial E}{\partial m}\right)$$

As $E > 0$ using the update rule below to reduce the err However $S^{(1)}(*) = S^{(2)}(*) = 0$
hence
$\Delta w = \Delta m = 0$
therefore **we have NO update**

2.1.2. Optimal method 01

As non-gradient method cannot be used so consider gradient method with algorithm belowgiven gradient $\mathbf{g} = \nabla E(\mathbf{wm})$,

where

reshape matrix $\mathbf{W}$ as row vector $\overline{\mathbf{w}}$

reshape matrix $\mathbf{M}$ as row vector $\overline{\mathbf{m}}$

$\mathbf{wm} = [\overline{\mathbf{w}} \quad \overline{\mathbf{m}}]$

Now E is a multivariate function ofvar of dim $\mathrm{P} * \mathrm{Q} * \mathrm{R}$

$\mathbf{x}_{[1,\mathrm{P}]}\ \mathbf{y}_{[1,\mathrm{P}]}\ \mathbf{z}_{[1,\mathrm{R}]}$

Remark 1

System err E is a multivariate function

$E(\mathbf{x})$

where

$\mathbf{x} = [\overline{\mathbf{w}} \quad \overline{\mathbf{m}}]$, NOT input $\mathbf{X}$

reshape matrix $\mathbf{W}$ as row vector $\overline{\mathbf{w}}$

reshape matrix $\mathbf{M}$ as row vector $\overline{\mathbf{m}}$

$\mathbf{wm} = [\overline{\mathbf{w}} \quad \overline{\mathbf{m}}]$

Now E is a multivariate function ofvar of dim $\mathrm{P} * \mathrm{Q} * \mathrm{R}$

$\mathbf{x}_{[1,\mathrm{P}]}\ \mathbf{y}_{[1,\mathrm{P}]}\ \mathbf{z}_{[1,\mathrm{R}]}$

REPEAT
1] starting with initial point $\mathbf{x}_0$ with

2] using Line search function ib direction $\mathbf{d} = \mathbf{g},\ \mathbf{g} = \nabla E(\mathbf{x})\ \ a = LS(\mathbf{x}, \mathbf{d}), a > 0$

3] next point $\mathbf{x}_{k+1} = \mathbf{x}_k + a\mathbf{d}$

4] DONE if $a\|\mathbf{d}\| < tol$

As in section of DElta rule as $S^{(1)}(*)=S^{(2)}(*)=0$ we have $\nabla E(*)=0$
Therefore the optimal method ether cannot be used

2.1.3. Numerical Newton Raphson

Given $f(x)=0$,by definition of derivative

$$f'(x_n)=\frac{f(x_{n+1})-f(x_n)}{x_{n+1}-x_n}=\frac{0-f(x_n)}{x_{n+1}-x_n}$$

If $f'(x_n)\neq 0$

thenwe have Newton graphson method below

$$x_{n+1}-x_n=-\frac{f(x_n)}{f'(x_n)}$$

$$P>Q>R$$

$$\mathbf{u}_{[1,Q]}=\mathbf{x}_{[1,P]}*\mathbf{W}_{[P.Q]}$$

$$\mathbf{y}_{[1,Q]}=\mathbf{S}(\mathbf{u}_{[1,Q]})$$

$$\mathbf{v}_{[1,R]}=\mathbf{y}_{[1,Q]}*\mathbf{M}_{[Q.R]}=\mathbf{S}(\mathbf{u})*\mathbf{M}=\sum_{j=1}^{Q}S(u_j)m_{jk},$$

$$v_k=S(u_j)m_{jk}$$

as

$$z_k=r_k$$

then

$$r_k=S(v_k)$$

This is an trancendental eq with unknown $v_k$numerical method Newton Graphson must be use

$$\mathrm{x}_{n+1}=x_n-\frac{S(x_n)}{S'(x_n)}$$

Once again $S'(x_n)=0$so the solution$v_k$cannot be found

3. Conclusion

We have explored all possible way to find weight in AI training using Activate function
However its derivative is ZERO so the solution cannot be found using grad-base optimal method
So we have proposea simplified structure not using Activate function However
this ZERO derivative of Activate function could have potitive impact for an robust AI system
Ideallyonce a AI stem trained with some iput data
and a input data with char **a** A wtitten by another person
The trained system should recognize it as char **a**
So a non grad baseoptimal method Bayessian shou be atemted seriously
Below is py implement of Bayes method
. **https://pypi.org/project/bayesian-optimization/**