T

10-2021 Monday

Week #6 lecture #7

loss function code:

$$L = \sum \max(0, s_j - s_y + 1)$$

```
def loss(sj, sy)
[
    Sum = 0
    for i in range(len(sj)):
        temp = max(0, sj[i] - sy[i] + 1)
        Sum += temp
    return Sum
]
```

```
def score(w, x):
    return w.dot(x)
def loss(w, x, y):
    s = score(w, x)
    d = s - s[y] + 1        -> // subtract Trueclass score from every element in s
    d[y] = 0                // true class difference is 0
    Mask = d > 0
    Loss = np.sum(d[Mask])
    return loss
```

Steps

① Score function

$$f = Wx$$

② Loss function

$$L = \sum_{j \neq y} \max(0, s_j - s_y + 1)$$

③ Optimization for weights

$$W = w - \alpha \frac{dL}{dW}$$

$$\begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ c_0 & c_1 & c_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \end{bmatrix} \leftarrow \text{True class}$$

$$L = \max(0, s_0 - s_1 + 1) + \max(0, s_2 - s_1 + 1)$$

$$= L = \max(0, (a_0x_0 + a_1x_1 + a_2x_2) - (b_0x_0 + b_1x_1 + b_2x_2) + 1)$$

$$+ \max(0, (c_0x_0 + c_1x_1 + c_2x_2) - (b_0x_0 + b_1x_1 + b_2x_2) + 1)$$

$$\frac{dL}{da_0} = x_0 [s_0 - s_1 + 1 > 0]$$

$$\frac{dL}{db_0} = -x_0 [s_0 - s_1 + 1 > 0] - x_0 [s_2 - s_1 + 1 > 0]$$

Gradient

evaluate function:

def Eval(W, X, Y): # Gradients

S = Score(W, X)

d = S - S[y] + 1

d[y] = 0

mask = d > 0

dW = np.zeros(W)

dW[mask] = X # a0 = x0, a1 = x1, a2 = x2
# c0 = x0, c1 = x1, c2 = x2

n = np.sum(mask) how many times the conditions are true

dW[y] = -n * X

Steps: SVM classifier

① Initialize W

W[K, d] K = no. of classes
d = dimension of Image.

② Update weights

for i in range(1000):
    dW = Gradients(W, X, Y)

Gradients functions takes Gradients for all imgs in dataset & returns avg Gradients value.

$W = W - \alpha * dW$

Loss (W, X, Y) To check Performance of weights of updated W's

& This checks cost of weights for all imgs in Train set & validation set

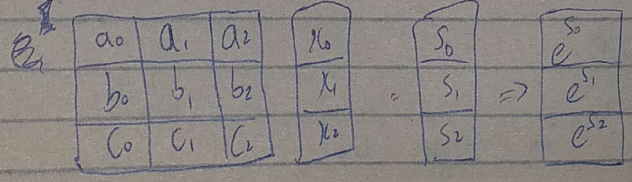
Gradients only updated bcz of Training set & not after checking on validation set

SoftMax classifier:-

Steps:

① Score function

as SVM only gives score, we want to convert these scores to Probabilities



$\alpha = \sum_i e^{s_i} = e^{s_0} + e^{s_1} + e^{s_2}$

$$\begin{bmatrix} e^{s_0} \\ e^{s_1} \\ e^{s_2} \end{bmatrix} \Rightarrow \begin{bmatrix} e^{s_0}/\alpha \\ e^{s_1}/\alpha \\ e^{s_2}/\alpha \end{bmatrix}$$

all Probabilities are Positive, in SUM Scores were sometimes -ve

Store function

$$\frac{e^{f_y}}{\sum_i e^{f_i}}$$

② Loss function:- Cross entropy loss Function:

$Loss = -\log(P[y])$

$\therefore \log(1) = 0$

$\therefore \log(0) = \infty$

bcz $\log_2 0 = \infty$

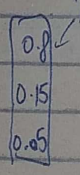
$2^{-\infty} = \frac{1}{2^{\infty}}$ approaches 0

True class Probability

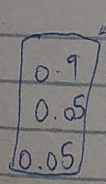
closer to 1 loss function is good

closer to 0 loss function is bad

e.g



$\rightarrow -\log(0.8) = 0.097$



$\rightarrow -\log(0.9) = 0.046$

True class

0.2
0.4
0.4

$-\log(0.2) = 0.7$

0.1
0.8
0.1

True class $-\log(0.1) = 1$

Probability closer to Zero means loss function/weights is bad

③ Gradient function: / Update weights.

$$L = -\log(P[y])$$

$$= -\log\left(\frac{e^{S_1}}{e^{S_0}+e^{S_1}+e^{S_2}}\right) \quad \text{// True class } S_1$$

$$= -\log\left(\frac{e^{(b_0 x_0 + b_1 x_1 + b_2 x_2)}}{e^{(a_0 x_0 + a_1}}\right)$$

$$= -\log(e^{S_1}) + \log\left(e^{S_0}+e^{S_1}+e^{S_2}\right)$$

$$L = -S_1 + \log \sum_j e^{S_j}$$

$\log x$

$\frac{1}{x}$

$L = \log\left(e^{(a_0x_0 + a_1x_1 + a_2x_2)} + e^{(b_0x_0 + b_1x_1 + b_2x_2)} + e^{(c_0x_0 + c_1x_1 + c_2x_2)}\right)$

Gradients

$\frac{dL}{da_0} \frac{da_0}{da_0} = \frac{+1}{\sum_j e^{s_j}} \left(x_0 e^{a_0x_0}\right)$

$\frac{dL}{da_0} = \frac{1}{\sum_j e^{s_j}} \left(x_0 e^{(a_0x_0 + a_1x_1 + a_2x_2)}\right)$

$\frac{dL}{db_0} = -x_0 + \frac{1}{\sum_j e^{s_j}} \left(x_0 e^{(b_0x_0 + b_1x_1 + b_2x_2)}\right)$

$\frac{dL}{dc_0} = \frac{1}{\sum_j e^{s_j}} \left(x_0 e^{(c_0x_0 + c_1x_1 + c_2x_2)}\right)$

$f = \log x$

$\frac{df}{dx} = \frac{1}{x} \cdot dx$

$f = e^{ax}$

$\frac{df}{dx} = ae^{ax} \, d(ax)$

$\frac{df}{dx} = ae^{ax}$

$L = -S_1 + \log\left(e^{S_0} + e^{S_1} + e^{S_2}\right)$

$L = -(b_0x_0 + b_1x_1 + b_2x_2) + \log\left(e^{(a_0x_0 + a_1x_1 + a_2x_2)} + e^{(b_0x_0 + b_1x_1 + b_2x_2)} + e^{(c_0x_0 + c_1x_1 + c_2x_2)}\right)$

$e^{S_0} = e^{a_0x_0 + a_1x_1 + a_2x_2}$

$\frac{dL}{da_0} = \frac{1}{\sum_{j=0}^{n-1} e^{s_j}} \left(x_0 e^{S_0}\right)$ & $\frac{dL}{db_0} = -x_0 + \frac{1}{\sum_j e^{s_j}} \left(x_0 e^{S_1}\right)$

use this for Gradient Decent for this

eg Gradients for

$$f = (x+y).z$$

$$\left[\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}\right] = \left[z, z, (x+y)\right]$$

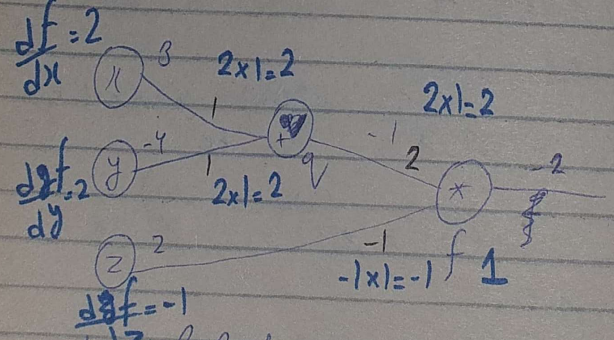
$$q = x+y$$

$$f = qz$$

$$\frac{df}{dq} = z \quad , \quad \frac{df}{dz} = q$$

$$\frac{dq}{dx} = 1 \quad , \quad \frac{dq}{dy} = 1$$

$$\frac{df}{dx} = \frac{df}{dq} \times \frac{dq}{dx} \quad \leftarrow \text{Chain Rule}$$



• Local Gradients

back Propagation, using graph

f = xy

$$\frac{df}{dx} = y$$

$$\frac{df}{dy} = x$$

$$\frac{df}{dx} \cdot \frac{df}{dy}$$

we always send 1 backwards to get respective Gradients

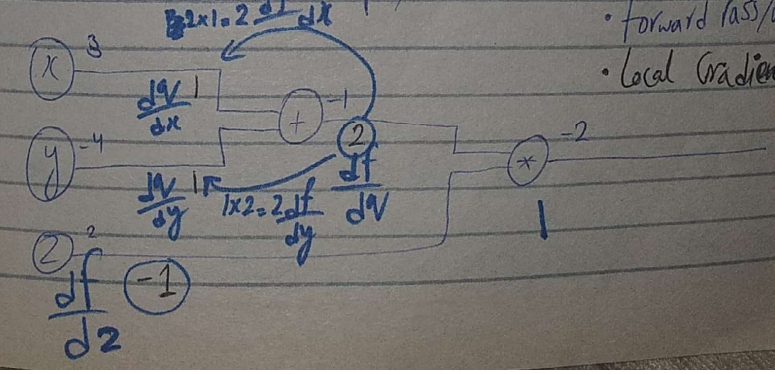
we make circuits to calculate gradients. To calculate multi level composition of functions & to reduce complexity/steps in using chain Rule.

Steps.

$$f = (x+y)z$$

① Identify variables in function

② Make Real valued Graph/circuit



- forward pass/output
- local Gradients

Local Gradients:

$q = x+y$

$\frac{\partial f}{\partial x} = 1$

$\frac{\partial q}{\partial y} = 1$

Example.

$$\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2x_2)}}$$

Make Circuit:

Solution.