Therefore, finally it can be said that -

$$\frac{\partial}{\partial \hat{\theta}_0^0} = \underbrace{\left[\underbrace{\left[1\ 0\ 0\ 0 \cdots\right]}_{\substack{1\times M \text{ one hot} \\ \text{encoded vector for} \\ 0 \text{ class label}}} - \frac{e^{(\hat{\theta}_0^0 + \hat{\theta}^{0T} x_m)}}{\sum_{k=0}^{M-1}} \right]}_{1\times M \text{ vector}} \begin{bmatrix} c_m^0 \\ \vdots \\ c_m^{M-1} \end{bmatrix} \Bigg\updownarrow \begin{matrix} M\times 1 \\ \text{vector} \end{matrix}$$

But we are taking $\frac{\partial}{\partial \hat{\theta}_0^0}\left[\log_e L(\hat{\theta}_0, \hat{\theta})\right]$ so —

final expression will be —

$$\frac{\partial}{\partial \hat{\theta}_0^0} = \frac{1}{N_{train}} \sum_{m=1}^{N_{train}} \left(\overbrace{\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots \end{bmatrix}}_{\text{One hot encoded representation of class 0 row}}}^{1 \times M \text{ vector}} - \underbrace{\frac{e^{(\hat{\theta}_0^0 + \hat{\theta}^{0T} x_m)}}{\sum_{k=0}^{M-1} e^{(\hat{\theta}_0^k + \hat{\theta}^{kT} x_m)}}}_{P(c^m = 0 \mid x_m)} \right) \cdot \underbrace{\begin{bmatrix} c_m^0 \\ c_m^1 \\ \vdots \\ c_m^{M-1} \end{bmatrix}}_{M \times 1 \text{ vector}}$$

Lets call the one hot encoded representation of class label of mth example, $c^m \Rightarrow \underbrace{\left[c_m^0 \; c_m^1 \; \ldots\ldots \; c_m^{M-1} \right]}_{1 \times M \text{ vector}}$ as $\underline{\underline{O_m}}$

Therefore, now

$$O_m = \underbrace{\left[c_m^0 \; c_m^1 \; \ldots\ldots \; c_m^{M-1} \right]}_{1 \times \underline{\underline{M}} \text{ one hot encoded representation of class label for } m^{th} \text{ } \underline{\underline{\text{example}}}}$$

So, finally we can write —

Scalar Quantity

$$\frac{\partial}{\partial \hat{\theta}_0}\left[\log_e L(\hat{\theta}_0, \hat{\theta})\right] = \frac{1}{N_{train}}\left[\sum_{m=1}^{N_{train}} \left(\underbrace{\left[1\ 0\ 0\ 0 \ldots\right]}_{1 \times M} - P(c^m = 0 \mid X_m)\right) \cdot O_m^T\right.$$

Lets consider an identity matrix of $\underline{\underline{M \times M}}$

$I[0,:] \rightarrow$ Numpy Notation

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & 0 & \cdots \\ & & & \vdots & & \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

← one hot encoded vector for class 0

⋮

← for class M-1

Identity Matrix of Size, $M \times M$

Therefore, it can be finally said that —

$$\frac{\partial}{\partial \hat{\theta}_0^0}\left[\log_e L(\hat{\theta}_0, \hat{\theta})\right] = \frac{1}{N_{train}} \sum_{m=1}^{N_{train}} \left(I[0,:] - P(c^m = 0 \mid x_m) \right) \cdot \theta_m^T$$

$$O = \begin{bmatrix} O_1 \\ O_2 \\ \vdots \\ O_{N_{train}} \end{bmatrix}$$

$1 \times M$

$1 \times M$

One hot encoded representation of class label of 1st Example

$\Rightarrow$

One hot encoded representation of class label of last Example

$N_{train} \times M$

Matrix

Lets consider a matrix of S-softmax probabilities evaluated for every example for every class —

$$P = \begin{bmatrix} P(c^1=0|x_1) & P(c^1=1|x_1) & \cdots & P(c^1=M-1|x_1) \\ \longleftarrow & 1 \times M \text{ vector} & \longrightarrow & \\ \vdots & & & \\ P(c^{N_{train}}=0|x_{N_{train}}) & \cdots & \vdots & P(c^{N_{train}}=M-1|x_{N_{train}}) \end{bmatrix}$$

$N_{train} \times M$ Matrix

If we now do —

$$I[0,:] - P = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots \end{bmatrix}}_{1 \times M \text{ vector}} - P$$

We get —

$$
\begin{bmatrix}
1 - P(C^1 = 0 \mid X_0) & 0 - P(C^1 = 1 \mid X_1) & \cdots & 0 - P(C^1 = M-1 \mid X_1) \\
1 - P(C^2 = 0 \mid X_2) & \vdots & & \vdots \\
\vdots & \vdots & & \vdots \\
1 - P(C^{N_{train}} = 0 \mid X_{N_{train}}) & 0 - P(C^{N_{train}} = 1 \mid X_{N_{train}}) & \cdots & 0 - P(C^{N_{train}} = M-1 \mid X_{N_{train}})
\end{bmatrix}
$$

If we simply element by element multiply —

$$\left(\mathbb{1}[0,:] - P\right) * O \quad =$$

$$\begin{bmatrix} c_1^0\left(1 - P(c^1 = 0 \mid x_1)\right) & & c_1^{M-1}\left(0 - P(c^1 = M-1 \mid x_1)\right) \\ \vdots & \cdots & \vdots \\ c_{N_{train}}^0\left(1 - P(c^{N_{train}} = 0 \mid x_{N_{train}})\right) & & c_{N_{train}}^{M-1}\left(0 - P(c^{N_{train}} = M-1 \mid x_{N_{train}})\right) \end{bmatrix}$$

If we take np.sum of the above matrix with axis = 1 then we get

$$\begin{bmatrix} np.matmul\left(\left(I[0,:]-P[0,:]\right), 0_1^T\right) \\ \vdots \\ np.matmul\left(\left(I[0,:]-P[N_{train},:]\right) 0_{N_{train}}^T\right) \end{bmatrix} \Bigg\updownarrow N_{train} \times 1$$

<u>vector</u>

Now, if we take the np. mean of the above vector we get -

$$\frac{\partial}{\partial \hat{\theta}_0} \left[\log_e L(\hat{\theta}_0, \hat{\theta}) \right] = \frac{1}{N_{train}} \sum_{m=1}^{N_{train}} \left(I[0, :] - P[m, :] \right) \cdot O_m^T$$

So, we have derived the formula for all the derivatives in —

$$\frac{\partial}{\partial \hat{\theta}_0} = \left[\frac{\partial}{\partial \hat{\theta}_0^0}, \frac{\partial}{\partial \hat{\theta}_0^1}, \ldots, \frac{\partial}{\partial \hat{\theta}_0^{M-1}} \right]$$

Any of the derivatives in the above gradient vector, $\frac{\partial}{\partial \hat{\theta}_0^k}$ can be directly computed using numpy as —

$$\frac{\partial}{\partial \hat{\theta}_0^k} = np.mean\left(np.sum\left(\left(\left(I[k,:] - P[:,k]\right) * D\right), axis=1\right)\right)$$

1 X M gradient vector

So, the overall gradient vector is given as –

$$\begin{bmatrix} \frac{\partial}{\partial \hat{\theta}_0^0} \log_e L(\hat{\theta}_0, \hat{\theta}) & \cdots & \frac{\partial}{\partial \hat{\theta}_0^{M-1}} \log_e L(\hat{\theta}_0, \hat{\theta}) \\ np.mean(np.sum(((I[0,:] - P)*O), axis=1)) & \cdots & \\ \cdots & np.mean(np.sum(((I[M-1,:] - P)*O), axis=1)) & \end{bmatrix}$$

Notice that —

$$np.mean\left(np.sum\left(\left(\left(I[0,:] - P\right) * O\right), axis=1\right)\right)$$

is giving us —

$$\frac{1}{N_{train}} \sum_{m=0}^{N_{train}} \left(I[0,:] - P[m,:]\right) . O_m^T$$

If we multiply O_m^T inside what do we get

$$\frac{1}{N_{train}} \sum_{m=0}^{N_{train}} \left(\underbrace{I[0,:]}_{1\times M \text{ vector}} \cdot \underbrace{O_m^T}_{M\times 1} - \underbrace{P[m,:]}_{1\times M \text{ vector}} \cdot O_m^T \right)$$

$$\boxed{\left[P(c^m = 0 \mid x_m) \; \ldots\ldots \; P(c^m = M-1 \mid x_m) \right]}$$

one hot encoded vector of mth example

Note that for any m^{th} example, $I[0,:]$ will be either orthogonal to O_m or it will be in the same direction as $\underline{\underline{O_m}}$

When $I[0,:]$ is orthogonal to O_m then

$$I[0,:] \cdot O_m^T = 0$$

otherwise —

$$I[0,:] \cdot O_m^T = 1$$

Only these two cases will occur

So, we can therefore write the derivative as -

It is important to understand this

$$\boxed{\frac{1}{N_{train}} \sum_{m=1}^{N_{train}} \left(O[m, O] - P[m, :] \cdot O_m^T \right)} \rightarrow 7$$

Stop here and think about how I have written this thing !!!

$$P = \begin{bmatrix} P(c^1 = 0 \mid x_1) & & P(c^1 = M-1 \mid x_1) \\ \vdots & & \vdots \\ \vdots & \cdots & \vdots \\ P(c^{N_{train}} = 0 \mid x_1) & & P(c^{N_{train}} = M-1 \mid x_{N_{train}}) \end{bmatrix}$$

$N_{train} \times M$ matrix.

$$O^T = \begin{bmatrix} c_1^0 & & \\ c_1^1 & & \\ c_1^2 & & \\ \vdots & & \\ c_1^{M-1} & & \end{bmatrix}$$

$M \times N_{train}$ matrix

If we matrix multiply both then we get -

$$P \cdot O^T = \begin{bmatrix} P[1,:] \cdot O_1^T & - & - & - & - & - & - & - & - & - \\ | & P[2,:] \cdot O_2^T & \cdot & - & - & - & - & - & - & - \\ | & | & P[1,:] \cdot O_3^T & \cdot & - & - & - & - & - & - \\ | & | & | & \ddots & - & - & - & - & - & - \\ | & | & | & | & \ddots & - & - & - & - & - \\ | & | & | & | & - & \ddots & - & - & - & - \\ | & | & | & | & - & - & \ddots & - & - & - \\ | & | & | & | & - & - & - & \ddots & - & - \\ | & | & | & | & - & - & - & - & \ddots & - \\ | & | & | & | & - & - & - & - & - & P[N_{train},:] \cdot O_{N_{train}}^T \end{bmatrix}$$

$$np.diag(np.matmul(P, O^T)) = 1 \times N_{train} \text{ vector}$$

$$\left[P[1,:] \cdot O_1^T,\ P[2,:] \cdot O_2^T \ \cdots \ \cdots \ P[N_{train},:] \cdot O_{N_{train}}^T \right]$$

$1 \times M$ vector

$M \times 1$ vector

Therefore,

$$\frac{\partial}{\partial \hat{\theta}_0^0} \log_e L(\hat{\theta}_0, \hat{\theta}) = \frac{1}{N_{train}} \sum_{m=1}^{N_{train}} \left(O[m, 0] - np.diag(np.matmul(P, O^T))[m] \right)$$

This sum can be further simplified as —

$$\frac{\partial}{\partial \hat{\theta}_0} \log_e L(\hat{\theta}_0, \hat{\theta}) =$$

$N_{train} \times 1$ vector

$1 \times N_{train}$ vector

$$\text{np.mean}\left(O[:, 0] - \overbrace{\text{diag}\left(\text{np.matmul}\left(P, O^T\right)\right)}^{T} \right)$$

Simplified even further

Now others can also be simplified

Lets see how

In general $-\frac{\partial}{\partial \hat{\theta}_0^k} \log L(\hat{\theta}_0, \hat{\theta}) =$

$$\text{np.mean}\left(O[:, k] - \text{diag}\left(\text{np.matmul}\left(P, O^T \right) \right)^T \right)$$

This can be calculated <u>superfast</u> further

$$\frac{\partial}{\partial \hat{\theta}_0} \log_e L(\hat{\theta}_0, \hat{\theta}) = \overbrace{\left[\frac{\partial}{\partial \hat{\theta}_0^0}, \frac{\partial}{\partial \hat{\theta}_0^1}, \cdots \frac{\partial}{\partial \hat{\theta}_0^k} \cdots \frac{\partial}{\partial \hat{\theta}_0^{M-1}} \right]}^{1 \times M \text{ vector}}$$

Q.E.D

$$= \text{np.mean}\left(\left(\overset{N_{train} \times M \text{ matrix}}{O} - \overbrace{\text{diag}\left(\text{np.matmul}\left(P, O^T \right) \right)}^{1 \times N_{train}} \right), \text{axis} = 0 \right)$$

1st Objective

Achieved

Lets now compute the derivative matrix inversed in second update —

$$\nabla_{\hat{\theta}} \log_c L(\hat{\theta}_1, \hat{\theta}) = \begin{bmatrix} \frac{\partial}{\partial \hat{\theta}_1^0} & \cdots & \frac{\partial}{\partial \hat{\theta}_1^k} & \cdots & \frac{\partial}{\partial \hat{\theta}_1^{M-1}} \\ \vdots & & \vdots & & \vdots \\ \cdots & \cdots & \frac{\partial}{\partial \hat{\theta}_N^k} & \cdots & \vdots \\ \vdots & & \vdots & & \vdots \\ \frac{\partial}{\partial \hat{\theta}_N^0} & \cdots & \frac{\partial}{\partial \hat{\theta}_N^k} & \cdots & \frac{\partial}{\partial \hat{\theta}_N^{M-1}} \end{bmatrix}$$

We know that — By (5) and (6)

$$\frac{\partial}{\partial \hat{\theta}_n^k} \log_e L(\hat{\theta}, \hat{\gamma}) = \frac{1}{N_{train}} \sum_{m=0}^{N_{train}} \left(I[k, :] - P[m, :] \right) \cdot O_m^T \cdot x_n^m$$

Furthermore, we can write this derivative as -

X_h is the column vector of the hth feature in the dataset -

$$= \frac{1}{N_{train}} \cdot np.matmul\left(X_h^T, \left(\underbrace{O[:,k]}_{N_{train} \times 1 \text{ vector}} - \underbrace{diag\left(np.matmul\left(P, O^T \right)^T \right)}_{1 \times N_{train} \text{ vector and transpose will give } N_{train} \times 1} \right) \right)$$

$X_h^T \longrightarrow 1 \times N_{train}$ vector

Therefore all the derivatives are given as –

$$= \frac{1}{N_{train}} \text{np.matmul}\Big(\underbrace{(X^T)^T}_{N \times N_{train} \text{ matrix}}, \underbrace{\big(O - \text{diag}(\text{np.matmul}(P, O^T))\big)^T}_{N_{train} \times M \text{ matrix}} \Big)$$

The answer is $N \times M$ matrix of derivatives

Now the final thing, how to get the P matrix. —

$$U = np.exp\left(\underbrace{\hat{\theta}_0}_{1 \times N \text{ matrix}} + np.matmul\left(\underbrace{X^T}_{N_{train} \times N \text{ matrix}}, \underbrace{\hat{\theta}}_{N \times M \text{ matrix}}\right)\right)$$

$$S = np.sum(U, axis = 1)$$

$$S = S.reshape(S.shape[0], 1)$$

$$P = U / S$$ $\longrightarrow$ Treats our P matrix of size $\underline{\underline{N_{train} \times M}}$

Finally Gradient-Descent Algorithm for Minimizing

$-\log_e L(\hat{\theta}_0, \hat{\theta})$:-

theta0_initial = np.random.randn(1, M)

theta_initial = np.random.randn(N, M)

step_size = 10**(-3), tol = 10**(-5)

We are assuming that the training_data is having all the training examples along with one hot encoded vector representations of their labels.

So, O = training_data[labels] $\Rightarrow$ O.shape = (N_{train}, M)

Furthermore, we are assuming that normalized_data is only having features of training_data = X^T = normalized_data

X^T

```
while (True):
    U_initial = theta0_initial + np.matmul(normalized_data, theta_initial)
    S_initial = np.mean(U_initial, axis=1)
    P_initial = U_initial / S_initial
```

Now lets work the two updates —

$\nabla_{\hat{\theta}_0} \log_c L(\hat{\theta}_0, \hat{\theta}) \longrightarrow 1 \times M$ gradient-vector

$$\text{theta0_final} = \text{theta0_initial} + \text{step_size} * \boxed{np.mean\left(O - \left(np.diag\left(np.matmul\left(P_initial, O^T\right)\right)\right)^T, axis = 0\right)}$$

$$\nabla_{\hat{\theta}} \log_e L(\hat{\theta}_0, \hat{\theta}) = N \times M$$ gradient-

$$\text{theta_final} = \text{theta_initial} + \underset{\text{fraction}}{\text{Step_size}} *$$

$$\left(\frac{1}{N_{train}} * np.matmul\left(normalized_data.T\right.\right.$$

$$\left.\left.\left.\left.\left.\left(O - \left(np.diag\left(np.matmul\left(P_initial, O^T\right)\right)\right)^T\right)\right)\right)\right)\right)$$

```
neg_log_loss_initial = -np.mean(np.sum((0 *
np.log(P_initial)), axis=1))
```

```
U_final = np.exp(theta0_final + np.matmul(normalized_data,
theta_final))
```

```
S_final = np.sum(U_final, axis=1)
P_final = U_final / S_final
neg_log_loss_final = -np.mean(np.sum((O*
np.log(P_final)), axis=1))
```

if abs(neg_log_loss_final − neg_log_loss_initial) < tol:

break

theta0_initial = theta0_final

theta_initial = theta_final

Q.E.D