Grad Problem Solving

Submitted By: Nafis Neehal

661990881

Problem 1:

- Image size = 3xNxN (RGB)
- L layers with h node/hidden layer, n nodes in the output layer [typically, input layers are not considered as layers as they don't have any learnable parameters to optimize like weights or bias. But, I'm considering input layer as a separate layer here for mathematical convenience].
- Input –> Hidden = [(3xNxN + 1) x h] number of learnable parameters, 3xNxNxh weights and h biases
- For (L-2) number of Hidden Layers = [(L-2)x{(hxh)+h}] number of total learnable parameters, where each hidden layer has hxh number of weights as input and h number of biases.
- $\text{Hidden}^{[L-1]}$ -> Output = [(hxn) + n] number of total learnable parameters.
- Final Formula:
  [(3xNxN + 1) x h] + [(L-2)x{(hxh)+h}] + [(hxn) + n]

Problem 2:

- 3xNxN at input
- Conv filters, kxk size, number of channels = number of filters = d
- Padding = 1, Stride = 1 will keep the image dimension same:
  - **Proof:**
    Let image size nxn = 6x6, filter size = 3x3, padding
    Output image dimension = $\left(\frac{n+2p-f}{s}+1\right)$ x $\left(\frac{n+2p-f}{s}+1\right)$ = 6x6
- L #of conv layers, F #of FC layers, $n^{(o)}$ #of nodes in output layer
- Theory -> Conv layers will only change depth if number of filter increases from d, image size will remain NxN as padding=1 is set.
- Theory -> Max-pool layers will pull down the image size (height and width only, not depth) into half of the input (as filter size 2x2, so stride will be 2 as non-overlapping regions).
- Each conv is followed by max-pool! So L number of Convs and L number of Max-pools.
- Filters in Conv layers can only be considered as learnable parameters, Max-Pool doesn't have any of that, it just pulls out the max value of the selected image region.
- As, in each layer image size will reduce by a factor of 2,
  - At i^{th} layer, image dimension will be $\frac{dxNxN}{2^{2i}}$
  - Number of learnable parameters in each CONV layer will be [dx3x3 + d] = [d x {(3x3)+1}], when number of bias parameters is d (1 bias for each filter).
- So, to generalize, if the CONV filter size is fxf, number of filters is d, and there are L #of CONV filters, then total learnable all CONV parameters are =

L x [d x {(fxf)+1}]

- For FC layers, number of learnable paramers =

  Number of activations in previous layer x number of activations in current layer + 1

- So if we have F number of FC layers, and h nodes in each FC layers and $n^{(o)}$ number of nodes in the output layer, then:
  - FC->FC = (F-2) x h
  - Conv-last-layer->FC-1st-layer = [d x {(fxf)+1}] x h
  - Total for all FC layers: [(F-2) x h] + [[d x {(fxf)+1}] x h]
- Total number of learnable parameters in output layer:
  - FC-Last-Layer->Output = $n^{(o)}$ x h

Problem 3(a):

The proof is a little bit unorthodox and uses inference rather than a solid mathematical proof. But the logic is perfect.

Let's say-

Output at final layer = [1, 3.5, 757, -5]

#this output vector covers normal small number, decimal, large number and negative number

For softmax, for each output node Z_i ,

Final output will be = Softmax(Z_i) = $\frac{e^{Zi}}{\sum_{i=1}^{k} e^{Zk}}$

Now, e^{1} = 2.718, $e^{3.5}$ = 33.1545, e^{757} = 5.76, e^{-5} = 0.007

So, while applying Softmax to each of the output node, denominator always will be =

2.718 + 33.1545 + 5.76 + 0.001 = 41.639

And for each softmax, one of the 4 exponential values will be as numerator.

So, $\frac{e^{Zi}}{\sum_{i=1}^{k} e^{Zk}}$ can never be greater than 1 and can be at **max very close to 1** if one of the four values is actually equal to denominator.

Again, as it is an exponential function, whatever the input is, the output will always have to be **greater than 0**. So, the value of softmax is always lying in between 0 and 1 which represents probability distribution.

Problem 3(b):

Let $\sum_{i=1}^{k} e^{\mathrm{Zk}}$ = SUM

Pj = $\frac{e^{\mathrm{Zj}}}{\sum_{i=1}^{k} e^{\mathrm{Zk}}}$ = $\frac{e^{\mathrm{Zj}}}{SUM}$

$\frac{\partial Pi}{\partial Zk} = \frac{\partial}{\partial Zk}\left(\frac{e^{\mathrm{Zi}}}{SUM}\right)$

By applying, f'(U/V) = $\frac{Vf'(U)-U\,f'(V)}{V\text{^}2}$,

$\frac{\partial Pi}{\partial Zk} = \frac{e^{\mathrm{Zi}} * SUM - e^{\mathrm{Zi}} e^{\mathrm{Zk}}}{SUM\text{^}2}$ [for i=j case]

= $\frac{e^{\mathrm{Zi}}\,(SUM - e^{\mathrm{Zk}})}{SUM*SUM}$ = Pi * (1-Pi)

$\frac{\partial Pi}{\partial Zk} = \frac{e^{\mathrm{Zi}} * SUM - e^{\mathrm{Zi}} e^{\mathrm{Zk}}}{SUM\text{^}2}$ [for i≠j case]

= $\frac{0 - e^{\mathrm{Zi}} e^{\mathrm{Zk}}}{SUM\text{^}2}$ [for i=j case]

= - Pi*Pk

We know, Kronecker delta function, $\delta ij = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$

Combining both (i=j) and (i≠ j) cases,

$$\frac{\partial Pi}{\partial Zk} = \begin{cases} \mathrm{Pi} * (1 - \mathrm{Pi}), & i = j \\ \quad - \mathrm{Pi} * \mathrm{Pk}, & i \neq j \end{cases}$$

= Pi (δik - Pk)

Problem 3(C)

Cross Entropy Loss FN: L (p,y) = $-\sum_i Yi * \log{(Pi)}$ = -Y1*log(P1) - Y2*log(P2) - … … … -Yk*log(Pk)

Net Output at output layer = Zi

Pi = Softmax(Zi)

$\frac{\partial L}{\partial \mathrm{Zi}} = \frac{\partial Pi}{\partial Zi} X \ \frac{\partial L}{\partial Pi}$ [Applied chain rule]

$\frac{\partial L}{\partial Pi} = -\frac{Yi}{Pi} \quad for\ \forall i = 1 \ldots k$

$\frac{\partial Pi}{\partial Zk} = \begin{cases} \mathrm{Pi} * (1 - \mathrm{Pi}), & i = j \\ - \mathrm{Pi} * \mathrm{Pk}, & i \neq j \end{cases}$ [from 3(b)]

Putting it together,

$$\frac{\partial Pi}{\partial Zi} X \ \frac{\partial L}{\partial Pi} = \begin{bmatrix} P1(1-P1) & \cdots & -P1Pk \\ \vdots & \ddots & \vdots \\ -P1Pk & \cdots & Pk(1-Pk) \end{bmatrix} \begin{bmatrix} -\frac{Y1}{P1} \\ \ldots \\ -\frac{Yk}{Pk} \end{bmatrix}$$

$$= \begin{bmatrix} -Y1 + Y1.P1 + \sum_{i\neq 1}^{k} YiP1 \\ \ldots \\ -Yk + YK.PK + \sum_{i\neq 1}^{k} Yi.Pk \end{bmatrix}$$

$$= \begin{bmatrix} -Y1 + P1\sum_{i=1}^{k} Yi \\ \ldots \\ -Yk + PK\sum_{i=1}^{k} Yi \end{bmatrix}$$

= $\begin{bmatrix} -Y1 + P1 \\ \ldots \\ -YK + PK \end{bmatrix}$, since $\sum_{i=1}^{k} Yi$ = 1 [softmax is probability distribution]

= Pi – Yi $\quad for\ \forall i = 1 \ldots k$