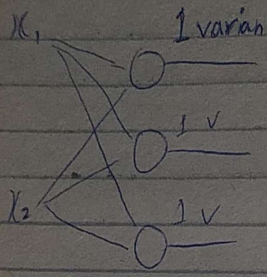
15-11-2021

Week#12 Lecture#15 Monday

Batch Normalization:-



no. of inputs — no. of outputs

fc1 = dense (inp_shape, [3, "relu"])

3 neurons — Activation function

fully connected layer.

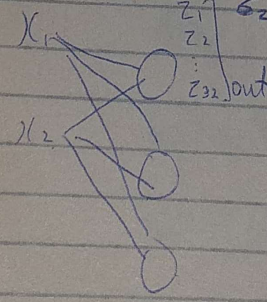
Above is in Xavier Initialization.

Batch Normalization:-

called Batch Norm.

We add a layer after each layer in our NN.

we send
So ~~X_{32}~~ ↓ batches



μ_2 = mean

σ_2 = standardized output

apply Standardization on each neuron & their input on each neuron is 32 images

Their mean is Zero & variance is 1.

32 - images = batch

If we use this, then our weights initialization does affect our output that much.

we also update μ_2, σ_2

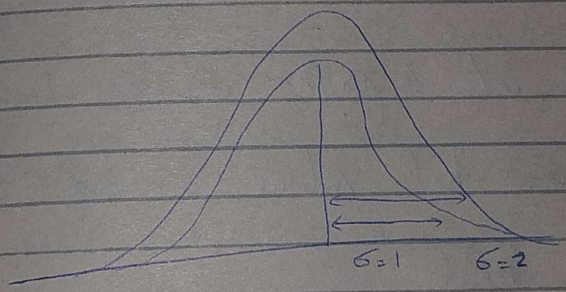
Z_i

$$Z_i = \frac{Z_i - \mu_2}{\sigma_2}$$] standard deviation = 1, Mean = 0

$$Z_i = \underbrace{\alpha Z_i}_{\text{new standard deviation}} + \underbrace{\beta}_{\text{new mean}}$$

instead of having $\sigma = 1$, $\mu = 0$

we multiply it with another standard deviation



So some input requires $\sigma = 1$, & some require $\sigma = 2$ or some other value.

we learn these α & β.

α = New Standard deviation

β = New Mean

Reason:

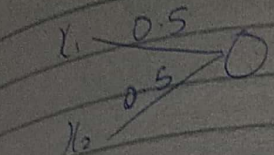
some input comes from different distribution so these images have different outputs at neuron & it affects our learning.

So what we do is send 32 images at each neuron & apply standardization at each neuron. Now, there standard deviation = 1 & Mean = 0 ~~Error~~ our assumption was these $\sigma = 1$, $\mu = 0$ will be good.

But Now, we let algorithm decide which input requires ~~what~~ what values for ~~output~~ σ & ~~μ~~ for current input.

Each Neuron will have its own α & β value which it will be learning.

Example: $w_1 = 0.5$	$w_2 = 0.5$	Z	~~$(Z-\mu)^2$~~	$Z_i = \frac{Z_i - \mu}{\sigma}$	$\hat{Z}_i = \alpha Z_i + \beta$
20	10	15	$(15-14)^2 = 1^2$	$1/6.633$	
30	20	25	$(25-14)^2 = 11^2$	$121/6.633$	
20	10	15	$(15-14)^2 = 1^2$	$1/6.633$	
10	10	10	$(10-14)^2 = 16$	$16/6.633$	
5	5	5	$(5-14)^2 = 81$	$81/6.633$	
x_1	x_2				



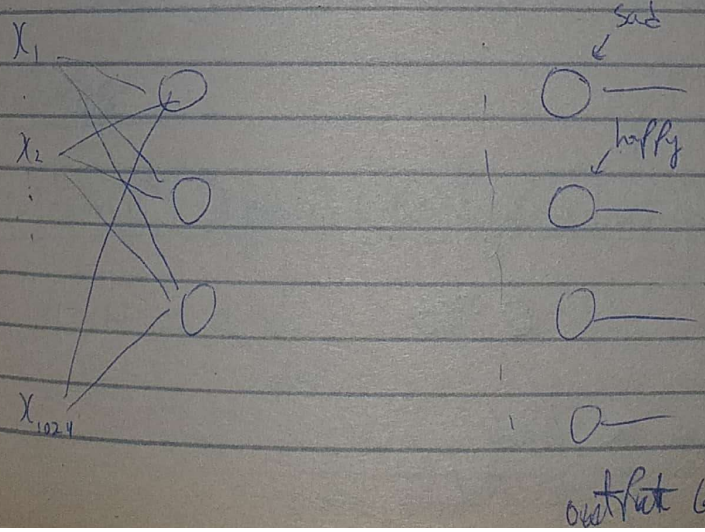
$\mu = 14$

Loss Functions:

$$L_{svm} = \sum_{y \neq j=\infty}^{N-1} (0, S_j - S_y + 1)$$

$$L_{softmax} = -\log\left(\frac{e^{S_y}}{\sum_{j=0}^{N-1} e^{S_j}}\right)$$

But Now, if we have multi-label classification, then our output layers would be total no. of unique labels. Now, every output has probability 0-1. So we use Sigmoid Activation function



We can use Softmax Loss function,

$$L = \sum_{j \neq y} -y * \log(P_i)$$

1	belongs to this class
0	Doesn't belong to this class.
1	
0	
0	

5 labels

e.g.:-

0.8
0.1
0.2
0.75
0.1

After applying sigmoid at output layer.

Now we need loss of this output so we do

0.8	→	1
0.1		0
0.2		0
0.75		1
0.1		0

So

$= -1 * \log(0.8) - 1 * \log(0.75)$

$= 0.735$

This ones the labels
y not Predicted labels.

Regression Problem is more difficult to solve than classification Problem.

3 abs
Next Friday deadline

Assignment:

Sklearn / keras

Take images dataset 1 image = 1 label MLP fully connected NN

& Do image classification using sklearn

Tensorflow

Multi-label classification

pytorch

Regression

input images

we try to convert Regression Problem to classification.

- nestro
- Adam

] optimizer.

0.79

0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1.0
2.0
2.1

optimizer:-

① Gradient Descent / Vanilla Gradient Descent:

$$w = w - \alpha \frac{dL}{dw}$$

In this case, throughout our learning rates doesn't change.

for each weight, learning rate is same.

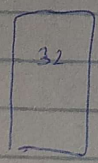
So those weights whose gradient is large & those weights whose gradient is small has different learning rate in each step so small gradient is learning slowly large gradients are learning quickly

② AdaGrad:-

Adaptive Gradient

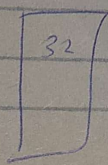
$$\alpha_{w_i} = \frac{\alpha}{\sqrt{Sum}} \quad , \alpha = 0.01$$

e.g.:-



$\rightarrow \frac{dL^{(1)}}{dw_1}$ → batch-1

next batch



$\rightarrow \frac{dL^{(2)}}{dw_1}$ → batch-2

Sum $\frac{dL}{dw_1}^{2(1)} + \frac{dL}{dw_1}^{2(2)}$

We take squared sum of these gradients.

We are taking square bcz gradients could be -ve. Then we take square root to bring magnitude down.

1st Problem.

a) Throughout learning sum is increasing in magnitude & so learning rate gets closer to zero, so our weights don't get updated as we go through our training. bcz α gets closer to zero.

b) Sum if closer to zero, our α gets increased, ~~$\frac{1}{0}$~~ or if sum gets = 0, so division by zero error could occur