optimization - most weighted

PAGE No.
DATE / /

Stochastic Gradient Descent (SGD)

$y = w * x + b$

update rule can be written as:

$w = w - learning\_rate * gradient\_w$

$b = b - learning\_rate * gradient\_b$

initialize weight parameter w & bias parameter b to small random value $w = 0.5$, $b = 0.1$

$$\text{weight gradient} = \frac{\partial L}{\partial w} = -x(y - \hat{y})$$

$$\text{bias gradient} = \frac{\partial L}{\partial b} = -y(y - \hat{y})$$

$$\text{weight update} = w = w - \alpha \frac{\partial L}{\partial w}$$

$$\text{bias update} = b - \alpha \frac{\partial L}{\partial w}$$

Q.

x	y
1	3
2	5
3	7
4	9
5	11

$y = wx + b$

$\hat{y} = (0.5 \times 1) + 0.1$

$\hat{y} = (0.5 \times 2) + 0.1$

$\hat{y} = (0.5 \times 3) + 0.1$

$\hat{y} = (0.5 \times 4) + 0.1$

$\hat{y} = (0.5 \times 5) + 0.1$

$$MSE = \frac{1}{2m} \sum_{i=1}^{n} (y - \hat{y})^2$$

$$= \frac{1}{2(5)} \sum_{i=1}^{n} \left((3 - 0.6)^2 + (5 - 1.1)^2 + (7 - 1.6)^2 + (9 - 2)^2 + (11 - 2.6)^2 \right)$$

$$\boxed{MSE = 16.83}$$

PAGE No.
DATE / /

$y = wx + b$

- for $x=1$ $\quad w=0.5, b=0.1$

$\hat{y} = wx + b$

$\hat{y} = (0.5 \times 1) + 0.1 = 0.6$

gradient (weight) $= \frac{\partial L}{\partial w} = -(y - \hat{y})x$

$= -(3 - 0.6) \times 1$

$= -2.4$

gradient bais $= \frac{\partial L}{\partial b} = -(3 - 0.6) = -2.4$

updated weight $= w - \alpha \frac{\partial L}{\partial w} = 0.5 - (0.01)(-2.4)$

$= 0.524$

updated bais $= b - \alpha \frac{\partial L}{\partial w} = 0.1 - (0.01)(-2.4)$

$= 0.124$

- for $x2$

$\hat{y} = wx + b$

$\hat{y} = 0.524 \times 2 + 0.124 = 1.172$

gradient weight $= \frac{\partial L}{\partial w} = -(y - \hat{y})x$

$= -(5 - 1.172) \times 2 = -7.656$

gradient bias $= \frac{\partial L}{\partial b} = -(5 - 1.172) = -3.828$

updated weight $= w - \alpha \frac{\partial L}{\partial w} = 0.524 - 0.01$

$= 0.600$

PAGE No.
DATE / /

Updated bias: $b - \alpha \frac{\partial L}{\partial w} = 0.124 - 0.01(-3.824)$

$= 0.162$

for $X_2 = 3$

$\hat{y} = wx + b$

$\hat{y} = (0.6 \times 3) + 0.162 = 1.962$

gradient weight $= \frac{\partial L}{\partial w} = -(y - \hat{y})x$

$= -(7 - 1.962) \times 3$

$= -15.114$

Gradient bias $= -(7 - 1.962) = -5.038$

updated weight $= 0.600 - 0.01(-15.114)$

$= 0.7512$

updated bias $= 0.162 - 0.01(-5.038) = 0.212$

for $X = 4$

$\hat{y} = wx + b$

$= (0.7512 \times 4) + 0.212 = 3.216$

Gradient weight $= \frac{\partial L}{\partial w} = -(y - \hat{y})x$

$= -(9 - 3.216)4$

$= 3.864$

Gradient bias $= -(9 - 3.864) = -5.136$

updated weight $= 0.7512 - 0.01(-3.864)$

$=$

PAGE No.
DATE / /

MSE after applying SGD optimization Algo

$x=1$	3	$\hat{y} = 1 \times 2.737 + 0.3287$	$(3-1.602)^2$
$x=2$	5	$\hat{y} = 2 \times 2.737 + 0.3287$	$(5-2.873)^2$
$x=3$	7	$\hat{y} = 3 \times 2.737 + 0.3287$	$(7-4.146)^2$
$x=4$	9	$\hat{y} = 4 \times 2.737 + 0.3287$	$(9-5.419)^2$
$x=5$	11	$\hat{y} = 5 \times 2.737 + 0.3287$	$(11-6.692)^2$

MSE
After SGD $= \frac{1}{2 \times 5} (1.954 + 4.524 + 8.152 + 12.880 + 18.558$

$= \frac{1}{10} (46.005$

RMSE $= 4.6005$

Momentum Gradient Descent

momentum coefficient $\beta = 0.9$

$w = 0.5$, $\alpha = 0.01$ $b = 0.1$ initial velocity $v_w = 0$, $v_b = 0$

Q Given dataset

x	y
1	3
2	5
3	7

$\hat{y} = (0.5 \times 1) + 0.1$

$\hat{y} = (0.5 \times 2) + 0.1$

$\hat{y} = (0.5 \times 3) + 0.1$

$$MSE = \frac{1}{2n} \sum_{i=1}^{n} (y - \hat{y})^2$$

$$= \frac{1}{2 \times 3} \left((3 - 0.6)^2 + (5 - 1.1)^2 + (7 - 1.6)^2 \right)$$

$$= \frac{1}{6} \left(5.76 + 15.21 + 29.16 \right)$$

$= 8.35$

Velocity updates:

$$v_w = \beta v_w + \alpha \frac{\partial L}{\partial w} \qquad v_b = \beta v_b + \alpha \frac{\partial L}{\partial b}$$

new weight update

$w = w - v_w$, $b = b - v_b$

Now for $x = 1$

$$\frac{\partial L}{\partial w} = -(y - \hat{y}) x, \quad \frac{\partial L}{\partial b} = -(y - \hat{y}) = -2.4$$

$= -2.4$

$v_w = (0.9 \times 0) + 0.01 (-2.4) = -0.024$

$v_b = (0.9 \times 0) + 0.01) (-2.4) = -0.024$

$w = 0.5 + 0.024 = 0.524$

$b = 0.1 + 0.024 = 0.124$

for $x = 2$

$\hat{y} = (0.524 \times 2) + 0.124$

$= 1.17$

PAGE No.
DATE / /

$\frac{\partial L}{\partial w} = -(5-1.17) \times 2 = -7.66$

$\frac{\partial L}{\partial b} = -(5-1.17) = -3.82$

$Vw = (0.9 \times (-0.025) + 0.01 (-7.66) = -0.099$

$Vb = (0.9 \times -0.024) + 0.01 (-3.82) =$

$w = 0.525 - (-0.099) = 0.623$

$b = 0.12 - (0.060) = 0.184$

for $x = 3$.

$\hat{y} = (0.623 \times 3) + 0.184 = 2.053$

$\frac{\partial L}{\partial w} = (7 - 2.053) \times 3 = -14.8$

$\frac{\partial L}{\partial b} = (7 - 2.053) = -4.9$

$Vw = (0.9 \times (-0.099) + 0.01 (-14.9) = -0.237$

$Vb = (0.9 \times (-0.060) + 0.01 (-4.9) = -0.103$

final weight $w = 0.623 - (-0.237) = 0.860$

$b = 0.184 - (-0.103) = 0.287$

Calculate mse for 1 & 2

$MSE = \frac{1}{2m} \sum_{i=1}^{n} (y - \hat{y})^2$

$= \frac{1}{2 \times 3} ((3 - 1.457)^2 + (5 - 2.001)^2 + (7 - 2.867)^2$

$MSE = 9.912$

$\hat{y} = (0.860) \times 1 + 0.287 = 1.147$

$\hat{y} = (0.860 \times 2 + 0.287 = 2.007$

$\hat{y} = (0.860 \times 3 + 0.287 = 2.867$

AdaGrad (Adaptive Gradient)

Q.1 Given $w = 0.5$ input $(x, y) = (3, 5)$
learning rate $= 0.1$ $\epsilon = 10^{-8}$

$\Rightarrow$ ~~$L = 1/n * (y\_pred - y)^2$~~

$\hat{y} = y_{pred} = 0.5 \times 3 = 1.5$

$$\frac{\partial L}{\partial w} = (-(y - \hat{y})x)$$

$$\frac{\partial L}{\partial w} = (-(5 - 1.5) \times 3) = -7.5$$

$$\alpha t = \sum_{i=1}^{t} (g_i)^2 = (-7.5)^2 = 56.25$$

$$n't = \frac{n}{\sqrt{\alpha_t + \epsilon}} = \frac{0.1}{\sqrt{56.25 + 10^{-8}}} = 0.013$$

$$wt = 0.5 - (0.013)(-7.5) = 0.5975$$

$$= w_{t-1} - n't \frac{\partial L}{\partial w(t-1)}$$

Q2 $w = 0.597$ input $(4, 5)$

$\hat{y} = y_{pred} = 0.597 \times 4 = 2.388$

$$\frac{\partial L}{\partial w} = -(y - \hat{y})x = (-(5 - 2.388) \times 4)$$
$$= -10.448$$

$$\frac{\partial L}{\partial w} = -10.448$$

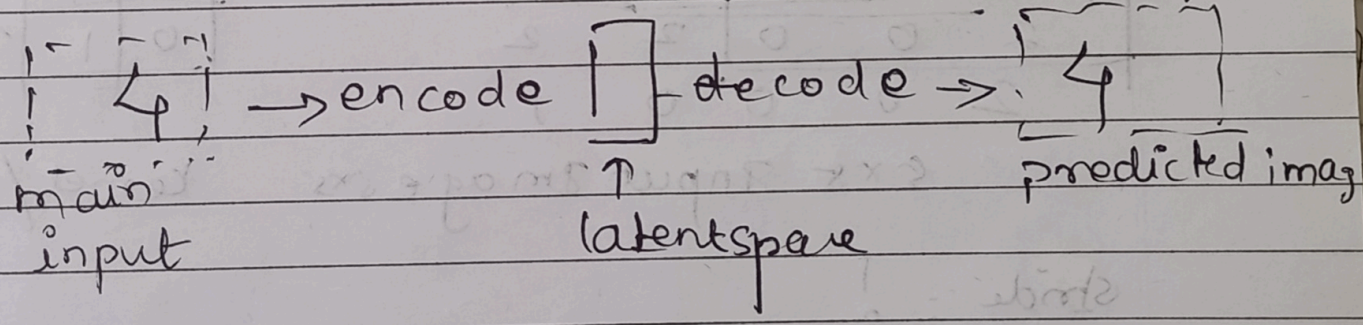
$$\alpha t = \sum_{i=1}^{t} (gi)^2 = (-10.448)^2 = 109.160$$

$$\eta t = \frac{\eta}{\sqrt{\alpha t + \varepsilon}} = \frac{0.1}{\sqrt{109.160 + 10^{-8}}} = 0.0095$$

$$wt = 0.597 - (0.0095)(-10.448)$$
$$= 0.597 + 0.09925$$
$$wt = 0.69625$$

◉ Auto encoders : PCA ∈ dimensional deduction

encoder : $h = f(x)$
decoder : $r = g(f(x))$ r as close x as possible



encoding : $z = w1 * x + b1$

decoding : $x' = w2 * z + b2$

$$loss = \frac{1}{n} * \Sigma (x - x')^2$$

1) Under complete autoencoder $n_h < d$
2) sperse Autoencoder
3) overcomplete encoder

Applications

1) Dimensionality Reduction
2) Anomaly detection
3) Denoising Images
4) feature learning
5) Generative modeling

CNN (Convolution Neural Network)

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

0	1	2
2	2	0
0	1	2

5x5 Input Image 5x5 Kernel / filter 3x3

stride = 1
= 2
= 3
;

12	12	17
10	17	19
9	6	14

← feature map

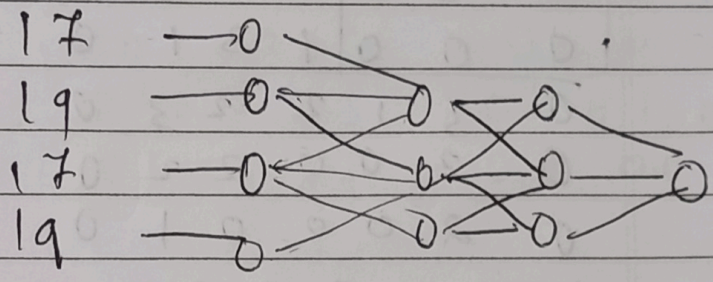
17	19
17	19

12.57	16.25
10.5	14

Avg max pooling Avg pooling

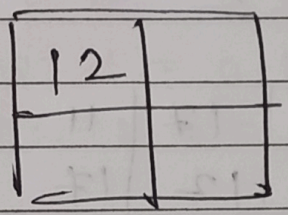
flattening

17 19
17 19

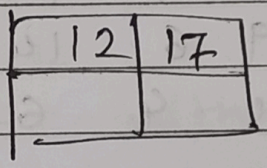


Now stride = 2

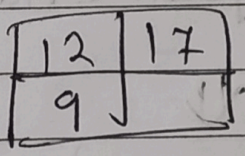
3 3 2
0 0 1
3 1 2



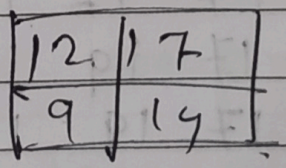
2 1 0
1 3 1
2 2 3



3 1 2
2 0 0
2 0 0

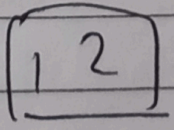


2 2 3
0 2 2
0 0 1



Now stride = 3

3 3 2
0 0 1
3 1 2



PAGE No.
DATE / /

solve by maxpooling

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0	1	3	1	0
0	3	1	2	2	3	0
0	2	0	0	2	2	0
0	2	0	0	0	1	0
0	0	0	0	0	0	0

7×7

0	1	2
2	2	0
0	1	2

3×3 kernel

6	14	17	11	13
14	12	12	17	11
8	10	17	19	13
11	9	6	14	12
6	4	4	6	4

5×5 feature matrix

↓

maxpooling

14	17	17	17
14	17	19	19
11	17	19	19
11	9	14	14

⇒ flattening

14	0
17	0
17	0
17	0
14	0
17	0
19	0
19	0
11	0
17	0
19	0
19	0
11	0
9	0
14	0
14	0

PAGE No. / DATE / /

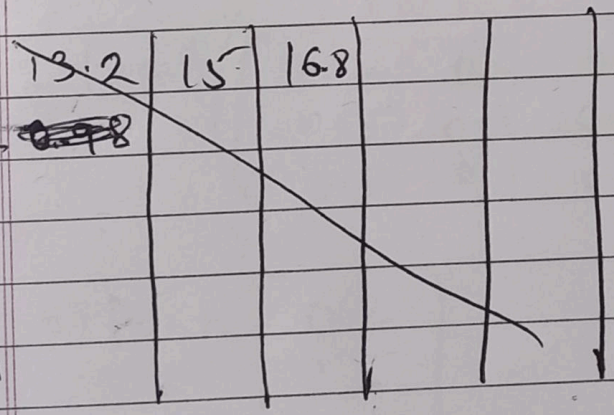
Q. 1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25

Kernel 1			Kernel 2		
0.1	0.2	0.3	-0.1	-0.1	-0.1
0.1	0.2	0.3	0.1	0.1	0.1
0.1	0.2	0.3	0.0	0.0	0.0

Input 5X5

1) Stride = 1
2) Bias Term for kernel 1 = 0.1
   Bias Term for kernel 2 = 0.2
3) Activation f^n is Relu.
4) Max pooling is 2X2
5) Flattering & FC layer with 2hidden layer with 2 neurons.

=> 22.2



13.2	15	16.8
22.2	24	25.8
31.2	33	34.8

feature map 1

class 1
class 2
class 3

1.5 1.5 1.5
1.5 1.5 1.5
1.5 1.5 1.5

feature map 2

Adding bias for feature

13.3	15.1	16.9
22.3	24.1	25.9
31.3	33.1	34.9

bias

1.7 1.7 1.7
1.7 1.7 1.7
1.7 1.7 1.7

Activation f^n convert
- to + value

, 1st row , 1st row

13.3	15.1	16.9	1.7	1.7	1.7
22.3	24.1	25.9	1.7	1.7	1.7
31.3	33.1	34.9	1.7	1.7	1.7

Apply Relu Activation function

after apply Relu Matrix will be same as above

stride 2 max pooling 2×2

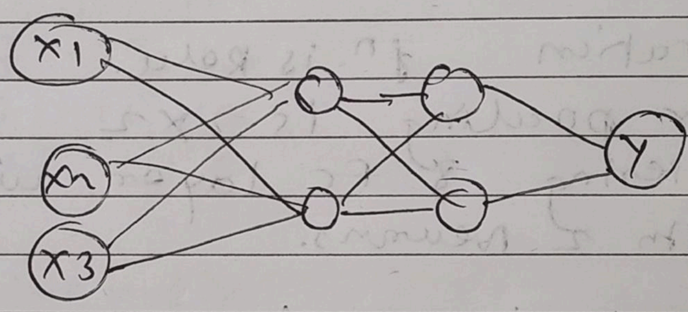
24.1 25.9 1.7

FC

24.1

25.9

1.7



flattening ⇓

24.1

25.9

1.7

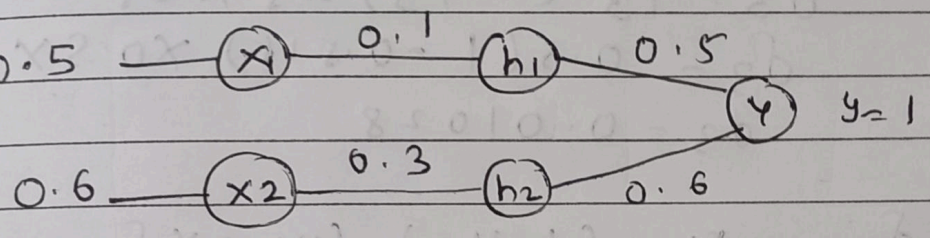
Imp: LeNet-5 Architecture, Autoencoder (theory)
CNN optimization, backpropagation (Numerical)

5×5, 3×3 max pooling

Assignment Class Test-8

Q. Perform one step back Propogation in simple neuron network with one I/P layer having 2 networks 1 hidden layer with 2N, $lr = 0.1$
Af = sigmoid Torget O/P: $y = 1$
$X = [0.5 \ 0.6]^T$ $w_1 = [0.1 \ 0.3]$

⇒



$H_3 = x_1 w_1 = 0.5 \times 0.1 = 0.05$
$H_3 = 0.5125$

$H_4 = x_2 w_2 = 0.6 \times 0.3 = 0.18$
$H_4 = 0.544$

$O_5 = H_3 w_3 + H_4 w_4$
$= 0.5125 \times 0.5 + 0.544 \times 0.6$
$= 0.58313$

$O_5 = 0.64178$
error $= y_1 - 0.5 = 1 - 0.64178 = 0.35822$

$S_5 = y(1-y)(y_2 - y)$
$S_5 = 0.0823$

$\Delta w_{45} = 0.1 \times 0.0823 \times 0.5448$
$w_{45} = 0.6 + 0.00448$
$= 0.60448$

$\Delta w_{35} = 0.1 \times 0.0823 \times 0.5125$

$= 0.004521$

$w_{35} = 0.5 + 0.004521 = 0.5$

for hidden layer,

$\delta_3 = y_3 (1 - y_3) w_5 \times \delta_5$

$\delta_3 = 0.5(1 - 0.518) \times 0.5 \times 0.0823$

$\delta_3 = 0.01028$

$\delta_4 = y_4 (1 - y_4) w_{45} \times \delta_5$

$\delta_4 = 0.554 (1 - 0.564) \times 0.6$

$\delta_4 = 0.01222$

$\Delta w_{13} = 0.1 \times \delta_3 \times 0.1 = 0.1 \times 0.01028$

$w_{13} = 0.1 \times 0.0005$

$= 0.100054$

$\Delta w_{24} = 0.1 \times \delta_4 \times 02 = 0.1 \times 0.012 \times 0.6$

$= 0.00073$

$w_{24} = 0.0007349$

Iteration 2

$H_3 = 0.5 \times 0.1005$

$= 0.05$

$H_3 = 0.512$

$H_4 = 0.6 \times 0.300$

$= 0.18$

$H_4 = 0.5549$

$O_5 = H_3 w_{35} + H_4 w_{45}$

$= 0.58787$

$O_5 = 1.80015$

Q.2 perform CNN output kernel matrix by 5x5 Input & 3x3 kernel

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \\ 3 & 4 & 5 & 1 & 2 \\ 4 & 5 & 1 & 2 & 3 \\ 5 & 1 & 2 & 3 & 4 \end{bmatrix} \qquad \begin{bmatrix} 0.2 & 0.2 & 0.3 \\ 0.2 & 0.2 & 0.3 \\ 0.2 & 0.2 & 0.3 \end{bmatrix}$$

feature map 1

$$\begin{matrix} 6.6 & 7.2 & 6.8 \\ 7.2 & 6.8 & 5.4 \\ 6.8 & 5.4 & 5.5 \end{matrix}$$

AF = Relu

As all vals are +ve same matrix

max pooling

$$\begin{matrix} 7.2 & 7.2 \\ 7.2 & 6.8 \end{matrix}$$

⇓

flattering

[7.2 7.2 7.2 6.8]