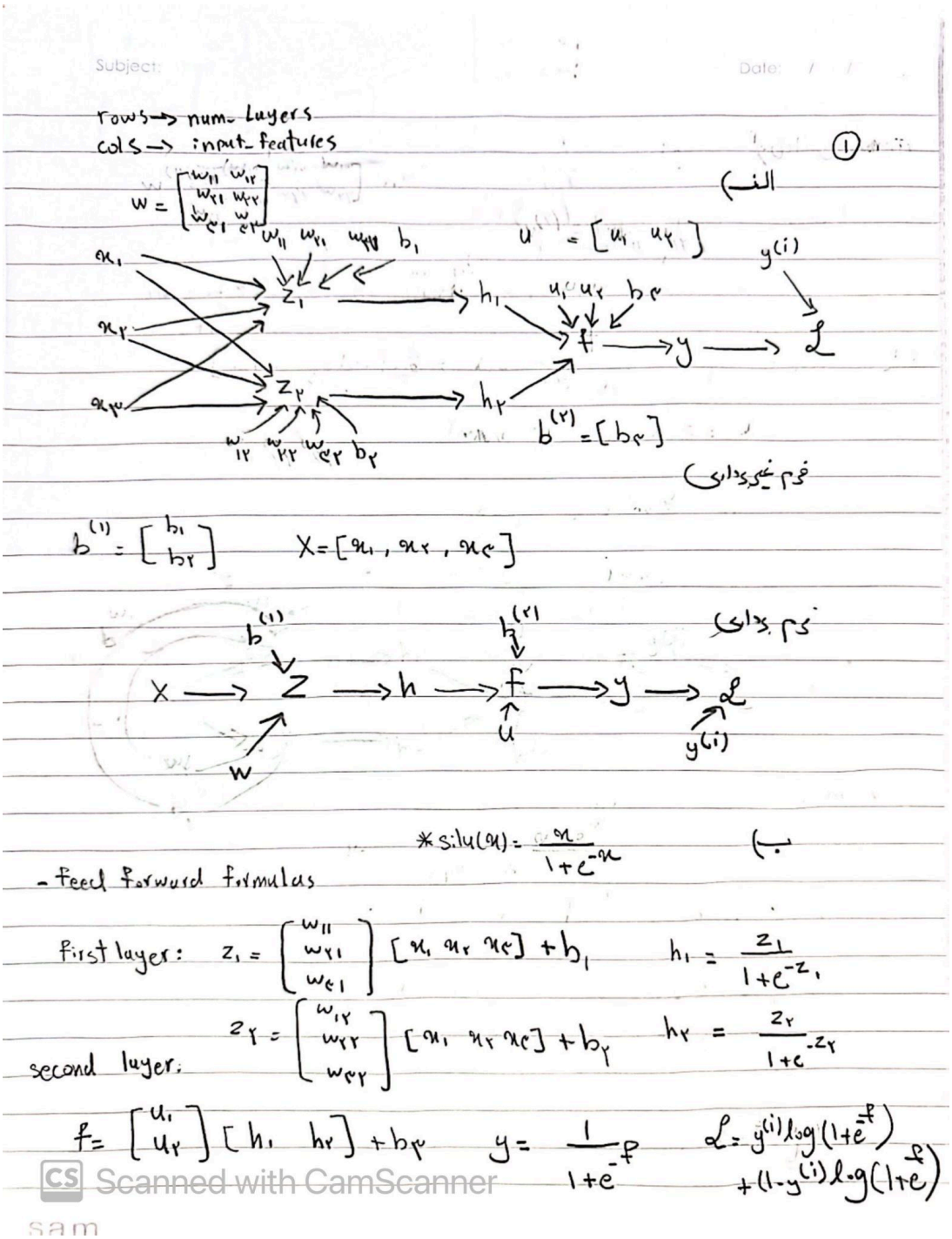
پوریا یزدانی ۴۰۰۲۴۳۰۸۲

تمرین دوم - بخش تئوری

سوال اول)

([matrix calculation notebook])

الف)



ب)

- Feed forward formulas (matrix form):

First layer: $Z = \underset{1\times 2}{X}\underset{2\times 2}{W} = [z_1 \quad z_2]$

$h = z \circ \sigma(z) = [z_1\sigma(z_1) \quad z_2\sigma(z_2)]$

second layer:

$[u_1 \quad u_2]\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$ $\quad f = \underset{1\times 2}{u}\ \underset{2\times 1}{h^T}$ $\quad y = \frac{1}{1+e^{-f}}$ $\quad \mathcal{L} = -y^{(i)}\log(y) - (1-y^{(i)})\log(1-y)$

- Backward pass (matrix form)

$\bar{\mathcal{L}} = 1$

$\bar{y} = \bar{\mathcal{L}}\left[\frac{-y^{(i)}}{y} + \frac{1-y^{(i)}}{1-y}\right]$

$\bar{f} = \bar{y}\frac{dy}{df} = \bar{y}\circ\sigma'(f)$

$\bar{h} = \bar{f}\frac{df}{dh} = \bar{f}\,u^T$

$\bar{z} = \bar{h}\frac{dh}{dz} = \bar{h}\circ(z\circ\sigma(z))' = \bar{h}\circ(\sigma(z) + z\sigma(z)')$

* $\bar{W} = \bar{z}\,x^T$
* $\bar{b}^{(1)} = \bar{z}$
* $\bar{u} = \bar{f}\,h^T$
* $\bar{b}^{(2)} = \bar{f}$

iteration 1

$$z = [x_1\ x_2\ x_3] \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} + [b_1\ \ b_2] \rightarrow b^{(1)}$$

$$\begin{bmatrix} 0.4 & -0.3 \\ 0.2 & 0.2 \\ -0.5 & 0.1 \end{bmatrix} \quad [\ 0 \quad -0.4\ ]$$

forward pass: First layer = $z_1 = X^{(1)} W^{(1)} + b^{(1)}$ $\rightarrow z_1$ $\rightarrow z_2$

$$[0.15 \quad -0.45]$$

$$z_2 = X^{(2)} W^{(1)} + b^{(1)} = [-0.34 \quad -0.24]$$

$$z_3 = X^{(3)} W^{(1)} + b^{(1)} = [-0.15 \quad -0.15]$$

$X^{(1)} = [1 \quad 0.3]$
$X^{(2)} = [0.1 \quad 0.5 \quad 1]$
$X^{(3)} = [0 \quad 0.7]$

$h_1 \rightarrow$ $\leftarrow h_2$

$$h_1 = z_1 \circ g(z_1) = [0.75 \quad -0.22]$$
$$h_2 = z_2 \circ g(z_2) = [0.10 \quad -0.14]$$
$$h_3 = z_3 \circ g(z_3) = [-0.07 \quad -0.15]$$

$$[\text{silu}(z_1) \quad \text{silu}(z_2)]$$

second layer:

$$f_1 = u h_1^T + b_2 = 0.15$$
$$f_2 = u h_2^T + b_2 = 0.07$$
$$f_3 = u h_3^T + b_2 = 0.09$$

$$f = [u_1\ u_2] \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} + b_2$$

$b_2 \rightarrow 0.1$

$u \rightarrow [0.3 \quad -0.1]$

$$y_1 = \frac{1}{1+e^{-f_1}} = 0.537$$
$$y_2 = \frac{1}{1+e^{-f_2}} = 0.517$$
$$y_3 = \frac{1}{1+e^{-f_3}} = 0.521$$

$\rightarrow y = \text{sigmoid}(f)$

$$L\_iter1 = \frac{L_1 + L_2 + L_3}{3}$$

$L_1 = 0.622$
$L_2 = 0.722$
$L_3 = 0.737$

$\rightarrow -y^{(i)} \log(y) - (1-t) \log(1-y)$

$$\boxed{L\_iter1 = 0.695}$$

s.a.m

Subject: Date: / /

Backward pass: $\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial f}\frac{\partial f}{\partial h}\frac{\partial h}{\partial z}\frac{\partial z}{\partial w}$ ($\frac{\partial h}{\partial z}$)

Gradients of W $\frac{\partial \mathcal{L}}{\partial w} = \bar{\mathcal{L}}\left[\frac{-y^{(i)}}{y} + \frac{1-y^{(i)}}{1-y}\right] \cdot \sigma'(P)\, U_0'(z\sigma(z))' x^T$

$= \left[\bar{\mathcal{L}}\left[\frac{-y^{(i)}}{y} + \frac{1-y^{(i)}}{1-y}\right] \cdot (y(1-y))\, U_0(\sigma(z) + z\sigma(z)(1-\sigma(z)))\right] x^T$

$= \left[-\left[\frac{-y^{(i)}}{y} + \frac{1-y^{(i)}}{1-y}\right] \cdot (y(1-y)) \cdot U_{1\times 1} \circ \begin{bmatrix} \sigma(z_1) + z_1\sigma(z_1)(1-\sigma(z_1)) \\ \sigma(z_2) + z_2\sigma(z_2)(1-\sigma(z_2)) \end{bmatrix}^T\right] [x_1\ x_2\ x_3]$

scalar ← element wise ← 1×2

طبق تعریف انجام شده

$u \to 1\times 2$, $\frac{dh}{dz} \to 1\times 2$ → element wise → $1\times 2 \to [1\times 2]^T \to 2\times 1$. $1\times 3 \to 2\times 3$

$\frac{\partial \mathcal{L}}{\partial w} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_{11}} & \frac{\partial \mathcal{L}}{\partial w_{12}} & \frac{\partial \mathcal{L}}{\partial w_{13}} \\ \frac{\partial \mathcal{L}}{\partial w_{21}} & \frac{\partial \mathcal{L}}{\partial w_{22}} & \frac{\partial \mathcal{L}}{\partial w_{23}} \end{bmatrix}$

For $x^{(1)}$ $= \begin{bmatrix} -0.05 & 0 & -0.03 \\ 0.0009 & 0 & 0.0006 \end{bmatrix}$ $x^{(1)} = [1\ 0\ 1]$ $y^{(1)} = 1$

For $x^{(2)}$ $= \begin{bmatrix} 0.04 & 0.05 & 0.06 \\ -0.02 & -0.01 & -0.03 \end{bmatrix}$ $x^{(2)} = [1\ 1\ 1]$ $y^{(2)} = 0$

For $x^{(3)}$ $= \begin{bmatrix} 0.04 & 0.05 & 0 \\ -0.02 & -0.01 & 0 \end{bmatrix}$ $x^{(3)} = [0\ 1\ 0]$ $y^{(3)} = 0$

average:

$n=3 \to \frac{1}{n}\sum_{i=1}^{n} \frac{\partial \mathcal{L}_i}{\partial w} = \begin{bmatrix} -0.01 & -0.01 & -0.02 \\ 0.0002 & -0.01 & 0.03 \end{bmatrix}$

Gradients of $b^{(1)}$ – first layer

$$\frac{\partial \mathcal{L}}{\partial b^{(1)}} = \frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial f} \cdot \frac{\partial f}{\partial h} \cdot \frac{\partial h}{\partial z}$$

$$= \left[\left(\frac{-y^{(i)}}{y} + \frac{1-y^{(i)}}{y} \right) \cdot y(1-y) \right] \cdot U \circ (\sigma(z) + z\sigma(z)(1-\sigma(z)))$$

(element-wise)

scalar ← 1×1 ; U : 1×2 ; $\sigma(z) + z\sigma(z)(1-\sigma(z))$: 1×2

$$= \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial b_1^{(1)}} & \frac{\partial \mathcal{L}}{\partial b_2^{(1)}} \end{bmatrix} = \left[\left(\frac{-y^{(i)}}{y} + \frac{1-y^{(i)}}{y} \right) \cdot y(1-y) \right] U \circ \begin{bmatrix} \sigma(z_1) + z_1\sigma(z_1)(1-\sigma(z_1)) \\ \sigma(z_2) + z_2\sigma(z_2)(1-\sigma(z_2)) \end{bmatrix}^T$$

for $X^{(1)} = [0.9 \quad -0.8]$ $\quad y^{(1)} = 1$

for $X^{(2)} = [0.5 \quad -0.4]$ $\quad y^{(2)} = 0$

for $X^{(3)} = [0.4 \quad -0.2]$ $\quad y^{(3)} = 0$

average

$n = 3 \rightarrow \frac{1}{n} \sum_{i=1}^{n} \frac{\partial \mathcal{L}^{(i)}}{\partial b^{(1)}} = [0.12 \quad -0.10]$

Subject: Date: / /

Gradients of U

$$\frac{\partial \mathcal{L}}{\partial U} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial f} h^T$$

$$= \left[\left[\frac{-y^{(i)}}{y} + \frac{1-y^{(i)}}{y}\right] \cdot y(1-y)\right] h^T \quad (1\times 2)$$

scalar

$$\frac{\partial \mathcal{L}}{\partial U} = \left[\frac{\partial \mathcal{L}}{\partial u_1} \quad \frac{\partial \mathcal{L}}{\partial u_2}\right] = \left[\left(\frac{-y^{(i)}}{y} + \frac{1-y^{(i)}}{y}\right) y(1-y)\right] [h_1 \quad h_2]$$

for $x^{(1)}$ = [illegible]

for $x^{(2)}$ = [illegible]

for $x^{(3)}$ = [illegible]

$n=3 \rightarrow \frac{1}{n}\sum_{i=1}^{n}\frac{\partial \mathcal{L}}{\partial u_i}$ = [illegible]

Gradients of $b^{(2)}$ _ second layer

$$\frac{\partial \mathcal{L}}{\partial b^{(2)}} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial f}$$

$$= \left[\frac{-y^{(i)}}{y} + \frac{1-y^{(i)}}{y} \cdot y(1-y)\right] \rightarrow \text{scalar}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(2)}} = \left[\frac{\partial \mathcal{L}}{\partial b_2}\right]$$

for $x^{(1)}$ = -0,64

for $x^{(2)}$ = 0,02

for $x^{(3)}$ = 0,02

$n=3 \quad \frac{1}{n}\sum_{i=1}^{n}\frac{\partial \mathcal{L}_i}{\partial b^{(2)}} = [0,19]$

sam

First iteration weight updates

GD with momentum → update velocity, → update weights

$v_0 = \vec{0}$

$\gamma = 0.9$

$\eta = 0.8$

$$W \Rightarrow \begin{cases} v_1 \leftarrow \gamma v_0 - \frac{\eta}{n}\left[\sum_{i=1}^{n} \frac{\partial \mathcal{L}_i}{W}\right] \\ W_{new} \leftarrow W_{old} + v_1 \end{cases}$$

$$\Rightarrow \begin{cases} v_1 \leftarrow \vec{0} - 0.8 \begin{bmatrix} -0.024 & 0.03 & 0.019 \\ 0.002 & -0.01 & -0.01 \end{bmatrix} \\ W_{new} \leftarrow \begin{bmatrix} 0.4 & 0.2 & -0.5 \\ -0.3 & 0.2 & 0.1 \end{bmatrix} + \begin{bmatrix} 0.02 & -0.024 & -0.015 \\ -0.001 & 0.009 & 0.008 \end{bmatrix} \end{cases}$$

$$= \begin{bmatrix} 0.41 & 0.17 & -0.51 \\ -0.30 & 0.20 & 0.10 \end{bmatrix}$$

$$b^{(1)} \Rightarrow \begin{cases} v_1 \leftarrow \gamma v_0 - \frac{\eta}{n}\left[\sum_{i=1}^{n} \frac{\partial \mathcal{L}_i}{b^{(1)}}\right] \\ b_{new} \leftarrow b_{old} + v_1 \end{cases}$$

$$\Rightarrow \begin{cases} v_1 \leftarrow \vec{0} - 0.8 \begin{bmatrix} 0.012 & -0.01 \end{bmatrix} \\ b_{new} \leftarrow \begin{bmatrix} 0 & -0.4 \end{bmatrix} + \begin{bmatrix} -0.01 & 0.009 \end{bmatrix} \end{cases}$$

$$= \begin{bmatrix} -0.01 & -0.39 \end{bmatrix}$$

$$U \Rightarrow \begin{cases} v_1 \leftarrow \gamma - \frac{\eta}{n}\left[\sum_{i=1}^{n} \frac{\partial \mathcal{L}_i}{\partial U}\right] \\ U_{new} \leftarrow U_{old} + v_1 \end{cases}$$

$$\Rightarrow \begin{cases} v_1 \leftarrow \vec{0} - 0.8 \begin{bmatrix} -0.05 & 0.006 \end{bmatrix} \\ U_{new} \leftarrow \begin{bmatrix} 0.3 & -0.1 \end{bmatrix} + \begin{bmatrix} 0.04 & -0.005 \end{bmatrix} \end{cases}$$

$$= [\,۰,۳۴ \quad -۰,۱۰۵\,]$$

$$b^{(۲)} \Rightarrow \begin{cases} v_1 \leftarrow \gamma v_0 - \frac{\eta}{n}\left[\sum_{i=1}^{n} \frac{\partial \mathcal{L}_i}{\partial b^{(۲)}}\right] \\ b^{(۲)}_{new} \leftarrow b^{(۲)}_{old} + v_1 \end{cases}$$

$$\Rightarrow \begin{cases} v_1 \leftarrow \vec{0} - ۰,۸[\,۰,۱۹\,] \\ b^{(۲)}_{new} \leftarrow [\,۰,۱\,] + [\,-۰,۱۵\,] \end{cases}$$

$$= [\,-۰,۰۵\,]$$

weights after first iteration

$$W_1 = \begin{bmatrix} ۰,۴۱ & ۰,۱۷ & -۰,۵۱ \\ -۰,۲۰ & ۰,۲۵ & ۰,۱۰ \end{bmatrix} \qquad U = [\,۰,۲۴ \quad -۰,۱۰۵\,]$$

$$b^{(1)} = [\,-۰,۰۱ \quad -۰,۲۹\,] \qquad b^{(۲)} = [\,-۰,۰۵\,]$$

iteration2

forward pass: first-layer:

$$z_1 = x^{(1)} W^T + b^{(1)} = [\,۰,۱۵ \quad -۰,۴۳\,]$$
$$z_۲ = x^{(۲)} W^T + b^{(1)} = [\,-۰,۴۰ \quad -۰,۲۵\,]$$
$$z_۳ = x^{(۳)} W^T + b^{(1)} = [\,-۰,۱۹ \quad -۰,۱۱\,]$$
$$h_1 = z_1 \sigma(z_1) = [\,۰,۰۸ \quad -۰,۲۳\,]$$
$$h_۲ = z_۲ \sigma(z_۲) = [\,-۰,۱۷ \quad -۰,۰۹\,]$$
$$h_۳ = z_۳ \sigma(z_۳) = [\,-۰,۰۸ \quad -۰,۰۵\,]$$

second layer:

$$f_1 = U h_1^T = -۰,۰۰۲ \qquad y_1 = ۰,۴۹۹$$
$$f_۲ = U h_۲^T = -۰,۰۹۷ \xrightarrow{\text{sigmoid}} y_۲ = ۰,۴۷۴$$
$$f_۳ = U h_۳^T = -۰,۰۷۷ \qquad y_۳ = ۰,۴۸۱$$

$\Rightarrow \mathcal{L}_1 = 0.694$

$\mathcal{L}_2 = 0.646 \longrightarrow \boxed{\mathcal{L}_{\text{iter }2} = 0.665}$

$\mathcal{L}_3 = 0.655$

Backward pass:

بدلیل جاگیری فرمول‌ها را بازنویسی نمی‌کنیم زیرا در iteration اول برای همه نورون‌ها فرمول‌ها نوشته شده است

Gradients of w:

For $X^{(1)} \rightsquigarrow \begin{bmatrix} -0.11 & 0 & -0.05 \\ 0.01 & 0 & 0.005 \end{bmatrix}$

For $X^{(2)} \rightsquigarrow \begin{bmatrix} 0.005 & 0.02 & 0.05 \\ -0.002 & -0.01 & -0.02 \end{bmatrix} \xrightarrow{avg} \begin{bmatrix} -0.03 & 0.03 & 0.02 \\ 0.002 & -0.01 & -0.01 \end{bmatrix}$

For $X^{(3)} \rightsquigarrow \begin{bmatrix} 0 & 0.66 & 0.05 \\ 0 & -0.22 & -0.02 \end{bmatrix}$

Gradients of $b^{(1)}$ - First layer

for $X^{(1)} \rightsquigarrow [-0.10 \quad 0.10]$

for $X^{(2)} \rightsquigarrow [0.05 \quad -0.02] \xrightarrow{avg} [0.006 \quad -0.106]$

For $X^{(3)} \rightsquigarrow [0.07 \quad -0.02]$

Gradients of U

for $X^{(1)} \rightsquigarrow [-0.04 \quad 0.11]$

for $X^{(2)} \rightsquigarrow [-0.08 \quad 0.04] \xrightarrow{avg} [-0.05 \quad 0.01]$

For $X^{(3)} \rightsquigarrow [-0.04 \quad -0.02]$

Subject: Date: / /

Gradient of $b^{(2)}$ - second layer

for $x^{(1)}$ $\rightsquigarrow$ $\begin{bmatrix} -0.25 \end{bmatrix}$

for $x^{(2)}$ $\rightsquigarrow$ $\begin{bmatrix} 0.48 \end{bmatrix}$ $\xrightarrow{avg}$ $\begin{bmatrix} 0.15 \end{bmatrix}$

for $x^{(3)}$ $\rightsquigarrow$ $\begin{bmatrix} 0.48 \end{bmatrix}$

second iteration weight updates

$$W \Rightarrow \begin{cases} v_2 \leftarrow 2v_1 - \frac{\eta}{N}\left[\sum \frac{\partial E_i}{\partial W}\right] \\ W_{new} \leftarrow W_{old} + v_2 \end{cases}$$

$$\Rightarrow \begin{cases} v_2 \leftarrow 0.9\begin{bmatrix} 0.04 & -0.24 & -0.16 \\ -0.001 & 0.009 & 0.005 \end{bmatrix} - 0.1\begin{bmatrix} -0.04 & 0.06 & 0.02 \\ 0.002 & -0.01 & -0.01 \end{bmatrix} \\ W_{new} \leftarrow \begin{bmatrix} -0.02 & 0.03 & 0.02 \\ 0.002 & -0.01 & -0.01 \end{bmatrix} + \begin{bmatrix} -0.04 & 0.06 & 0.02 \\ -0.002 & 0.01 & 0.01 \end{bmatrix} \end{cases}$$

$$W_{new} = \begin{bmatrix} 0.04 & 0.01 & -0.03 \\ -0.02 & 0.02 & 0.01 \end{bmatrix}$$

$$b^{(1)} \Rightarrow \begin{cases} v_2 \leftarrow 2v_1 - \frac{\eta}{N}\left[\sum \frac{\partial E_i}{\partial b^{(1)}}\right] \\ b^{(1)}_{new} \leftarrow b^{(1)}_{old} + v_2 \end{cases}$$

$$\Rightarrow \begin{cases} v_2 \leftarrow 0.9\begin{bmatrix} -0.01 & 0.01 \end{bmatrix} - 0.1\begin{bmatrix} 0.006 & -0.04 \end{bmatrix} \\ b^{(1)}_{new} \leftarrow \begin{bmatrix} -0.01 & -0.03 \end{bmatrix} + \begin{bmatrix} -0.01 & 0.02 \end{bmatrix} \end{cases}$$

$$b_{new} = \begin{bmatrix} -0.02 & -0.08 \end{bmatrix}$$

sam

Subject: Date: / /

$$U \Rightarrow \begin{cases} v_1 \leftarrow \gamma v_1 - \frac{\eta}{n}\left[\sum \frac{\partial E_i}{\partial U}\right] \\ U_{new} \leftarrow U_{old} + v_1 \end{cases}$$

$$\begin{cases} v_1 \leftarrow 0.9\left[0.04 \quad -0.05\right] - 0.1\left[-0.06 \quad 0.01\right] \\ U_{new} \leftarrow \left[0.4 \quad -0.1\right] + \left[0.07 \quad -0.02\right] \end{cases}$$

$$\boxed{U_{new} = \begin{bmatrix} 0.41 & -0.12 \end{bmatrix}}$$

$$b \Rightarrow \begin{cases} v_1 \leftarrow \gamma v_1 \leftarrow \frac{\eta}{n}\left[\sum \frac{\partial E_i}{\partial b^{(1)}}\right] \\ b^{(1)}_{new} \leftarrow b^{(1)}_{old} + v_1 \end{cases}$$

$$\Rightarrow \begin{cases} v_1 \leftarrow 0.9\left[-0.05\right] + 0.1\left[0.05\right] \\ b^{(1)}_{new} \leftarrow \left[-0.05\right] + \left[-0.24\right] \end{cases}$$

$$\boxed{b^{(1)}_{new} = \begin{bmatrix} -0.31 \end{bmatrix}}$$

Final weights after 2 iterations

$$W = \begin{bmatrix} 0.24 & 0.13 & -0.04 \\ -0.36 & 0.23 & 0.12 \end{bmatrix} \qquad U = \begin{bmatrix} 0.41 & -0.12 \end{bmatrix}$$

$$b^{(1)} = \begin{bmatrix} -0.02 & -0.27 \end{bmatrix} \qquad b^{(2)} = \begin{bmatrix} -0.31 \end{bmatrix}$$

سوال دوم)

الف)

- می‌دانیم در GD با momentum پارامتر سرعت محاسبه می‌شود که جهت گرادیان‌های قبلی را در نظر می‌گیرد و درواقع exponentially decaying average گرادیان‌های قبلی را حساب می‌کند، این کار باعث می‌شود oscillation ها کمتر شود زیرا گرادیان‌های قبلی را در نظر می‌گیرد(هرچه گرادیان ها برای پیمایش‌های قدیمی‌تر باشند تاثیرشان کمتر است.) با توجه به اینکه گفته شده نمودار آبی رنگ GD است، می‌توان فهمید که نمودار سبز رنگ مربوط به momentum است زیرا بسیار شبیه به GD است ولی با stability کمتر. از طرفی در nestrov-momentum روی momentum کار شده تا آن‌را از momentum هم بهتر کند و این‌کار را با تغییر نحوه بروزرسانی پارامترهای مدل و محاسبه گرادیان پارامترها در نقطه‌ای کمی جلوتر از حالت فعلی (lookahead) با کمک جهت سرعت محاسبه شده مانند حالت قبل می‌کند. در نتیجه با این نگاه به جلو باز هم از oscillation ها کاسته می‌شود. این تغییر نحوه بروزرسانی پارامترها و خود سرعت را هوشمندتر می‌کند. درواقع تفاوت اصلی آن با momentum در محاسبه گرادیان در هر پیمایش است که در nestrov-momentum عنصر آینده‌نگری به آن اضافه شده است و گرادیان پارامترها با فرض جهت حرکت کنونی حساب می‌شود.

SGD with momentum

$$v \leftarrow \beta v - \alpha \nabla L(w)$$
$$w \leftarrow w + v$$

SGD with nestrov-momentum

$$w_{lookahead} = w + \beta v$$
$$v \leftarrow \beta v - \alpha \nabla L(w)$$

$$w \leftarrow w + v$$

بنابراین نمودار قرمز رنگ برای nestrov-momentum است زیرا دارای oscillation های بسیار کمتر است.

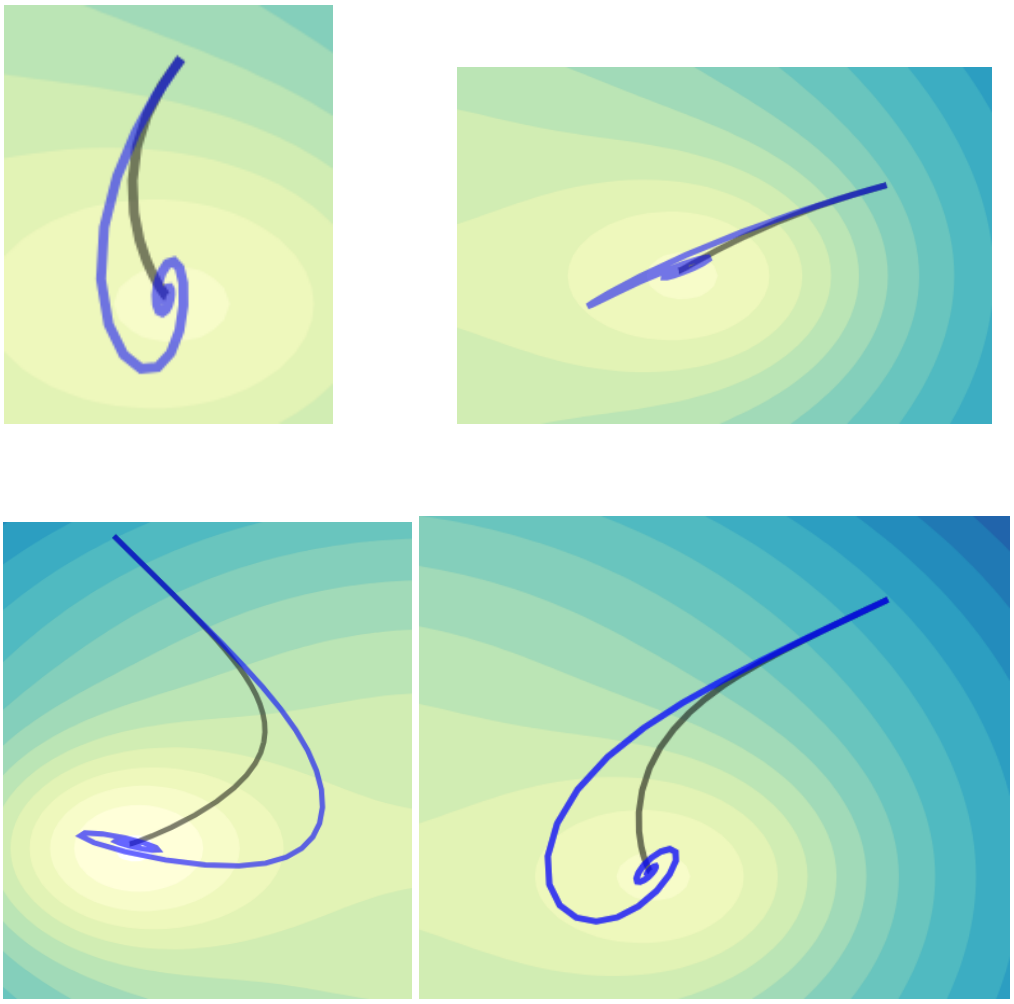
دو روش بهینه‌سازی بر اساس momentum دارای نرخ یادگیری ثابت هستند اما RMSProp در حوزه الگوریتم‌های adaptive learning است و نرخ یادگیری آن با توجه به پیشروی گرادیان‌ها کاهش پیدا می‌کند. در واقع نرخ یادگیری هر پارامتر به صورت جداگانه با توجه به exponential moving average گرادیان‌های آن پارامتر از اول تا آن پیمایش به صورت معکوس کاهش می‌یابد. در نتیجه با نزدیک شدن به نقطه optimal نرخ یادگیری نیز کاهش پیدا می‌کند و پرش‌ها بسیار کمتر می‌شوند. در نتیجه RMSProp در این مثال از بروزرسانی‌های اضافی و با گام‌های بزرگ پیشگیری می‌کند و باعث می‌شود به صورت مستقیم به سمت نقطه بهینه حرکت کند. در واقع از حرکات زیگزاگی و پرش‌های بلند که جلوگیری می‌شود مسیر بسیار هموار و direct تر می‌شود زیرا همواره در جهت و مسیر درست می‌ماند.

RMSProp

$$r_t \leftarrow \rho r_{t-1} + (1 - \rho) g^2$$
$$w \leftarrow w - \alpha/(\sqrt{r_t + \delta}) g$$

- **مزایا و معایب روش momentum:** در این روش سعی شده تا با تعریف پارامتر سرعت که با توجه به گرادیان‌های پیمایش‌های قبلی محاسبه می‌شود فرایند بروزرسانی در هر پیمایش هموارتر و پایدارتر شود تا از oscillation ها تا حدی جلوگیری شود و نقطه بهینه overshoot نشود. در نتیجه باعث می‌شود نسبت به GD معمولی نرخ همگرایی بالاتری داشته باشد. از معایب این روش می‌توان به نیازمندی دو ابرپارمتر نرخ یادگیری و ضریب momentum اشاره کرد. اما مورد مهم تر در این روش که باعث شده تا در شکل داده شده عملکرد آن از GD بدون momentum بدتر شود، وابستگی و حساسیت زیاد این روش نسبت به نقطه اولیه گرادیان‌ها و چند گام اولیه می‌باشد با توجه به اینکه در روش momentum پارامتر سرعت نیز در نظر گرفته می‌شود در نتیجه هر

بروزرسانی گرادیان‌ها **بایاسی** به سمت گام‌های قبلی دارد و همین مورد باعث می‌شود بعضی‌جاها با اینکه نباید جلو برود می‌رود و پارامترها را به اشتباه بروزرسانی می‌کند. در پایین این وابستگی به شرایط اولیه نشان داده شده است. (آبی مربوط به Imomentum است و مشکی SGD معمولی)



مزایا و معایب روش nestrov-momentum: این روش دارای مزایای روش momentum نیز می‌باشد اما پیش از بروزرسانی گرادیان‌ها یک further adjustment روی پارامتر سرعت انجام می‌شود به این صورت که در جهت سرعت گرادیان نقطه‌ای کمی جلوتر محاسبه می‌شود (به جای گرادیان نقطه کنونی) و درواقع **جلویش را نگاه**

می‌کند، این تکنیک دقیقا مشکل آخری که در momentum در بخش معایب آن مطرح شد را نشانه قرار می‌دهد و آن را حل می‌کند. حاصل این تغییر بروزرسانی گرادیان با توجه به نقطه‌ای کمی جلوتر بروزرسانی هوشمندانه‌تر پارامتر هاست، مخصوصا در محیط‌هایی که گرادیان ها با کمی تغییر بسیار oscillate می‌کنند. در نتیجه این روش نسبت به روش momentum دارای پایداری و در نتیجه نرخ یادگیری بالاتری می‌باشد. از معایب آن ثابت بودن نرخ یادگیری است که می‌تواند منجر به overshoot کردن نقطه بهینه شود زیرا به طور کلی ما با نزدیکتر شدن به نقطه بهینه به گام‌های کوچک‌تری در بروزرسانی پارامترها نیاز داریم همینطور همگرایی این روش بسیار حساس به نرخ یادگیری می‌باشد و نیاز به تنظیم دقیق دارد. از دیگر معایب این روش که شامل روش قبل هم می‌شود، می‌توان به گیر کردن در نقطه بهینه محلی اشاره کرد زیرا سعی می‌کند جهت بروزرسانی پارامترها را حفظ کند و این مهم می‌تواند باعث oscillation حول نقطه بهینه شود زیرا نرخ یادگیری نیز ثابت است.

مزایا و معایب روش RMSProp: هدف اصلی این روش پیاده سازی adaptive learning است و این کار را با کوچک کردن نرخ یادگیری هر پارامتر به صورت جداگانه با توجه به میانگین نمایی مربع گرادیان‌های گذشته آن انجام می‌دهد. این کار باعث می‌شود تا ما در نقاط نزدیک به بهینه آن‌را overshoot نکنیم زیرا گام‌های بروزرسانی ما به مرور زمان کوچک‌تر می‌شود. همینطور RMSProp اهمیت کمتری به گرادیان های محاسبه‌شده در گام‌های قدیمی می‌دهد که باعث بروزرسانی هوشمندنه‌تر می‌شود و پایداری و همواری بیش‌تر می‌شود در نتیجه پارامترهایی که دارای گرادیان‌های کوچک‌تری بوده‌اند بروزرسانی‌های قوی‌تری روی آن‌ها انجام می‌شود و برعکس. از معایب این روش می‌توان به حساس بودن به نرخ‌یادگیری اولیه و نبود momentum و پارامتر سرعت و در نتیجه در نظر گرفتن جهت تغییر گرادیان‌ها اشاره کرد. همچنان تنظیم غیر مناسب پارامتر وزن در میانگین نمایی نیز می‌تواند نرخ همگرایی را به شدت پایین آورد. بار محاسباتی آن نیز کمی بالاتر است زیرا با میانگین گرادیان‌ها را در هر مرحله برای همه پارامترها حساب کند.

- الگوریتم AdaGrad نیز مانند RMSProp از روش‌های adaptive learning می‌باشد ولی نحوه کوچک‌تر کردن پارامتری‌های متفاوت است. در این روش کوچک کردن با توجه به

به حاصلی جمع مربع‌های گرادیان‌های گذشته رخ می‌دهد و میانگینی گرفته نمی‌شود. همین مورد باعث می‌شود این الگوریتم برای فرایندهای یادگیری طولانی مناسب نباشد زیرا نرخ‌یادگیری بدون هیچ limitای می‌تواند کوچک شود که موجب جلوگیری از پیشروی پارامترها می‌شود یا سرعت آن را بسیار کند می‌کند. این مهم در RMSProp با میانگین گیری مورد توجه قرار گرفته است. با توجه به نکات گفته شده در بخش‌های قبل به نظر می‌رسد مسیر آن بسیار شبیه به RMSProp باشد با این تفاوت که نرخ همگرایی آن کند تر می‌شود و حتی ممکن است به نقطه بهینه نرسد زیرا ممکن است گام‌ها بسیار کوچک شوند، البته این موضوع به نرخ یادگیری اولیه ارتباط دارد همینطور با توجه به مشخص بودن تعداد استپ‌ها در مسیرهای رسم شده در GD به نظر می‌رسد فرایند یادگیری طولاتی نباشد و AdaGrad نیز به نقطه بهینه برسد.(قبل از بسیار کوچک‌تر شدن استپ‌ها)

ب)

$$s_t = \beta_2 s_{t-1} + (1-\beta_2) g_t^2$$

$$s_{t-1} = \beta_2 s_{t-2} + (1-\beta_2) g_{t-1}^2$$

$$* \quad \beta s_{t-1} = (\beta_2)^2 s_{t-2} + \beta_2 (1-\beta_2) g_{t-1}^2$$

$$** \quad (\beta_2)^2 s_{t-2} = (\beta_2)^3 s_{t-3} + (\beta_2)^2 (1-\beta_2) g_{t-2}^2$$

$$\vdots$$

$$* \quad s_t = (\beta_2)^2 s_{t-2} + (1-\beta_2) g_{t-1}^2 + (1-\beta_2) g_t^2$$

$$** \quad s_t = (\beta_2)^3 s_{t-3} + (\beta_2)^2 (1-\beta_2) g_{t-2}^2 + \beta_2 (1-\beta_2) g_{t-1}^2 + (1-\beta_2) g_t^2$$

$$\vdots \quad (t=n)$$

$$s_t = (\beta_2)^n s_{t-n} + (1-\beta_2) g_t^2 + \beta_2 (1-\beta_2) g_{t-1}^2 + \beta_2^2 (1-\beta_2) g_{t-2}^2$$

$$+ \cdots + \beta_2^{n-1} (1-\beta_2) g_{t-n+1}^2$$

$$s_0 = 0 \Rightarrow s_t = \sum_{i=0}^{t-1} \beta_2^i (1-\beta_2) g_{t-i}^2 = (1-\beta_2) \sum_{i=0}^{t-1} \beta_2^i g_{t-i}^2 \quad \blacksquare$$

$$s_t = (1-\beta_2) \sum_{i=0}^{t-1} \beta_2^i g_{t-i}^2$$

$$E[s_t] = (1-\beta_2) \sum_{i=0}^{t-1} \beta_2^i \, E[g_{t-i}^2]$$

$\mu = E[g_{t-i}^2]$ → average of g_{t-i}^2's

$$= (1-\beta_2) \mu \sum_{i=0}^{t-1} \beta_2^i \quad \blacksquare$$

$$\sum_{i=0}^{n-1} ar^i = \frac{a(r^n - 1)}{r-1} \Rightarrow E[s_t] = (1-\beta_2) \mu \frac{\beta_2^t - 1}{\beta_2 - 1} = \mu (1 - \beta_2^t)$$

- در الگوریتم Adam تصحیح بایاس روی هردوی گشتاورهای اول و دوم انجام می‌شود. دلیل اصلی این کار این است که هر دوی این گشتاورها با ۰ در پیمایش اولیه مقداردهی می‌شوند. بدون تصحیح بایاس در پیمایش‌های اول الگوریتم مقادیر محاسبه شده برای این گشتاورها حول ۰ بایاس هستند. در نظر داریم که ما در هر مرحله تخمینی از این گشتاورها را محاسبه می‌کنیم، پس برای محاسبه یک unbiased estimator از این گشتاورها از فرمول زیر استفاده می‌کنیم.

$$\widehat{s} = s/(1 - \beta_2)$$

با توجه به نتیجه حاصل‌شده در بخش قبل در شروع فرایند یادگیری و در پیمایش‌های اولیه s_t و v_t هر دو بسیار به سمت ۰ بایاس هستند زیرا $\beta^t \approx 1$.

اما با پیشروی یادگیری و بزرگ شدن t ، $\beta^t \to 0$ در نتیجه دیگر بایاسی نداریم.

$$E[s_t] = \mu(1 - \beta_2^{t-1})$$

در نتیجه بدون این تصحیح بایاس الگوریتم ما در پیمایش‌های اولیه به اشتباه گرادیان‌ها را بسیار کوچک و نزدیک به ۰ فرض می‌کند که منجر به گام‌های بزرگ در این مراحل می‌شود. این مهم باعث ناپایداری و غیرهموار بودن الگوریتم در مراحل اولیه می‌شود که می‌تواند باعث کاهش نرخ همگرایی و ضعف الگوریتم در پیدا کردن مسیر بهینه به سمت نقطه بهینه شود.

- می‌دانیم که Adam ایده‌های هردوی RMSProp و momentum را با هم ادغام کرده است. گشتاور دوم از RMSProp و گشتاور اول از momentum. از مزایای Adam نسبت به RMSProp به درنظر گرفتن جهت بروزرسانی گرادیان‌ها با بردار سرعت محاسبه شده اشاره می‌کنیم که منجر به بروزرسانی هوشمندانه تر می‌شود.(جزئیات در توضیحات مربوط به GD با مومنتوم ذکر شده.) همچنین در Adam بر خلاف RMSProp تصحیح بایاس وجود دارد که جزئیات آن‌را نیز گفتیم. اما از طرفی از مزایای RMSProp می‌توان به سریع‌تر بودن آن اشاره کرد زیرا گشتاور اول را در نظر نمی‌گیرد و در نتیجه محاسبات کمتری دارد. همچنین در محیط‌های convex بررسی شده است که RMSProp به خوبی Adam عمل می‌کند که با توجه به سریع‌تر بودن، استفاده کمتر از حافظه و ۱ ابرپارامتر کمتر از Adam می‌تواند انتخاب بهتری باشد.

سوال سوم)

الف)

اگر به جای y، $\widehat{x}$ بگذاریم درواقع مثل این می‌ماند که در تمام شرایط داده‌های ورودی هر لایه شبکه ما دارای میانگین ۰ و واریانس ۱ است زیرا در هر مرحله صرفا روی هر دسته z-score normalization انجام می‌دهیم و پارامترهای نرمالیزیشن γ و β آموزشی نمی‌بینند و shift و scale رخ نمی‌دهد در نتیجه داده در هر لایه محدود در بازه [1,1-] یا [0,1]می‌شود (بستگی به تابع فعال‌ساز دارد) که اتفاق خوشایندی نیست زیرا flexibility مورد نیاز را برای مسائل غیر خطی را از شبکه گرفته و ممکن است جلوی این را بگیرد که شبکه نحوه نمایش داده‌ها را در مراحل مختلف به صورت بهینه یاد بگیرد. همچنین در بسیاری از مسائل نیاز به این است که داده در range زیادی کشیده شده باشد تا فرایند یادگیری بتواند انجام شود زیرا تاثیر یک پارامتر را نمی‌توانیم به اندازه کافی کم و زیاد کنیم و فیدبک بگیریم. پس در نهایت با اعمال تغییر گفته شده قدرت یادگیری مدل کاهش می‌یابد. در نتیجه باید پارامترهای γ و β را یادبگیریم تا به صورت بهینه از BN بتوانیم استفاده کنیم.

ب)

- اگر در BN سایز هر دسته از حدی کوچکتر باشد برآورد و تخمین‌هایی که ما از میانگین و واریانس داده داریم به unrepresentative خواهد بود و درست نخواهد بود. به همین دلیل این تخمین‌ها در هر دسته بسیار با هم متفاوت خواهند بود و درواقع این نرمالسازی با یک سری آماره noisy انجام شده است. با وجود این نویز خروجی توابع فعال‌ساز می‌توانند بسیار غیر قابل پیش‌بینی باشند و شیفت زیادی در هر دسته نسبت به هم داشته باشند. این موجب می‌شود که تا گرادیان‌ها پایدار نباشند و فرایند یادگیری پارامترها مختل می‌شود و یا نرخ همگرایی آن پایین می‌آید. همچنین می‌تواند موجب overfit روی این نویزها هم بشود.
- اگر اندازه هر دسته برابر ۱ باشد و مقدار پارامتر اولیه شیفت ۰ باشد، طبق الگوریتم در هر مرحله y_i برابر صفر می‌شود زیرا میانگین x_i با x برابر خواهد بود و واریانس دسته نیز ۰ خواهد بود (البته تقسیم بر ۰ رخ نمی‌دهد زیرا e در نظر گرفته شده است) و همچنین چون $\beta = 0$ در نتیجه شیفتی هم نداریم. در نتیجه همه y_i ها صفر می‌شوند

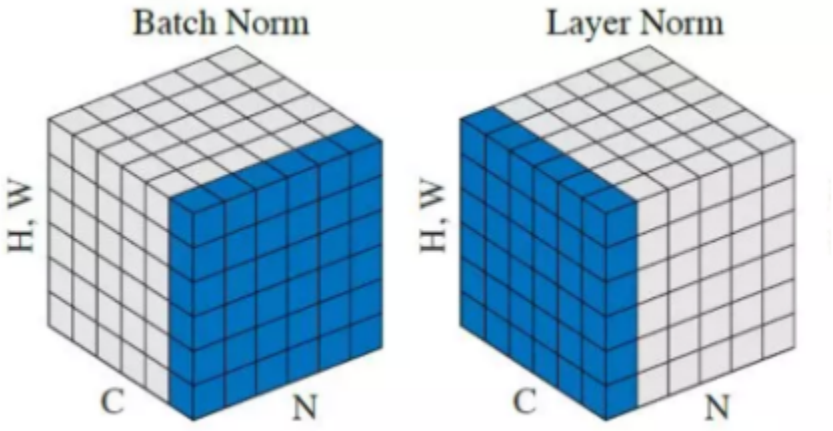
و مقادیر توابع فعال‌ساز به ازای تمام ورودی‌ها با هم برابر می‌شوند و عملا فرایند یادگیری‌ای نمیتواند رخ دهد زیرا تمام اطلاعات قابل یادگیری از داده از بین می‌رود.

ج)

با توجه به اینکه در این لایه ۲۰ نورون داریم و ورودی ما ۱۰ بعدی است و لایه ما تماما متصل است، هر نورون تمام ابعاد ورودی را دریافت می‌کند در نتیجه هر نورون به ۲۰+۱ پارامتر وزن نیاز دارد. با احتساب بایاس. از طرفی BN در سطح هر نورون (ویژگی) و قبل از اعمال تابع فعال‌ساز و مستقل از دیگر نورون‌ها انجام می‌شود تا به ازای ویژگی‌های مختلف فرایندهای نرمالیزشن بهینه را داشته باشیم پس هر نورون ۲ پارامتر BN را نیز دارد، یعنی هر نورون ۲۳ پارامتر قابل یادگیری دارد. ما در کل ۲۰ نورون داریم پس ۲۰*۲۳ یعنی ۲۶۰ پارامتر آموزشی داریم.

د)

بزرگ‌ترین تفاوت بین LN و BN این است که در اولی نرمال‌سازی در **سطح یک نمونه** ولی در بازه تمام نورون‌های یک لایه و در دومی در سطح چند نمونه (سایز دسته) و برای هر **ویژگی** (نورون) به صورت جداگانه انجام می‌شود. در نتیجه پارامترهای γ و β در LN در هر لایه مشترک هستند ولی در BN هر نورون به صورت جداگانه γ و β دارد. در LN میانگین و واریانس روی ویژگی‌های نمونه انجام می‌شود. در نتیجه دیگر وابستگی به سایز دسته در این روش وجود ندارد.



در شرایطی BN بهتر است که اندازه دسته‌ها بزرگ باشد زیرا اینگونه برآورد ما از میانگین و واریانس داده واقعی‌تر خواهد بود و همینطور نشان داده شده است[لینک مقاله] که در شبکه‌های عصبی feedforward, و به ویژه CNNها نرمال‌سازی در ابعاد دسته بهتر است زیرا نرمال‌سازی دسته‌ای ورودی‌های هر لایه را به گونه‌ای نرمال می‌کند که توزیع ثابتی داشته باشند، و این باعث کاهش (Internal Covariate Shift) می‌شود، جایی که ورودی‌های لایه‌ها در طول آموزش تغییر می‌کنند. این پایداری امکان استفاده از نرخ یادگیری بالاتر و همگرایی سریع‌تر را فراهم می‌کند.

از طرفی زمانی LN بهتر است که شبکه ما از نوع RNN باشد زیرا LN آماره‌های نرمال‌سازی را بر روی ویژگی‌های یک نمونه تکی محاسبه می‌کند، که آن را برای RNNها مناسب می‌سازد زیرا طول sequence ممکن است متغیر باشد و وابستگی‌ها بین گام‌های زمانی و نمونه‌ها اهمیت دارند.استفاده از نرمال‌سازی دسته‌ای (Batch Normalization یا BN) در RNNها به دلیل طول متغیر توالی‌ها و نیاز به آماره‌های یکنواخت در تمام گام‌های زمانی، چالش‌برانگیز است. در کل در زمانی که داده ما **temporal** است یعنی وابسته به زمان، بهتر است از LN استفاده کنیم. همچنین در سناریوهایی با اندازه‌های کوچک دسته اتکای BN به آماره‌های دسته‌ای باعث ایجاد واریانس بالا و ناپایداری می‌شود. LN که مستقل از اندازه‌ی دسته است، در این شرایط پایدار باقی می‌ماند.LN در حین آموزش و استنتاج همان محاسبات را انجام می‌دهد، زیرا وابسته به آماره‌های دسته‌ای نیست. این سازگاری می‌تواند در سناریوهای پیاده‌سازی که حفظ ساختار یکسان بین آموزش و inference مطلوب است، مزیت داشته باشد.

سوال چهارم)

الف)

درآپ اوت و **منظمساز** هر دو به منظور جلوگیری از Overfitting استفاده می‌شوند، اما از روش‌های متفاوتی برای دستیابی به این هدف بهره می‌برند.

- **تضعیف وابستگی به نورون‌های خاص:** دراپ‌اوت به صورت تصادفی تعدادی از نورون‌ها را در طول آموزش غیرفعال می‌کند. این باعث می‌شود که مدل به ترکیب خاصی از

وزن‌ها یا نورون‌ها وابسته نباشد. منظم‌ساز نیز وابستگی مدل به مقادیر بزرگ وزن‌ها را کاهش می‌دهند.

- **ایجاد یکپارچگی در یادگیری:** دراپوت عملاً ترکیبات مختلفی از نورون‌ها را در هر مرحله از آموزش ایجاد می‌کنداین کار به نوعی نقش عبارت تنظیمی را ایفا می‌کند زیرا پیچیدگی مدل را کاهش می‌دهد.
- **کاهش پیچیدگی مدل:** منظم‌سازها وزن‌ها را کنترل می‌کنند تا از رشد بیش از حد پارامترها جلوگیری کنند. دراپوت هم از طریق غیرفعال کردن نورون‌ها به نوعی تعداد پارامترهای فعال را کاهش می‌دهد، که اثری مشابه دارد.

این روش از co-adapting بیش از حد نورون‌ها جلوگیری می‌کند و شبکه را مجبور می‌سازد تا ویژگی‌های مقاوم‌تر و مستقل‌تری را یاد بگیرد.همچنین یک اثر مجموعه‌ای (Ensemble Effect) دارد، زیرا هر تکرار آموزشی را می‌توان به عنوان آموزش یک زیرشبکه متفاوت در نظر گرفت.

بنابراین، دراپ اوت به طور غیرمستقیم شبیه منظم‌ساز عمل می‌کند، زیرا هر دو به کاهش وابستگی مدل به ویژگی‌ها یا پارامترهای خاص کمک کرده و احتمال بیش‌برازش را کاهش می‌دهند.

ب)

بله، می‌توان گفت دراپ‌اوت عملکردی شبیه **یادگیری جمعی (Ensemble Learning)** دارد. در زیر توضیح این شباهت را ارائه می‌کنم:

- **ایجاد زیرشبکه‌های متفاوت:** در هر بار استفاده از دراپ‌اوت در طول آموزش، به‌صورت تصادفی بخشی از نورون‌ها غیرفعال می‌شوند. این کار باعث می‌شود که هر بار یک **زیرشبکه متفاوت** از شبکه اصلی آموزش ببیند. در نتیجه، فرآیند آموزش مشابه آموزش چندین مدل (یا مجموعه‌ای از مدل‌ها) است.
- **تاثیر ترکیب مدل‌ها:** در **یادگیری جمعی**، ترکیب خروجی‌های چندین مدل برای بهبود عملکرد استفاده می‌شود. در دراپ‌اوت نیز، هنگام پیش‌بینی، وزن‌های نورون‌های

غیرفعال شده به‌صورت میانگین‌گیری مقیاس‌بندی می‌شوند. این کار باعث می‌شود شبکه نهایی رفتاری مشابه ترکیب چندین مدل داشته باشد.

- **افزایش تعمیم‌پذیری:** همانطور که در بخش قبل توضیح داده شده این روش تعمیم‌پذیری مدل را افزایش می‌دهد که باز هم شبیه به خواص یادگیری جمعی است.

در نتیجه، می‌توان گفت که دراپ‌اوت به‌طور ضمنی مانند یادگیری جمعی عمل می‌کند، زیرا عملکرد کلی شبکه حاصل از ترکیب زیرشبکه‌های مختلفی است که در طی فرآیند آموزش ایجاد می‌شوند.

ج)

با توجه به اینکه گفته شده هر نود به صورت **مستقل** می‌تواند روشن یا خاموش باشد به صورت زیر عمل می‌کنیم.

یکبار فرض می‌کنیم از N گره هیچ کدام خاموش نشده،

یکبار فرض می‌کنیم از N گره 1 گره خاموش شده،

یکبار فرض می‌کنیم از N گره ۲ گره خاموش شده،

و ...

در نتیجه داریم:

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

د)

در زمان آزمایش، **دراپ‌اوت** معمولاً اعمال نمی‌شود و خروجی شبکه با احتمال دراپ‌اوت (p) **مقیاس‌بندی** می‌شود تا سازگاری با مقدار مورد انتظار در زمان آموزش حفظ شود.

دلیل این مقیاس‌بندی:در طول **آموزش**، نورون‌ها به‌صورت تصادفی غیرفعال می‌شوند و شبکه فقط بخشی از وزن‌ها را استفاده می‌کند.در زمان **تست**، تمام نورون‌ها فعال هستند. اگر

خروجی شبکه بدون مقیاس‌بندی استفاده شود، مقدار خروجی به‌طور قابل‌توجهی بزرگ‌تر از مقدار مورد انتظار در زمان آموزش خواهد شد.

برای جلوگیری از این مشکل، وزن‌های نورون‌ها در زمان آزمایش در مقدار $1 - p$ ضرب می‌شوند. این کار باعث می‌شود مقدار خروجی شبکه با مقدار میانگین محاسبه‌شده در زمان آموزش برابر باشد.