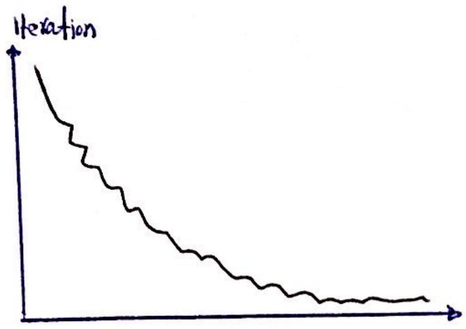
به نام خدا

محمد جواد امینی - 40121193

تمرین سری 2 - یادگیری عمیق

Problem 1:

a)



با افزایش تعداد داده هر بار ما تخمین بهتری نسبت به توزیع کلی داده داریم و در نتیجه، گرادیان ما دقیق‌تر بوده و سریع‌تر به نقطه optimal می‌رسیم (همگرایی سریع‌تر است)

b)

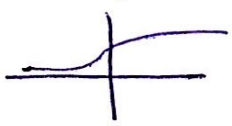
بله - در هنگام آموزش، در لایه Batch normali مقادیر نویز تزریق شده تا مدل ما عملکرد بهتری داشته باشد

$\underline{x} = \{x_1, \dots, x_m\}^T$ input of mini batch $\rightarrow$ normalize $\hat{x}_i = \dfrac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$

then $\rightarrow y_i = \gamma \hat{x}_i + \beta$; γ, β hyperprameters

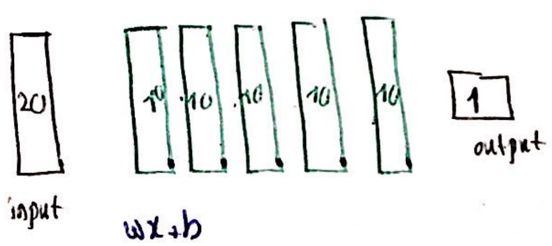
به دلیل این که تخمین میانگین (μ) و واریانس (σ^2) برای هر mini batch، متفاوت بوده تا حدودی و در هر batch یکسان نیست مانند این است که نویزی به داده‌ها اضافه کرده باشیم!

c) $z = \sigma(w^T \underline{x} + b)$



خیر - زیرا تابع سیگموید به اشباع افتد و قابلیت یادگیری خود را از دست می‌دهد و می‌تواند عملکرد خوبی برای ورودی های مختلف قائل شود (Vanishig Gradient)

d)



$10[20+1] + 10[10+1] \times 4 + 1[10+1] = 661$

e)

خیر - بله، softmax، 2 کلاس، دقیقاً مشابه Logistic Regression ساده عمل می‌کند

In Logistic Regression

class 0: $\hat{y} = \dfrac{1}{1 + e^{-(\omega_e x + b_e)}} \leq 0.5 \rightarrow e^{-(\omega_e x + b_e)} \geq 1 \rightarrow -(\omega_e x + b_e) \geq 0 \rightarrow \omega_e x + b_e \leq 0$

class 1: $\hat{y} = \dfrac{1}{1 + e^{-(\omega_e x + b_e)}} \geq 0.5 \rightarrow -(\omega_e x + b_e) \leq 0 \rightarrow \omega_e x + b_e \geq 0$

Softmax

$$\hat{y}_1 = \frac{e^{\omega_{s1}x + b_{s1}}}{e^{\omega_{s1}x + b_{s1}} + e^{\omega_{s2}x + b_{s2}}} \quad , \quad \hat{y}_2 = \frac{e^{\omega_{s2}x + b_{s2}}}{e^{\omega_{s1}x + b_{s1}} + e^{\omega_{s2}x + b_{s2}}}$$

class 0 if $\hat{y}_1 > \hat{y}_2 \rightarrow e^{\omega_{s1}x + b_{s1}} > e^{\omega_{s2}x + b_{s2}} \rightarrow e^{x(\omega_{s1} - \omega_{s2}) + b_{s1} - b_{s2}} > 1$

$$\rightarrow x(\omega_{s1} - \omega_{s2}) + b_{s1} - b_{s2} > 0 \rightarrow x(\omega_{s2} - \omega_{s1}) + b_{s2} - b_{s1} < 0$$

$$\rightsquigarrow \begin{cases} \omega_l = \omega_{s2} - \omega_{s1} \\ b_l = b_{s2} - b_{s1} \end{cases}$$

Problem 2: a)

- در روش کمینه ، ما چندین مدل را روی دیتاست‌های مختلف آموزش می‌دهیم و درنهایت ، طبق روش مانند رای اکثریت ، یا میانگین یا medan ، خروجی مدل‌های مختلف ، پیش‌بینی نهایی را مشخص می‌کنیم. برای این‌که کمینه بهتر عمل کند خطای هرکدام از مدل‌ها نسبت به هم ناهمبسته باشند و در این مقاله برای اطمینان به این منظور در pre pro های مختلفی انجام شده.

مزایا: واریانس کمتر شده ، دقت افزایش پیدا می‌کند و دقت جلوگیری از overfitting می‌شود به دلیل این‌که خطاهای مدل‌ها مستقل اند

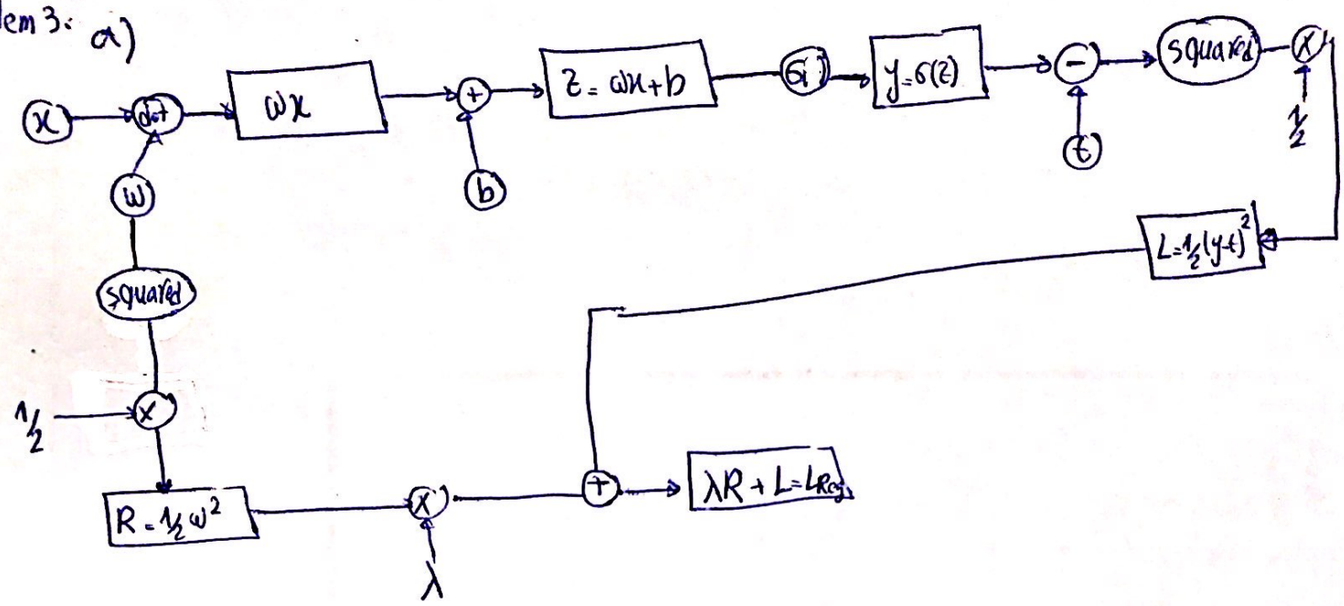
b)

1. Scaled data set to $[0,1]$
2. Elastic Deformations
3. Affine Transformations
4. normalized the data set with that the width and height of the boundig box equals 20 pixels. Due to a large variation bounding box is normalized to 8 to 20
5. Deslanted Training Set. Generation of a deslanted trainy set by horizontally shearing the digit. The shearing magnitude is determined by $\tan(\alpha) \times d$, α is angle of the first principal component of pixel intesities with respect to the vertical axis, and d is the vertical distance from the image center

These proprocessing steps may contribute to preventing model errors.

1. Robustness of Deformation
2. Affine Trans. for variability
3. Normalization for consistency
4. Deslanted Trainy set for improved Generalization

Problem 3: a)



- $\frac{\partial L_{reg}}{\partial \lambda} = R \quad ; \quad \frac{\partial L_{reg}}{\partial R} = \lambda \quad ; \quad \frac{\partial L_{reg}}{\partial L} = 1$ ،

- $\frac{\partial L_{reg}}{\partial t} = \frac{\partial L_{reg}}{\partial L} \cdot \frac{\partial L}{\partial t} = -1 \cdot (y-t) = t-y$

$z=\sigma(x) \rightsquigarrow z' = z(1-z)$

- $\frac{\partial L_{reg}}{\partial y} = \frac{\partial L_{reg}}{\partial L} \cdot \frac{\partial L}{\partial y} = y-t$

- $\frac{\partial L_{reg}}{\partial z} = \frac{\partial L_{reg}}{\partial L} \cdot \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z} = 1 \cdot (y-t) \cdot y(1-y)$

- $\frac{\partial L_{reg}}{\partial w} = \lambda \frac{\partial R}{\partial w} + \frac{\partial L_{reg}}{\partial z} \cdot \frac{\partial z}{\partial w} = w\lambda + [(y-t)\, y(1-y)]\, x$

- $\frac{\partial L_{reg}}{\partial b} = \frac{\partial L_{reg}}{\partial z} \cdot \frac{\partial z}{\partial b} = (y-t)\, y(1-y)$

- $\frac{\partial L_{reg}}{\partial x} = \frac{\partial L_{reg}}{\partial z} \cdot \frac{\partial z}{\partial x} = w[(y-t)\, y(1-y)]$

b) اگر همه وزن ها یکی باشند و یا تصادفی نباشند، ممکن است در یادگیری وزن ها دچار اختلال شوند به طوری که از خود رفتارهای متفاوتی (ابتدایی) و تنوع در یادگیری وجود ندارد. و ممکن است در نهایی که متشکل از نورون هستند به دلیل رفتار مشابه نورون ها، مدل مختل شود.

اگر مقدار اولیه وزن ها زیاد باشد ممکن است activation function ها در حالت اشباع رفته و دیده نشود Vanishing Gradient رخ دهد و یادگیری بسیار کند یا کلا متوقف

c) $\underline{X} = [1, 2]^T$, $\underline{w} = [0.1, 0.3]^T$, $b = 0.2$, $\lambda = 0.1$, $t = 1$, Learning Rate $= 0.1$

$z = 0.1 + 0.6 + 0.2 = 0.9 \rightarrow y = 0.7109 \rightarrow L = \frac{1}{2}(t-y)^2 = 0.0418$

$R = 0.05 \rightsquigarrow L_{req} = 0.0468$, $\frac{\partial L_{req}}{\partial w} = [-0.0494, -0.088]^T$, $\frac{\partial L_{req}}{\partial b} = -0.0594$

$\rightsquigarrow \underline{w} = \underline{w} - \frac{\partial L_{req}}{\partial w} \times 0.1 = [0.1049, 0.3089]^T$

$b = b - \frac{\partial L_{req}}{\partial b} \times 0.1 = 0.2059$

Problem 4:

a)

- $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1}) \rightarrow$ گرادیان تابع هدف (تابع Loss) (f) نسبت به وزن ها در گام (زمان) t برای آپدیت کردن وزن ها در زمان t

- $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t \rightsquigarrow$ (میانگین متحرک گرادیان ها) بروز رسانی momentum برای گام t

- $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \rightsquigarrow$ (میانگین متحرک مربعات گرادیان) بروز رسانی سرعت برای گام t

- $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$ تصحیح میانگین گرادیان

- $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$ تصحیح میانگین مربع گرادیان

} برای تصحیح بایاس

- $\theta_t \leftarrow \theta_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \rightsquigarrow$ آپدیت کردن وزن ها برای گام t

- ϵ را برای جلوگیری از تقسیم بر صفر می گذارند. و α نرخ یادگیری است.

- β_1 و β_2 را معمولا مقادیر بالایی گذارند تا الگوریتم m_t ها را زیاد به تغییرات لحظه حساس نباشند ،
الگوریتم آدام باعث می شود که ما سریع‌تر همگرا شویم زیرا سرعت یادگیری ما به طور هوشمند تعیین شد.

b)

$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$

$\beta_1 = 0.9$ $\quad m_0 = 0$

$m_1 = 0.1 g_1 + 0.9(0) = 0.1 g_1 \qquad \hat{m}_1 = g_1$

$m_2 = 0.9[0.1 g_1] + 0.1 g_2 = 0.09 g_1 + 0.1 g_2 \qquad \hat{m}_2 = \frac{0.09 g_1 + 0.1 g_2}{0.1 + 0.09}$

$m_3 = 0.9[0.1 g_2 + 0.09 g_1] + 0.1 g_3 = 0.09 g_2 + 0.081 g_1 + 0.1 g_3$

در گام‌های نخست m_1 و m_2 بطور مثال m_t ها به صفر بایاس زیادی دارند چون تا پیش m_0 در این گام‌ها زیاد بوده به همین دلیل m_t ها را باید نرمالایز کرد تا کاپیر نقطه شروع در این گام‌ها کمتر بشود. مثلاً $m_1 = 0.1 g_1$ ولی $\hat{m}_1 = g_1$ است که تاثیر g_1 در $\hat{m}_1$ بیشتر شده [تصحیح بایاس صورت گرفته است] و هرچه جلوتر برویم β_1^t ها بزرگ‌تر شده $1-\beta_1^t$ ها همان یک شده این کار مشکل بایاس در t های کوچک را حل می‌کند.

Problem 5: $\quad w^T H w = L(w)$

a) $\nabla_w (w^T H w) = Hw + H^T w \Big|_{H=H^T} = 2Hw$ و $\nabla_w(\cdot) = 0 \rightarrow Hw = 0$

update rule SGd $\rightarrow$ $w_t = w_{t-1} - \alpha \nabla_{w_{t-1}}(L) \rightsquigarrow w_{t-1} - \alpha 2(Hw_{t-1})$

$\rightarrow w_t = (I - 2\alpha H) w_{t-1} \rightsquigarrow w_t = (I - 2\alpha Q \lambda Q^T) w_{t-1}$

b) $w_1 = (I - 2\alpha \overbrace{Q\Lambda Q^T}^{QQ^T}) w_0 = Q(I - 2\alpha\lambda) Q^T w_0$

$w_2 = (I - 2\alpha Q\lambda Q^T)(I - 2\alpha Q\lambda Q^T) w_0 = (I - 2\alpha Q\lambda Q^T)^2 w_0$

$= Q(I - 2\alpha\lambda) Q^T \; Q(I - 2\alpha\lambda) Q^T w_0 = Q(I - 2\alpha\lambda)^2 w_0$

$\rightsquigarrow w_t = Q(I - 2\alpha\lambda)^t w_0$

c) $|1-2\alpha\lambda_i| < 1 \rightarrow -1 < 1-2\alpha\lambda_i < 1 \rightarrow 0 < 2-2\alpha\lambda_i < 2$

$\rightarrow 2 \geq 2\alpha\lambda_i \geq 0 \rightarrow \frac{2}{\lambda_i} \geq 2\alpha \geq 0 \rightarrow \frac{1}{\lambda_{max}} \geq \alpha$

که ما در بعضی جاها به جای α، 2α گرفته و این مقدار به صورت $\frac{2}{\lambda_{max}} \geq \alpha$ در می‌آید و همچنین الگوریتم آپدیت به صورت $\omega_t = \omega_{t-1} - \alpha H \omega_{t-1}$ می‌شود

d) $L(\omega) = \omega^T H \omega \quad \nabla_\omega (L(\omega)) = 2H\omega$ و $\nabla^2_\omega (L(\omega)) = 2H$

الگوریتم نیوتن $\quad \omega_t = \omega_{t-1} - [\nabla^2_{\omega_{t-1}}(L)]^{-1} \nabla_\omega (L(\omega))$

- این الگوریتم در یک مرحله به جواب می‌رسد

$\omega_t = \omega_{t-1} - \frac{1}{2}H^{-1} 2H \omega_{t-1} \rightarrow \omega_t = 0$

می‌توان این الگوریتم را با α نیز پیاده سازی کرد

$\omega_t = \omega_{t-1} - \alpha \frac{1}{2} H^{-1} 2H\omega_{t-1} \rightarrow \omega_t = (I-\alpha)\omega_{t-1}$

شرط همگرایی $\alpha \leq 1$

e) الگوریتم نیوتن به دلیل استفاده از معکوس ماتریس هسیان (H^{-1}) بسیار پرهزینه بوده و برای مسائل با ابعاد بزرگ کاربردی نبوده و همچنین بعضی مواقع ماتریس H، ill condition بوده و حساب کردن معکوس آن مشکل ساز باشد پس در مسائل عملی نمی‌توان از آن استفاده کرد.

Problem 6:

a)

$$J_1 = \frac{1}{2}\left(y_d - \sum_{k=1}^{n}(\omega_k + \delta_k)x_k\right)^2$$

$$E\left[\frac{\partial J_1}{\partial \omega_i}\right] = \frac{\partial E\{J_1\}}{\partial \omega_i}$$

$$E\{J_1\} = E\left\{\frac{1}{2}\Big(\underbrace{y_d - \sum_{k=1}^{n}(\omega_k x_k)}_{J_0} - \sum_{k=1}^{n}\delta_k x_k\Big)^2\right\}$$

$$\Rightarrow E\{J_1\} = \frac{J_0^2}{2} + \frac{1}{2}E\left\{\Big(\sum_{k=1}^{n}\delta_k x_k\Big)^2\right\} - J_0 E\Big\{\sum_{k=1}^{n}\delta_k x_k\Big\}$$

($E\{\delta_k\}=0$ — با فرض استقلال ω_k و δ_k)

$$\Rightarrow E\{J_1\} = \frac{J_0^2}{2} + \frac{1}{2}\sum_{k=1}^{n}E\{(\delta_k x_k)^2\} + \frac{1}{2}E\Big\{\sum_{i}\sum_{j\neq i}\delta_i\delta_j x_i x_j\Big\}$$

$$= \frac{J_0^2}{2} + \frac{1}{2}\sum_{k=1}^{n}\alpha\,\omega_k^2 x_k^2$$

$$\Rightarrow E\{J_1\} = \frac{1}{2}\underbrace{\Big(y_d - \sum_{k=1}^{n}\omega_k x_k\Big)^2}_{J_{1/2}} + \frac{1}{2}\alpha\sum_{k=1}^{n}\omega_k^2 x_k^2$$

$$\rightsquigarrow \frac{\partial E\{J_1\}}{\partial \omega_i} = -\frac{1}{2}x_i + \frac{1}{2}\alpha\, 2\omega_i x_i^2 = \alpha\,\omega_i x_i^2 - 0.5x_i$$

b) می‌توان به صورت رگولاسیون نوشت

$$E\{J_1\} = J_{1/2} + \frac{\alpha}{2}\sum_{k=1}^{n}\omega_k^2 x_k^2$$

$$= J_0' + \lambda\sum_{k=1}^{n}(\omega_k')^2 \quad \rightarrow \text{تنبیه رگولاسیون شد}$$

$\lambda = \alpha/2$, $\omega_k' = \omega_k x_k$