سوال ۳ الف

قسمت اول: بطور کلی رگولارایز کردن بایاس، مقدار آن را نزدیک به صفر می کند. حال رگرسیون لاجستیک را در نظر بگیرید. با صفر شدن بایاس، ابرصحفه مرز تصمیم گیری حتما از مبداء فضای ویژگی می گذرد که این واریانس مدل را کاهش داده و می تواند پیشبینی مدل را با خطای زیادی مواجه کند. حال یک شبکه عصبی دو لایه را در نظر بگیرید. در این حالت صفر کردن بایاس ها باعث می شود، برای نمونه هایی که مقدار بردار ویژگیشان نزدیک به صفر است، مدل خروجی نزدیک به صفر بدهد، زیرا خود وزن ها نیز رگولارایز می شوند. بطور کلی این اتفاق می تواند برای شبکه های عمیق تر نیز رخ دهد ولی با افزایش عمق شبکه تاثیر صفر شدن بایاس ها در کاهش واریانس مدل کمتر شده و عملا در شبکه های عمیق و کانولوشنی بایاس ها در واریانس مدل تاثیر ناچیزی دارند. هرچند رگولارایز کردن آن ها می تواند تاثیری منفی بگذارد اما بطور کلی در شبکه های عمیق رگولارایز کردن یا نکردن بایاس معمولا تاثیر چندانی در عملکرد شبکه ندارد. با توجه به این نکات و اینکه از تحلیل شبکه های سطحی می فهمیم که رگولارایز کردن بایاس می تواند تاثیر منفی داشته باشد (درصورتی که اگر آن را رگولارایز نکنیم، اگر با توجه به داده ها بهینه باشد، خودش به صفر می رسد)، لذا در شبکه های عصبی معمولا بایاس رگولارایز نمی شود.

مشتق رگولارایز L1 برای یک وزن $\mathrm{sign}(\mathrm{w})$ می شود. رابطه بروزرسانی وزن به صورت زیر نوشته می شود:

$$w \coloneqq w - \eta\,\mathrm{sign}(w) - \frac{\partial \mathcal{L}}{\partial w}$$

قسمت دوم: فارغ از تاثیر گرادیان هزینه، در این رابطه اگر وزن مثبت باشد در مرحله بعدی وزن به اندازه نرخ یادگیری به سمت صفر سوق داده می شود و اگر منفی باشد دوباره به همان اندازه به سمت صفر می رود. این موضوع که این اتفاق با یک مقدار مشخص برای هر به روز رسانی اتفاق می افتد، باعث می شود تعداد زیادی از وزن ها صفر شده و مدل بدست آمده اصطلاحا به جواب تنکی برسد. البته شهودهای هندسی نیز برای این موضوع قابل ارائه است.

سوال ۳ ج

قسمت اول: بچ نرمالیزیشن سرعت پردازش هر بچ را کاهش می دهد ولی چون می تواند باعث شود مدل زودتر همگرا شود، در مجموع فرایند آموزش را سرعت می بخشد.

قسمت دوم: اگر سایز بچ کوچک باشد، میانگین و واریانسی که در بچ نرمالیزیشن بدست می آید نویز بالایی خواهند داشت و نمایانگر توزیع واقعی نخواهند بود و در زمان آزمون هم می تواند پیشبینی شبکه را دچار مشکل کند.

سوال ۴ الف

خروجی شبکه ۰/۵ می شود و با یک بار به روز رسانی وزن ها می توانند مثبت یا منفی شوند و نمی توان بطور قطعی اظهار نظر کرد.

Q1. a

$$w_j := w_j - \alpha \frac{\partial \mathcal{L}}{\partial w_j}$$

$$b := b - \alpha \frac{\partial \mathcal{L}}{\partial b}$$

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{1}{2n} \sum_i \frac{\partial \mathcal{L}}{\partial w_j} (y^{(i)} - \hat{y}^{(i)})^2 = \frac{1}{2n} \sum_i 2x_j (\hat{y}^{(i)} - y^{(i)})$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{2n} \sum_i 2 (\hat{y}^{(i)} - y^{(i)})$$

Q1. b

Set $X := \begin{bmatrix} 1 & -x^{(1)T}- \\ & \vdots \\ 1 & -x^{(n)T}- \end{bmatrix}_{n \times (d+1)}$, $W := \begin{bmatrix} b \\ | \\ w \\ | \end{bmatrix}_{(d+1) \times 1}$, $Y := \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}_{n \times 1}$

then $\mathcal{L}(w,b) = \frac{1}{2n} \| Y - XW \|_2^2$

$$\Rightarrow \frac{\partial \mathcal{L}}{\partial W} = \frac{-2}{2n} X^T (Y - XW) = 0 \Rightarrow \boxed{W = (X^T X)^{-1} X^T Y}$$

Q1. c

Gradient Descent $= O(nm)$

closed form $= O(n^3 + n^2 d)$

Scanned with CamScanner

Q2.a

به علت اینکه در backprop طبق قانون زنجیره‌ای گرادیان‌ها در هم ضرب می‌شوند، اگر مقادیر گرادیان‌ها کوچک باشند (<1) به صورت نمایی گرادیان‌ها به صفر میل می‌کنند و باعث می‌شود وزن مربوطه آپدیت نشود. $\leftarrow$ vanishing grad

$$\sigma'(x) = \sigma(x)(1-\sigma(x)) \Rightarrow \sigma'(x) < 1$$

لذا ضرب گرادیان نورون‌های سیگموئیدی می‌تواند منجر به این مشکل شود.

$$Relu'(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

لذا برای Relu این مشکل پیش نمی‌آید چون یا ($1 \times 1 \times 1 \ldots$) یا ($1 \times 1 \times 0 \times \ldots$) یا ($0 \times 0 \times \ldots$) می‌شود و به صورت نمایی گرادیان را کوچک نمی‌کند.

Q2.c

مقدار بزرگ وزن‌ها باعث می‌شود مقدار ورودی به تابع فعال‌ساز سیگموئید نورون‌ها بزرگ باشد که در نتیجه گرادیان بازگشتی کوچک شده و در نهایت باعث مشکل vanishing grad می‌شود.

Scanned with CamScanner

Q3.b

$$\mathcal{L}'(w) = \frac{1}{2n}\sum_i \left(y^{(i)} - (x^{(i)} + \delta^{(i)})^T w\right)^2$$

where $\delta^{(i)} \sim \mathcal{N}(0, \sigma^2 I)$.

$$= \frac{1}{2n}\sum_i \left(y^{(i)} - x^{(i)T}w - \delta^{(i)T}w\right)^2$$

$$= \frac{1}{2n}\sum_i \left(y^{(i)} - x^{(i)T}w\right)^2 - 2\left(y^{(i)} - x^{(i)T}w\right)\left(\delta^{(i)T}w\right) + \left(\delta^{(i)T}w\right)^2$$

$$= \underbrace{\frac{1}{2n}\sum_i \left(y^{(i)} - x^{(i)T}w\right)^2}_{\mathcal{L}(w)} + \frac{1}{2n}\left[\sum_i -2\left(y^{(i)} - x^{(i)T}w\right)\left(\delta^{(i)T}w\right) + \left(\delta^{(i)T}w\right)^2\right]$$

$$\Rightarrow \mathbb{E}_{\delta\sim\mathcal{N}}\left[\mathcal{L}'(w)\right] = \mathbb{E}_{\delta\sim\mathcal{N}}\left[\mathcal{L}(w)\right] + \frac{1}{2n}\sum_i \mathbb{E}_{\delta\sim\mathcal{N}}[\cdots]$$

we have that
$$\begin{cases} \mathbb{E}_{\delta\sim\mathcal{N}}\left[-2\left(y^{(i)} - x^{(i)T}w\right)\left(\delta^{(i)T}w\right)\right] = -2\left(y^{(i)} - x^{(i)T}w\right)\underbrace{\mathbb{E}_{\delta\sim\mathcal{N}}\left[\delta^{(i)T}w\right]}_{=0} \\ \mathbb{E}_{\delta\sim\mathcal{N}}\left[\left(\delta^{(i)T}w\right)^2\right] = \sigma^2\|w\|_2^2 \end{cases}$$

$$\Rightarrow \boxed{\mathbb{E}_{\delta\sim\mathcal{N}}\left[\mathcal{L}'(w)\right] = \mathcal{L}(w) + \sigma^2\|w\|_2^2}$$

Scanned with CamScanner

Q3.d

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta_t)$$

$$\Rightarrow m_t = (1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i}\nabla_\theta J(\theta_i)$$

$$\Rightarrow \mathbb{E}[m_t] = \mathbb{E}\left[(1-\beta_1)\sum_{i=1}^{t}\beta_1^{t-i}\nabla_\theta J(\theta_i)\right]$$

$$= (1-\beta_1)\sum_{i=1}^{t}\beta_1^{t-i}\,\mathbb{E}[\nabla_\theta J(\theta_i)]$$

$$\underset{\text{iid}}{\overset{\text{since}}{=}} (1-\beta_1)\,\mathbb{E}[\nabla_\theta J(\theta_t)]\sum_{i=1}^{t}\beta_1^{t-i}$$

$$= \mathbb{E}[\nabla_\theta J(\theta_t)](1-\beta_1)\frac{\beta_1^{t-1}}{\beta_1-1}$$

$$= \mathbb{E}[\nabla_\theta J(\theta_t)](1-\beta_1^t)$$

لذا به ازای مقادیر β_1 نزدیک به ۱ مقدار $1-\beta_1^t$ برای t های
اول به صفر میل می‌کند (نزدیک‌تر است) و در نتیجه مقدار امید m_t هم به صفر
نزدیک‌تر می‌شود لذا به جای m_t از $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ استفاده
می‌شود که:

$\mathbb{E}[\hat{m}_t] = \mathbb{E}[\nabla_\theta J(\theta_t)]$ که همان چیزی است که می‌خواهیم.

Scanned with CamScanner

Q4.b

این شبکه word2vec است که بعداً با آن آشنا خواهید شد.

دقت شود که $W \in \mathbb{R}^{d\times H}$ و $W' \in \mathbb{R}^{H\times d}$ و y یک بردار وان‌هات است که فرض می‌کنیم عنصر $\underline{m}$ آن یک است.

داریم:

$$\frac{\partial \mathcal{L}}{\partial u_j} = \hat{y}_j - \mathbb{I}\{j=m\} = \varnothing$$

$$\frac{\partial \mathcal{L}}{\partial w'_{ij}} = \sum_{k=1}^{d} \frac{\partial \mathcal{L}}{\partial u_k}\frac{\partial u_k}{\partial w'_{ij}} = \frac{\partial \mathcal{L}}{\partial u_j}\frac{\partial u_j}{\partial w'_{ij}}$$

چون w'_{ij} فقط در u_j حضور دارد.

$$\frac{\partial u_j}{\partial w'_{ij}} = \sum_{k=1}^{d} W_{ik} x_k = \varnothing \longrightarrow \frac{\partial \mathcal{L}}{\partial w'_{ij}} = \varnothing$$

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \sum_{k=1}^{d} \frac{\partial \mathcal{L}}{\partial u_k}\frac{\partial u_k}{\partial w_{ij}} = \sum_{k=1}^{d}\left(\hat{y}_k - \mathbb{I}\{k=m\}\right)\frac{\partial u_k}{\partial w_{ij}}$$

$$\frac{\partial u_k}{\partial w_{ij}} = w'_{jk} x_i = \varnothing \longrightarrow \frac{\partial \mathcal{L}}{\partial w_{ij}} = \varnothing$$

منظور از $\varnothing$ این است که قابل محاسبه است.

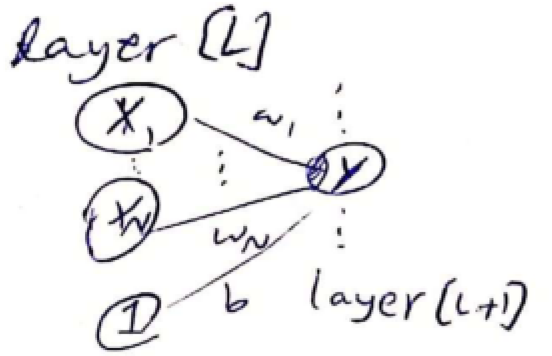
Scanned with CamScanner

Xavier Initialization

if the weights are too small, then the variance of the input signal starts diminishing as it passes through layers of the network. this finally becomes so small that won't be useful. (Causing Vanishing Gradients).

Now if the weights are too large, then the variance of the input data tends to rapidly increase with each passing layer; finally becomming too large to be useful anymore. (Causing exploding gradients).

So the main idea is that we want the variance of input data not to change when it passes through the network layers.



$$y = g(z)$$

$$z = w_1 x_1 + \cdots + w_N x_N + b$$

we want the $var(z) = var(x_i)$ for all $i \in \{1, \cdots, N\}$.

$$Var(z) = Var(w_1 x_1) + \cdots + Var(w_N x_N) + \cancel{Var(b)}^{0}$$

assume $w_i \sim Gaussian(0, \sigma)$ for all $i \in \{1, \cdots, N\}$

also assume that $E[x_i] = 0$

then

$$\cancel{var(z)} = N * var(w_k) \cancel{var(x_k)} \quad \text{for any } k.$$

(Layer [L])

↑

(number of input layer units)

$$\Rightarrow \quad var(w_k) = \frac{1}{N} \quad \text{for all } k \in \{1, \cdots, N\}$$

Xavier Initialization: Initialize weights from Gaussian of mean=0 & var= $\frac{1}{N}$

So in Xavier Initialization, we will initialize like so:

$$W^{[L]} \sim Gaussian\left(0, \frac{1}{N^{[L]}}\right)$$

units at layer [L-1]