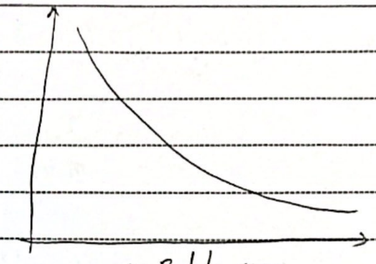
تمرین سری دوم _ امیرعباس اصغری ۴۰۰۱۰۰۶۶۲

Subject : Date :

سؤال ۱)



همانطور که با افزایش سایز mini_batch، distribution دیتای درون mini_batch با احتمال خوبی به distribution کل دیتا میل می‌کند و در نتیجه وزن‌ها، از گرادیان بدست آمده از loss در هر mini_batch، به طور مناسبی آپدیت می‌شوند و performance کل مدل افزایش می‌یابد و احتمال convergence شبکه افزایش می‌یابد. به بیانی دیگر، با افزایش سایز mini_batch، واریانس خطای تخمین توزیع کل داده‌ها توسط داده‌های mini_batch کاهش می‌یابد.

و به طورکلی با افزایش سایز mini_batch حافظه‌ی مورد نیاز برای train شبکه افزایش می‌یابد و بین حافظه‌ی مورد نیاز و تعداد step‌های لازم برای همگرایی یک tradeoff وجود دارد.

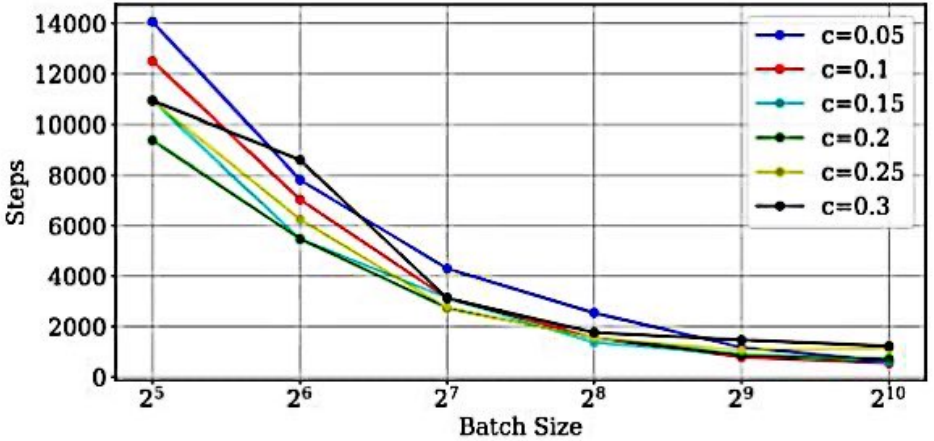


Figure 1: Number of steps for Algorithm 1 versus batch size needed to train ResNet-34 on CIFAR-10

Subject : Date :

ج) با توجه به الگوریتم Batch Normalization داریم:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2 \leftarrow \text{m.b. variance}$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \leftarrow \text{normalize}$$

$$y_i = \gamma \hat{x}_i + \beta = BN_{\gamma,\beta}(x_i) \leftarrow \text{scale and shift}$$

که γ و β، پارامترهای learnable هستند. size of mini batch ← m

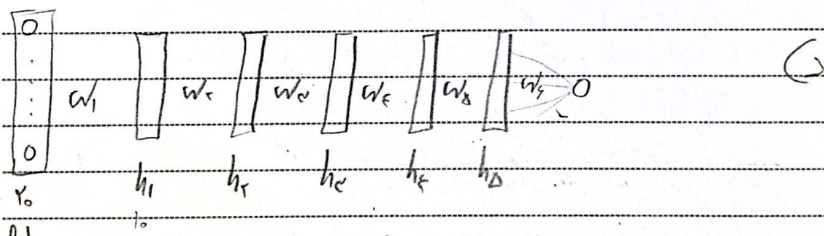
توجه بکنید در رابطه‌ی بالا، x_i خروجی نورون i ام لایه‌ی قبل BN layer است و y_i خروجی i ام BN layer. البته پارامترهای learnable γ و β، برای همه‌ی x_i ها یکسان (share) هستند.

در اینجا، با انتخاب یک mini batch به صورت تصادفی، چون که μ_B و σ_B چون که تا حدی از دیتاهای mini batch دارند هم تصادفی خواهند بود و با این کار نویز (Randomness) به شبکه اضافه شده است که همین Randomness می‌تواند در Regularization شبکه مفید باشد.

Subject : Date :

ج) اگر مقادیر اولیه‌ی وزن‌ها خیلی بزرگ باشند، فرایند training به مشکل numerical instability منجر می‌شود و همچنین با توجه به فرم تابع سیگموید، این تابع به ازای ورودی‌های extreme، در اشباع (saturation) خواهد رفت و باعث Gradient vanishing خواهد شد و در اینصورت احتمالا شبکه عملکرد خوبی نخواهد داشت، زیرا با صفر شدن گرادیان، وزن‌ها آپدیت نخواهند شد و performance شبکه افزایش نخواهد یافت. برای جلوگیری از این مشکل می‌توانیم از متودهای مختلف weight initialization مانند Xavier و HE استفاده کرد.

د)



number of learnable parameters: تعداد پارامترهای بایاس

$n_{w_1} + n_{w_2} + \cdots + n_{w_6} + n_{b_1} + \cdots + n_{b_6}$ ← تعداد پارامترهای ماتریس وزن

= ۲۰×۱۰ + ۴×(۱۰×۱۰) + ۱۰×۱ + ۵×۱۰ + ۱

= ۶۶۱

Subject : Date :

چون از آنجا که خروجی لایه ی softmax توزیع احتمالاتی دارد، بنابراین

$$\hat{y} = \text{Softmax}(W_s x + b_s) = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix}$$

$$\rightarrow \hat{y}_1 + \hat{y}_2 = 1 \rightarrow \hat{y}_1 > \hat{y}_2 \equiv \hat{y}_1 > 1 - \hat{y}_1 \rightarrow \hat{y}_1 > 0.5 \quad (1)$$

معادل است با

در رابطه اول:

$$\hat{y} = \sigma(W_x x + b_x) = \frac{1}{1 + e^{-(W_x x + b_x)}}$$

$$\rightarrow \begin{cases} \text{if } \hat{y} \leq 0.5 \rightarrow \text{label } 0 \\ \text{if } \hat{y} > 0.5 \rightarrow \text{label } 1 \end{cases}$$

در رابطه دوم:

$$\hat{y}_1 = \frac{e^{(W_{s_1} x + b_{s_1})}}{e^{(W_{s_1} x + b_{s_1})} + e^{(W_{s_2} x + b_{s_2})}}$$

$$= \frac{1}{1 + e^{((W_{s_2} - W_{s_1}) x + (b_{s_2} - b_{s_1}))}}$$

حال با توجه به (1) اگر $\hat{y}_1 > 0.5$ باشد، label predicted برابر 1 و در غیر اینصورت برابر 0 خواهد بود. یعنی دو الگوریتم یکسان عمل خواهند کرد.

$$W = W_{s_1} - W_{s_2}$$

$$b = b_{s_1} - b_{s_2}$$

Subject: Date:

توجه شود که در روش اول:

$$\underset{1\times n}{W} \; \underset{n\times 1}{x}$$

و در روش دوم چون که تعداد پارامترها دو برابر است داریم:

$$\underset{2\times n}{W_s} \; \underset{n\times 1}{x}$$

در روابط قبل
و ماتریس W_s را به صورت روبرو باز کردیم:

$$W_s = \begin{bmatrix} W_{s1} \\ W_{s2} \end{bmatrix}$$

سؤال ۳) همانطور که در مقاله گفته شده، یک کمیته، یک مجموعه از شبکه های

است. و بجای این که یک از (subset) دیتا را بر روی شبکه های مختلف train کنیم، training data

ی هر مدل را به نحوی preprocess می کنیم که مدل های مختلف، error های

خاصی داشته باشند. decorrelated و هدف از استقلال این است که هر مدل بخشی از اطلاعات

را capture کند و در کل سه مدل مدل های کمکی مناسب باشد. برای همین یک کمیته به صورت ترکیب خطی از

مدل های مختلف تعریف می شود:

$$y^{k}_{com}(x) = \sum_{n=1}^{N} w_{nk}\, y_{nk}(x)$$

↳ combination weight depends on class k.

ب) پیش پردازش مقاله: در ابتدا دیتاست به بازه‌ی [0,1] نرمالایز می‌شود.

سپس تصاویر با bounding box های به عرض و ارتفاع ۲۰ پیکسل نرمالایز می‌شوند.

همچنین با این حال از bounding box های به طول و عرض های متفاوت از ۸ تا ۲۰ پیکسل با step size = 2

استفاده در نرمالایز می‌شود و باعث تنوعی در نسبت ابعاد دیتاست می‌شود و منجر به ۷ مجموعه‌ی آموزشی

مختلف می‌شود. همچنین یک training set ایجاد deslanted تولید کردند به نحوی که باعث کاهش

انحنای ارقام می‌شوند (با انجام horizontal shearing). این کار با یک transformation

با رابطه‌ی tan(α) d انجام می‌شود که α زاویه component اول تصویر و d

vertical distance تا مرکز تصویر باشد. که تصویر زیر نشان دهنده مثالی از این transformation می‌باشد.

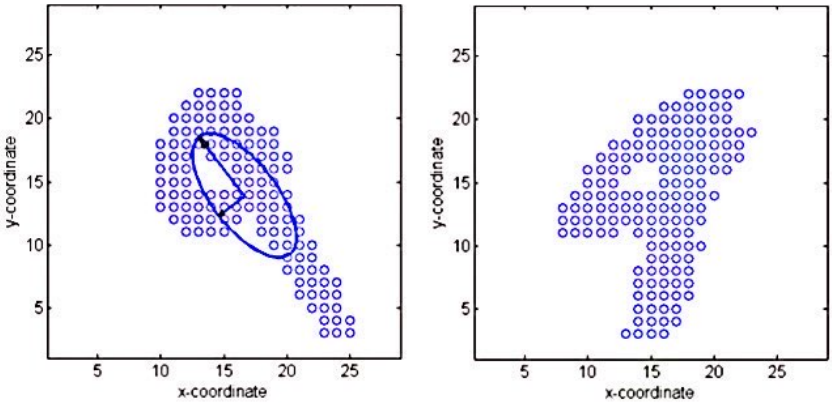


Figure 1. (Left panel) x-, y- coordinates (circles) of the original image together with the eigenvectors scaled by the corresponding eigenvalues. (Right panel) x-, y-coordinates (circles) of the deslanted image after horizontal shearing.

Subject :

Date :

همچنین با اعمال انحراف الاستیک ، یعنی با اعمال انحراف های مختلف خطی و غیرخطی بر روی Training data ، مدل ها را نسبت به تغییرات جزئی مقاوم می کنیم.

- و همچنین در برخی از موارد از زیرمجموعه های مختلفی از دیتاست اصلی برای Train کردن مدل های مختلف استفاده می کنیم.

- به طور کلی با انجام Predictions های گفته شده ، سعی می کنیم که مدل های مختلف (و خطاهای آن ها) را Decorrelated کنیم تا بتوانند وزن های مختلف را یاد بگیرند و خطاهای غیر همبسته در دیتای تست ایجاد کنند. و ترکیب (کردن) این مدل ها (Ensemble) مختلف باعث پوشش خطاهای یکدیگر می شوند و عملکرد کلی بهبود می یابد.

سوال ۳

Sigmoid activation function — Hidden layer

$z = wx + b$

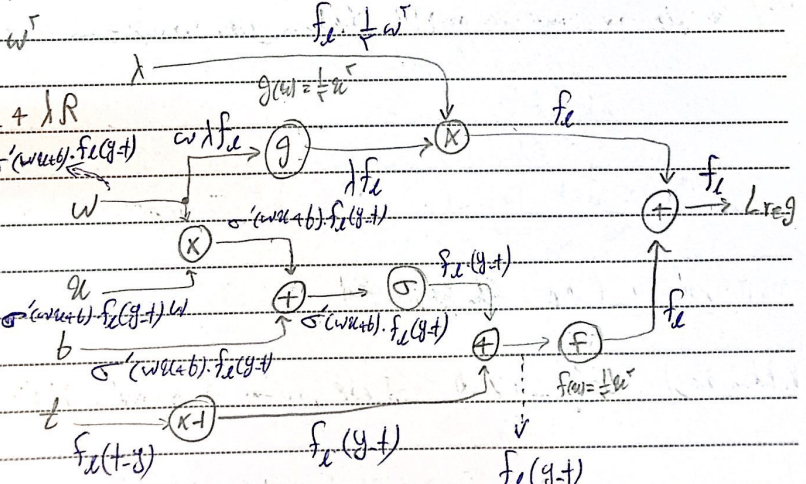
$y = \sigma(z)$

$L = \frac{1}{2}(y-t)^2$

$R = \frac{1}{2}w^2$

$L_{reg} = L + \lambda R$

$L(x;w,b) = \frac{1}{2}(g(x;w,b)-t)^2 + \frac{\lambda}{2}w^2$



با توجه به گراف صفحه‌ی قبل :

$$\frac{\partial L_{reg}}{\partial \lambda} = \frac{1}{2} w^2 \qquad (f_x = 1)$$

$$\frac{\partial L_{reg}}{\partial w} = w\lambda + \sigma'(wx+b)(y-t)$$

$$\frac{\partial L_{reg}}{\partial x} = \sigma'(wx+b)(y-t)\, w$$

$$\frac{\partial L_{reg}}{\partial b} = \sigma'(wx+b)(y-t)$$

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

$$\sigma'(x) = \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \sigma(x)\left(1-\sigma(x)\right)$$

ب) در صورت بزرگ بودن وزن ها، احتمالا حاصل linear combination ورودی ها و وزن ها ($w^T x$) بزرگ می شود و اگر از توابعی فعالی چون sigmoid و tanh استفاده شده، در ناحیه اشباع آن خواهیم رفت و گرادیان خیلی کوچک خواهد شد و پارامترها optimize نمی شوند.

همچنین در صورت وجود ترم های regularization مانند L2 Reg. ، از آنجا که $\frac{\lambda}{2}\|w\|^2$ به تابع loss اضافه می شود، ترم $2\lambda w$ به گرادیان وزن ها اضافه می شوند و از آنجا که w ها بزرگ است، converge کردن سخت تر می شود.

همچنین با رندوم initialize نکردن وزن ها ، مشکلات زیر ایجاد می شود :

symmetry Problem : اگر پارامتر ها یکسان initialize شوند ، نورون های مختلف یک لایه ،

output های یکسان خواهد داشت که یعنی یک feature یکسان توسط نورون های مختلف capture

می شود که باعث کاهش ظرفیت شبکه خواهد شد.

- همچنین باعث ناپایدار شدن شبکه و value یا pattern خاصی می شود.

بطور کلی اضافه کردن Randomness به شبکه باعث می شود که شبکه نواحی مختلف از

parameter space را explore کند و پاسخ به پاسخ optimal نزدیک تر می شود ،

و convergance بهتر خواهد بود و واضحاً از overfitting شبکه جلوگیری می شود.

C) مقادیر اولیه :

$$
\begin{cases}
\lambda = 0.1 \\
\text{learning rate} \leftarrow \alpha = 0.1 \\
\omega_0 = 1 \\
b_0 = 1 \\
\text{iteration} \leftarrow t
\end{cases}
$$

from G.D. , we know: $\omega_{i+1} = \omega_i - \alpha \frac{\partial L_{reg}}{\partial \omega}$

$t=0$:

$$\frac{\partial L_{reg}}{\partial \omega} = \lambda \omega + x(y-t)\,\sigma'(\omega x + b) = 0.01 + x(\sigma(x+1) - 0)\,\sigma'(x+1)$$

$$= 0.01 + x\,\sigma(x+1)\,\sigma(x+1)[1 - \sigma(x+1)]$$

Subject : Date :

$$\frac{\partial L_{reg}}{\partial b} = (y-t)\,\sigma'(w_2 a_1 + b) = \sigma(z_{L+1})\,\sigma(z_{L+1})\left[1-\sigma(z_{L+1})\right]$$

$$\rightarrow \begin{cases} w_1 = w_0 - \alpha \frac{\partial L_{reg}}{\partial w} = 1 - 0.1 \times \frac{\partial L_{reg}}{\partial w} = 1 - 0.1\left[0.1 + 2\sigma^2(z_{L+1})\left(1-\sigma(z_{L+1})\right)\right] \\ b_1 = b_0 - \alpha \frac{\partial L_{reg}}{\partial b} = 1 - 0.1 \times \frac{\partial L_{reg}}{\partial b} = 1 - 0.1\left[\sigma^2(z_{L+1})\left(1-\sigma(z_{L+1})\right)\right] \end{cases}$$

Subject : Date :

$$\frac{\partial L_{reg}}{\partial b} = (y-\hat{y})\,\sigma'(wx+b) = \sigma(x_{n+1})\,\sigma(x_{n+1})\left[1-\sigma(x_{n+1})\right]$$

$$\rightarrow \begin{cases} w_1 = w_0 - \alpha \dfrac{\partial L_{reg}}{\partial w} = 1 - 0.1 \times \dfrac{\partial L_{reg}}{\partial w} = 1 - 0.1\left[0.1 + x\,\sigma^2(x_{n+1})\left(1-\sigma(x_{n+1})\right)\right] \\ b_1 = b_0 - \alpha \dfrac{\partial L_{reg}}{\partial b} = 1 - 0.1 \times \dfrac{\partial L_{reg}}{\partial b} = 1 - 0.1\left[\sigma^2(x_{n+1})\left(1-\sigma(x_{n+1})\right)\right] \end{cases}$$

Subject : Date :

سوال ۴) گرادیان تابع هدف نسبت به وزن ها محاسبه می‌شود → $g_t = \nabla_\theta f_t(\theta_{t-1})$

در این مرحله اول momentum اول آپدیت می‌شود به صورت میانگین وزن دار از گرادیان فعلی و مقدار قبلی momentum

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

$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$

$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ (۳)

$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$ (۴)

در این مرحله momentum دوم با میانگین وزن دار از توان دوم گرادیان و مقدار قبلی v آپدیت می‌شود و این به تضمین واریانس گرادیان کمک می‌کند

$\theta_t = \theta_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$ (۵)

به خاطر این که بردار m و v روی صفر initialize شدند، بایاس به صفر دارند و برای تصحیح این بایاس (bias correction)، خط (۳) و (۴) الگوریتم انجام می‌شود.

(۵): در نهایت optimization انجام می‌شود که در اینجا α، learning rate است و $\hat{m}_t$ و $\hat{v}_t$ به ترتیب momentum اول و دوم اصلاح شده هستند و ϵ نیز تضمین می‌کند که در صورتی که $\hat{v}_t$ کوچک بود، از تغییرات زیاد پارامتر (θ_t) جلوگیری شود.

Subject :

Date :

ب) m_t بدلیل صفر initialize شدن و همچنین تاثیر exponential moving average

به سمت صفر بایاس دارد. مخصوصاً در timestep های اولیه.

برای مومنتوم دوم داریم:

$v_0 = 0$

$v_t = (1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} g_i^2$

$$\rightarrow \mathbb{E}[v_t] = \mathbb{E}\left[(1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} \cdot g_i^2\right]$$

$$= \mathbb{E}[g_t^2] \cdot (1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} + \zeta$$

$$= \mathbb{E}[g_t^2] \cdot (1-\beta_2^t) + \zeta$$

که ترم $(1-\beta_2^t)$ باعث running average ، صفر خواهد شد.

زیرا $\beta_2 > 0$ است و در t های کوچک مقدار $(1-\beta_2^t)$ به صفر نزدیک است و

با افزایش t به 1 میل می‌کند. و برای حل این مشکل (خصوصاً در t های کوچک)،

bias correction انجام می‌شود یعنی $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$

مشاهده می‌شود که همین مشکل دقیقاً برای momentum اول نیز وجود دارد:

$$\mathbb{E}[m_t] = \mathbb{E}\left[(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} \cdot g_i\right] = \mathbb{E}[g_t] \cdot (1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} + \zeta$$

$$= \mathbb{E}[g_t](1-\beta_1^t) + \zeta$$

Subject : Date :

سوال ۵

$$\frac{\partial w^T H w}{\partial w} = (H + H^T) w$$

$$\rightarrow H = Q \lambda Q^T \rightarrow H^T = Q \lambda^T Q^T = Q \lambda Q^T = H$$

$$\rightarrow \frac{\partial w^T H w}{\partial w} = 2Hw$$

Gradient descent: $w_t = w_{t-1} - \epsilon 2 H w_{t-1} = (I - 2\epsilon H) w_{t-1}$

$$= (I - 2\epsilon Q \lambda Q^T) w_{t-1} = Q(I - 2\epsilon \lambda) Q^T w_{t-1}$$

$$w_t = Q(I - 2\epsilon \lambda) Q^T w_{t-1}$$

$$\rightarrow w_1 = Q(I - 2\epsilon \lambda) Q^T w_0$$

$$\vdots$$

$$w_t = [Q(I - 2\epsilon \lambda) Q^T]^t w_0 \qquad Q \text{ is unitary matrix.}$$

$$= [Q(I - 2\epsilon \lambda) Q^T][Q(I - 2\epsilon \lambda) Q^T] \cdots [Q(I - 2\epsilon \lambda) Q^T] w_0$$

$$= Q (I - 2\epsilon \lambda)^t Q^T w_0$$

$$\hookrightarrow \begin{bmatrix} (1 - 2\epsilon \lambda_1)^t & & 0 \\ & \ddots & \\ 0 & & (1 - 2\epsilon \lambda_n)^t \end{bmatrix}$$

MICRO

Subject:　　　　　　　　　　　　　　　　Date:

ج) برای همگرایی باید مقادیر ویژه‌ی ماتریس قطری $(I-2\epsilon\lambda)^{+}$ کوچکتر مساوی یک باشند

یعنی: $\forall i \in \{1,\dots,n\}: \quad |1-2\lambda_i| \leq 1 \rightarrow -1 \leq 1-2\epsilon\lambda_i \leq 1$

$\forall i \rightarrow -2 \leq -2\epsilon\lambda_i \leq 0 \rightarrow \begin{cases} \lambda_i \geq 0 & \rightarrow \text{ماتریس } H \text{، PSD باشد.} \\ \epsilon \leq \frac{1}{\lambda_{max}} \end{cases}$

د) در روش نیوتن داریم:

$$w_t \leftarrow w_{t-1} - \left(\nabla^2 f(w_{t-1})\right)^{-1} \nabla f(w_{t-1})$$

($\nabla^2 f$: ماتریس ژاکوبین (J))

$$J = \left[\frac{\partial\left(\nabla_w (w^T H w)\right)}{\partial w}\right]^T = \left[\frac{\partial}{\partial w}(2Hw)\right]^T = 2H^T = 2H$$

$$w_{t+1} \leftarrow w_t - J^{-1} \nabla_{w=w_t}(w^T H w) = w_t - \frac{1}{2} H^{-1}(2Hw_t) = w_t - w_t = 0$$

بنابراین در اولین step، مسئله همگرا می‌شود زیرا تابع هدف مرتبه دو $(w^T H w)$ برحسب w است و روش نیوتن نیز حل با تقریب درجه دوم برای تابع هدف ارائه می‌دهد.

ه) زیرا محاسبه ماتریس computation cost بالایی دارد $O(n^3)$ است و چون به در شبکه‌های عصبی تعداد پارامترها (n) خیلی زیاد است، عملا استفاده از این روش نشدنی است.

همچنین برای ذخیره سازی ماتریس ژاکوبین، پر هزینه است $(O(n^2))$.

MICRO

Subject: Date:

$$J_1 = \frac{1}{2}\left(y_j - \sum_{k=1}^{n}(w_k+\delta_k)x_{ik}\right)^2 \qquad \delta_k \sim N(0, \alpha\sigma_k^2)$$

$$E[J_1] = \frac{1}{2}E[(y_j - z)^2] = \frac{1}{2}E[y_j^2 + z^2 - 2zy_j]$$

$$\frac{1}{2}\left(y_j^2 + E[z^2] - 2y_j E[z]\right)$$

$$E[z] = E\left[\sum_{k=1}^{n}(w_k+\delta_k)x_{ik}\right] = \sum_{k=1}^{n} w_k x_{ik}$$

$$E[z^2] = E\left[\sum_{a=1}^{n}(w_a+\delta_a)x_{ia}\sum_{b=1}^{n}(w_b+\delta_b)x_{ib}\right]$$

$$= E\left[\sum_{a=1}^{n}(w_a+\delta_a)^2 x_{ia}^2 + 2\sum_{a=1}^{n}\sum_{\substack{b=1\\ b\neq a}}^{n}(w_a+\delta_a)(w_b+\delta_b)x_{ia}x_{ib}\right]$$

$$= \sum_{a=1}^{n} E\left[(w_a+\delta_a)^2\right]x_{ia}^2 + 2\sum_{\substack{a,b\\ a\neq b}}^{n} w_a w_b x_{ia}x_{ib}$$

$$= \sum_{a=1}^{n}(1+\alpha)w_a^2 x_{ia}^2 + 2\sum_{\substack{a,b\\ a\neq b}}^{n} w_a w_b x_{ia}x_{ib}$$

$$\Rightarrow E[J_1] = \frac{1}{2}\left[y_j^2 + \sum_{a=1}^{n}(1+\alpha)w_a^2 x_{ia}^2 + 2\sum_{\substack{a,b\\ a\neq b}}^{n} w_a w_b x_{ia}x_{ib} - 2y_j\sum_{k} w_k x_{ik}\right]$$

$$= \frac{1}{2}\left(y_j - \sum_{i=1}^{n} w_i x_{ij}\right)^2 + \frac{1}{2}\alpha\sum_{i=1}^{n} w_i^2 x_{ij}^2$$

$$\Rightarrow E\left[\frac{\partial J_1}{\partial w_k}\right] = \frac{\partial E[J_1]}{\partial w_k} = -\left(y_j - \sum_{i=1}^{n} w_i x_{ij}\right)x_{ik} + \alpha w_k x_{ik}^2$$

Subject :   Date :

ب) از رابطه‌ی بدست آمده برای متوسط مقدار E داریم: $E[J_1] = \frac{1}{2}(y_1 - \sum w_i x_i)^2 + \frac{1}{2}\alpha \sum_{i=1}^{n} w_i^2 x_i^2$

می‌توان این برداشت را کرد که این نوع dropout مانند یک L2_regularizer

عمل می‌کند و با اضافه کردن ترم $\frac{1}{2}\alpha \sum_{i=1}^{n} w_i^2 x_i^2$ به Loss func.، باعث کوچک‌تر شدن اندازه ضرایب

در فرایند optimization می‌شود و در نتیجه باعث می‌شود که generative error

کاهش یابد. البته تفاوت آن با L2_reg. در این است که به هر w_i وزن خاصی نسبت

می‌دهد.