$\boxed{1.a}$

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix} \overset{\circledast}{=} \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} = A$$

$$y_i = \sum_{k=1}^{n} a_{ik} x_k \longrightarrow \frac{\partial y_i}{\partial x_j} = a_{ij} \quad \circledast$$

$\boxed{1.b}$

مسئله را در حالت کلی حل می‌کنیم که z بردار باشد و تمام اعضای x به تمام اعضای z وابسته باشد
با طول ℓ

$$y_i = \sum_{k=1}^{n} a_{ik} x_k \longrightarrow \frac{\partial y_i}{\partial z_j} = \sum_{k=1}^{n} a_{ik} \frac{\partial x_k}{\partial z_j} \quad \circledast$$

$$\frac{\partial y}{\partial z} = \begin{bmatrix} \frac{\partial y_1}{\partial z_1} & \cdots & \frac{\partial y_1}{\partial z_\ell} \\ \vdots & & \vdots \\ \frac{\partial y_m}{\partial z_1} & \cdots & \frac{\partial y_m}{\partial z_\ell} \end{bmatrix}_{m\times \ell} \overset{\circledast}{=} \begin{bmatrix} \sum_{k=1}^{n} a_{1k} \frac{\partial x_k}{\partial z_1} & \cdots & \sum_{k=1}^{n} a_{1k} \frac{\partial x_k}{\partial z_\ell} \\ \vdots & & \vdots \\ \sum_{k=1}^{n} a_{mk} \frac{\partial x_k}{\partial z_1} & \cdots & \sum_{k=1}^{n} a_{mk} \frac{\partial x_k}{\partial z_\ell} \end{bmatrix}_{m\times \ell} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}_{m\times n} \times \begin{bmatrix} \frac{\partial x_1}{\partial z_1} & \cdots & \frac{\partial x_1}{\partial z_\ell} \\ \vdots & & \vdots \\ \frac{\partial x_n}{\partial z_1} & \cdots & \frac{\partial x_n}{\partial z_\ell} \end{bmatrix}_{n\times \ell}$$

$$= A \times \frac{\partial x}{\partial z}$$

$\boxed{1.c}$

اگر فرض کنیم $y = Ax$ ، داریم :

$$y^T A x = \|y\|^2 = \left(\sum_{k=1}^{n} a_{1k} x_k\right)^2 + \cdots + \left(\sum_{k=1}^{n} a_{mk} x_k\right)^2$$

$$\longrightarrow \frac{\partial y^T A x}{\partial x} = \begin{bmatrix} \sum_{j=1}^{m} 2 a_{j1} \sum_{k=1}^{n} a_{jk} x_k \\ \vdots \\ \sum_{j=1}^{m} 2 a_{jn} \sum_{k=1}^{n} a_{jk} x_k \end{bmatrix} = 2 y^T A$$

در کوئرا گفته شده که x، y را مستقل از هم در نظر بگیریم . در این صورت می‌توان $y^T A$ را ماتریسی مستقل از x دانست ، بنابراین آنرا S می‌نامیم و طبق بخش $\boxed{1.a}$ می‌توان نوشت :

$$\frac{\partial \left[\overbrace{S}^{y^T A}\right]_{1\times n} [x]_{n\times 1}}{\partial [x]_{n\times 1}} \overset{\text{بخش a}}{=} S = y^T A$$

$\circledast$ البته معمولاً مشتق اسکالر نسبت به بردار را به صورت column vector می‌نویسند اما در این سوال به صورت row vector در نظر گرفته شده

$\boxed{1.c}$ ادامه

$\alpha = y^T A x$ را می‌توان به صورت dot product دو بردار y و Ax درنظر گرفت و می‌دانیم که ترتیب در dot product بی اثر است

$$\alpha = y^T A x = (Ax)^T y = x^T A^T y$$

مشابه قبل $x^T A^T$ را یک ماتریس مستقل از y درنظر می‌گیریم و طبق بخش a داریم:

$$\frac{\partial \alpha}{\partial y} = x^T A^T$$

⊛ البته می‌دانیم که در dot product دو بردار مستقل، مشتق نسبت به یکی برابر دیگری است
اما نخواستیم از آن استفاده نکنیم

$\boxed{1.d}$

z را مشابه بخش b فرض می‌کنیم و داریم:

$$\alpha = \sum_{i=1}^{n} x_i y_i \longrightarrow \frac{\partial \alpha}{\partial z_j} = \sum_{i=1}^{n} \frac{\partial x_i}{\partial z_j} y_i + \sum_{i=1}^{n} \frac{\partial y_i}{\partial z_j} x_i \quad ⊛$$

$$\frac{\partial \alpha}{\partial z} = \begin{bmatrix} \frac{\partial \alpha}{\partial z_1} \\ \vdots \\ \frac{\partial \alpha}{\partial z_l} \end{bmatrix}^T \overset{⊛}{=} \begin{bmatrix} \sum_{i=1}^{n} \frac{\partial x_i}{\partial z_1} y_i \\ \vdots \\ \sum_{i=1}^{n} \frac{\partial x_i}{\partial z_l} y_i \end{bmatrix}^T + \begin{bmatrix} \sum_{i=1}^{n} \frac{\partial y_i}{\partial z_1} x_i \\ \vdots \\ \sum_{i=1}^{n} \frac{\partial y_i}{\partial z_l} x_i \end{bmatrix}^T = \begin{bmatrix} y_1 \cdots y_n \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial z_1} & \cdots & \frac{\partial x_1}{\partial z_l} \\ \vdots & & \vdots \\ \frac{\partial x_n}{\partial z_1} & \cdots & \frac{\partial x_n}{\partial z_l} \end{bmatrix} + \begin{bmatrix} x_1 \cdots x_n \end{bmatrix} \begin{bmatrix} \frac{\partial y_1}{\partial z_1} & \cdots & \frac{\partial y_1}{\partial z_l} \\ \vdots & & \vdots \\ \frac{\partial y_n}{\partial z_1} & \cdots & \frac{\partial y_n}{\partial z_l} \end{bmatrix}$$

$$= y^T \frac{\partial x}{\partial z} + x^T \frac{\partial y}{\partial z}$$

در این سوال مشتق اسکالر نسبت به بردار به صورت row vector فرض شده است

$\boxed{1.e}$

$$AA^{-1} = I_m \xrightarrow{\frac{\partial}{\partial \alpha}} \frac{\partial A}{\partial \alpha} A^{-1} + A \frac{\partial A^{-1}}{\partial \alpha} = 0 \longrightarrow A \frac{\partial A^{-1}}{\partial \alpha} = -\frac{\partial A}{\partial \alpha} A^{-1}$$

$$\xrightarrow[\text{از چپ}]{\times A^{-1}} \frac{\partial A^{-1}}{\partial \alpha} = -A^{-1} \frac{\partial A}{\partial \alpha} A^{-1}$$

2

گرادیان : بردار مشتق های جزئی تابع با خروجی اسکالر، نسبت به هر کدام از ورودی هایش را می دهد

ماتریس ژاکوبین : ماتریسی از مشتق های جزئی که المان ij آن مشتق جزئی خروجی i ام نسبت به ورودی j ام است (برای تابع با ورودی و خروجی برداری)

$$\nabla y = \begin{bmatrix} \frac{\partial y}{\partial u} \\ \frac{\partial y}{\partial v} \\ \frac{\partial y}{\partial z} \end{bmatrix} \qquad \text{Jacobian}(\nabla y) = \begin{bmatrix} \frac{\partial^2 y}{\partial u^2} & \frac{\partial^2 y}{\partial u \partial v} & \frac{\partial^2 y}{\partial u \partial z} \\ \frac{\partial^2 y}{\partial v \partial u} & \frac{\partial^2 y}{\partial v^2} & \frac{\partial^2 y}{\partial v \partial z} \\ \frac{\partial^2 y}{\partial z \partial u} & \frac{\partial^2 y}{\partial z \partial v} & \frac{\partial^2 y}{\partial z^2} \end{bmatrix}$$

مشاهده می شود که دقیقاً به تعریف Hessian رسیدیم

3

در تسک های متفاوت همواره می توان بر اساس domain آن تسک فرض هایی انجام داد تا فضای حالت را محدود کرده و inductive bias مناسبی را بر اساس شرایط آن مسئله، در راه حل مان تزریق کنیم. به عنوان مثال اگر بدانیم که در صورت مسئله ما همواره در داده ها نوعی تقارن وجود دارد، می توانیم این فرض را در راه حل مان استفاده کنیم. استفاده از تقارن در تسک های متنوعی مانند metric learning (یافتن representation مناسبی که similarity در آن معنا داشته باشد)، خوشه بندی، classification و ... می تواند کاربرد داشته باشد.

این تقارن در حالت کلی می تواند یک group structure را تعیین کند و یا در حالت صورت سوال ما، تقارن در یک تصویر باشد

هدف آن است که تأثیر فرض تقارن را در loss شبکه ایجاد کنیم. برای این کار می توان regularization term را به نحوی تغییر داد که در نظر نگرفتن تقارن ورودی را penalize کند.

طبق شرایط صورت سوال فرض می کنیم بخواهیم یک تسک binary classification انجام دهیم و ورودی مان یک تصویر 1x2 باشد. در این حالت باید بردار وزن ها نیز طول 2 داشته باشد.

مطلوب ما آن است که در بردار وزن $W = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ مقادیر w_1 و w_2 به هم نزدیک باشند زیرا ورودی دارای تقارن است و بنابراین تأثیر 2 المان آن در خروجی شبکه نیز شبیه به هم باشد.

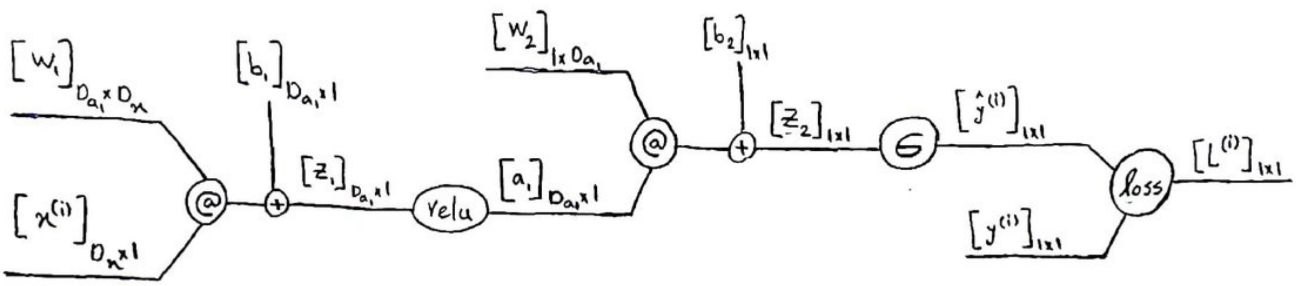
ادعا می کنیم اگر S را $\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ در نظر بگیریم، w_1 و w_2 نزدیک به هم می مانند :

$$R(w) = W^T S W = \underbrace{\begin{bmatrix} w_1 , w_2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}}_{\begin{bmatrix} w_1 - w_2 , & w_2 - w_1 \end{bmatrix}} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = w_1 (w_1 - w_2) + w_2 (w_2 - w_1)$$

$$= (w_1 - w_2)(w_1 - w_2) = \boxed{(w_1 - w_2)^2}$$

بنابراین هرچه فاصله w_1 و w_2 بیشتر شود لاس بیشتری خواهیم داشت $\longleftarrow$ مدل در جهتی وزن ها را آپدیت می کند که تقارن ایجاد شود

$\boxed{4.a}$



$$J = \frac{-1}{m}\sum_{i=1}^{m} L^{(i)}$$

روی تصویر تمام ابعاد مشخص شده است

$\boxed{4.b}$

$$\frac{\partial J}{\partial \hat{y}^{(i)}} = \delta_1 = \frac{-1}{m}\left(\frac{y^{(i)}}{\hat{y}^{(i)}} + \frac{(y^{(i)}-1)}{1-\hat{y}^{(i)}}\right) \Longrightarrow [\delta_1]_{1\times 1}$$

$\boxed{4.c}$

$$\frac{\partial \hat{y}^{(i)}}{\partial z_2} = \delta_2 = \underbrace{\sigma}_{\text{sigmoid}}(z_2)\times\left(1-\sigma(z_2)\right) \Longrightarrow [\delta_2]_{1\times 1}$$

$\boxed{4.d}$

$$\frac{\partial z_2}{\partial a_1} = \delta_3 = W_2 \Longrightarrow [\delta_3]_{1\times D_{a_1}}$$

$\boxed{4.e}$

$$\frac{\partial a_1}{\partial z_1} = \delta_4 = \begin{cases} 0 & z_1 < 0 \\ 1 & z_1 \geq 0 \end{cases}$$

1 ⟵ هر المان z_1 که صفر یا مثبت باشد

0 ⟵ هر المان z_1 که منفی باشد

$$= [\delta_4]_{D_{a_1}\times 1}$$

$\boxed{4.f}$

$$\frac{\partial z_1}{\partial W_1} = \delta_5 = x^{(i)T} \Longrightarrow [\delta_5]_{1\times D_x}$$

$\boxed{4.g}$

$$\frac{\partial J}{\partial W_1} = \delta_1 * \delta_2 * [\delta_3^T]_{D_{a_1}\times 1} \underbrace{*}_{\text{elemenwise}} [\delta_4]_{D_{a_1}\times 1} \underbrace{@}_{\text{matmul}} [\delta_5]_{1\times D_x}$$

$\boxed{5.1}$ Beale

$$f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$$

$$\frac{\partial f}{\partial x} = 2(1.5 - x + xy)(-1+y) + 2(2.25 - x + xy^2)(-1+y^2) + 2(2.625 - x + xy^3)(-1+y^3)$$

$$\frac{\partial f}{\partial y} = 2(1.5 - x + xy)(x) + 2(2.25 - x + xy^2)(2xy) + 2(2.625 - x + xy^3)(3xy^2)$$

$$\frac{\partial^2 f}{\partial x^2} = 2(-1+y)(-1+y) + 2(-1+y^2)(-1+y^2) + 2(-1+y^3)(-1+y^3)$$

$$\frac{\partial^2 f}{\partial y^2} = 2x(x) + 2\left[(2xy)(2xy) + (2x)(2.25 - x + xy^2)\right] + 2\left[(3xy^2)(3xy^2) + (6xy)(2.625 - x + xy^3)\right]$$

$$\frac{\partial^2 f}{\partial x \partial y} = 2\left[(x)(-1+y) + (1.5 - x + xy)\right] + 2\left[(2xy)(-1+y^2) + (2y)(2.25 - x + xy^2)\right] + 2\left[(3xy^2)(-1+y^3) + (3y^2)(2.625 - x + xy^3)\right]$$

$$=$$

$$\frac{\partial^2 f}{\partial y \partial x} = 2\left[(-1+y)(x) + (1.5 - x + xy)\right] + 2\left[(-1+y^2)(2xy) + (2y)(2.25 - x + xy^2)\right] + 2\left[(-1+y^3)(3xy^2) + (3y^2)(2.625 - x + xy^3)\right]$$

$$\text{gradient} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \qquad \text{hessian} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial xy} \\ \frac{\partial^2 f}{\partial yx} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

$\boxed{5.1}$ ادامه

می دانیم که تابع $f(x,y)$ که دو بار مشتق پذیر است ، محدب است اگر و تنها اگر ماتریس hessian آن psd باشد (در تمام نقاط)

نشان می دهیم که به ازای نقطه $(0,0)$ ماتریس hessian برابر است با

$$\text{hessian} = \begin{bmatrix} 2+2+2 & 3+0+0 \\ 3+0+0 & 0+0+0 \end{bmatrix} = \begin{bmatrix} 6 & 3 \\ 3 & 0 \end{bmatrix} \xrightarrow{\text{eigen values}} \begin{cases} \lambda_1 = 3+3\sqrt{2} & \text{مثبت} \\ \lambda_2 = 3-3\sqrt{2} & \text{منفی} \end{cases}$$

تمام eigen val ها بزرگتر مساوی صفر نشد

پس ماتریس hessian به ازای نقطه $(0,0)$

psd نیست

پس تابع Beale تابع محدب نیست

⊛ به طور کلی توابع غیر محدب برای optimization سخت تر هستند زیرا می توانند دارای local minima های مختلف ، saddle point ، نواحی flat بزرگ و curvature های متفاوت باشند . این موارد برای الگوریتم های بهینه سازی چالش زا هستند و باعث می شوند که الگوریتم نتواند global minimum را پیدا کند

اکثر مسائلی که شبکه های عصبی با آن رو به رو هستند از این جنس است و بنابراین optimizer های متنوعی در درس معرفی شد که سعی دارند این چالش ها را رفع کنند

$$\text{gradient } (0,1) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(0,1) \\ \frac{\partial f}{\partial x_2}(0,1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$x^1 = x^0 - \eta \begin{bmatrix} 0 \\ 0 \end{bmatrix} \longrightarrow \text{نقطه شروع پس از یک بار به روز رسانی} = x^1 = x^0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$\boxed{5.2}$

الگوریتم gradient descent در هر قدم در جهت منفی گرادیان $h(x)$ حرکت می‌کند. می‌دانیم Q ماتریس PD و بنابراین متقارن است. پس:

$$\nabla_x h(x) = \frac{1}{2}(Q+Q^T)x + c = Qx + c$$

جواب بهینه را x^* می‌نامیم $\longrightarrow Qx^* + c = 0 \longrightarrow c = -Qx^*$

بردار r^t را به صورت $r^t = x^t - x^*$ تعریف می‌کنیم و داریم:

$$\left.\begin{array}{l} x^{t+1} = x^t - \eta\,(Qx^t + c) \\ r^t = x^t - x^* \longrightarrow x^t = r^t + x^* \end{array}\right\} \longrightarrow r^{t+1} + x^* = r^t + x^* - \eta\,(\underbrace{Qx^t - Qx^*}_{Qr^t})$$

$$\longrightarrow r^{t+1} = (I - \eta Q)\, r^t$$

از جبر خطی می‌دانیم که چون Q متقارن است پس diagonalizable است
پس می‌توان Q را به صورت مقابل decompose کرد:

$$Q = U\Sigma U^T \longrightarrow \Sigma = U^T Q U$$

که Σ ماتریس قطری است که eigen val های Q اعضای قطر آن هستند
و U ماتریس unitary است که ستون های آن eigen vector های Q هستند

$$r^{t+1} = U\underbrace{\big(I - \eta\underbrace{U^TQU}_{\Sigma}\big)}_{\begin{bmatrix} 1-\eta\lambda_1 & & 0 \\ & \ddots & \\ 0 & & 1-\eta\lambda_n \end{bmatrix}} U^T r^t \quad \text{(I)}$$

$$\longrightarrow r^{t+1} = U \begin{bmatrix} (1-\eta\lambda_1)^{t+1} & & 0 \\ & \ddots & \\ 0 & & (1-\eta\lambda_n)^{t+1} \end{bmatrix} U^T r^0 \quad \text{(II)}$$

طبق تعریف بردار r^t جهت حرکت به سمت نقطه بهینه را نشان می‌دهد.
می‌دانیم که ماتریس های unitary یا reflection اند یا rotation و اندازه ها را تغییر نمی‌دهند. همچنین می‌دانیم که ماتریس قطری تنها در راستای محور ها scale می‌کند و جهت را تغییر نمی‌دهد.
در حقیقت فرمول (I) بیانگر آن است که r^t در هر مرحله change of basis روی آن اعمال شده و توسط ماتریس U^T به فضایی می‌رود که eigen vector های Q محور های مختصات اند، سپس در آن فضا توسط ماتریس قطری در راستای این

r^t

eigen vector ها scale می‌شود و سپس توسط U به فضای قبلی برمی‌گردد. بنابراین می‌توان گفت که جهت حرکت به سمت x^* در هر مرحله، در راستای eigen vector های Q است. فرمول (II) نشان می‌دهد که اگر قرار باشد converge کنیم باید ماتریس قطری به مرور به ماتریس 0 میل کند پس $0 < 1-\eta\lambda_i < 1 \longrightarrow 0 < \eta < \frac{1}{\lambda_i}$
بدون از دست رفتن کلیت می‌توان فرض کرد که λ_n کوچکترین eigen val باشد و بنابراین $(1-\eta\lambda_n)^{t+1}$ دیرتر از بقیه المان های ماتریس به صفر میل می‌کند و بنابراین در گام های نهایی تنها در جهت eigen vector ای حرکت می‌کنیم که کمترین eigen val را دارد

* البته حرکت کردن در راستای یک eigen vector در گام های نهایی معنای بیشتری دارد تا حرکت کردن در راستای تمام eigen vector ها زیرا n تا eigen vector داریم و eigen space بُعدی به اندازه بُعد فضای اولیه دارد

5.3

a

Adam

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

$$V_t = \beta_2 V_{t-1} + (1-\beta_2) g_t^2 \longrightarrow \text{element wise}$$

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

$$\hat{V}_t = \frac{V_t}{1-\beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{V}_t} + \varepsilon} \hat{m}_t$$

در الگوریتم Adam برای بدست آوردن V_t به جای نورم 2 می توان از حالت کلی تر نورم p استفاده کرد که به صورت زیر در می آید:

$$V_t = \beta_2^p V_{t-1} + (1-\beta_2^p) |g_t|^p \longrightarrow \text{element wise}$$

حال p را به ∞ میل می دهیم و u_t را به صورت زیر تعریف می کنیم:

$$u_t = \lim_{p \to \infty} (V_t)^{\frac{1}{p}}$$

$$= \lim_{p \to \infty} \left((1-\beta_2^p) \sum_{i=1}^{t} \beta_2^{p(t-i)} \cdot |g_i|^p \right)^{\frac{1}{p}}$$

$$= \lim_{p \to \infty} \left(\sum_{i=1}^{t} \left(\beta_2^{(t-i)} \cdot |g_i| \right)^p \right)^{\frac{1}{p}}$$

$$= \max \left(\beta_2^{t-1} |g_1| , \dots , \beta_2 |g_{t-1}| , |g_t| \right)$$

$$= \max \left(\beta_2 \cdot u_{t-1} , |g_t| \right)$$

Adamax

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

$$u_t = \max(\beta_2 \cdot u_{t-1} , |g_t|) \longrightarrow \text{elementwise max}$$

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

$$\theta_{t+1} = \theta_t - \frac{\eta}{u_t} \hat{m}_t$$

بنابراین به varient ای از Adam به نام Adamax رسیدیم که به صورت تعامل است و بر اساس infinity norm عمل می کند:

b

با توجه به اینکه در Adamax بین مقدار قبلی و گرادیان ماکسیمم گرفته می شود، این روش در مواردی که گرادیان noisy باشد robust تر عمل می کند و عملاً مقادیر کوچک گرادیان را ignore می کند. یک نمونه از این حالت زمانی است که آپدیت وزن ها به صورت sparse باشد. در این حالت ابعادی که در آن ها نباید آپدیتی انجام شود تحت تأثیر نویز گرادیان قرار نمی گیرند. به طور کلی نرمالایز کردن با max نسبت به L2 norm در برابر نویز robust تر است.

همچنین در مواقعی که پارامتر ها با فرکانس کمتری آپدیت می شوند نیز adamax می تواند بهتر باشد

البته در کل Adam معروف تر و مورد استفاده تر است زیرا به رنج مسائل بیشتری apply می شود و محاسبات efficient تری دارد

⊛ هر دو روش ایده های momentum و adaptive بودن را استفاده می کنند اما Adamax برای adaptive کردن از max استفاده می کند و Adam از L2 norm