بسم الله الرحمن الرحیم

دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

یادگیری عمیق

نیم‌سال دوم ۰۴-۰۳

مدرس: مهدیه سلیمانی

تمرین اول | ددلاین تمرین : ۱۹ اسفند

- برای ارسال هر تمرین تا ساعت ۲۳:۵۹ روز ددلاین فرصت دارید. مهلت تاخیر (مجاز و غیر مجاز) برای این تمرین، ۷ روز است (یعنی حداکثر تاریخ ارسال تمرین ۲۶ اسفند است)
- در هر کدام از سوالات، اگر از منابع خارجی استفاده کرده‌اید باید آن را ذکر کنید. در صورت همفکری با افراد دیگر هم باید نام ایشان را در سوال مورد نظر ذکر نمایید.
- پاسخ تمرین باید ماحصل دانسته‌های خود شما باشد. در صورت رعایت این موضوع، استفاده از ابزارهای هوش مصنوعی با ذکر نحوه و مصداق استفاده بلامانع است.
- پاسخ ارسالی واضح و خوانا باشد. در غیر این صورت ممکن است منجر به از دست دادن نمره شود.
- پاسخ ارسالی باید توسط خود شما نوشته شده باشد. به اسکرین‌شات از منابع یا پاسخ افراد دیگر نمره‌ای تعلق نمی‌گیرد.
- در صورتی که بخشی از سوال‌ها را جای دیگری آپلود کرده و لینک آن را قرار داده باشید، حتما باید تاریخ آپلود مشخص و قابل اتکا باشد.
- محل بارگذاری سوالات نظری و عملی در هر تمرین مجزا خواهد بود. به منظور بارگذاری بایستی تمارین تئوری در یک فایل pdf با نام HW1_[First-Name]_[Last-Name]_[Student-Id].pdf و تمارین عملی نیز در یک فایل مجزای زیپ با نام HW1_[First-Name]_[Last-Name]_[Student-Id].zip بارگذاری شوند.
- در صورت وجود هرگونه ابهام یا مشکل، در کوئرای درس آن مشکل را بیان کنید و از پیغام دادن مستقیم به دستیاران آموزشی خودداری کنید.
- طراحان این تمرین : پیام تائبی-رامتین مسلمی-امیرمهدی میقانی-کسری ملیحی

بخش نظری (۱۰۰ نمره)

پرسش ۱. Matrix Differentiation (۲۰ نمره)

برای یک تابع $f: \mathbb{R}^n \to \mathbb{R}^m$ می‌توان مشتق آن را به ازای یک ورودی نظیر $x \in \mathbb{R}^n$ به صورت زیر تعریف کرد:

$$J_{i,j} = \frac{\partial f_i}{\partial x_j}, \quad J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

اولین نکته قابل توجه این است که این ماتریس $J \in \mathbb{R}^{m\times n}$ بوده و سطرهای آن ترانهاده گرادیان هر یک از ابعاد خروجی نسبت به ورودی می‌باشند. برخی منابع ترانهاده این ماتریس را به عنوان مشتق در نظر می‌گیرند و شما باید همواره به این نکته دقت داشته باشید. در صورتی که قرار باشد از یک تابع مانند $f(X) \in \mathbb{R}^{n\times m}$ بر حسب یک ماتریس همچون $X \in \mathbb{R}^{k\times p}$ مشتق بگیریم، حاصل این کار یک تنسور[۱] $\frac{\partial f(X)}{\partial X} \in \mathbb{R}^{n\times m\times k\times p}$ از مرتبه چهار خواهد شد.

[۱] Tensor

به صورت مشابه می‌توان برای ابعاد بالاتر نیز مشتق‌گیری را انجام داد. به یاد داشته باشید که برای مشتق‌گیری از هر تابعی با هر ابعادی نسبت به هر ورودی با هر ابعادی، کافیست نسبت به المان‌های آن‌ها، نظیر به نظیر مشتق جزئی را محاسبه نماییم و مقادیر بدست آمده را کنار هم قرار دهیم. یک روش ساده برای جلوگیری از مواجهه با تنسورها این است که در صورت نیاز، ماتریسی همچون $A \in \mathbb{R}^{m\times n}$ را به شکل یک بردار تخت نظیر $a \in \mathbb{R}^{mn}$ درآورد و نسبت به آن مشتق بگیریم. به زبان ریاضی خواهیم داشت:

$$A \in \mathbb{R}^{n\times m}, a \in \mathbb{R}^{nm} \to A_{ij} = a_{(i-1)n+j}$$

با استفاده از این تعریف تلاش کنید تا به پرسش‌های زیر پاسخ دهید و هرجا که نیاز شد، بجای ایجاد تنسور از تخت‌سازی ماتریس‌ها استفاده کنید. (ذکر دقیق مراحل برای کسب نمره ضروری است)

۱. اگر $a, x \in \mathbb{R}^n$ باشند، نشان دهید که:

$$\frac{\partial(a^\top x)}{\partial x} = \frac{\partial(x^\top a)}{\partial x} = a^\top.$$

۲. برای $x \in \mathbb{R}^n$ و $A \in \mathbb{R}^{m\times n}$، مقدار

$$\frac{\partial(Ax)}{\partial x}$$

را بیابید.

۳. برای $A \in \mathbb{R}^{n\times n}$ و $x \in \mathbb{R}^n$، عبارت

$$\frac{\partial(x^\top Ax)}{\partial x}$$

را محاسبه کنید. همچنین مشتق نسبت به A به صورت

$$\frac{\partial(x^\top Ax)}{\partial A}$$

را نیز تعیین کنید.

۴. برای $A, X \in \mathbb{R}^{n\times n}$، مقدار

$$\frac{\partial \operatorname{tr}(X^\top AX)}{\partial X}$$

را محاسبه کنید.

پاسخ.

۱. فرض کنید $a, x \in \mathbb{R}^n$ باشند. می‌خواهیم مشتق عبارت $f(x) = a^\top x$ را نسبت به x پیدا کنیم. ابتدا تابع را به‌صورت مؤلفه‌ای می‌نویسیم:

$$f(x) = \sum_{i=1}^{n} a_i x_i$$

حال مشتق این تابع را نسبت به x_j محاسبه می‌کنیم:

$$\frac{\partial f}{\partial x_j} = \frac{\partial}{\partial x_j}\sum_{i=1}^{n} a_i x_i = a_j$$

بنابراین، ماتریس ژاکوبین به‌صورت زیر خواهد بود:

$$J = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} = a^\top$$

چون $x^\top a = a^\top x$ است، مشتق آن نیز مشابه خواهد بود.

۲. اگر $A \in \mathbb{R}^{m\times n}$ و $x \in \mathbb{R}^n$ باشد، تابع به صورت زیر تعریف می‌شود:

$$f(x) = Ax$$

مولفه iام این تابع برابر است با:

$$f_i(x) = \sum_{j=1}^{n} A_{ij}x_j$$

با مشتق‌گیری از این عبارت نسبت به x_k داریم:

$$J_{ik} = \frac{\partial f_i}{\partial x_k} = A_{ij} \to J = A$$

۳. اگر $A \in \mathbb{R}^{n\times n}$ و $x \in \mathbb{R}^n$ باشد، تابع به‌صورت زیر تعریف می‌شود:

$$f(x) = x^\top Ax = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i A_{ij} x_j$$

حال مشتق این عبارت نسبت به x_k را می‌گیریم:

$$\frac{\partial f}{\partial x_k} = \sum_{j=1}^{n} A_{kj}x_j + \sum_{i=1}^{n} A_{ik}x_i$$

که به صورت ماتریسی نوشته می‌شود:

$$\nabla f(x) = (A + A^\top)x \to J = x^\top(A^\top + A)$$

اگر ماتریس داده شده متقارن بود، یعنی $A = A^\top$، در این صورت گرادیان برابر با $2Ax$ می‌شد.
مشتق این تابع نسبت به A برابر است با:

$$\frac{\partial x^\top Ax}{\partial A_{ij}} = x_i x_j \to \frac{\partial x^\top Ax}{\partial A} = xx^\top$$

ابعاد اصلی این خروجی $\mathbb{R}^{1\times n\times n}$ می‌باشد ولی برای سادگی آن را به صورت یک ماتریس $\mathbb{R}^{n\times n}$ در نظر می‌گیریم.

۴. با استفاده از خاصیت مشتق‌گیری از ردگیری ماتریس داریم:

$$\frac{\partial \mathrm{tr}(X^\top AX)}{\partial X} = \frac{\partial \mathrm{tr}(AXX^\top)}{\partial X} = \frac{\partial \mathrm{tr}(AXX^\top)}{\partial XX^\top}\frac{\partial XX^\top}{\partial X} = AX + A^\top X$$

$\triangleright$

پرسش ۲. Backpropagation (۲۵ نمره)

در این سوال، با یک مسئله دسته بندی سه کلاسه روبه‌رو هستیم. معماری شبکه‌ی عصبی مورد استفاده به صورت زیر است:

$$l^{(1)} = \mathrm{ReLU}(W^{(1)}x), \quad l^{(21)} = \mathrm{ReLU}(W^{(21)}l^{(1)}), \quad l^{(22)} = \sigma(W^{(22)}l^{(1)}), \quad z = \max(l^{(21)}, l^{(22)})$$

علاوه بر این، از لایه‌ی Softmax برای خروجی شبکه استفاده شده است:

$$\hat{y} = \text{softmax}(z)$$

ابعاد متغیرها به‌صورت زیر داده شده‌اند:

$$x \in \mathbb{R}^4, \quad W^{(1)} \in \mathbb{R}^{2\times 4}, \quad W^{(21)}, W^{(22)} \in \mathbb{R}^{3\times 2}$$

۱. گراف محاسباتی این شبکه را رسم کنید. در این گراف، هر گره نشان‌دهنده‌ی یک عملیات (نظیر ReLU، sigmoid، ضرب ماتریسی، و انتخاب ماکسیمم) است و یال‌ها وابستگی بین این مقادیر را نشان می‌دهند.

۲. در مرحله‌ی Backward Pass، گرادیان تابع هزینه $\mathcal{L}$ نسبت به وزن‌های شبکه را محاسبه کنید. توجه کنید که در Forward Pass برخی مقادیر محاسبه شده و ذخیره می‌شوند و نیازی به محاسبه‌ی مجدد آن‌ها نیست. در پاسخ، از گراف محاسباتی استفاده کنید و روابط زیر را به‌دست آورید:

$$\frac{\partial \mathcal{L}}{\partial z}, \quad \frac{\partial \mathcal{L}}{\partial l^{(21)}}, \quad \frac{\partial \mathcal{L}}{\partial l^{(22)}}, \quad \frac{\partial \mathcal{L}}{\partial W^{(21)}}, \quad \frac{\partial \mathcal{L}}{\partial W^{(22)}}, \quad \frac{\partial \mathcal{L}}{\partial l^{(1)}}, \quad \frac{\partial \mathcal{L}}{\partial W^{(1)}}$$

در این بخش، مشتق‌ها را گام‌به‌گام بر اساس زنجیره‌ی محاسباتی به‌دست آورید.

۳. مقدار خروجی و گرادیان‌ها را بر اساس مقداردهی اولیه‌ی زیر محاسبه کنید. فرض کنید که تابع هزینه‌ی مورد استفاده Cross Entropy است و داده‌ی ورودی به کلاس دوم تعلق دارد.

$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}, \quad W^{(1)} = \begin{bmatrix} -1 & 0 & 1 & -1 \\ 0 & -1 & 1 & 1 \end{bmatrix}, \quad W^{(21)} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad W^{(22)} = \begin{bmatrix} 2 & -1 \\ 4 & -2 \\ -2 & 1 \end{bmatrix}$$

مراحل مورد نیاز برای محاسبه‌ی Forward Pass را به طور کامل نمایش دهید. در پایان، مقدار $\hat{y}$ را محاسبه کرده و لاجیت‌ها را تا دو رقم اعشار گرد کنید.

سپس، با استفاده از قواعد به‌دست‌آمده در قسمت قبل، Backward Pass را اجرا کنید و گرادیان‌های وزن‌ها را تعیین کنید.

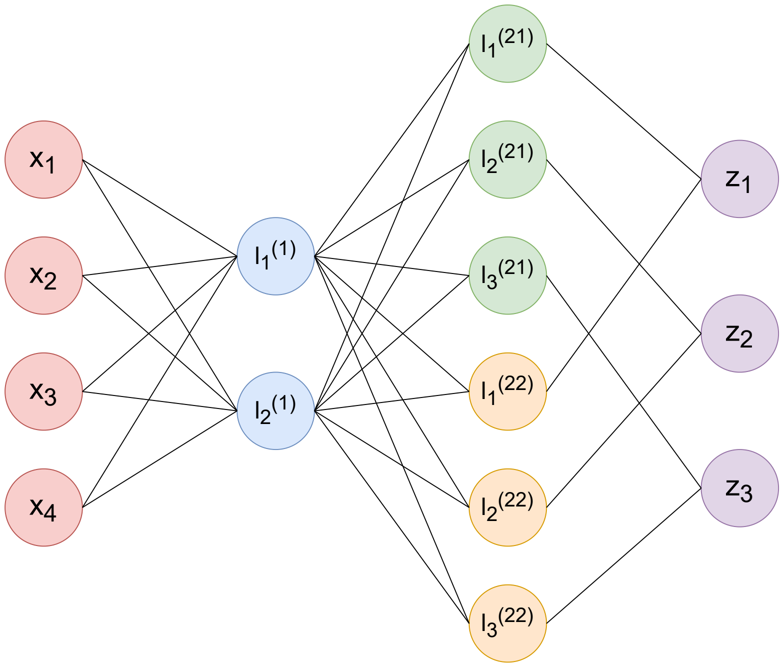
پاسخ.

۱. داریم:

۲. با استفاده از قانون زنجیره‌ای می‌دانیم:

$$\frac{\partial \mathcal{L}}{\partial l^{(21)}} = \begin{cases} \frac{\partial \mathcal{L}}{\partial z} & \text{if } l^{(21)} > l^{(22)} \\ 0 & \text{otherwise} \end{cases} \qquad \frac{\partial \mathcal{L}}{\partial l^{(22)}} = \begin{cases} \frac{\partial \mathcal{L}}{\partial z} & \text{if } l^{(22)} > l^{(21)} \\ 0 & \text{otherwise} \end{cases}$$

از آنجا که داریم:



$$l^{(21)} = \mathrm{ReLU}(W^{(21)} l^{(1)})$$

$$l^{(22)} = \sigma(W^{(22)} l^{(1)})$$

با استفاده از مشتق ReLU و سیگموئید:

$$\frac{\partial l^{(21)}}{\partial W^{(21)}} = \mathbf{diag}(\mathbb{1}(l^{(21)} > 0)) \cdot l^{(1)\top}$$

$$\frac{\partial l^{(22)}}{\partial W^{(22)}} = \mathbf{diag}(\sigma(W^{(22)} l^{(1)})(1 - \sigma(W^{(22)} l^{(1)}))) \cdot l^{(1)\top}$$

در نتیجه:

$$\frac{\partial \mathcal{L}}{\partial W^{(21)}} = \frac{\partial \mathcal{L}}{\partial l^{(21)}} \cdot l^{(1)\top}$$
$$\frac{\partial \mathcal{L}}{\partial W^{(22)}} = \frac{\partial \mathcal{L}}{\partial l^{(22)}} \cdot l^{(1)\top}$$

با توجه به روابط قبلی، مشتق نسبت به $l^{(1)}$ به‌صورت زیر محاسبه می‌شود:

$$\frac{\partial \mathcal{L}}{\partial l^{(1)}} = (W^{(21)})^\top \frac{\partial \mathcal{L}}{\partial l^{(21)}} + (W^{(22)})^\top \frac{\partial \mathcal{L}}{\partial l^{(22)}}$$

چون داریم:

$$l^{(1)} = \mathrm{ReLU}(W^{(1)} x)$$

پس:

$$\frac{\partial l^{(1)}}{\partial W^{(1)}} = \text{diag}(\mathbb{1}(l^{(1)} > 0)) \cdot x^\top$$

در نتیجه:

$$\frac{\partial \mathcal{L}}{\partial W^{(1)}} = \frac{\partial \mathcal{L}}{\partial l^{(1)}} \cdot x^\top$$

۳. داریم:

$$l^{(1)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad l^{(21)} = \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix}, \quad l^{(2)} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}, \quad z = \begin{bmatrix} 3 \\ 0.5 \\ 1 \end{bmatrix}$$

حال می‌توان گفت:

$$\text{softmax}(z) \approx \begin{bmatrix} 0.82 \\ 0.07 \\ 0.11 \end{bmatrix} \to \text{CE}(\hat{y}, y) = -\log(\hat{y}_2) \approx 2.7$$

پس:

$$\frac{\partial \mathcal{L}}{\partial W^{(1)}} = \begin{bmatrix} -0.22 & -0.44 & -0.66 & -0.22 \\ 1.4 & 2.8 & 4.2 & 1.4 \end{bmatrix}$$

$$\frac{\partial \mathcal{L}}{\partial W^{(21)}} = \begin{bmatrix} 0.82 & 1.64 \\ 0 & 0 \\ 0.11 & 0.22 \end{bmatrix}$$

$$\frac{\partial \mathcal{L}}{\partial W^{(22)}} = \begin{bmatrix} 0 & 0 \\ -0.23 & -0.46 \\ 0 & 0 \end{bmatrix}$$

$\triangleright$

پرسش ۳. Backtracking Line Search (۱۰ نمره)

در روش‌های بهینه‌سازی مانند Gradient Descent، مقدار مناسب برای t نقش مهمی در سرعت و همگرایی الگوریتم دارد. یکی از روش‌های رایج برای انتخاب مقدار t، روش Backtracking Line Search است که به‌صورت زیر تعریف می‌شود:

$$x_i = x_{i-1} - t_i \nabla f(x_{i-1})$$

Backtracking Line Search Algorithm:

Require: $\alpha \in (0, 0.5)$ (Sufficient decrease parameter)
Require: $\beta \in (0, 1)$ (Step size reduction factor)
Require: $t_0 > 0$ (Initial step size)

Initialize:
$t \leftarrow t_0$ (Set initial step size)

while $f(x+t\Delta x) > f(x) + \alpha t \nabla f(x)^T \Delta x$ **do**
$t \leftarrow \beta \cdot t$ (Reduce step size)
end while

return t

فرض کنید تابع f دارای مشتق دوم پیوسته بوده و کران ماتریسی زیر برای هسین آن برقرار باشد:

$$mI \preceq \nabla^2 f(x) \preceq MI$$

که در آن I ماتریس همانی است و نماد $\preceq$ نشان‌دهنده‌ی ترتیب جزئی بین ماتریس‌های متقارن (نیمه‌مثبت معین[۲]) می‌باشد. این شرط نشان می‌دهد که تمامی مقادیر ویژه‌ی $\nabla^2 f(x)$ در بازه‌ی $[m, M]$ قرار دارند.
به عبارت دیگر، از نامساوی $mI \preceq \nabla^2 f(x) \preceq MI$ نتیجه می‌شود که ماتریس $MI - \nabla^2 f(x)$ نیمه‌مثبت معین است، یعنی برای هر بردار v داریم:

$$v^T(MI - \nabla^2 f(x))v \geq 0, \quad \forall v \in \mathbb{R}^n.$$

علاوه بر این، بردار Δx یک جهت کاهش در نقطه x است اما لزوماً برابر با $\nabla f(x)$ نیست.

۱. نشان دهید که اگر مقدار t در بازه‌ی زیر قرار گیرد، شرط توقف در Backtracking برقرار خواهد بود:

$$0 < t \leq -\frac{\nabla f(x)^T \Delta x}{M\|\Delta x\|_2^2}$$

راهنمایی: از بسط تیلور مرتبه دوم برای تابع $f(x)$ در جهت Δx استفاده کنید. سپس با استفاده از کران بالای $\nabla^2 f(x)$ نامساوی مناسب را استخراج کنید.

۲. با استفاده از نتیجه‌ی بخش قبل، یک کران بالا برای تعداد تکرارهای مورد نیاز در فرآیند Backtracking Line Search به‌دست آورید.
راهنمایی: مقدار t در هر تکرار با ضریب β کاهش می‌یابد. از این خاصیت استفاده کرده و لگاریتم بگیرید تا تعداد تکرارهای مورد نیاز را به‌دست آورید.

پاسخ. با استفاده از بسط تیلور و همچنین کران بالای $\nabla^2 f(x)$ $MI \succeq$ داریم:

$$f(x+t\Delta x) \leq f(x) + t\nabla f(x)^T \Delta x + (M/2)t^2 \Delta x^T \Delta x$$

با جایگذاری $\alpha + (1-\alpha)$ داریم:

$$\overbrace{f(x+t\Delta x) \leq f(x) + \alpha t\nabla f(x)^T \Delta x}^{\text{Stopping Condition}} + \underbrace{(1-\alpha)t\nabla f(x)^T \Delta x + (M/2)t^2 \Delta x^T \Delta x}_{\leq 0}$$

$$0 < t \leq -2(1-\alpha)\frac{\nabla f(x)^T \Delta x}{M\|\Delta x\|_2^2}$$

$$0 < t \leq -\frac{\nabla f(x)^T \Delta x}{M\|\Delta x\|_2^2}$$

[۲] Positive Semi-Definite (PSD) Ordering

برای محاسبه تکرارهای جست‌وجوی بازگشتی داریم:

$$0 < t_k \leq -\frac{\nabla f(x)^T \Delta x}{M\|\Delta x\|_2^2}$$

$$0 < t_0 \beta^k \leq -\frac{\nabla f(x)^T \Delta x}{M\|\Delta x\|_2^2}$$

$$0 < \beta^k \leq \frac{-1}{t_0} \frac{\nabla f(x)^T \Delta x}{M\|\Delta x\|_2^2}$$

$$k \geq \frac{\log \frac{-1}{t_0} \frac{\nabla f(x)^T \Delta x}{M\|\Delta x\|_2^2}}{\log \beta}$$

▷

پرسش ۴. Adam (۱۵ نمره)

در درس، الگوریتم بهینه‌سازی Adam معرفی شده است که با استفاده از میانگین‌های نمایی از گرادیان‌ها و مربع‌های آن‌ها، نرخ به‌روزرسانی پارامترها را بهبود می‌بخشد. الگوریتم Adam به صورت زیر ارائه شده است:

Adam Algorithm:

Require: α : Stepsize
Require: $\beta_1, \beta_2 \in [0, 1]$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0 : Initial parameter vector

Initialize:
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)

while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Obtain gradient at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second moment estimate)
$\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ (Compute bias-corrected second moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

۱. ابتدا توجه کنید که در الگوریتم Adam رابطه میانگین نمایی برای مربع گرادیان‌ها به صورت بازگشتی تعریف شده است:

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

حالا این رابطه را به صورت غیر بازگشتی بازنویسی کنید تا نشان داده شود که

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

۲. فرض کنید گرادیان‌ها به صورت مستقل و با توزیع یکسان ($i.i.d.$) باشند. از رابطه‌ی غیر بازگشتی بدست آمده، امید ریاضی v_t را محاسبه کنید. منظور از $i.i.d.$ این است که

$$\mathbb{E}[v_t] = (1-\beta_2)\sum_{i=1}^{t}\beta_2^{t-i}\,\mathbb{E}[g_i^2].$$

نتیجه نهایی نشان می‌دهد که

$$\mathbb{E}[v_t] = \mathbb{E}[g_t^2]\,(1-\beta_2^t).$$

در این بخش توضیح دهید که این نتیجه چه مفهومی دارد و چرا نشان‌دهنده بایاس در تخمین مربع گرادیان‌ها است.

۳. بر اساس نتایج بخش قبل توضیح دهید که چرا اعمال مرحله‌ی Bias Correction در الگوریتم Adam ضروری است. به طور خلاصه بیان کنید که در ابتدای اجرای الگوریتم، مقادیر v_t (و همچنین m_t) به دلیل مقدار اولیه صفر به سمت صفر سوگیری دارند و Bias Correction باعث می‌شود که این سوگیری اصلاح شده و تخمین‌های به‌دست آمده دقیق‌تر باشند. همچنین توضیح دهید که تاثیر مقدار β_2 چگونه این سوگیری به سمت صفر را تشدید می‌کند.

۴. در الگوریتم Adam، به‌روزرسانی هر وزن بر مبنای مقیاس کردن گرادیان‌ها با معکوس «نرم‌دو» گرادیان‌های فعلی و قبلی انجام می‌شود. حال با جایگزینی «نرم‌دو» با «نرم‌بی‌نهایت» (به این صورت که توان β_2 را برابر p در نظر گرفته و سپس $p\to\infty$ را اعمال کنید) می‌توانید الگوریتم بهینه‌سازی جدیدی به نام AdaMax را به دست آورید. راهنمایی: به مقاله اصلی Adam مراجعه کنید. در آن توضیح داده شده که

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

که منجر به رابطه‌ی بازگشتی

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

می‌شود. الگوریتم AdaMax به صورت زیر ارائه شده است:

AdaMax Algorithm:

Require: α : Stepsize
Require: $\beta_1, \beta_2 \in [0,1]$
Require: $f(\theta)$: Stochastic objective function
Require: θ_0 : Initial parameter vector

Initialize:
$m_0 \leftarrow 0$
$u_0 \leftarrow 0$ (Initialize the exponentially weighted infinity norm)
$t \leftarrow 0$

while θ_t not converged **do**
$t \leftarrow t+1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
$m_t \leftarrow \beta_1\cdot m_{t-1} + (1-\beta_1)\cdot g_t$
$u_t \leftarrow \max\big(\beta_2\cdot u_{t-1},\ |g_t|\big)$
$\theta_t \leftarrow \theta_{t-1} - \Big(\alpha/(1-\beta_1^t)\Big)\cdot m_t/u_t$

end while
return θ_t

---

۵. الگوریتم حاصل را با الگوریتم Adam مقایسه کنید. به صورت مستقیم بیان کنید که در چه شرایطی استفاده از الگوریتم حاصل‌شده نسبت به Adam بهتر عمل می‌کند؟

پاسخ.

- ابتدا به محاسبه نرم‌بی‌نهایت می‌پردازیم:

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

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

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

$$= \max\left(\beta_2^{t-1}|g_1|, \beta_2^{t-2}|g_2|, \ldots, |g_t| \right)$$

که برابر است با فرمول بازگشتی زیر:

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

در نهایت الگوریتم جدید به شکل زیر خواهد بود:

AdaMax Algorithm:

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0,1]$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector

Initialize:
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$u_0 \leftarrow 0$ (Initialize the exponentially weighted infinity norm)
$t \leftarrow 0$ (Initialize timestep)

while θ_t not converged **do**
$t \leftarrow t+1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1)\cdot g_t$ (Update biased first moment estimate)
$u_t \leftarrow \max(\beta_2 \cdot u_{t-1}, |g_t|)$ (Update the exponentially weighted infinity norm)
$\theta_t \leftarrow \theta_{t-1} - (\alpha/(1-\beta_1^t)) \cdot \hat{m}_t/u_t$ (Update parameters)
end while
return θ_t (Resulting parameters)

- Adam ممکن است به دلیل Scaling نرم‌دو در مواجهه با گرادیان‌های بسیار بزرگ دچار ناپایداری شود، در حالی که AdaMax در چنین شرایطی پایدارتر است زیرا نرم‌بی‌نهایت کاهش کمتری دارد. همچنین، AdaMax در شرایطی که تفاوت بزرگی در مقدار گرادیان‌ها وجود دارد، پایداری بیشتری ارائه می‌دهد. استفاده از نرم‌بی‌نهایت در AdaMax از مشکل gradients vanishing جلوگیری کرده و آن را برای شبکه‌های عمیق مانند DNN، GAN و Transformer مناسب می‌سازد.

-

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

با یک مرحله جایگذاری داریم:

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

و با تکرار آن داریم:

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

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

-

$$\mathbb{E}[v_t] = \mathbb{E}\left[(1-\beta_2) \sum_{i=1}^{t} \beta_2^{t-i} g_i^2\right]$$
$$\mathbb{E}[v_t] = (1-\beta_2) \sum_{i=1}^{t} \beta_2^{t-i} \mathbb{E}\left[g_i^2\right]$$
$$\text{Under i.i.d. assumption:} \mathbb{E}[v_t] = \mathbb{E}[g_t^2](1-\beta_2) \sum_{i=1}^{t} \beta_2^{t-i}$$
$$\mathbb{E}[v_t] = \mathbb{E}[g_t^2](1-\beta_2) \frac{1-\beta_2^t}{1-\beta_2}$$
$$\mathbb{E}[v_t] = \mathbb{E}[g_t^2](1-\beta_2^t)$$

با تقسیم v_t بر عبارت $(1-\beta_2^t)$ امید ریاضی آن برابر با امید ریاضی گرادیان‌ها می‌شود، که دلیل انجام مرحله‌ی bias correction است.

- در بلندمدت، هر دو امید ریاضی همگرا خواهند شد. با این حال، در ابتدا مقدار به سمت صفر سوگیری دارد. این سوگیری با مقادیر بزرگ‌تر β_2 بدتر می‌شود.

$\triangleright$

پرسش ۵. بهینه‌سازی مرتبه دوم (۱۰ نمره)

برای یک تابع دلخواه $f(x)$ می‌توان از سری Taylor Series مرتبه دوم در نزدیکی نقطه x_0 استفاده کرد:

$$f(x) \approx f(x_0) + (x-x_0)^\top \nabla f(x_0) + \frac{1}{2}(x-x_0)^\top H (x-x_0),$$

که در اینجا:

- $\nabla f(x_0)$ بردار gradient تابع در نقطه x_0 است.
- H ماتریس هسین[۳] تابع f در نقطه x_0 است که به صورت

$$H_{i,j} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$$

تعریف می‌شود.

۱. فرض کنید نرخ یادگیری ϵ در الگوریتم Descent Gradient به صورت $x = x_0 - \epsilon\, g,$ به کار می‌رود، که در آن $g = \nabla f(x_0)$ می‌باشد. طبق بسط تیلور، مقدار $f(x_0 - \epsilon\, g)$ را به صورت تقریبی تا جمله مرتبه سوم (یعنی با باقی‌مانده‌ای از مرتبه $O(\epsilon^3)$) بنویسید.

۲. در صورتی که $g^\top H g$ منفی یا صفر باشد، افزایش یا کاهش نرخ یادگیری ϵ چه تأثیری بر مقدار تقریبی $f(x)$ خواهد داشت؟ چرا استفاده از این روش بهینه‌سازی در چنین شرایطی مناسب نیست؟

۳. اگر $g^\top H g$ مثبت باشد، با مشتق‌گیری از عبارت ارائه‌شده نسبت به ϵ و قرار دادن آن برابر صفر، نرخ یادگیری بهینه $\epsilon^\star$ را به دست آورید.

۴. با توجه به رابطه به دست آمده برای $\epsilon^\star$، توضیح دهید که مقدار نرخ یادگیری بهینه $\epsilon^\star$ در چه شرایطی بیشینه و کمینه خواهد شد.
راهنمایی: Hx را به‌صورت ترکیب خطی از بردارهای ویژه‌ی H و مقدار ویژه‌های متناظر آن بنویسید.

پاسخ.

۱. در این صورت داریم:

$$x = x_0 - \epsilon g \to f(x_0 - \epsilon g) \approx f(x_0) - \epsilon g^\top g + \frac{1}{2}\epsilon^2 g^\top H g$$

۲. در صورتی که این مقدار منفی یا صفر باشد، تخمین سری تیلور پیش‌بینی می‌کند که با افزایش مقدار ϵ همواره منجر به کاهش f خواهد شد. در عمل این تخمین مناسب نخواهد بود و باید از روش‌های دیگری استفاده کنیم.

۳. در این صورت داریم:

$$0 - g^\top g + \epsilon g^\top H g = 0 \to \epsilon^\star = \frac{g^\top g}{g^\top H g}$$

۴. برای سادگی در ابتدا صرفا به عبارت Hg توجه داشته باشید. اگر ماتریس هسین دارای مقادیر ویژه $\lambda_1, \cdots, \lambda_n$ باشد، در این صورت می‌توان g را به صورت جمع وزن‌داری از بردارهای ویژه متناظر با این مقادیر در نظر گرفت.

$$Hg = H(\sum_{i=1}^{n} \alpha_i v_i) = \sum_{i=1}^{n} \alpha_i (H v_i) = \sum_{i=1}^{n} \alpha_i \lambda_i v_i$$

بنابراین می‌توان گفت که هنگامی این عبارت کمینه یا بیشینه می‌شود که g هم‌راستا با بردار ویژه‌ای که دارای کوچک‌ترین مقدار یا بزرگ‌ترین مقدار است:

$$Hg = \lambda g \to \begin{cases} \max \ Hg = \lambda_{\max} g \\ \min \ Hg = \lambda_{\min} g \end{cases}$$

[۳] Hessian Matrix

در این صورت بیش‌ترین نرخ یادگیری ممکن هنگامی رخ خواهد داد که گرادیان هم‌راستا با بردار متناظر با کوچک‌ترین مقدار ویژه آن باشد و در این صورت داریم:

$$\epsilon^\star = \frac{g^\top g}{g^\top H g} = \frac{g^\top g}{\lambda_{\min} g^\top g} = \frac{1}{\lambda_{\min}}$$

به روش مشابه می‌توان نشان که کمترین مقدار ممکن برای نرخ یادگیری هنگامی است که گرادیان هم راستا با بردار متناظر با بزرگ‌ترین مقدار ویژه باشد و مقدار آن برابر با $\frac{1}{\lambda_{\max}}$ خواهد بود.

$\triangleright$

پرسش ۶. Regularization (۲۰ نمره)

اگر برای تابعی نظیر $f : \mathbb{R}^n \to \mathbb{R}$ داشته باشیم:

$$\|f(x_1) - f(x_2)\| \leq L\|x_1 - x_2\|, \qquad \forall x_1,\ x_2 \in \mathbb{R}^n$$

می‌گوییم این تابع L-Lipschitz است. برای درک بهتر این مفهوم $x_1 = x_2 + \delta$ را در نظر داشته باشید. اکنون می‌توان ادعا کرد که برای هر x_2 و به ازای هر δ دلخواه داریم:

$$\|f(x_2 + \delta) - f(x_2)\| \leq L\|\delta\|$$

یعنی اگر به ازای هر مقدار ورودی نظیر x، آن را به اندازه δ تغییر دهیم، میزان تغییرات در خروجی تابع کمتر از $L\|\delta\|$ خواهیم داشت و به بیانی یک کران بالا برای میزان تغییرات تعریف کردیم.
می‌دانیم که در هر لایه از یک شبکه عصبی داریم:

$$x^{(\ell+1)} = h^{(\ell)}(W^{(\ell)} x^{(\ell)})$$

که در اینجا $h(\cdot)$ یک تابع فعال‌سازی[۴] دلخواه است. حال به پرسش‌های زیر پاسخ دهید.

۱. با نوشتن رابطه فوق برای ضرب ماتریسی Wx نشان دهید که ثابت Lipschitz برای این ماتریس برابر با نرم الحاقی[۵] آن است. یعنی:

$$L = \max_{\delta \neq 0} \frac{\|W\delta\|}{\|\delta\|} = \|W\|$$

۲. با استفاده از تجزیه مقادیر تکین[۶] این ماتریس نشان دهید که ثابت Lipschitz این ماتریس برابر است با بزرگ‌ترین مقدار تکین آن.

۳. در صورتی که پس از ضرب ماتریس از یک تابع فعال‌سازی نظیر $h(\cdot)$ استفاده کنیم، ثابت Lipschitz آن چه تغییری خواهد کرد؟ (کافیست برای توابع داده شده مقدار جدید را برای $h(Wx)$ بدست آورید.)

$$h_1(z) = \mathrm{ReLU}(z) = \max(0, z)$$
$$h_2(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$
$$h_3(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

Activation Function[۴]
Spectral Norm[۵]
Singular Value Decomposition (SVD)[۶]

۴. با استفاده از نتایج بخش‌های قبل، با فرض استفاده از توابع فعال‌سازی ReLU در تمامی لایه‌های یه شبکه عصبی دارای n لایه، کران بالایی برای ثابت Lipschitz آن ارائه دهید.

۵. در صورتی که داده‌های ما دارای مقداری نویز باشد، یعنی $\tilde{x} = x + \epsilon$ باشد، توضیح دهید چگونه Weight Decay می‌تواند به ما در رسیدن به نتایج بهتر کمک کند؟ (راهنمایی: با استفاده از نتایج بخش‌های قبل و با مقایسه پیش‌بینی مدل برای $y = Wx$ و $\tilde{y} = W\tilde{x}$ به این سوال پاسخ دهید.)

پاسخ.

۱. داریم:

$$\|Wx_1 - Wx_2\| \leq L\|x_1 - x_2\|$$

برای سهولت تغییر متغیر $x_1 = x_2 + \delta$ را انجام می‌دهیم:

$$\|W(x_2 + \delta) - Wx_2\| \leq L\|x_2 + \delta - x_2\| \leftrightarrow \|W\delta\| \leq L\|\delta\|$$

اکنون باید بزرگ‌ترین مقدار ممکن L را پیدا کنیم. یعنی:

$$L = \max_{\delta \neq 0} \frac{\|W\delta\|}{\|\delta\|} = \|W\|$$

۲. با استفاده از $\|\delta\|_2 = 1$ می‌توانیم (اگر بخواهیم می‌توانیم هر مقدار دیگری را نیز در نظر بگیریم ولی در نهایت این مقدار در صورت و مخرج با هم ساده می‌شوند و پاسخ مسئله تفاوتی با یک نخواهد داشت) نشان دهیم:

$$\|W\|_2 = \max_{\|\delta\|=1} \frac{\sqrt{\delta^\top W^\top W\delta}}{\sqrt{\delta^\top \delta}} = \max_{\|\delta\|=1} \sqrt{\delta^\top W^\top W\delta}$$

برای یافتن مقدار بیشینه‌ی $\|W\|_2$ می‌توانیم ابتدا $\|W\|_2^2$ را به کمک ضرایب لاگرانژ و قید $g(\delta) = \delta^\top\delta - 1$ بیشینه کنیم:

$$\max_{\delta^\top\delta=1} \left[\sqrt{\delta^\top W^\top W\delta}\right] \leftrightarrow \max_{\delta^\top\delta=1} \left[\delta^\top W^\top W\delta\right] \leftrightarrow \max_{\delta} \left[\delta^\top W^\top W\delta - \lambda(\delta^\top\delta - 1)\right]$$

با مشتق گرفتن از تابع فوق نسبت به δ و برابر با صفر قرار دادن حاصل آن خواهیم داشت:

$$\frac{\partial}{\partial\delta}\left[\delta^\top W^\top W\delta - \lambda(\delta^\top\delta - 1)\right] = 0$$
$$2W^\top W\delta - 2\lambda\delta = 0$$
$$W^\top W\delta - \lambda\delta = 0$$
$$W^\top W\delta = \lambda\delta$$

با کمی دقت متوجه می‌شویم که δ بردار ویژه و λ مقدار ویژه‌ی متناظر با آن، برای ماتریس $W^\top W$ می‌باشد. در این صورت با جایگذاری این مقدار در رابطه‌ی اصلی داریم:

$$\|W\|_2 = \max_{\|\delta\|=1} \sqrt{\delta^\top W^\top W\delta} = \max_{\|\delta\|=1} \sqrt{\delta^\top \lambda\delta} = \max_{\|\delta\|=1} \sqrt{\lambda}\sqrt{\delta^\top\delta} = \max_{\delta} \sqrt{\lambda}$$

پس بردار δ ای که منجر به بیشینه شدن $\|W\delta\|$ می‌شود، بردار ویژه‌ی متناظر با بزرگ‌ترین مقدار ویژه $W^\top W$ است که همان بزرگ‌ترین مقدار تکین W می‌باشد.

۳. بیشینه مقدار تغییرات برابر است با بزرگ‌ترین مقدار ممکن برای مشتق هر یک از توابع:

- برای تابع ReLU این مقدار برابر با یک است.
  برای تابع tanh داریم؛
  $$\tanh'(z) = 1 - \tanh^2(z)$$
  و مقدار بیشینه دوباره برابر با یک است.
- برای این تابع داریم؛
  $$\sigma(z)' = \sigma(z)\,(1 - \sigma(z))$$
  که مقدار بیشینه آن برابر با 0.25 است که متناظر با ورودی صفر می‌شود.

۴. با استفاده از نتایج بخش پیش داریم:
$$L = \prod_{\ell=1}^{n} \|W^{(\ell)}\|$$

۵. هنگامی که داده‌ها دارای نویز باشند ($\tilde{x} = x + \epsilon$)، پیش‌بینی مدل برای داده‌های نویزی $\tilde{y} = W\tilde{x}$ و داده‌های اصلی $y = Wx$ متفاوت خواهد بود. اختلاف بین این دو پیش‌بینی به صورت زیر است:
$$\|\tilde{y} - y\| = \|W\tilde{x} - Wx\| = \|W\epsilon\|$$
با استفاده از ثابت Lipschitz، داریم:
$$\|W\epsilon\| \leq \|W\|\|\epsilon\|$$
اگر نرم ماتریس W کوچک باشد، اختلاف بین پیش‌بینی‌ها نیز کوچک خواهد بود. Weight Decay با اضافه کردن یک جمله تنظیم‌کننده به تابع هزینه، نرم ماتریس وزن‌ها را کاهش می‌دهد. این کار باعث می‌شود مدل در برابر نویز مقاوم‌تر شود و از بیش‌برازش[۷] جلوگیری کند. به عبارت دیگر، Weight Decay به مدل کمک می‌کند تا تغییرات کوچک در ورودی (ناشی از نویز) تأثیر کمتری بر خروجی داشته باشند.

▷

بخش عملی (۱۰۰ نمره)

در این مجموعه چهار تمرین ارائه شده است که آخرین تمرین دارای نمره اضافی می‌باشد. از شما خواسته می‌شود هر تمرین را به‌صورت جداگانه در سامانهٔ Quera بارگذاری نمایید. لازم به ذکر است که به استثنای تمرین ۳ (تحویل **حضوری**)، کلیه کدها باید از قبل اجرا شده باشند؛ در غیر این صورت نمره مربوطه تعلق نخواهد گرفت.

پرسش ۷. Basics (۴۰ نمره)

در این تمرین با مفاهیم اولیه PyTorch آشنا شده و پس از کسب تسلط کافی بر این کتابخانه، به پیاده‌سازی یک شبکه عصبی برای یک تسک دسته‌بندی (classification) مبتنی بر توابع ساده خواهید پرداخت. لازم است هر دو فایل مربوط به این تمرین (فایل Basics.ipynb و فایل pytorch_basic.py) پس از کامل اجرا شدن، به‌صورت یک فایل فشرده (zip) آپلود شوند. همچنین حجم فایل فشرده باید حداکثر ۲۰ مگابایت باشد؛ در صورتی که حجم فایل از این

[۷] Overfitting

مقدار تجاوز کند، می‌بایست آن را در گوگل درایو به‌صورت پرایوت قرار داده و لینک دسترسی را در زمان تحویل ارسال نمایید. مصحح، در زمان تحویل، از لینک ارسال‌شده فایل را بررسی خواهد کرد.

پرسش ۸. NN_Scratch (۲۵ نمره)

هدف این تمرین، ایجاد ماژول‌های شبکه عصبی به‌صورت دستی و ترکیب آن‌ها جهت ساخت یک شبکه کامل است. لازم است بدانید که در هنگام اجرا، به پوشهٔ utils نیاز دارید؛ لذا از هرگونه تغییر یا دستکاری در آن اجتناب فرمایید.

همچنین تأکید می‌شود که این بخش نیز باید به‌صورت کامل اجرا شده و سپس آپلود گردد (در هنگام آپلود نیازی به ارسال پوشهٔ utils نمی‌باشد).

پرسش ۹. Optimization (۳۵ نمره)

این تمرین به‌صورت **حضوری** تحویل داده می‌شود. پس از اتمام کار، کلیه خروجی‌های نوت‌بوک باید پاکسازی شود؛ اما توضیحات، بخش‌های مختلف و کدهایی که قابلیت اجرا دارند، بدون تغییر باقی بمانند و به‌صورت نوت‌بوک آپلود شوند. زمان تحویل به شما اعلام خواهد شد. در زمان تحویل، مسئول مربوطه فایل را اجرا کرده و در صورت بروز هرگونه خطا (به‌عنوان مثال، کامند ارور)، نمره صفر تعلق خواهد گرفت.

پرسش ۱۰. Lazy_Gradient (تمرین امتیازی)

این تمرین به عنوان تمرین امتیازی در نظر گرفته شده و موضوع آن بررسی چالش‌های مربوط به حافظه هنگام کار با شبکه‌های عصبی است.