



یادگیری عمیق

نیم سال دوم ۰۴-۰۳
مدرس: مهدیه سلیمانی

دولاین تمرین : ۱۹ اسفند

تمرین اول

- برای ارسال هر تمرین تا ساعت ۲۳:۵۹ روز دولاین 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 کرده اید 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 دانسته های 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 اسکرین شات از منابع یا پاسخ افراد دیگر نمره ای تعلق نمی گیرد.
- در صورتی که بخشی از سوال ها 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 کوثرای 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 $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ می توان مشتق آن را به ازای یک ورودی نظیر $x \in \mathbb{R}^n$ به صورت زیر تعریف کرد:

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

اولین نکته قابل توجه این است که این ماتریس $J \in \mathbb{R}^{m \times n}$ بوده و سطرهاى آن ترانهاده گرادیان هر یک از ابعاد خروجی نسبت به ورودی می باشند. برخی منابع ترانهاده این ماتریس را به عنوان مشتق در نظر می گیرند 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 تانسور^۱ $\frac{\partial f(X)}{\partial X} \in \mathbb{R}^{n \times m \times k \times p}$ از مرتبه چهار خواهد شد.

Tensor^۱

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، کافیت 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 تانسورها 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:

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

با استفاده از این تعریف تلاش کنید تا به پرسش‌های زیر پاسخ دهید و هر جا که نیاز شد، بجای ایجاد تانسور 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

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

را محاسبه کنید.

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:

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

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

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

ابعاد متغیرها به صورت 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:

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

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

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.

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.

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**

Positive Semi-Definite (PSD) Ordering^۲

```

 $t \leftarrow t + 1$ 
 $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$  (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  $\theta_t$  (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 $p \rightarrow \infty$ 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

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

که منجر به رابطه‌ی بازگشتی

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

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**

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$

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

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

$\theta_t \leftarrow \theta_{t-1} - \left(\alpha / (1 - \beta_1^t) \right) \cdot m_t / u_t$

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 حاصل شده نسبت به Adam بهتر عمل می‌کند؟

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.

^۳Hessian Matrix

۲. در صورتی که $g^T H g$ 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 $g^T H g$ مثبت باشد، با مشتق‌گیری از عبارت ارائه‌شده نسبت به ϵ و قرار دادن آن برابر صفر، نرخ یادگیری بهینه ϵ^* را به دست آورید.
۴. با توجه به رابطه به دست آمده برای ϵ^* ، توضیح دهید که مقدار نرخ یادگیری بهینه ϵ^* در چه شرایطی بیشینه و کمینه خواهد شد.
- راهنمایی:** Hx را به صورت ترکیب خطی از بردارهای ویژه H و مقدار ویژه‌های متناظر آن بنویسید.

پرسش ۶. Regularization (۲۰ نمره)

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

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

می‌گوییم این تابع L -Lipschitz است. برای درک بهتر این مفهوم δ 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.)

$$h_1(z) = \text{ReLU}(z) = \max(0, z)$$

$$h_2(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$h_3(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

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 یک 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.)

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 سامانه 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 مقدار تجاوز کند، می‌بایست آن را در گوگل درایو به‌صورت پراپورت 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 پوشه 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 پوشه 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.