پاسخ تمرین اول **مهدیه نادری: ۹۸۵۲۲۰۷۶**

سوال اول)

۱.الف تشخیص کلاس خبر:

کلاس	متن خبر	نوع داده
۰	فناوری فرهنگی علمی اقتصادی	آموزش
۰	فناوری فرهنگی علمی اجتماعی سیاسی	آموزش
۱	فناوری فرهنگی اجتماعی سیاسی	آموزش
۱	علمی اجتماعی سیاسی اقتصادی	آموزش

باید برای هر داده تست احتمال تعلق به هر کلاس را محاسبه و کلاسی که احتمال بیشتری دارد برایش انتخاب کنیم.

فرمول کلی بیز را داریم:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \rightarrow P(A) = \frac{P(B|A)P(A)}{P(B)}$$

قضیه بیز ساده بر اساس احتمالات پیشامدهای پیشین، پسین، درست نمایی و شواهد:

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)} = \frac{1}{z}\, P(C_k) \prod_{i=1}^{n} P(X_i|C_k) \propto P(C_k) \prod_{i=1}^{n} P(X_i|C_k)$$

$$P(X_i|C_k) = \frac{number\ of\ X_i\ in\ C_k}{number\ of\ all\ X_i s\ in\ C_k}$$

پسین : $P(C_k|X)$

پیشین: $P(X|C_k)$

درستنمایی : $P(C_k)$

شواهد : $P(X)$ که با توجه به ثابت بودن در حاصل احتمال تمام کلاس ها، از آن صرف نظر شد.

$$Z = P(X) = \sum_{k} P(C_k)\, P(X|C_k)$$

برای اولین داده تست (فناوری، فرهنگی، علمی، اجتماعی) از بیز ساده استفاده میکنیم:

محسابه احتمال کلاس ۰:

$$P\left(C_0 \middle| (\text{فناوری، فرهنگی، علمی، اجتماعی})\right) = P(C_0) \prod_{i=1}^{4} P(X_i|C_0)$$

با توجه به جدول داده شده، ۴ داده آموزشی داریم که خروجی ۲ مورد ۰ و خروجی دو مورد دیگر ۱ است. در نتیجه داریم:

$$P(C_0) = 0.25\,, P(C_1) = 0.25$$

$$P\left(C_0 \middle| (\text{فناوری، فرهنگی، علمی، اجتماعی})\right)$$

$$= 0.25 \left(P\left(\text{اجتماعی}|C_0\right)\right)\left(P\left(\text{علمی}|C_0\right)\right)\left(P\left(\text{فرهنگی}|C_0\right)\right)\left(P\left(\text{فناوری}|C_0\right)\right)$$

$$= 0.25 \left(\frac{1}{9}\right)\left(\frac{2}{9}\right)\left(\frac{2}{9}\right)\left(\frac{3}{9}\right) = 0.00045$$

محسابه احتمال کلاس ۱:

$$P\left(C_1 \middle| (\text{فناوری، فرهنگی، علمی، اجتماعی})\right) = P(C_1) \prod_{i=1}^{4} P(X_i|C_1)$$

$$P\left(C_1\middle|(\text{اجتماعی ،علمی ،فرهنگی ،فناوری})\right)$$
$$= 0.25\left(P\left(\text{اجتماعی}|C_1\right)\right)\left(P\left(\text{علمی}|C_1\right)\right)\left(P\left(\text{فرهنگی}|C_1\right)\right)\left(P\left(\text{فناوری}|C_1\right)\right)$$
$$= 0.25\left(\frac{2}{8}\right)\left(\frac{1}{8}\right)\left(\frac{1}{8}\right)\left(\frac{1}{8}\right) = 0.00012$$

احتمال کلاس ۰ بودن بیشتر است. **پس کلاس تست اول ۰ است.**

برای دومین داده تست (فناوری، فرهنگی، علمی، اجتماعی، ورزشی) نمیتوانیم از بیز ساده استفاده کنیم چون:

محسابه احتمال کلاس ۰:

$$P\left(C_0\middle|(\text{ورزشی ،اجتماعی ،علمی ،فرهنگی ،فناوری})\right) = P(C_0)\prod_{i=1}^{5} P(X_i|C_0)$$
$$P\left(C_0\middle|(\text{ورزشی ،اجتماعی ،علمی ،فرهنگی ،فناوری})\right)$$
$$= 0.25\left(P\left(\text{ورزشی}|C_0\right)\right)\left(P\left(\text{اجتماعی}|C_0\right)\right)\left(P\left(\text{علمی}|C_0\right)\right)\left(P\left(\text{فرهنگی}|C_0\right)\right)\left(P\left(\text{فناوری}|C_0\right)\right)$$
$$= 0.25\left(\frac{0}{9}\right)\left(\frac{1}{9}\right)\left(\frac{2}{9}\right)\left(\frac{2}{9}\right)\left(\frac{3}{9}\right) = 0$$

محسابه احتمال کلاس ۱:

$$P\left(C_1\middle|(\text{ورزشی ،اجتماعی ،علمی ،فرهنگی ،فناوری})\right) = P(C_1)\prod_{i=1}^{4} P(X_i|C_1)$$
$$P\left(C_1\middle|(\text{ورزشی ،اجتماعی ،علمی ،فرهنگی ،فناوری})\right)$$
$$= 0.25\left(P\left(\text{ورزشی}|C_1\right)\right)\left(P\left(\text{اجتماعی}|C_1\right)\right)\left(P\left(\text{علمی}|C_1\right)\right)\left(P\left(\text{فرهنگی}|C_1\right)\right)\left(P\left(\text{فناوری}|C_1\right)\right)$$
$$= 0.25\left(\frac{0}{8}\right)\left(\frac{2}{8}\right)\left(\frac{1}{8}\right)\left(\frac{1}{8}\right)\left(\frac{1}{8}\right) = 0$$

چون احتمال‌ها صفر میشود نمیتوان پیشبینی درستی داشت.

به همین دلیل از هموارسازی لاپلاسین با ضریب آلفای ۱ استفاده میکنیم تا مشکل احتمال صفر حل شود:

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)} = \frac{1}{z}P(C_k)\prod_{i=1}^{n} P(X_i|C_k) \propto P(C_k)\prod_{i=1}^{n} P(X_i|C_k)$$
$$P(X_i|C_k) = \frac{(number\ of\ rewiews\ with\ X_i\ and\ y = C_k) + \alpha}{N + \alpha * K}$$

پارامتر هموارسازی : α

تعداد ابعاد ویژگی‌ها در داده‌ها: K

برای اولین داده تست (فناوری، فرهنگی، علمی، اجتماعی) با فرض $\alpha = 1$:

محسابه احتمال کلاس ۰:

$$P\left(C_0\middle|(\text{اجتماعی ،علمی ،فرهنگی ،فناوری})\right) = P(C_0)\prod_{i=1}^{4} P(X_i|C_0)$$
$$P\left(C_0\middle|(\text{اجتماعی ،علمی ،فرهنگی ،فناوری})\right)$$
$$= 0.25\left(P\left(\text{اجتماعی}|C_0\right)\right)\left(P\left(\text{علمی}|C_0\right)\right)\left(P\left(\text{فرهنگی}|C_0\right)\right)\left(P\left(\text{فناوری}|C_0\right)\right)$$
$$= 0.25\left(\frac{1+1}{9+1*6}\right)\left(\frac{2+1}{9+1*6}\right)\left(\frac{2+1}{9+1*6}\right)\left(\frac{3+1}{9+1*6}\right) = 0.00035$$

محسابه احتمال کلاس ۱:

$$P\left(C_1\middle|(\text{اجتماعی ،علمی ،فرهنگی ،فناوری})\right) = P(C_1)\prod_{i=1}^{4} P(X_i|C_1)$$

$$P\left(C_1\middle|(\text{اجتماعی ،علمی ،فرهنگی ،فناوری})\right)$$
$$= 0.25\left(P\left(\text{اجتماعی}|C_1\right)\right)\left(P\left(\text{علمی}|C_1\right)\right)\left(P\left(\text{فرهنگی}|C_1\right)\right)\left(P\left(\text{فناوری}|C_1\right)\right)$$
$$= 0.25\left(\frac{2+1}{8+1*6}\right)\left(\frac{1+1}{8+1*6}\right)\left(\frac{1+1}{8+1*6}\right)\left(\frac{1+1}{8+1*6}\right) = 0.00015$$

احتمال کلاس ۰ بودن بیشتر است. **پس بازهم کلاس تست اول ۰ است.**

برای دومین داده تست (فناوری، فرهنگی، علمی، اجتماعی، ورزشی) با فرض $\alpha = 1$:

محسابه احتمال کلاس ۰:

$$P\left(C_0\middle|(\text{ورزشی ،اجتماعی ،علمی ،فرهنگی ،فناوری})\right)$$
$$= 0.25\left(P\left(\text{ورزشی}|C_0\right)\right)\left(P\left(\text{اجتماعی}|C_0\right)\right)\left(P\left(\text{علمی}|C_0\right)\right)\left(P\left(\text{فرهنگی}|C_0\right)\right)\left(P\left(\text{فناوری}|C_0\right)\right)$$
$$= 0.25\left(\frac{0+1}{9+1*6}\right)\left(\frac{1+1}{9+1*6}\right)\left(\frac{2+1}{9+1*6}\right)\left(\frac{2+1}{9+1*6}\right)\left(\frac{3+1}{9+1*6}\right) = 0.00035$$

محسابه احتمال کلاس ۱:

$$P\left(C_1\middle|(\text{ورزشی ،اجتماعی ،علمی ،فرهنگی ،فناوری})\right) = P(C_1)\prod_{i=1}^{4} P(X_i|C_1)$$

$$P\left(C_1\middle|(\text{ورزشی ،اجتماعی ،علمی ،فرهنگی ،فناوری})\right)$$
$$= 0.25\left(P\left(\text{ورزشی}|C_1\right)\right)\left(P\left(\text{اجتماعی}|C_1\right)\right)\left(P\left(\text{علمی}|C_1\right)\right)\left(P\left(\text{فرهنگی}|C_1\right)\right)\left(P\left(\text{فناوری}|C_1\right)\right)$$
$$= 0.25\left(\frac{0+1}{8+1*6}\right)\left(\frac{2+1}{8+1*6}\right)\left(\frac{1+1}{8+1*6}\right)\left(\frac{1+1}{8+1*6}\right)\left(\frac{1+1}{8+1*6}\right) = 0.00015$$

احتمال کلاس ۰ بودن بیشتر است. **پس کلاس تست دوم ۰ است.**

۱.ب تشخیص کلاس خبر:

این سوال را هم در قسمت الف جواب دادیم.

برای دومین داده تست (فناوری، فرهنگی، علمی، اجتماعی، ورزشی):

محسابه احتمال کلاس ۰:

$$P\left(C_0\middle|(\text{ورزشی ،اجتماعی ،علمی ،فرهنگی ،فناوری})\right)$$
$$= 0.25\left(P\left(\text{ورزشی}|C_0\right)\right)\left(P\left(\text{اجتماعی}|C_0\right)\right)\left(P\left(\text{علمی}|C_0\right)\right)\left(P\left(\text{فرهنگی}|C_0\right)\right)\left(P\left(\text{فناوری}|C_0\right)\right)$$
$$= 0.25\left(\frac{0+1}{9+1*6}\right)\left(\frac{1+1}{9+1*6}\right)\left(\frac{2+1}{9+1*6}\right)\left(\frac{2+1}{9+1*6}\right)\left(\frac{3+1}{9+1*6}\right) = 0.00035$$

محسابه احتمال کلاس ۱:

$$P\left(C_1\middle|(\text{ورزشی ،اجتماعی ،علمی ،فرهنگی ،فناوری})\right) = P(C_1)\prod_{i=1}^{4} P(X_i|C_1)$$

$$P\left(C_1\middle|(\text{ورزشی ،اجتماعی ،علمی ،فرهنگی ،فناوری})\right)$$
$$= 0.25\left(P\left(\text{ورزشی}|C_1\right)\right)\left(P\left(\text{اجتماعی}|C_1\right)\right)\left(P\left(\text{علمی}|C_1\right)\right)\left(P\left(\text{فرهنگی}|C_1\right)\right)\left(P\left(\text{فناوری}|C_1\right)\right)$$
$$= 0.25\left(\frac{0+1}{8+1*6}\right)\left(\frac{2+1}{8+1*6}\right)\left(\frac{1+1}{8+1*6}\right)\left(\frac{1+1}{8+1*6}\right)\left(\frac{1+1}{8+1*6}\right) = 0.00015$$

برای اینکه احتمال هیچ فیچری صفر نشود با استفاده از هموار سازی صورت و مخرج کسر احتمال هرکلمه به شرط کلاسش با مقادیری جمع می‌شود.

منبع:

https://blog.faradars.org/naive-bayes-classifier/

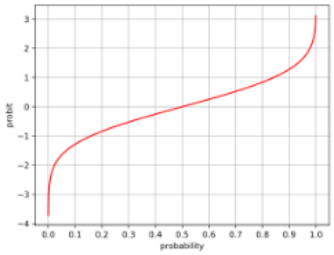
سوال دوم:

این سوال صرفا برای آموزش بود.

سوال سوم:

برای محاسبه ی احتمال منفی لگاریتم برای رگرسیون پرابیت باید تابع احتمالی که انتخاب کردیم تعریف کنیم:

$$\Phi(a) = \int_{-\infty}^{a} N(\theta|0,1)d\theta$$



این تابع قرار است خروجی مدل را تعیین کند پس باید a را بر حسب ورودی های x و پارامترهای مدل تعیین کنیم.

$$a = W^T X_i + b$$

Bو W پارامترهای مدل و X_i ورودی ها هستند.

$$\Phi(W^T X_i + b) = \int_{-\infty}^{W^T X_i + b} N(\theta|0,1)d\theta$$

سپس تابع ضرر احتمال منفی لگاریتم را تعریف میکنیم:

$$L(\theta) = -\sum_{i=1}^{n} (y_i log\hat{y}_{\theta,i} + (1-y_i)\log(1-\hat{y}_{\theta,i}))$$

در این رابطه y_i مقدار خروجی iام برای ورودی iام است. این رابطه برای طبقه بندی باینری است پس مقدار آن دو مقدار ۰ یا ۱ را دارد.

همچنین $\hat{y}_{\theta,i}$ بیانگر خروجی پیشبینی شده مدل است که قرار است با استفاده از تابع $\Phi(a)$ محاسبه شود.

در ادامه رابطه $\Phi(W^T X_i + b)$ را در $L(\theta)$ جایگذاری می‌کنیم تا خواسته سوال بدست بیاید.

$$L(\theta) = -\sum_{i=1}^{n} (y_i \log(\int_{-\infty}^{W^T X_i + b} N(\theta|0,1)d\theta) + (1-y_i)\log\left(1-(\int_{-\infty}^{W^T X_i + b} N(\theta|0,1)d\theta)\right))$$

n تعداد ورودیهاست.

چت جی پی تی:

پرسش: negative log likelihood for probit regression

https://en.wikipedia.org/wiki/Probit

پاسخ تمرین اول **مهدیه نادری: ۹۸۵۲۲۰۷۶**

سوال چهارم:

۴.الف دلیل استفاده از تابع فعالسازی در MLP:

بطور کلی در مسائل ساده که صرفت یک عدد خروجی داریم مثل رگرسیون میتوانیم از تابع فعالسازی استفاده نکنیم اما چون در MLP ما چند لایه پشت هم داریم اگر از تابع فعالسازی استفاده نکنیم کاری که انجام میدهیم معادل عملکرد یک لایه پرسپترون ولی با محاسبات بیشتر است و باعث میشود به نتیجه ی دلخواهی که با استفاده از عمیق کردن مدل قرار بود داشته باشیم نرسیم. که میتوانیم آن را اثبات کنیم. در زیر رابطه را برای دو لایه مینویسیم که قابل تعمیم برای لایه های بیشتر است.

پارامترهای لایه۱ : W1, b1 پارامترهای لایه۲: W2,b2

$$y1 = w1.x + b1$$
$$y2 = w2.y1 + b2 = w2.(w1x + b1) + b2 = w2.w1.x + w2.b1 + b2 = Wx + B$$

که در آن $W = w2.w1$ و $B = w2.b1 + b2$ هستند و یعنی نتیجه معادل تابع خطی است. برای nلایه با صرف نظر کردن از پارامتر ثابتb در هر لایه داریم:

$$y_n = w_n w_{nn-1} \dots w_2 w_1 x$$

که با هدف ما از استفاده از MLP برای مدل مغایرت دارد.

در شبکه‌های عصبی چندلایه(MLP) ، توابع فعال‌سازی یا توابع انتقال (Activation Functions) برای اضافه کردن نوعی غیرخطیت به مدل استفاده می‌شوند. این توابع نقش بسیار مهمی در افزایش قدرت انعطاف‌پذیری و توانایی یادگیری عمیق (Deep Learning) شبکه‌های عصبی دارند. در ادامه، دلیل استفاده از توابع فعال‌سازی در MLP را بررسی می‌کنیم:

1. **غیرخطیت مدل :**توابع فعال‌سازی افزودن غیرخطیت به مدل کمک می‌کنند. اگر از توابع خطی به عنوان توابع فعال‌سازی استفاده شود، مدل به یک مجموعه خطی از لایه‌ها تبدیل می‌شود. این باعث می‌شود که توانایی مدل در نمایش الگوهای پیچیده و غیرخطی محدود شود. با اضافه کردن توابع غیرخطی به لایه‌های مخفی، MLPقادر است الگوهای پیچیده‌تر را نمایش دهد.
2. **ضرب‌های وزن نزدیک به صفر :**در فرآیند یادگیری، وزن‌های لایه‌های مخفی به روز رسانی می‌شوند. اگر از توابع خطی به عنوان توابع فعال‌سازی استفاده شود، در صورت افترا به وزن‌های کم، مشتق تابع خطی نسبت به ورودی همیشه ثابت است و این می‌تواند منجر به آن شود که وزن‌ها نزدیک به صفر شوند، که در نهایت باعث از دست رفتن توانایی یادگیری مدل می‌شود. از توابع غیرخطی مانند ReLU (Rectified Linear Unit) که دارای مشتق غیرصفر است، می‌توان برای حل این مشکل استفاده کرد.
3. **جلوگیری از اشباع :**در توابع فعال‌سازی خاصی، مانند sigmoid وtanh ، وقتی ورودی به مقدارهای خیلی بزرگ) در صورت استفاده از (sigmoid یا خیلی کوچک) در صورت استفاده از (tanh نزدیک می‌شود، مشتق توابع به سمت صفر می‌رود. این اتفاق باعث می‌شود که شبکه به شدت اشباع شود و از دست رفتن اطلاعات در فرآیند یادگیری را ایجاد کند. توابعی مانند ReLU این مشکل را در حداکثر میزان کاهش می‌دهند.

به طور کلی، استفاده از توابع فعال‌سازی غیرخطی در MLP به شبکه اجازه می‌دهد تا الگوهای پیچیده‌تری را یاد بگیرد و مشکلات مربوط به اشباع و از دست رفتن توانایی یادگیری را کاهش دهد.

۴.ب آیا هرتابع غیرخطی را میتوان به عنوان تابع فعالسازی استفاده کرد؟:

ازجهتی میتوان گفت خیرهر تابع غیرخطی نمی‌تواند به عنوان تابع فعال‌سازی در شبکه‌های عصبی مصنوعی استفاده شود. مثلا ما هیچوقت از تابع x^2 برای فعالسازی نورن استفاده نمیکنیم چون مفهوم خاصی را منتقل نمیکند فقط ورودی را به توان دو میرساند یعنی میتواند شدت تاثیر ورودی و وزن را بیشتر کند و مقادیر را مثبت کند که برای مثبت کردن مقادیر توابع بهتری داریم بدی این تابع این است که اثر منفی مقادیر ورودی را درنظرنمیگیرد. البته مسائلی هستند که این تابع بتواند در حل آن کمک کند اما آنها را میتوان با توابع پرطرفدار دیگر هم حل کرد. در تئوری می‌توانید از هر تابع غیرخطی‌ای به عنوان تابع فعال‌سازی استفاده کنید، اما انتخاب تابع فعال‌سازی بر اساس خصوصیات مسئله

و رفتار مدل در فرآیند یادگیری بسیار مهم است .استفاده از توابع فعال‌سازی در شبکه‌های عصبی معمولاً مشخص می‌کند که چگونه یک نورون به ورودی‌ها وزن دهد و خروجی تولید کند. برخی از توابع فعال‌سازی معمولی عبارتند از:

1. **ReLU (Rectified Linear Unit)**: یک تابع غیرخطی است که در صورتی که ورودی مثبت باشد، خود ورودی را خروجی می‌دهد، و در صورتی که ورودی منفی باشد، صفر را خروجی می‌دهد.
2. **Sigmoid Function**: یک تابع لگیستیک (سیگموئید) که ورودی را به مقداری بین ۰ و ۱ نگاشت می‌کند.
3. **Tanh Function**: یک تابع لگیستیک دیگر که ورودی را به مقداری بین -۱ و ۱ نگاشت می‌کند .

استفاده از این توابع معمولاً به دلیل ویژگی‌های خاص خود، مانند افزایش غیرخطیت و جلوگیری از مشکل اشباع در شبکه‌های عصبی، مرسوم است. انتخاب تابع فعال‌سازی بستگی به مسئله مورد نظر و ویژگی‌های داده دارد. در برخی موارد، امکان استفاده از توابع فعال‌سازی خاص دیگر نیز وجود دارد، اما انتخاب یک تابع مناسب بر اساس ویژگی‌های مسئله و شرایط می‌تواند به کارایی و عملکرد مدل کمک کند.

منبع: https://blog.faradars.org/%D8%AA%D8%A7%D8%A8%D8%B9-%D9%81%D8%B9%D8%A7%D9%84%D8%B3%D8%A7%D8%B2%DB%8C-%D8%AF%D8%B1-%D8%B4%D8%A8%DA%A9%D9%87-%D9%87%D8%A7%DB%8C-%D8%B9%D8%B5%D8%A8%DB%8C/

سوال پنجم:

۵.الف توضیح و مقایسه توابع فعالسازی:

تابع سیگموید:

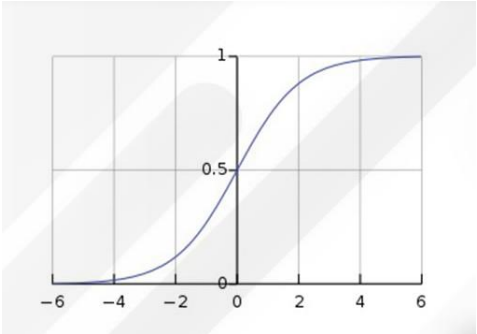
این تابع یک منحنی S شکل است. زمانی‌که می‌خواهیم خروجی مدل احتمال باشد، از تابع سیگموید استفاده می‌کنیم؛ چون تابع سیگموید مقادیر را به بازه صفر تا ۱ می‌برد و احتمالات هم میان همین بازه قرار دارند.

مزایا

- این تابع تمایزپذیر (Differentiable) است؛ یعنی در هر قسمت از منحنی می‌توانیم شیب میان دو نقطه را حساب کنیم.
- از آنجا که این تابع مقادیر را میان صفر و یک قرار می‌دهد، نوعی عادی‌سازی را برای خروجی هر نورون انجام می‌دهد.

معایب

- با محوشدگی گرادیان (Vanishing Gradient) مقادیر بسیار بزرگ یا بسیار کوچکx ، مشتق بسیار کوچک می‌شود و درواقع شبکه دیگر آموزش نمی‌بیند و پیش‌بینی‌هایش در خروجی ثابت می‌ماند.
- به‌دلیل مشکل محوشدگی‌گرادیان، تابع سیگموید هم‌گرایی کند دارد.
- خروجی تابع سیگموید صفرمحور (Zero-Centered) نیست؛ این امر کارایی به‌روزرسانی وزن‌ها را کم می‌کند.
- از آنجا که این تابع عملیات نمایی (Exponential Operations) دارد، می‌توان گفت هزینه‌ی محاسباتی بالایی دارد و کندتر پیش می‌رود.



معادله: $f(x) = s = 1/(1+e^{-x})$

پاسخ تمرین اول

مهدیه نادری: ۹۸۵۲۲۰۷۶

تابع softmax:

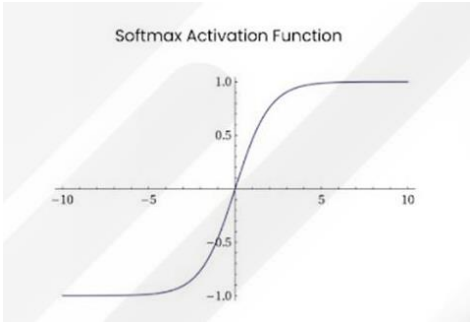
این تابع فعالساز از جمله توابع فعالساز (Activation Functions) است که در طبقه‌بندی‌های چندکلاسه استفاده می‌شود. زمانی‌که احتیاج داشته باشیم در خروجی احتمال عضویت بیشتر دو کلاس را پیش‌بینی کنیم، می‌توانیم به‌سراغ این تابع برویم. تابع سافت‌مکس تمامی مقادیر یک بردار با طول K را به بازه‌ی صفر تا ۱ می‌برد، به‌طوری که جمع تمامی مقادیر این بردار با هم ۱ می‌شود. این تابع برای نورون‌های لایه‌ی خروجی استفاده می‌شود؛ زیرا در شبکه‌های عصبی در آخرین لایه (خروجی) به طبقه‌بندی ورودی‌ها در کلاس‌های مختلف نیاز داریم.

مزایا

- این تابع قابلیت استفاده در تسک های چندکلاسه را دارد. خروجی هر کلاس را میان صفر تا ۱ عادی‌سازی می‌کند؛ سپس آن‌ها را بر مجموعه‌شان تقسیم و احتمال عضویت مقادیر ورودی را در هر کلاس به ما در خروجی ارائه می‌کند.

معایب

- مقدار گرادیان برای مقادیر منفی صفر است؛ به‌این معنا که وزن‌ها در حین عملیات پس‌انتشار به‌روزرسانی نمی‌شوند و این می‌تواند مشکل مرگ نورون را ایجاد کند.



$$f(x) = e^{x_i} / (\Sigma_{j=0} e^{x_i})$$

تابع RELU:

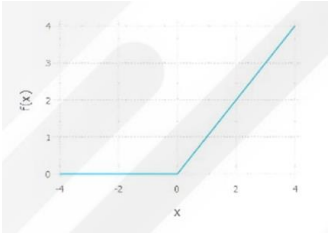
تابع فعالساز واحد یک‌سوشده‌ی خطی در زمینه‌ی یادگیری عمیق بسیار مشهور است و در بیشتر مواقع استفاده می‌شود. این تابع به‌این صورت عمل می‌کند که مقادیر منفی (زیر صفر) را صفر و مقادیر مثبت (بیشتر از صفر) و مقادیر برابر با صفر را همان مقدار خودش در نظر می‌گیرد.

مزایا:

- از نظر محاسباتی بسیار کارآمد است و به شبکه اجازه می‌دهد به‌سرعت همگرا شود؛ زیرا رابطه‌ی آن خطی است و به‌همین دلیل، در مقایسه با تابع‌های سیگموید وTanh ، سریع‌تر است.

معایب:

- مشکل مرگ نورون یا مرگ ReLU دارد؛ یعنی زمانی‌که ورودی صفر یا نزدیک به صفر باشد، تابع ReLU دیگر عملکردی ندارد و به‌بیان دیگر، می‌میرد. در این صورت، مقدار گرادیان تابع صفر می‌شود و شبکه نمی‌تواند عملیات پس انتشار (Backpropagation)را انجام دهد و آموزش ببیند.
- خروجی این تابع صفر یا مثبت است و این یعنی صفرمحور نیست.



$$f(x) = a = \max(0,x)$$

پاسخ تمرین اول
مهدیه نادری: ۹۸۵۲۲۰۷۶

تابع فعالساز Leaky ReLU :

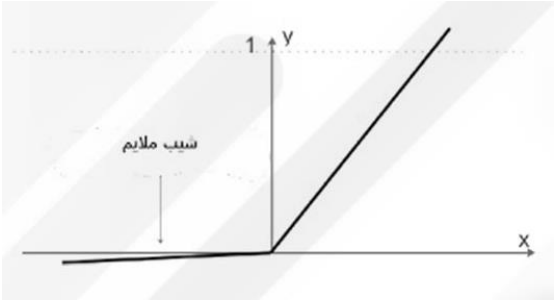
این تابع فعالساز برای حل مشکل اصلی تابع ReLU ارائه شده است. در شکل بعدی نمایی از این تابع را مشاهده می‌کنیم:

مزایا

- از مشکل مرگ ReLU جلوگیری می‌کند. این تابع یک شیب مثبت ملایم به‌سمت مقادیر منفی دارد که این امر باعث می‌شود عملیات پس انتشار (Backpropagation) حتی برای مقادیر منفی هم انجام شود.

معایب

- برای مقادیر منفی پیش‌بینی (خروجی) ثابتی را ارائه نمی‌کند.
- در حین عملیات انتشار روبه‌جلو (Forward Propagation) اگر نرخ یادگیری (Learning Rate) را خیلی بالا در نظر بگیریم، مشکل مرگ نورون‌ها را رقم می‌زند.



$f(x)= a = max(0.01x, x)$

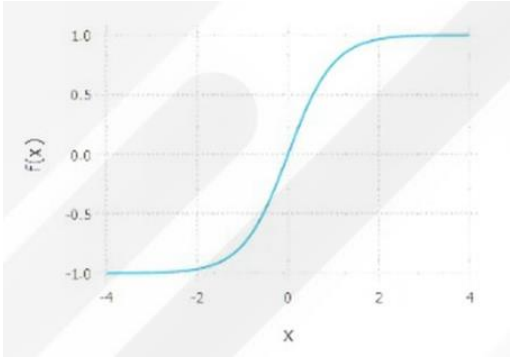
تابع tanh:

مزایا

- این تابع صفرمحور است؛ بنابراین به مدل کمک می‌کند تا مقادیر ورودی منفی، خنثی و مثبت داشته باشد؛ به‌عبارت دیگر، مقادیر منفی، به‌شدت منفی و مقادیر صفر در گراف تانژانت هایپربولیک نزدیک به صفر نگاشت می‌شوند.
- تابع آن یکنواخت(Monotonic) ، اما مشتق آن یکنواخت نیست.

معایب

- محوشدگی گرادیان
- هم‌گرایی کند



$f(x) = a =\tanh(x) =(e^x - e^{-x})/(e^x +e^{-x})$

- تابع سیگموید (Sigmoid) در مسائل طبقه‌بندی معمولاً خیلی خوب عمل می‌کند.
- توابع سیگموید (Sigmoid) و تانژانت هایپربولیک(Tanh) ، به‌دلیل مشکل محوشدگی گرادیان، در بعضی مواقع استفاده نمی‌شوند.
- تابع فعالساز واحد یک‌سوشده‌ی خطی (ReLU) بیشتر از باقی استفاده می‌شود و نتایج خوبی را در خروجی ارائه می‌کند.

- تابع فعالساز واحد یک‌سوشده‌ی خطی (ReLU) فقط در لایه‌های نهان (Hidden Layers) استفاده می‌شود.
- اگر با مشکل مرگ نورون در شبکه مواجه هستیم، تابع Leaky ReLU می‌تواند گزینه‌ی بسیار خوبی باشد.
- تابع تانژانت هایپربولیک(Tanh) ، به‌دلیل مشکل مرگ نورون، کمتر استفاده می‌شود.

منبع:https://cafetadris.com/blog/%D8%AA%D9%88%D8%A7%D8%A8%D8%B9-%D9%81%D8%B9%D8%A7%D9%84%D8%B3%D8%A7%D8%B2-activation-functions/

۵.ب

فرمول تابع سیگموید را $\sigma(a) = \frac{1}{1+\exp(-a)}$ داریم ، برای قسمت تابع exp کتابخانه numpy را ایمپورت و سپس آن را در خروجی یک تابع با ورودی x ریترن کردیم:

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

سپس نام تابع طراحی شده را در بخش کد زیر قرار دادیم تا با تابع پیاده سازی شده در پایتورچ مقایسه شود.

```
result_sigmoid, output_sigmoid = test(test_data_torch, sigmoid, nn.Sigmoid())
```

برای بقیه توابع خواسته شده هم به همین صورت فرمول ها را در توابع پیاده سازی و تست کردیم.

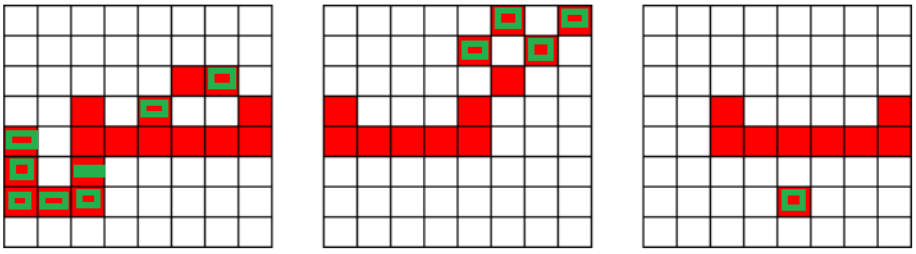
۵.ج

معماری MLP مناسب:

- تعداد لایه: ۳ تعداد نورون لایه ۱: ۶۴ تعداد نورون لایه میانی : ۱۲۸ تعداد نورون لایه خروجی: ۳

لایه ورودی: چون ابعاد ورودی ها ۸ در ۸ است ۶۴ نورون دارد.

لایه میانی : اگر مدل صرفا باید همین ۳ حرف را تشخیص دهد و حروف دیگر را لازم نیست با دقت در تصاویر میبینیم که این تصاویر در نقاط سبز رنگ باهم تفاوت دارند. یعنی مثلا اگر مدل فعال بودن خانه اول از بالا و پایین را تشخیص دهد میتواند حرف گ را پیدا کند. هر تصویر حداقل یک مولفه ی یونیک را دارد پس با تعداد ایپاک خیلی بالا ما با فقط ۱ نورون میتوانیم مسئله را حل کنیم.



اما برای تشخیص همه ی حروف ۸ در ۸ یعنی ۶۴ نورون لازم است که حداکثر تعداد است.

من برای صرفه جویی در زمان (تعداد ایپاک) و اینکه مشکل حافظه نداریم ۱۲۸ نورون انتخاب میکنم.

لایه خروجی: ۳ حالت خروجی داریم و من هم ۳ نورون در نظر میگیرم که هرکدام فعال شد یعنی آن کلاس مربوط به تصویر است البته میتوان یک نورون گرفت و ازتابع سافتمکس برای ۳ کلاس استفاده کرد.

- تابع فعالسازی:

برای لایه میانی: از ReLU استفاده میکنیم که معمولا استفاده میشود صرفا برای غیرخطی کردن خروجی این لایه جهت آپدیت بهتر پارامترها و پیداکردن پترن ها ی پیچیده تر

برای لایه خروجی: از تابع خاصی استفاده نمیکنیم چون تابع ضرری که انتخاب کردیم ما را از تابع فعالسازی بینیاز میکند.

- تابع ضرر:

برای حل مسائل چند کلاسه از کراس-انتروپی استفاده میکنیم. این تابع softmax را با negative log-likelihood ترکیب میکند. کمک میکند تا مدل بیشترین احتمال را به کلاس درست بدهد.

در مسائل چند کلاسی خروجی One-hot کد میشود و با یک وکتور نمایش داده میشود مثلا در اینجا میتوانیم خروجی را برای کلاس (ب) ۱۰۰ ، (گ) کلاس ۰۱۰ و کلاس (ص) ۰۰۱ و خروجی تابع ضرر هم یک وکتور ۳ تایی است که در آن احتمال هر کلاس در یک درایه ذخیره میشود.

فرمول تابع ضرر برای مسئله چند کلاسه :

$$L(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

۵.د پیاده سازی مدل MLP:

کدهای مربوطه در نوتبوک Q5.ipynb ابتدا کتابخانه های لازم را ایمپورت کردیم.

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

from PIL import Image
import os
import torch
import torch.nn.functional as F
import torchvision.transforms as transforms
```

سپس پیش پردازش هایی که لازم بود انجام دادیم. مثلا درست است که تصاویری که داشتیم به صورت یک جدول ۸در۸ دیده میشدند اما ابعاد واقعی تصویر بیشتر هستند و هر خانه ی جدول یک پیکسل نبود. برای همین لازم بود بعد از خواندن تصویر آن را grayscale کرده وابعداد را ۸در۸ کنیم و برای کار با پایتورچ به تنسور تبدیل کنیم. برای این تبدیل قطعه کد زیر را نوشتیم:

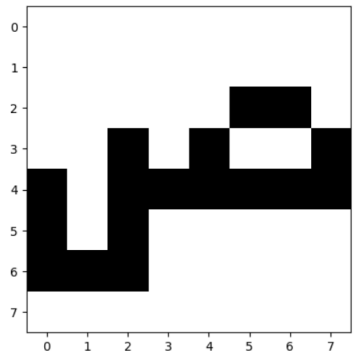
```
# Define a transformation to preprocess the images
transform = transforms.Compose([
    transforms.Grayscale(),  # Convert the image to grayscale (assuming your images are colored)
    transforms.Resize((8, 8)),  # Resize the image to 8x8
    transforms.ToTensor()  # Convert the image to a PyTorch tensor
])
```

سپس یک لیست از نام تصاویر ساختیم، با یک حلقه روی تصاویر، هر یک را میخوانیم و در متغیر ذخیره میکنیم سپس تبدیل را اعمال میکنیم و بعد به حالت باینری تبدیل میکنیم و بعد در یک لیست ذخیره میکنیم. در نهایت لیست را برای استفاده های بعدی به استک تبدیل کردیم.

```
# Load and preprocess your images
image_paths = ["Q5_1.png", "Q5_2.png", "Q5_3.png"]
images = []

for path in image_paths:
    # Load the image using PIL
    img = Image.open(os.path.join(path))
    # Apply the transformation
    img_tensor = transform(img)
    #convert to binary
    img_tensor = torch.where(img_tensor[0] > 0.5, torch.tensor(1.0), torch.tensor(0.0))
    # Add the preprocessed image tensor to the list
    images.append(img_tensor)

# Create a batch tensor by stacking the image tensors
batch_tensor = torch.stack(images)

#show one image
print(batch_tensor[0])
plt.imshow(batch_tensor[0].squeeze(), cmap='gray')
plt.show()
```

یک نمونه از تنسور باینری و نمایش تصویری آن را در زیر آوردیم:

```
tensor([[1., 1., 1., 1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1., 0., 0., 1.],
        [1., 1., 0., 1., 0., 1., 1., 0.],
        [0., 1., 0., 0., 0., 0., 0., 0.],
        [0., 1., 0., 1., 1., 1., 1., 1.],
        [0., 0., 0., 1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1., 1., 1., 1.]])
```



در ادامه داده های آموزشی و تست را مشخص کردیم و لیبل ها را هم خودمان دستی به صورت ۰و ۱و ۲ درنظر گرفتیم. هر سه تصویر و لیبل ها هم برای تست و هم آموزش درنظرگرفته شد چون داده دیگری نداشتیم.

سپس کلاس MLP را با استفاده از کتابخانه pytorchپیاده سازی کردیم.

```
# Create a custom MLP model
class MLP(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x
```

در آن لایه ورودی و میانی را به صورت خطی تعریف میکنیم که البته برای لایه اول تابع فعالسازی reluرا درنظرگرفتیم.

در ادامه تعداد نورون هر لایه و تعداد کلاسها را همانطور که در قسمت الف تعیین کردیم فرض کردیم همچنین از بهینه ساز Adam و نرخ یادگیری ۰.۰۰۱ استفاده کردیم. تابع ضرر را هم کراس انتروپی تعیین کردیم. دیتالودرهای ترین و تست را هم با همان دیتا ها ساختیم. با

توجه به کم بودن دیتا بچسایز معنا ندارد اما به هرحال ۳ درنظر گرفتیم. با تعداد ایپاک ۱۰۰ مدل را آموزش دادیم. در هر ایپاک به ازای هر تصویر ورودی بعد از محاسبه خروجی تابع ضرر را حساب کردیم، بکپروپگیشن روی مدل انجام دادیم و تابع بهینه سازی را اعمال کردیم و ضرر مجموع داده ها را محاسبه کرده و در پایان ایپاک حساب کردیم.

```
# Initialize the model, loss function, and optimizer
input_size = 64  # 8x8 images flattened
hidden_size = 128  # Number of neurons in the hidden layer
num_classes = 3  # Three classes for three alphabets (assuming 3 classes)

model = MLP(input_size, hidden_size, num_classes)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Create DataLoader for training and testing datasets
train_dataset = TensorDataset(x_train_tensor, y_train_tensor)
test_dataset = TensorDataset(x_test_tensor, y_test_tensor)
train_loader = DataLoader(train_dataset, batch_size=3, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=3, shuffle=False)

# Training the model
num_epochs = 100
for epoch in range(num_epochs):
    total_loss = 0
    for i, (inputs, labels) in enumerate(train_loader):
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
    print(f'Epoch {epoch + 1}, Loss: {total_loss / len(train_loader)}')

# Evaluate the model
model.eval()
```

در نهایت مدل را اعتبار سنجی کردیم و با بدست آوردن برچسب هایی که مدل به ازای داده ها پیشبینی میکند و مقایسه با برچسب درست دقت مدل را که ۱۰۰ شد، بدست آوردیم.

```
# Evaluate the model
model.eval()
correct = 0
total = 0
with torch.no_grad():
    for inputs, labels in test_loader:
        outputs = model(inputs)
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

accuracy = 100 * correct / total
print(f'Accuracy on test set: {accuracy:.2f}%')
```

```
Epoch 83, Loss: 0.04167379066348076
Epoch 84, Loss: 0.04034271836280823
Epoch 85, Loss: 0.039083559066057205
Epoch 86, Loss: 0.037880588322877884
Epoch 87, Loss: 0.036738861352205276
Epoch 88, Loss: 0.03564726188778877
Epoch 89, Loss: 0.034606870263814926
Epoch 90, Loss: 0.033612534403800964
Epoch 91, Loss: 0.03266273811459541
Epoch 92, Loss: 0.03176015987992287
Epoch 93, Loss: 0.030895931646227837
Epoch 94, Loss: 0.03006645292043686
Epoch 95, Loss: 0.029277952387928963
Epoch 96, Loss: 0.02852175198495388
Epoch 97, Loss: 0.027793658897280693
Epoch 98, Loss: 0.027093147858977318
Epoch 99, Loss: 0.026423586532473564
Epoch 100, Loss: 0.02578018046915531
Accuracy on test set: 100.00%
```

پاسخ تمرین اول **مهدیه نادری: ۹۸۵۲۲۰۷۶**

سوال ششم:

یک شبکه عصبی چند لایه را درنظر بگیرید که برای دسته بندی دو کلاس مورد استفاده قرار میگیرد. خروجی نورون آخر را z بگیرید و خروجی شبکه عصبی با سیگموید روی RELU(z) بدست می آید. MLP از یک آستانه ۰,۵ استفاده میکند و خروجی های بزرگتر یا مساوی ۰,۵ را به عنوان کلاس ۱ و خروجی های کمتر از ۰,۵ را به عنوان کلاس ۰ دسته بندی میکند.
در استفاده از این شبکه عصبی چه مشکلات و چالش هایی با توجه به ترکیب توابع فعال ساز و آستانه ۰,۵ ممکن است وجود داشته باشد؟

در این شبکه عصبی، ترکیب توابع فعال‌ساز) سیگموید و (ReLU و آستانه ۰,۵ می‌تواند باعث بروز مشکلات و چالش‌ها در دسته‌بندی شود:

1. **Gradients Vanishing Problem**:

وقتی که شبکه برای تعداد لایه‌های زیادی طراحی شده باشد، ممکن است مشکل کاهش گرادیان در سیگموید وجود داشته باشد. در زمان پس‌انتقال خطا(backpropagation) ، گرادیان‌ها به سرعت به صفر نزدیک می‌شوند، که موجب کم شدن فعالیت نورون‌ها و یادگیری نادرست می‌شود.

2. **ReLU Activation and Dead Neurons**:

اگر خروجی نورون آخر (z) عدد مثبت بزرگی باشد، از ReLU به عنوان فعال‌ساز استفاده می‌شود و به عنوان ورودی سیگموید داده میشود و خروجی سیگموید صفر می‌شود. این موضوع می‌تواند منجر به وجود نورون‌های مرده (dead neurons) در شبکه شود که به معنایی مفهومی در فرآیند یادگیری شرکت نمی‌کنند.

3. **Threshold Selection Sensitivity**:

انتخاب آستانه ۰,۵ برای تصمیم‌گیری در مورد کلاس‌ها ممکن است حساسیت‌هایی داشته باشد. اگر داده‌ها یک توزیع خاص داشته باشند، انتخاب اشتباهی از آستانه می‌تواند منجر به اشتباهات دسته‌بندی شود. مثلا اگر تعداد داده های کلاس یک خیلی بیشتر از نصف داده ها باشد با این انتخاب اکثر خروجی های مدل برای کلاس۱ پیشبینی میشوند چون مدل این داده ها را بیشتر دیده است و یکی از راه حل های این مشکل افزایش این آستانه است.

4. **Non-Smoothness of ReLU**:

ReLUیک تابع غیر صاف است. این ویژگی می‌تواند در برخی از مسائل به مشکلاتی در بهینه‌سازی و یادگیری منجر شود.

5. **Interpretability**:

استفاده از ترکیبی از سیگموید و ReLU با آستانه ۰,۵ ممکن است تعبیر مدل را سخت‌تر کند. درک از نحوه‌ی تصمیم‌گیری در مورد کلاس‌ها ممکن است پیچیده‌تر شود.

6. **آموزش:**

تابع فعالسازی relu برای تمام مقادیر منفی صفر میشود و مشتق تابع سیگموید در نقطه صفر بیشترین مقدار را دارد و سریعا از این نقطه آپدیت میشود و مقدار آپدیت هم زیاد است و اگر تابع به اندازه ی کافی پیچیده نباید و انتشار خطا به لایه های قبل همینقدر زیاد باشد میتواند منجر به همگرا نشدن مدل شود. همچنین برای تمام مقادیر منفی z خروجی نهایی مدل ۰.۵ و کلاس ۰ میشود که این خوب است ولی نوع آپدیت مناسب نیست.

برای حل این مشکلات، ممکن است نیاز باشد تا روش‌های بهینه‌تری برای انتخاب توابع فعال‌ساز، آستانه، یا حتی ساختار شبکه مورد استفاده قرار گیرد. انتخاب مناسب این پارامترها معمولاً نیازمند آزمایش و خطا در دسته‌بندی داده‌های خاص است.

منبع: اسلایدهای کلاس و چت جی پی تی

پاسخ تمرین اول **مهدیه نادری: ۹۸۵۲۲۰۷۶**

سوال هفتم:

۷.الف تفاوت یادگیری ماشین و یادگیری عمیق:

1.یادگیری ماشین:(Machine Learning)

- **پردازش فیچر :(Feature Engineering)** در یادگیری ماشین، طراحی و انتخاب ویژگی‌ها (فیچرها) از دسته کارهای اساسی است. انتخاب ویژگی‌های مناسب می‌تواند تأثیر زیادی در کارایی مدل داشته باشد.
- **استفاده از مدل‌های ساده :**در بسیاری از مسائل یادگیری ماشین، از مدل‌های ساده‌تر مانند رگرسیون لجستیک، ماشین‌های بردار پشتیبانی و درخت‌های تصمیم استفاده می‌شود.
- **نیاز به داده‌های کمتر :**معمولاً برای مدل‌های یادگیری ماشین، داده‌های کمتری برای آموزش مورد نیاز است.

2.یادگیری عمیق:(Deep Learning)

- **پیچیدگی مدل‌ها :**در یادگیری عمیق، از مدل‌های عمیق مانند شبکه‌های عصبی چند لایه استفاده می‌شود. این مدل‌ها دارای تعداد زیادی لایه هستند که به صورت خودکار ویژگی‌ها را استخراج می‌کنند.
- **پردازش اتوماتیک ویژگی‌ها :**در یادگیری عمیق، مدل‌ها قادرند ویژگی‌های مورد نیاز را به صورت خودکار از داده‌ها استخراج کنند، بدون نیاز به پیش‌پردازش دستی.
- **نیاز به داده‌های بیشتر :**برای آموزش مدل‌های عمیق، نیاز به داده‌های بیشتری نسبت به مدل‌های سنتی مانند یادگیری ماشین وجود دارد.

خلاصه :یادگیری ماشین و یادگیری عمیق در واقع دو رویکرد متفاوت به حل مسائل یادگیری ماشین هستند. یادگیری عمیق معمولاً برای مسائل پیچیده‌تر و با داده‌های بزرگ مورد استفاده قرار می‌گیرد، در حالی که یادگیری ماشین ممکن است برای مسائل کوچک‌تر و با داده‌های کمتر مناسب باشد.

۷.ب

در یک مسئله دسته‌بندی، لایه ۱۱ به خروجی نهایی نسبت به لایه ۷ نزدیک‌تر است. دلیل این امر این است که لایه‌های عمیق‌تر معمولاً ویژگی‌های پیچیده‌تری از داده‌ها استخراج می‌کنند که این ویژگی‌ها به تصمیم‌گیری در مورد دسته‌بندی کمک می‌کنند.
در یک شبکه عمیق، لایه‌های ابتدایی (مانند لایه ۷) ویژگی‌های مبتنی بر جزئیات و ویژگی‌های سطح پایین‌تری را استخراج می‌کنند، در حالی که لایه‌های عمیق‌تر (مانند لایه ۱۱) ویژگی‌های انتزاعی‌تر و پیچیده‌تری از داده‌ها استخراج می‌کنند. این ویژگی‌های پیچیده‌تر به مدل کمک می‌کنند که الگوهای پیچیده‌تری را در داده‌ها تشخیص دهد و برای دسته‌بندی بهتر عمل کند.
بنابراین، در مسائل دسته‌بندی، لایه‌های عمیق‌تر (مانند لایه ۱۱) معمولاً به خروجی نهایی نزدیک‌تر هستند و می‌توانند نقش مهمی در بهبود دقت و کارایی دسته‌بندی مدل داشته باشند.

۷.ج

بنظرم هر دو ویژگی عمیق و وسیع بودن شبکه میتواند کمک مهمی بکند. در هرصورت این مسئله با یک لایه عمیق هم با قدت خوبی قابل حل است اما چون کارایی مدنظر سوال است میتوان گفت با شبکه های عمیق تر میتوان سریعتر سوال را حل کرد. در این نوع شبکه‌ها،

پاسخ تمرین اول

مهدیه نادری: ۹۸۵۲۲۰۷۶

لایه‌های عمیق به معنای استخراج ویژگی‌های پیچیده از داده‌ها و لایه‌های وسیع به معنای ایجاد فضای نورونی بزرگ‌تر و انتزاعی‌تر برای تقریب توابع هستند. این ویژگی‌ها معمولاً به بهترین عملکرد در تقریب توابع پیچیده از داده‌ها منجر می‌شوند.

۷.د

مزایا:

1. **نمایندگی قوی‌تر:**
   - با افزودن لایه‌های بیشتر، شبکه قادر به یادگیری نمایندگی‌های پیچیده‌تر و انتزاعی‌تر از داده‌ها می‌شود.
2. **یادگیری ویژگی‌های سطح بالا:**
   - لایه‌های عمیق‌تر می‌توانند ویژگی‌های سطح بالا و پیچیده‌تری از داده‌ها را استخراج کنند، که می‌تواند به بهبود عملکرد در مسائل پیچیده کمک کند.
3. **منعطف‌تر بودن در یادگیری:**
   - با لایه‌های بیشتر، شبکه می‌تواند الگوها و اطلاعات مختلفی از داده‌ها یاد بگیرد و این امکان را فراهم کند که برای مسائل مختلف مناسب باشد.
4. **انجام یادگیری انتزاعی:**
   - لایه‌های بیشتر امکان ایجاد انتزاع بالاتری از داده‌ها و اجسام مختلف را دارند.

معایب:

1. **پیچیدگی مدل:**
   - با افزودن لایه‌های بیشتر، مدل پیچیده‌تر می‌شود و ممکن است زمان و منابع بیشتری برای آموزش و استفاده نیاز باشد.
2. **آموزش نیازمند داده بیشتر:**
   - شبکه‌های عمیق با تعداد لایه‌های بیشتر نیاز به حجم داده بیشتری برای آموزش دارند تا از اورفیتینگ جلوگیری شود.
3. **ممکن است به اورفیتینگ بیفزاید:**
   - در صورتی که تعداد لایه‌ها بیشتر از حد نیاز باشد، مدل ممکن است به اورفیتینگ بیفزاید و در مقابل داده‌های جدید نتواند به خوبی عمل کند.
4. **پیچیدگی در تنظیم پارامترها:**
   - شبکه‌های عمیق با تعداد لایه‌های بیشتر نیاز به تنظیم و بهینه‌سازی پارامترهای بیشتری دارند که این می‌تواند یک فرآیند زمان‌بر و پیچیده باشد.