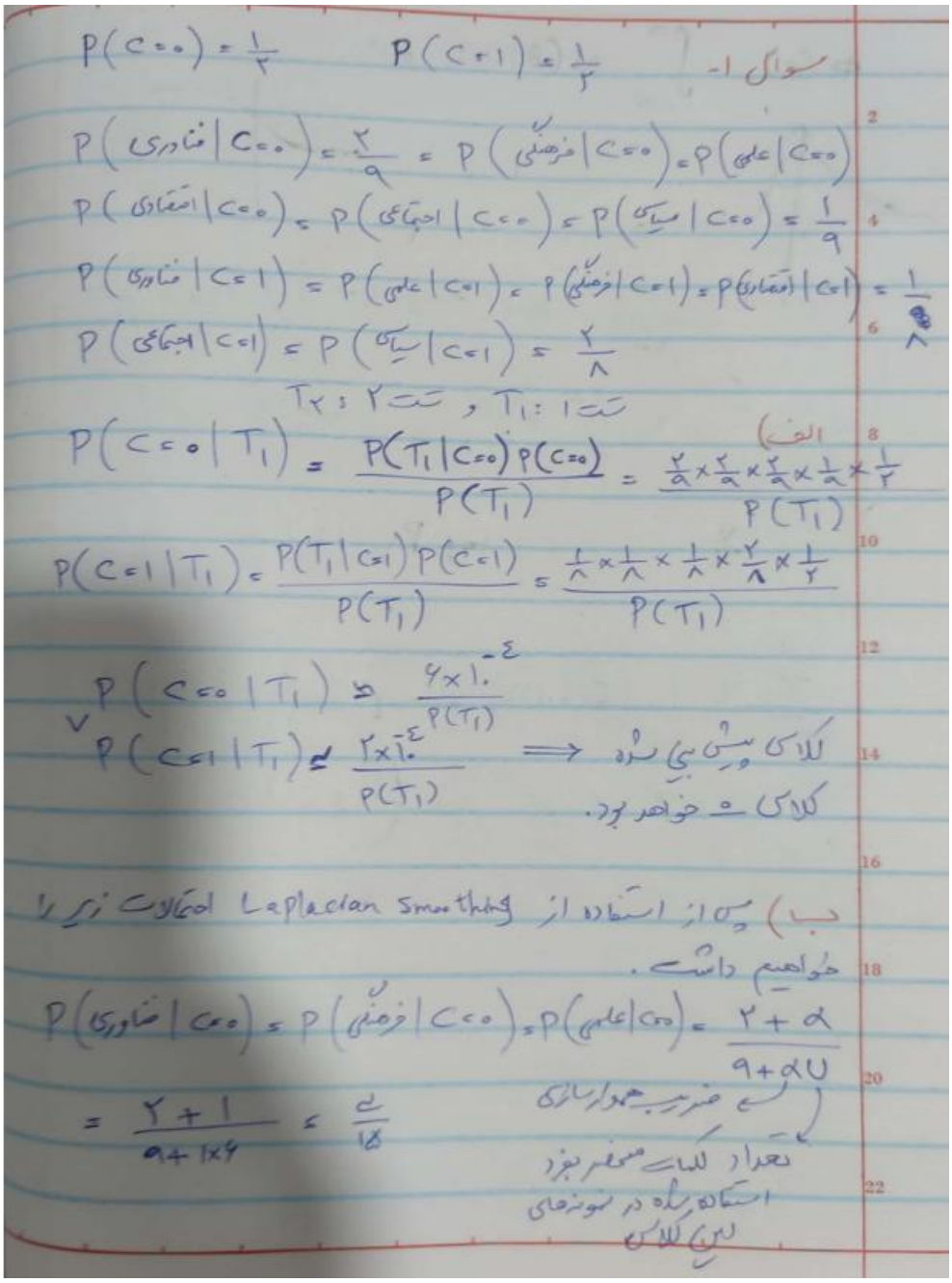
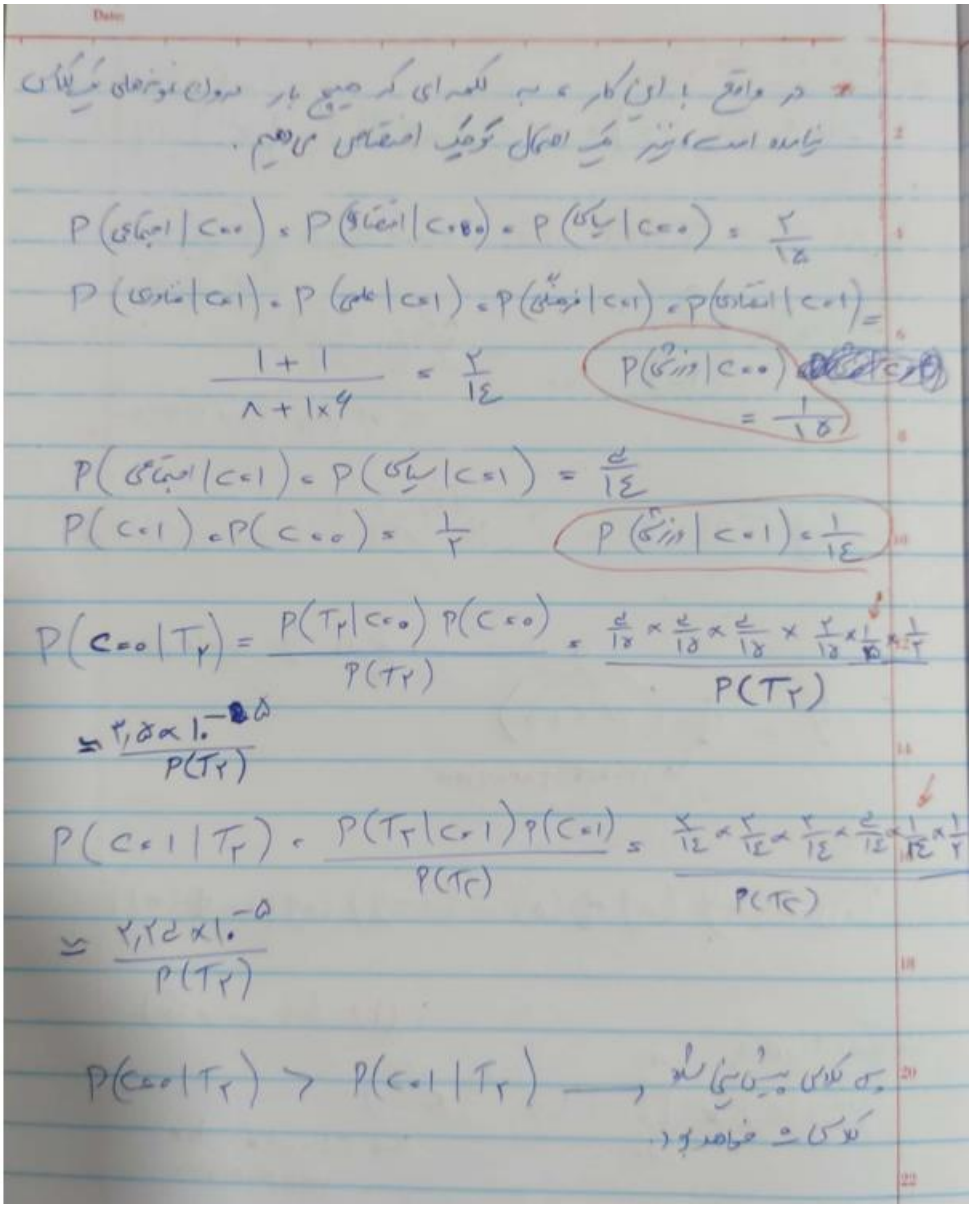
سوال ۱-


سوال ۱-
الف)
ب)
کلاس ۰ خواهد بود.
کلاس ۱ پیش بینی شده
تعداد کلمات صفر بردار
استفاده شده در نمونه های
این کلاس
ضریب هموارسازی



سوال ۲- فایل ها مطالعه شد.

سوال ۳-

این مقدار باید مینیمم شود

سوال ۲-

$$-\log P(y|x) = -\sum_{i=1}^{n} \log P(y^{(i)}|x^{(i)}) \quad (1)$$

و یک متغیر باینری است، احتمال $P(y|x)$ بصورت زیر محاسبه می‌شود:

$$P(y|x) = \begin{cases} \hat{y} & y=1 \\ 1-\hat{y} & y=0 \end{cases} = \hat{y}^{y}(1-\hat{y})^{1-y}$$

که همان توزیع برنولی است.

$$-\log P(y|x) = -\sum y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})$$

برای محاسبه $\hat{y}$ داریم:

$$\hat{y} = \Phi(\overbrace{w^T x + b}^{o})$$

↳ Probit function

$$\text{loss} = -y \log \Phi(o) - (1-y) \log(1-\Phi(o))$$

طریقه محاسبه رابطه (1):

objective:

$$P(y|x) = \prod_{i=1}^{n} P(y^{(i)}|x^{(i)})$$

↳ should be maximum

getting the Log

$$\log P(y|x) = \sum_{i=1}^{n} \log P(y^{(i)}|x^{(i)})$$

↳ should be maximum

در نتیجه $-\log P(y|x)$ باید مینیمم شود که همان رابطه (1) می‌شود.

سوال ۴-

الف) توابع فعالسازی خاصیت غیرخطی بودن را به توابعی که توسط شبکه عصبی ساخته میشود، اضافه میکند. یک شبکه عصبی بدون توابع فعالسازی، تنها میتواند مدلی خطی بر مسئله مورد نظر اعمال کند ولی اکثر مسائل را نمیتوان تنها با یک مدل خطی حل کرد. در مدل خطی فرض میشود که با زیاد شدن یک نورون ورودی به اندازه c ، خروجی هم به این اندازه زیاد میشود، برای مثال وقتی میخواهیم مستحقان وام بانکی را شناسایی کنیم، زیاد شدن حقوق از ۰ به ۵ میلیون با زیاد شدن حقوق از ۱۰۰ به ۱۰۵ میلیون تاثیر یکسانی بر روی خروجی میگذارد که در واقعیت درست نیست.

ب) خیر، هر تابع غیرخطی نمیتواند به عنوان تابع فعالسازی مورد استفاده قرار بگیرد. برای مثال یک تابع فعالسازی باید مشتق تعریف شده داشته باشد تا در فرایند back propagation به مشکل نخوریم.

همچنین این توابع باید، مقادیر ورودی را به محدوده مناسبی ببرند، برای مثال تابع سیگموئید مقادیر ورودی را به بازه 0 و 1 و تابع tanh این مقادیر را به بازه 1- و 1 می‌برد که برای مسئله binary classification مناسب است.

علاوه بر موارد فوق، این تابع بهتر است از لحاظ هزینه محاسباتی سبک باشد تا سرعت شبکه را کند نکند.

و در آخر، بسته به اینکه چه مسئله‌ای را میخواهیم حل بکنیم، باید تابع فعالسازی مناسب با آن را انتخاب کنیم، برای مثال معمولا تابع RELU را برای لایه های میانی استفاده میکنیم و از تابع Softmax در لایه آخر برای مسئله multi-class classification استفاده میکنیم.

سوال ۵-

الف)

تابع سیگموئید: این تابع مقادیر را به بازی 0 و 1 می‌برد که میتواند به عنوان احتمال برای binary classification تعبیر شود. این تابع مشتق پذیر است و تغییرات ناگهانی ندارد (smooth است) و محدود است.

معایب: باعث تشدید vanishing gradient میشود، زیرا برای مقادیر خیلی بزرگ یا خیلی کوچک، مشتق تقریبا صفر میشود و این باعث میشود فرایند یادگیری طول بکشد یا اصلا انجام نشود. مشکل دیگر این است که خروجی آن همواره مثبت است و این نیز منجر میشود که فرایند همگرایی بیشتر طول بکشد زیرا مجبور است

وزنهای مربوط به یک نورون را همواره زیاد یا کم کند و نمیتواند بعضی ها را زیاد و بعضی ها را کم کند. هرچند مشکل دوم نسبت به مشکل اول خیلی جدی نیست.

$$S(x) = \frac{1}{1 + e^{-x}}$$

تابع softmax: این تابع معمولا در لایه خروجی برای مسائل multi-class classification استفاده میشود، زیرا مقادیر ورودی را به یک توزیع احتمال تبدیل میکند و مطمئن میشود که جمع این مقادیر برابر با ۱ شود. این تابع مشتق پذیر نیز میباشد.

معایب: این تابع نیز همانند تابع سیگموئید از مشکل vanishing gradient رنج می‌برد. همچنین به مقادیر خیلی بزرگ حساس است که این موضوع، فرایند بهینه سازی را unstable میکند.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

تابع Relu: این تابع از لحاظ محاسباتی بسیار ساده و سریع است و مشکل vanishing gradient را تا حدود زیادی حل میکند و باعث میشود فرایند یادگیری بهتر صورت گیرد. این تابع در شبکه های عصبی محبوبیت کسب کرده و به عنوان تابع دیفالت در لایه های میانی استفاده میشود.

معایب: در صفر مشتق ندارد. همچنین مشتق برای نورونهایی که خروجی منفی دارند، صفر است که این باعث میشود، در نتیجه این نورونها در فرایند بهینه سازی آپدیت نمیشوند، به این نورونها به اصطلاح dead neurons گفته میشود. همچنین مقادیر خروجی این تابع نامحدود هستند و این باعث میشود به outlier ها حساس شود. برای بهبود معایب گفته شده توابع دیگری مثل leaky relu پیشنهاد میشود.

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

تابع Tanh: ورودی ها را در محدوده ای بین 1- و 1 می‌برد و یک خروجی به مرکزیت صفر ارائه می دهد. مشتق پذیر و smooth است و می تواند روابط پیچیده را مدل کند. این می تواند در لایه های پنهان برای ثبت الگوهای متقارن و متمرکز در داده ها در مقایسه با ReLU مفید باشد.

معایب: دارای محدودیت هایی همانند سیگموئید است، مانند مشکل vanishing gradient برای مقادیر ورودی شدید و همگرایی کند. محدودیت مقادیر به بازه‌ی [-1,1] می تواند ظرفیت شبکه را در مقایسه با ReLU محدود کند.

$$\varnothing(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

My prompt in Chatgpt:

what are following activation functions?(pros and cons) Compare them with each other.

sigmoid, softmax, relu, tanh

ب) سند نویسی به صورت کامنت در فایل کد

ج)

تعداد لایه‌ها و علت انتخاب این تعداد: اگر فقط تعداد لایه‌هایی که دارای پارامتر قابل آموزش هستند را در نظر بگیریم، به تعداد ۲ لایه از این نوع در معماری مدل استفاده کردم، یک لایه میانی با تعداد نورون ۲۰ و یک لایه نهایی با تعداد نورون ۳ (تعداد کلاس‌ها). دلیل کم بودن تعداد لایه‌ها این است که تعداد دیتای ما خیلی کم است (از هر کلاس ۱ عدد)، پس زیاد کردن تعداد لایه‌ها باعث پیچیدگی مدل شده و فرایند آموزش بسیار کند خواهد شد و در تعداد epoch معقول قابل انجام نیست. اگر هم تعداد epoch را خیلی بزرگ کنیم، نهایت مدل ما overfit میشود (به همان دلیل دیتای محدود).

تعداد نورون‌های هر لایه و علت انتخاب این تعداد: تعداد نورون لایه میانی ۲۰ عدد است و این با آزمون و خطا بدست آمد، مقادیر نزدیک به همین عدد نیز نتیجه تقریبا مشابهی می‌دهند. اگر این تعداد خیلی کمتر شود، طبیعتا ظرفیت مدل برای یادگیری کاهش پیدا میکند و اگر خیلی زیاد شود، مدل ما پیچیده شده و فرایند آموزش را مختل میکند.

تابع فعالسازی و علت انتخاب آن: از تابع Tanh برای تابع فعالسازی لایه میانی استفاده کردم. این تابع بر خلاف تابع سیگموئید خروجی منفی نیز دارد، مشتق پذیر نیز میباشد پس گزینه مناسبی به نظر میامد، و پس از

استفاده ازش متوجه شدم که دقت بهتری روی مسئله میدهد. برای لایه خروجی، از Softmax استفاده کردم زیرا یک مسئله multi-class classification داریم.

تابع ضرر و علت انتخاب آن: از Categorical Cross Entropy استفاده کردم زیرا یک مسئله classification داریم و برای این نوع مسائل با استفاده از تخمین Maximum Likelihood به این تابع ضرر می‌رسیم.

د) در فایل Q5_Pytorch_Model پیاده سازی شد.

سوال ۶-

اولین مشکل و مهم‌ترین آن این است که پیش‌بینی مدل ما همواره کلاس ۱ می‌باشد زیرا خروجی تابع Relu عددی مثبت است و تابع سیگموئید در x های مثبت خروجی بین [1/2, 1] دارد و در نتیجه پیش‌بینی مدل همواره کلاس ۱ خواهد بود.

مگر اینکه مرز 0.5 را افزایش دهیم. باز در آن صورت، ترکیب این دو تابع فعالسازی مشکلاتی را بوجود می‌آورد.

از آنجا که در بازه مثبت، تابع relu همان مقدار را برمیگرداند، بنابراین برای مقادیر بزرگ، ورودی تابع سیگموئید بزرگ خواهد بود و مشتق در این نقاط نزدیک صفر است و این موضوع باعث میشود فرایند آموزش بسیار کند شود و همچنین مشکل vanishing gradient رخ دهد.

همچنین آستانه 0.5 ممکن است همیشه خوب نباشد، مخصوصا وقتی از هر دو نوع کلاس به تعداد یکسان در دیتاست آموزش وجود نداشته باشد. معمولا این بازه را بزرگتر میگذراند تا در صورتی که مدل مطمئن بود، بگوید کلاس ۱ است و این معمولا وقتی است که کلاس ۱ به معنای وجود یک شئ و کلاس ۰ به معنای عدم وجود یک شئ در تصویر باشد.

سوال ۷-

الف) اگر بخواهم در یک جمله بگویم، مهم‌ترین تفاوت این است که در یادگیری ماشین feature engineering داریم و باید از دیتا، ویژگی استخراج کنیم ولی در یادگیری عمیق این ویژگیها توسط خود مدل شبکه عصبی استخراج میشوند.

توضیحات اضافه تر:

یادگیری ماشین یک فیلد گسترده تر است که دارای تنوعی از الگوریتمها میباشد. الگوریتم هایی مثل درخت تصمیم، SVM و Linear Regression.

یادگیری عمیق به خصوص برای دیتای غیرساختار یافته مثل عکس و متن مناسبتر است و دقت های لبه دانش را به خود اختصاص داده است.

به طور خلاصه یادگیری عمیق یک زیرشاخه از یادگیری ماشین میباشد که بر روی شبکه های عصبی با تعداد لایه های زیاد تمرکز دارد.

ب) اگر مسئله ما یک مسئله پیچیده است، به احتمال زیاد خروجی لایه یازدهم ویژگی های سطح بالاتری به ما خواهد داد، مثلا برای یک تصویر ماشین، لایه ششم ممکن است لبه‌ها و گوشه‌ها یا اشکال هندسی مثل دایره و ... را کشف کرده باشد و لایه یازدهم بگوید در این نقطه یک چرخ داریم یا در نقطه دیگر چراغ ماشین وجود دارد، در نتیجه خروجی این لایه برای تصمیم نهایی بسیار موثرتر خواهد بود. هرچند برای مسائل ساده تر ممکن است خروجی همان لایه ششم برای تصمیم نهایی کافی باشد. البته اگر مسئله ساده باشد، تعداد لایه بیشتر میتواند

ج) طبق قضیه universal approximator تنها با یک لایه میانی با تعداد نورون کافی و تابع فعالسازی مناسب میتوان هر تابعی برای هر مسئله ای را تخمین زد ولی متخصصان در عمل قادر به این کار نبوده‌اند و دست به دامان زیاد کردن تعداد لایه ها برای تغییر بازنمایی شده اند تا بتوانند مسئله را به یک بازنمایی جدید ببرند که در آن بازنمایی بتوانند مسئله را حل بکنند. در نتیجه جواب من این است که در حال حاضر شبکه های عمیق تر کاراتر بوده اند مگر اینکه روشی پیدا شود که بتوان با همان یک لایه میانی مسائل را با دقت بهتری حل کرد. البته این نکته نیز حائز اهمیت است که با عمیق کردن شبکه، تعداد نورونهای هر لایه نیز باید به اندازه کافی باشد وگرنه ظرفیت و توانایی مدل به شدت کاسته میشود.

د)

مزایا: یکی از مزایای افزایش تعداد لایه‌ها، بالارفتن قدرت و ظرفیت شبکه برای یادگیری مدلهای پیچیده‌تر است، در نتیجه میتوان دقت بهتری در مسائل پیچیده تر کسب کند. تعداد لایه های بیشتر به معنای یادگیری ویژگی‌های سطح بالاتر نیز می‌باشد که این در تصمیم گیری نهایی مدل بسیار تاثیرگذار است.

معایب: زیاد کردن تعداد لایه ها میتواند منجر به overfitting شود. مخصوصا وقتی دیتاست آموزش اندک است، ممکن است نویز را به جای ویژگی های معنایی واقعی حفظ کند.

مشکل دیگر، زیاد شدن هزینه محاسباتی است، این هزینه هم در Forward pass و هم در Back propagation زیادتر میشود، زیرا تعداد پارامترها زیادتر میشود.

علاوه بر آن، تعداد لایه های بیشتر میتواند مدل را در معرض vanishing gradient و exploding gradient قرار دهد.

همچنین با بیشتر شدن تعداد لایه ها، مدل ما پیچیده تر میشود و این کار طراحی و fine-tune کردن آن را میتواند سخت تر کند و همچنین hyperparameter tuning بیشتری را می‌طلبد.

و در نهایت میتوان به زیاد شدن زمان آموزش اشاره کرد.