به نام خدا

عنوان:

تکلیف اول درس شبکه های عصبی

استاد:

دکتر منصوری

دانشجو:

محمدعلی مجتهدسلیمانی

شماره دانشجویی:

٤٠٣٣٩٠٤٥٠٤

تاریخ:

١٤٠٣/٨/١٦

❖سوال ۱.

الف)

الگوریتم backpropagation یک تکنیک یادگیری نظارتی هست که در شبکه های عصبی برای کاهش خطا استفاده میشود. به این صورت هست که گرادیان تابع loss با توجه به وزن ها محاسبه میشود و اجازه میدهد که شبکه وزن ها را به نوعی تنظیم کند تا خطا به کمترین میزان برسد. شامل دو مرحله هست forward pass و backward pass.

- در forward pass:

با انتقال داده های ورودی از طریق لایه به لایه شبکه برای محاسبه خروجی شروع میکنیم.

هر نورون خروجی خود را با اعمال یک تابع فعال ساز (activation function) به مجموع وزنی ورودی های خود محاسبه می کنیم.

loss را با مقایسه خروجی پیش بینی شده با label واقعی با استفاده از یک تابع loss (به عنوان مثال، میانگین مربعات خطا برای رگرسیون) محاسبه میکنیم.

- در backward:

که همان کاهش گرادیان مد نظر ما هست شروع میکنیم و گرادیان را برای تابع loss محاسبه میکنیم یعنی مشتق جزئی میگیریم از این تابع نسبت به وزن ها. بعد از محاسبه گرادیان این را به لایه های قبلی منتقل میکنیم و در نهایت پارامتر ها را که همان وزن های ما هستند بروز رسانی میکنیم.

ب)

تابع loss در یک شبکه عصبی به عنوان معیاری برای سنجش میزان مطابقت پیش‌بینی‌های شبکه با مقادیر هدف واقعی عمل می‌کند. این خطا را برای هر پیش‌بینی محاسبه می‌کند و فرآیند یادگیری شبکه را جلو میبرد. این عملکرد برای بهبود شبکه بسیار مهم است، زیرا بازخورد عملکرد شبکه را ارائه می دهد و به تنظیم وزن برای کاهش خطاها در طول زمان کمک می کند. تنظیم وزن ها هم که همانطور میدانیم روندی هست که شبکه یادگیری دارد. در واقع تابع loss کمک میکند که ما متوجه شویم اپدیت کردن وزن های چه قدر درست بوده است و در جهت کاهش خطا بوده است همچنین یک بازخورد از روند یادگیری شبکه به ما میدهد و در نهایت کارایی شبکه را ارزیابی میکند.

برای این پروژه خاص که یک binary classification داریم یعنی شخص سرطان دارد/ندارد داریم، بهتر است از این فرمول برای تابع loss استفاده بکنیم.

$$Loss=1/-N \sum N[yi\cdot log(pi)+(1-yi)\cdot log(1-pi)]$$

که در اینجا yi هدف واقعی ما هست. pi پیشبینی یا همان خروجی شبکه ما هست و N تعداد نمونه ها هستند. و این یک خروجی بین ۰و۱ به ما میدهد. همچنین چون برای مصارف پزشکی هست و میخواهیم شبکه کاملا مطمئن باشد تا تشخیص کاذب یا اشتباه ندهد.

ج)

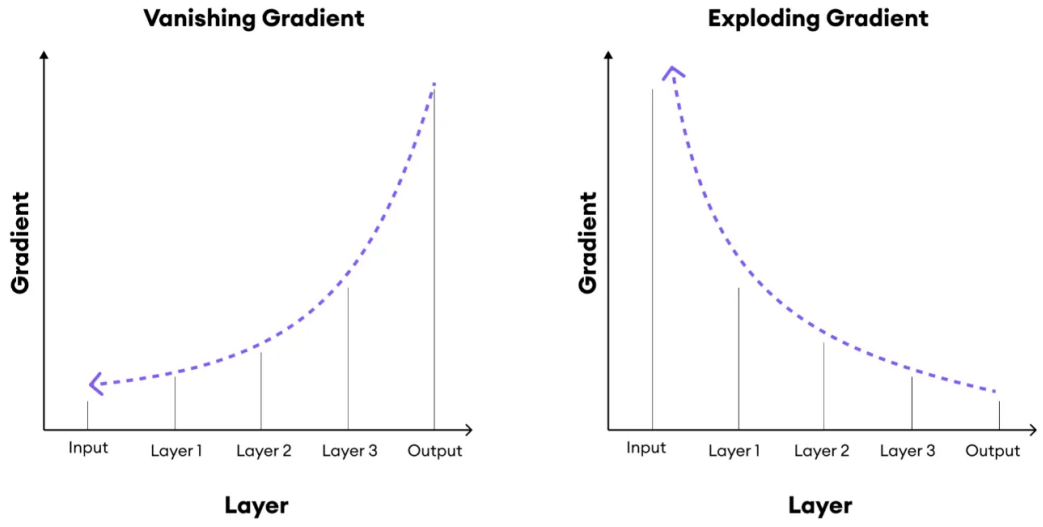
- محو شدن گرادیان:

این مشکل وقتی رخ میدهد که مقدار گرادیان به سبب عمق شبکه عصبی با انتشار مجدد در لایه ها بسیار کوچک میشوند. این منجر به بروز رسانی کوچک تر برای وزن های لایه اول میشود که باعث میشود روند یادگیری کند یا متوقف شود. از دلایل این اتفاق میتواند به activation function

ها و محاسبات گرادیان در back propagation بخاطر عمق شبکه بیان کرد. تاثیر آن هم که همانطور گفته شد منجر به توقف یا کند شدن سرعت یادگیری شبکه میشود. برای حل این مشکل میتوانیم از تابع activation، ReLU استفاده بکنیم یا یکبار دیگه سایز دسته ها را تنظیم بکنیم.

- انفجار گرادیان:

این مشکل دقیقا برعکس مسئله محو شدن گرادیان هست. در طول یادگیری شبکه عصبی زمانی رخ میدهد که گرادیان بیش از حد بزرگ میشود و به صورت تصاعدی رشد میکند که در لایه های عقبی منتشر میشوند. این میتواند باعث شود وزن های مدل به طور غیر قابل کنترلی رشد کنند و منجر به همگرایی ضعیف تر و بی ثباتی در روند یادگیری شود. از عوامل این اتفاق میتوان به استفاده از تابع فعالیت های مختلف، مقدار دهی اولیه وزن ها و نحوه محاسبه الگوریتم back propagation و نحوه محاسبه گرادیان اشاره کرد. در اثر این اتفاق ناپایداری و بی ثباتی در روند یادگیری شکل بگیرد و مدل نمیتواند الگوی درستی از داده ها یاد بگیرد. تابع loss بسیار میتواند بسیار زیاد شود. و در نهایت این انفجار باعث میشود که همگرایی بسیار بعید بنظر برسد و از دسترس دور شود. برای حل این مشکل میتواند یک حد آستانه مثلا برای گرادیان تعریف بکنیم یا تنظیم وزن ها را انجام بدهیم یا نرمال سازی کنیم یا نرخ یادگیری را کاهش بدهیم.



❖ سوال ۲.

الف)

در یک شبکه عصبی bias به مدل این اجازه را میدهد تا خروجی را مستقل از مقادیر ورودی تنظیم کند و به شبکه انعطاف بیشتری میدهد تا بتواند روی داده ها تناسب پیدا کند. که در اینجا منظور همان wo هست.

بدون bias ممکن است شبکه نتواند الگوهای خاصی را یاد بگیرد به عنوان مثال اگر همه ورودی ها صفر باشد خروجی که در اینجا s هست نیز صفر خواهد بود در صورتی که bias اجازه میدهد که شبکه یک مقدار غیر صفری تولید کند حتی وقتی همه ورودی ها صفر هستند که برای یادگیری شبکه مخصوصا برای الگو های پیچیده بسیار مناسب هست.
در نتیجه اثر bias برای تغییر تابع فعالیت بسیار مهم هست و علاوه بر مسائلی که گفته شد ممکن است کارایی ما بخاطر عدم وجود bias با مشکل مواجه شود یا حتی تعمیم پذیری شبکه عصبی ما محدود شود و قدرت شبکه برای یادگیری کاهش میابد و ممکن است زمان یادگیری آن پیچیده تر شود.

ب)

محمد علی محمدسلیمانی

۲ـ ب)

بیمار ① دیابت −

$S_1 = (0,4 \times 34) + (0,4 \times 135) = 94,4$ ضروری = −

$S_2 = (0,4 \times 34) + (0,4 \times 135) + 0,4 = 95,2$ ضروری = −

بیمار ② دیابت +

$S_1 = (0,4 \times 39) + (0,4 \times 145) = 99,4$ ضروری = −

$S_2 = (0,4 \times 39) + (0,4 \times 145) + 0,4 = 100,2$ ضروری = +

بیمار ③ دیابت +

$S_1 = (0,4 \times 33) + (0,4 \times 159) = 101,4$ ضروری = +

$S_2 = (0,4 \times 33) + (0,4 \times 159) + 0,4 = 101,2$ ضروری = +

بیمار ④ دیابت −

$S_1 = (0,4 \times 45) + (0,4 \times 145) = 102$ ضروری = +

$S_2 = (0,4 \times 45) + (0,4 \times 145) + 0,4 = 102,4$ ضروری = +

بیمار ⑤ دیابت +

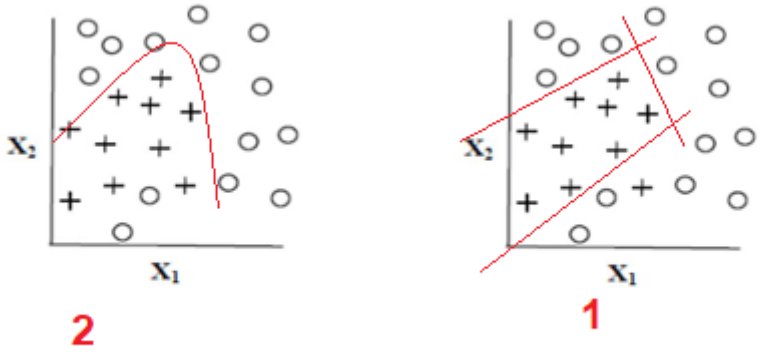
$S_1 = (0,4 \times 45) + (0,4 \times 139) = 99,4$ ضروری = −

$S_2 = (0,4 \times 45) + (0,4 \times 139) = 100$ ضروری = +

همانطور که در عکس بالا خروجی ها مشخص هست در ۲ جا مدل بدون داشتن bias جواب خطا داده است در صورتی وقتی که bias را اضافه کردیم مدل توانسته است به جوابی برسد که مورد انتظار ما هست و درست هست پس به طور کلی من اضافه کردن bias به مدل را مناسب میدانم تا کمک کند به مدل تا خروجی مورد انتظار ما را تولید کند.

سوال ۳.

الف)



برای داده های بالا یک داده خطی و یک داده غیر خطی پیشنهاد دادیم در داده خطی از ۳ پرسپترون استفاده کردیم و ۳ معادله خط درجه ۱ نوشتیم همانطور که میبینید این مدل overfit نشده است اگر قرار بود برای اون بعلاوه که در بیرون قرار گرفته نیز یک پرسپترون و مربع در نظر بگیریم مدل ما به سمت overfit شدن حرکت میکرد. در مدل دوم یک تابع غیر خطی پیشنهاد کردیم شبیه به تابع Gaussian که یک معادله درجه ۲ هست. خطا ما در اینجا تعداد مربع هایی هستند که در بیرون از تابع قرار گرفته اند و همانطور که مشخص هست شکل ۲ مستعد overfit شدن هست که میتوان نمونه بهتری هم ارائه کرد.

معادله برای دسته بندی جداساز خطی:

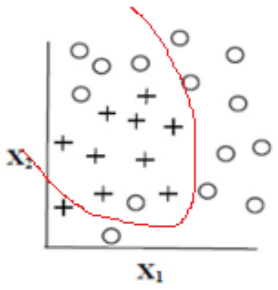
$$a \cdot X1 + b \cdot X2 + c = 0$$

که در اینجا برای feature ها x1 , x2 را در نظر گرفته ایم. این معادله خط هست برای هر کدوم از این ۳ پرسپترون ما.

معادله جداساز غیر خطی:

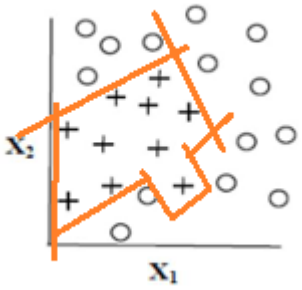
$$a \cdot X12 + b \cdot X22 + c \cdot X1 \cdot X2 + d \cdot X1 + e \cdot X2 + f = 0$$

یک تابع غیر خطی برای شکل بالا که مستعد overfit شدن نباشد به شکل زیر است:



که در اینجا خطا تعداد علامت + بیرون از تابع هستند. خطا در جداساز غیر خطی بیشتر است و مستعد است که ما به ویژگی به ورودی اضافه کنیم.

ب)



برای حل مشکل overfitting یک سری روش داریم. اول این است که معماری شبکه را تغییر بدهیم و از تعداد لایه های کمتر و نورون های کمتر استفاده کنیم مخصوصا وقتی حجم داده کم هست. روش دیگر dropout است که به صورت تصادفی بعضی نورون ها در حین هر دور یادگیری غیر فعال کنیم تا مدل فقط نسبت به یک سری نورون خاص وابسته نشود. روش اضافه کردن داده به training set و regularization نیز وجود دارند.

سوال ٤.

این سایت به ما کمک میکند تا روند یادگیری و خروجی شبکه های عصبی را بررسی کنیم و متوجه شویم که با افزایش یا کاهش ویژگی ها به عنوان ورودی، افزایش تعداد لایه های شبکه، تغییر نرخ یادگیری، تغییر noise، اندازه دسته ها و تعداد نورون در هر لایه چه تاثیری بر روند یادگیری شبکه و خروجی ما خواهد داشت. در این سوال به بررسی یک مسئله classification پرداخته ایم و نوع dataset ما از جنس XOR یا Exclusive OR است.

میدانیم با افزایش لایه های مخفی شبکه ما عمیق تر میشود این امر باعث میشود شبکه بتواند ویژگی های (feature) پیچیده تری استخراج کند و یاد بگیرد، لایه های بعدی میتوانند بر روی ویژگی

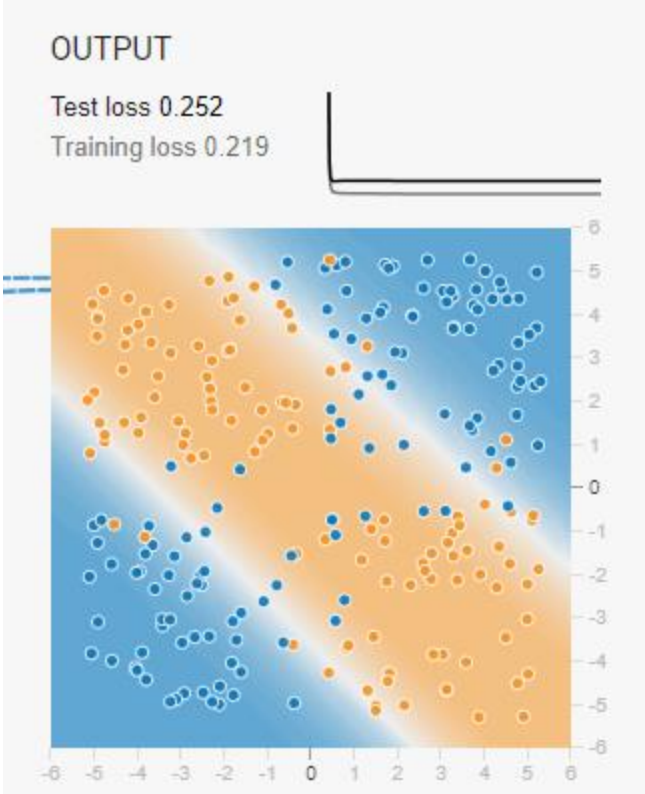
هایی که از لایه قبلی استخراج شده است یادگیری موفقی انجام دهد با این حال افزایش تعداد لایه ها ممکن است به محو شدن گرادیان صورت بگیرد بخاطر عمق شبکه، همچنین ممکن است موجب overfitting نسبت به training dataset شود و generality را از دست بدهد.

با افزایش تعداد نورون ها در هر لایه، شبکه فضای بیشتری برای ذخیره الگوها خواهد داشت و به همین دلیل میتواند feature های متنوع تری ذخیره کند و یاد بگیرد که به پارامتر های بیشتر برای یادگیری ختم میشود با اینحال افزایش هر نورون علاوه بر افزایش پیچیدگی ممکن است باعث overfitting نیز شود.

اینگونه میتوان گفت که عمق شبکه باعث ساخت ترکیب های مختلفی از feature ها شود در حالی که عرض آن اجازه میدهد که در هر گام feature های بیشتری را مورد توجه قرار بدهیم.

نوع مسئله ای که در سایت تعریف کردم از نوع Classification خواهد بود. از تابع سیگموید به عنوان تابع activation استفاده کردم بخاطر اینکه از اونجایی که مسئله ما کلاس بندی هست این تابع برای مقادیر بین ۰ و ۱ خیلی خوب عمل میکند.

در ابتدا تنها با یک لایه و ۲ ورودی شروع به یادگیری کرد که میتوانیم در لینک پایین خروجی را مشاهده کنیم، به سایر هایپر پارامتر ها مثل نرخ یادگیری تغییری صورت نگرفته.



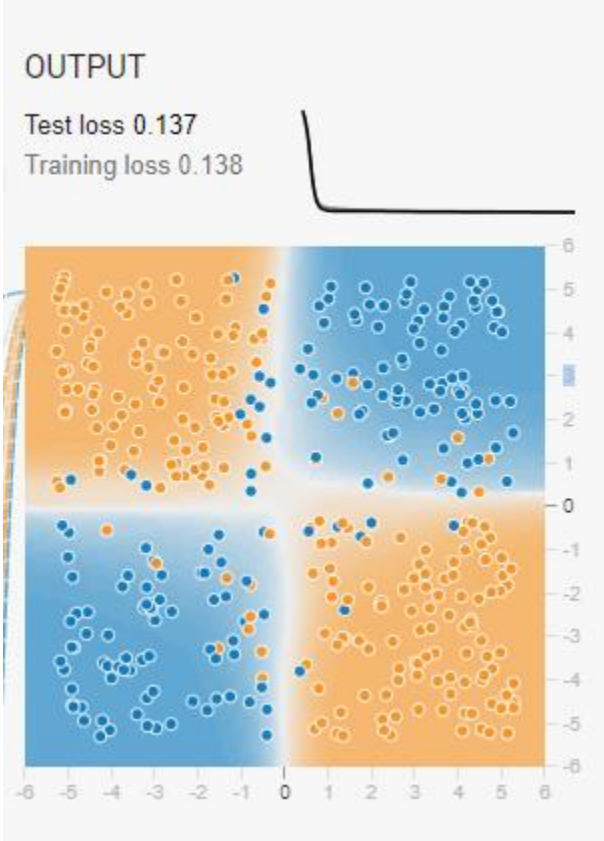
لینک خروجی.

در ادامه تعداد لایه ها را به ۲ لایه افزایش دادم و بعد از آن تعداد هر نورون در هر لایه را به ٤ نورون افزایش دادم. نرخ یادگیری را به ۰.۱ افزایش دادم و توانستم خروجی را کمی بهتر کنم و loss را کاهش دهم. لینک خروجی.

بعد تعداد لایه ها و همچنین تعداد نورون ها در هر لایه و همچنین تعداد ورودی را افزایش دادم با وجود اینکه تابع loss کمتر شد اما در ابتدا دچار overshooting شد و تابع ما زیگزاگی شد. لینک خروجی.

با اضافه کردن ٦ لایه مخفی و ترکیبی از تعداد نورون ها در لایه، همچنین کاهش نرخ یادگیری به 0.003 و همچنین استفاده از تابع ReLU توانستم تابع loss بر روی مجموعه آموزشی و تست

را تا حد خوبی کاهش بدهم و بنظرم این مجموعه متغیر ها برای این مجموعه دیتا مناسب خواهد بود.



لینک خروجی. با توجه به دو تابع میتوانیم بفهمیم مدل به خوبی توانسته روند یادگیری را طی بکند و با وجود خروجی خوب overfit هم نشود که این از نزدیکی تابع loss برای مجموعه تست و آموزش مشخص است.