به نام خدا

گزارش فاز دو پروژه ۵ هوش مصنوعی، محمد عظیم پور، ۸۱۰۱۹۷۶۵۷

نام پروژه: بررسی برخی از مسائل شبکه های عصبی به کمک TensorFlow

هدف: تشخیص سالم بودن یا ابتلای شخص به بیماری های Covid-19 یا ذات الریه با توجه به تصویر اسکن ریه.

شرح کلی: هر تصویر ابتدا مسطح شده و به صورت بردار به عنوان ورودی به شبکه عصبی داده می شود. هر درایه این بردار (معادل با یک پیکسل تصویر) یک ویژگی برای آن تصویر محسوب می شود. شبکه قرار است بر اساس این ویژگی ها و با ساختن ترکیبات غیرخطی از آن ها، وزن اتصالات بین لایه هایش را طوری تنظیم کند که خروجی آن ضمن داشتن کمترین خطا سالم بودن یا ابتلای شخص به بیماری های Covid-19 یا ذات الریه را به درستی تشخیص دهد.

داده ها:

در دو پوشه train و test به ترتیب داده های یادگیری و آزمون قرار داده شده اند. ساختار فایل ها به این صورت است که هر فایل سه پوشه دارد و در هر پوشه سه پوشه دیگر حاوی تصاویر اسکن ریه افراد سالم، مبتلا به کرونا و مبتلا به ذات الریه وجود دارد.

**** در ادامه پروژه منظور از دقت همان مقدار F1 است. ****

بخش های پروژه:

قسمت اول و دوم:

تصویر برای حالات مختلف در نوتبوک موجود است. پس از شمارش داده های موجود، مشخص شد که در ۵۱۴۴ تصویر داده های آموزش ۴۶۰ تصویر مربوط به افراد مبتلا به کووید-۱۹، ۱۲۶۶ تصویر مربوط به افراد سالم و ۳۴۱۸ تصویر مربوط به افراد مبتلا به ذات الریه است. همچنین در ۱۲۸۸ تصویر داده های آزمون ۱۱۶ تصویر مربوط به افراد مبتلا به کووید-۱۹، ۳۱۷ تصویر مربوط به افراد سالم و ۸۵۵ تصویر مربوط به افراد مبتلا به ذات الریه است.

قسمت سوم:

برای لایه ورودی و لایه Flattern پارامتر نداریم چرا که روی این لایه ها یادگیری انجام نمی شود.

برای لایه مخفی اول ۶۵۵۴۶۲۴ پارامتر داریم. این لایه ۱۰۲۴ نورون دارد پس ۱۰۲۴ مقدار بایاس باید داشته باشد. همچنین هر ورودی یک تصویر ۸۰ در ۸۰ است و ۶۴۰۰ ویژگی دارد که برای هر ویژگی یک وزن به ازای هر نورون نیاز است و در نتیجه ۶۵۵۳۶۰۰ پارامتر وزن داریم. پس کل پارامترهای این لایه از جمع پارامترهای بایاس و وزن ها به دست می آید که برابر ۶۵۵۴۶۲۴ است.

$$6554624 = 1024 \times (80 \times 80) + 1024$$

برای لایه مخفی دوم ۱۰۴۹۶۰۰ پارامتر داریم. . این لایه ۱۰۲۴ نورون دارد پس ۱۰۲۴ مقدار بایاس باید داشته باشد. همچنین این لایه ۱۰۲۴ ورودی (خروجی لایه قبل) دارد که برای هر ورودی یک وزن به ازای هر نورون نیاز است و در نتیجه ۱۰۴۸۵۷۹ پارامتر وزن داریم. پس کل پارامترهای این لایه از جمع پارامترهای بایاس و وزن ها به دست می آید که برابر ۱۰۴۹۶۰۰ است.

$$1049600 = 1024 \times 1024 + 1024$$

برای لایه خروجی ۳۰۷۵ پارامتر داریم. این لایه ۳ نورون دارد پس ۳ مقدار بایاس باید داشته باشد. همچنین این لایه ۱۰۲۴ ورودی (خروجی لایه قبل) دارد که برای هر ورودی یک وزن به ازای هر

نورون نیاز است و در نتیجه ۳۰۷۲ پارامتر وزن داریم. پس کل پارامترهای این لایه از جمع پارامترهای بایاس و وزن ها به دست می آید که برابر ۳۰۷۵ است.

$$3075 = 3 \times 1024 + 3$$

بنابراین تعداد کل پارامترهای شبکه از جمع تعداد پارامترهای سه لایه به دست می آید و برابر ۷۶۰۷۲۹۹ است.

$$7607299 = 6554624 + 1049600 + 3075$$

قسمت چهارم:

بعد از تمرین با تابع فعالساز relu مشاهده می شود که به دلیل اینکه مقادیرمان نرمالایز شده نیستند انفجار گرادیان رخ می دهد مقدار loss سرریز میکند. بنابراین شبکه نمی تواند تمرین داده شود و دقت مدل روی داده های آزمون ثابت و پایین (۹ درصد) است.

بعد از استفاده از تابع فعالساز tanh میبینیم که به دلیل اینکه برد تابع بین صفر و یک است سرریز قسمت قبل رخ نمی دهد، ولی به دلیل اینکه مشتق تابع در مقادیر بزرگ به صفر میل می کند باز هم عملیات آپدیت کردن وزن ها به درستی انجام نمی شود (مقدار اضافه یا کم شده به وزن هم به صفر میل می کند) و شبکه نمی تواند تمرین داده شود. در این حالت هم دقت مدل روی داده های آزمون ثابت است (این بار روی ۶۸ درصد).

در هر دو مدل عملیات یادگیری ناموفق است. برای اصلاح این مشکل باید از توابعی استفاده کنیم که حداقل در قسمت بزرگی از دامنه مشتق محسوس داشته باشند و همچنین باید داده هایمان را نرمالایز کنیم تا انفجار گرادیان رخ ندهد.

قسمت پنجم:

بعد از تقسیم کردن داده های اولیه به ۲۵۵، مدل اول قسمت چهار با تابع relu به دقت ۹۳ درصد رسید.

قسمت ششم:

۱- مومنتوم، جهشیست که به شبکه اعمال می شود در صورتی که در هنگام یادگیری به اکسترمم موضعی رسیده باشیم و مدل بیشتر آپدیت نشود. استفاده از آن در حالتی که امکان گیر افتادن مدل در چنین حالتی وجود دارد مفید است زیرا این حالت را از بین می برد.
۲- در حالتی که دقت بدون استفاده از مومنتوم ۹۳ درصد بود، با استفاده از مومنتوم ۰,۵ به دقت ۹۴ درصد، با استفاده از مومنتوم ۰,۹ به دقت درصد ۹۳ و با استفاده از مومنتوم ۰,۹۹ به دقت ۶۳ درصد می رسیم. هرقدر مقدار مومنتوم بیشتر می شود مشاهده می کنیم که نوسانات دقت حین یادگیری هم بیشتر می شود.
۳- اگر مقدار مومنتوم بسیار کم باشد، توانایی خارج کردن مدل از اکسترمم موضعی را ندارد. همچنین اگر این مقدار خیلی بالا باشد ممکن است پیشرفت های حاصل شده توسط مدل تا آنجای فرآیند یادگیری را هم به هم بزند. به این صورت که مدل را از اکسترممی (که ممکن است اکسترمم مطلق هم باشد) به اکسترمم محلی دیگر با دقت کمتر منتقل کند. به عنوان مثال در حالت ۰,۹۹ دیدیم که با وجود اینکه مدل در ایپاک اول به دقت بالای ۹۰ رسیده بود در نهایت در اکسترمم موضعی حول دقت ۶۶ درصد گیر افتاد.
۴- با استفاده از Adam (Adaptive moment estimation) لدون ثبت کردن مومنتوم و با نرخ یادگیری کمتر به نتیجه ای به خوبی قسمت اول بخش شش می رسیم. زیرا این الگوریتم در واقع بهینه شده ی SGD است، به این صورت که نرخ یادگیری و مومنتوم در هر مرحله با توجه به شرایط کنونی تعیین می شوند. همچنین در به روز رسانی وزن ها میانگینی از نتایج حاصل روی داده ها در مراحل فعلی و قبلی آزمون مهم است و نه فقط نتیجه آخر.

قسمت هفتم:

معمولا شبکه را در چند ایپاک تمرین می دهیم به این علت که تعداد داده هایی که داریم برای یادگیری کافی نیست و دقت مطلوب را نمی دهد. اگر تعداد داده ها به اندازه کافی زیاد باشد می توانیم به یک ایپاک هم بسنده کنیم.

اگر تعداد ایپاک های تمرین بیشتر از حد نیاز باشد، شبکه روی تمام استثنائات داده های آموزش هم حساس می شود که این اتفاق می تواند دقت مدل روی داده های آزمون را کم کند. به این فرایند overfiting می گویند. برای حل این مشکل چندین راه حل وجود دارد، به عنوان مثال می توانیم قبل از اینکه مدل به جایی برسد که دقت روی داده های آزمون با بیشتر شدن دقت روی داده های یادگیری کمتر شود فرآیدن را متوقف کنیم که به این روش early stopping می گویند. همچنین می توانیم با اعمال محدودیت هایی روی مقادیر وزن و بایاس، یا وارد کردن مقداری نویز به داده های یادگیری، یا حذف کردن برخی از مقادیر حاصل شده در خروجی یک لایه قبل از رسیدن به ورودی لایه بعد (به مجموعه این کارها regularization می گویند.) جلوی این اتفاق را بگیریم.

قسمت هشتم:

این تابع برای محاسبه خطا در تخمین کمیت های پیوسته مناسب است، نه برای مسائلی مانند دسته بندی. چرا که در این مسائل خروجی گسسته است و در نتیجه مقدار خطا هم مقداری گسسته خواهد بود که مشتق پذیر نیست. بنابراین همانطور که مشاهده می شود یادگیری انجام نمی گیرد و دقت ثابت می ماند.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(ObservedVal_i - PredictedVal_i)^2$$

قسمت نهم:

به طور کل Regularization روش هاییست که برای جلوگیری از overfitting استفاده می شود.

L2 جلوی زیاد شدن وزن ها و بایاس ها را می گیرد ولی نمی گذارد به صفر برسند. بعد از استفاده از این متد دیدیم که نوسانات دقت روی داده های آزمون خیلی زیاد بود (شبکه در نهایت به دقت ۹۱ درصد رسید) و بعضا با زیاد شدن دقت روی داده های یادگیری دقت روی داده های آزمون پایین می آمد. احتمالا به این علت که لزوما جلوگیری از زیاد شدن مقادیر وزن و بایاس و نگه داشتن

آن ها روی بازه مشخص باعث جلوگیری از overfitting نمی شود و ممکن است این اتفاق در مقادیر کوچک وزن و بایاس نیز بیفتد. همچنین شاید این عمل باعث بشود نتوانیم به دقت های بالاتری که در صورت افزایش مقادیر وزن و بایاس به دست می آمدند برسیم.

$$L(x,y) \equiv \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} \theta_i^2$$

L2 Regularization

Dropout به این صورت عمل می کند که درصدی از ورودی های هر لایه (که خروجی لایه قبلند) حذف می کند تا جلوی تمرین اضافه و اورفیت شدن را بگیرد. بعد از استفاده از این متد دقت تا ۹۵ درصد بالا رفت که می توان نتیجه گرفت این عمل برای جلوگیری از اورفیت شدن مناسب تر است و عوارض کمتری دارد.