به نام خدا

دانشکده‌ی مهندسی برق و کامپیوتر

درس هوش مصنوعی

تمرین کامپیوتری شماره ۵ فاز دوم

استاد: دکتر فدایی

علی پاکدل صمدی

۸۱۰۱۹۸۳۶۸

هدف پروژه:

هدف از انجام این پروژه آشنایی پیاده‌سازی neural network با استفاده از کتابخانه‌های Keras و TensorFlow می‌باشد. در این پروژه می‌خواهیم یک دیتاست از عکس حروف انگلیسی را طبقه‌بندی کنیم.

توضیح کلی پروژه:

در این پروژه ما یک فایل برای عکس‌های حروف انگلیسی در اختیار داریم. حال ابتدا پیش‌پردازش کرده و سپس با استفاده از network neuralای که با کتابخانه‌ها ساختیم، شبکه را آموزش می‌دهیم و سپس به طبقه‌بندی داده تست می‌پردازیم.

فاز اول: بررسی و پیش‌پردازش داده

۸. هنگامی که ترتیب در کلاس‌ها مهم نیستند، باید از روش one-hot-encoding استفاده کرد. این روش داده‌های train ما را مفیدتر و گویاتر بیان می‌کند و می‌توان مقیاس آن‌ها را به راحتی تغییر داد. با استفاده از مقادیر numeric، ما احتمالی را برای مقادیر خود تعیین می‌کنیم. این روش برای مقادیر خروجی ما استفاده می‌شود و پیش‌بینی‌های ظریف‌تری نسبت به برچسب‌های تنها دارد.

اگر از label encoding استفاده شود، لایه آخر ۱ نورون خواهد داشت، در binary encoding این تعداد ۲ می‌باشد و در one-hot encoding، به تعداد کلاس‌ها ما نورون خواهیم داشت.

فاز دوم: طراحی شبکه عصبی

در این قسمت ما یک شبکه عصبی طراحی کرده و hyper-parameterهای را به مقادیری که ذکر شده است ست می‌کنیم و دیتاست خود را به نحوی تغییر می‌دهیم که برای استفاده از توابع کتابخانه‌ها آماده باشد.

فاز سوم: طبقه‌بندی داده‌ها

در این قسمت ما ابتدا مقادیر را نرمالایز کرده و این کار را با تقسیم بر ۲۵۵ انجام می‌دهیم تا مقادیر بین ۰ و ۱ دربیایند. حال با استفاده از تابع fit، داده‌ها را طبقه‌بندی می‌کنیم و بررسی می‌کنیم کدام activation function بهترین F1 را خواهد داشت.

بررسی تاثیر تغییرات مختلف و سوالات:

قسمت اول) تاثیر optimizer:

- Momentum: momentum یک تکنیک ساده است که اغلب سرعت و دقت train را بهبود می‌بخشد و برای تسریع همگرایی تکنیک‌های بهینه‌سازی مبتنی بر گرادیان است. این پارامتر به جستجو اجازه می‌دهد که در یک جهت در فضای جستجو اینرسی ایجاد کند و بر نوسانات گرادیان‌های noisy غلبه کند. Momentum میانگین گرادیان هایی است که در طول زمان تغییر می کند و ما از آن برای به روزرسانی وزن در هر مرحله استفاده می کنیم. بنابراین منجر به همگرایی سریعتر می شود. در این روش بروزرسانی‌های بزرگتری خواهیم داشت. این مقدار معمولا بین ۰ تا ۱ می‌باشد در optimizer SGD باعث کاهش نوسانات می‌شود.

- همانطور که مشاهده می‌شود با momentum ۰,۵ و ۰,۹ ما با پیشرفت مواجه بودیم ولی در ۰,۹۸، نوسانات زیادی در هم train و هم test دیده می‌شود و همچنین دقت و f1 نیز کاهش پیدا کردند. افزایش momentum تا حدی می‌تواند موثر باشد اما اگر بیش‌از اندازه زیاد شود، باعث می‌شود بروزرسانی‌های ما زیاد باشد و این باعث می‌شود به خوبی آموزش صورت نگیرد و تاثیر منفی بگذارد.

- در optimizer adam سرعت training بیشتر می‌باشد ولی معمولا نسبت به SGD عملکرد بهتری ندارند، سرعت بهتر این optimizer باعث شده تا استفاده از Adam بسیار مرسوم شود.

قسمت دوم) تاثیر تعداد epoch:

- برای یادگیری شبکه، ما به نمونه‌های train احتیاج داریم. معمولا نمونه‌هایی که در اختیار داریم، برای رسیدن به همگرایی شبکه، به بیش از یک epoch نیاز داریم. اگر نمونه‌های زیادی داشته باشیم که شبیه به هم باشند و از یک توزیع نمونه‌بردای شده باشند، می‌توان با یک epoch هم انجام داد ولی در اکثر مواقع به بیش از یک epoch نیاز داریم. معمولا اکثر optimizerها در یک epoch به حداقل دقت دلخواه و وزن‌های بهینه نمی‌رسند و باید بیشتر از این باشد.

- افزایش epochها تنها در صورتی منطقی است که داده‌های زیادی در مجموعه داده خود داشته باشید. با این حال، مدل شما در نهایت به نقطه ای می رسد که افزایش دوره ها دقت را بهبود نمی بخشد. با افزایش تعداد دوره‌ها، تعداد دفعات تغییر وزن در شبکه عصبی افزایش می‌یابد و منحنی از

منحنی underfitting به منحنی overfitting می‌رود. ما باید هنگامی که مدل ما دیگر افزایش عملکرد خوبی ندارد و تقریبا ثابت شده است توقف کنیم و همان تعداد epoch کافی می‌باشد. تعداد مناسب epochها به پیچیدگی ذاتی مجموعه داده شما بستگی دارد. یک قانون خوب این است که با مقداری شروع کنید که 3 برابر تعداد ستون های داده شما باشد. اگر متوجه شدید که پس از اتمام تمام دوره‌ها، مدل همچنان در حال بهبود است، دوباره با مقدار بالاتری امتحان کنید.

قسمت سوم) تاثیر Loss Function:

همانطور که مشاهده می‌شود نتایج بدست‌آمده از طریق MSE مناسب نیستند.

دو دلیل کلی برای اینکه MSE برای طبقه‌بندی مناسب نیست:

- استفاده از MSE به این معنی است که فرض می‌کنیم داده‌های زیربنایی از یک توزیع نرمال (یک منحنی زنگ‌شکل) تولید شده‌اند. در اصطلاح بیزی این به این معنی است که ما یک پیشین گوسی را فرض می کنیم. در حالی که در واقعیت، مجموعه داده ای که می تواند به دو دسته طبقه بندی شود (یعنی باینری) از یک توزیع نرمال نیست، بلکه از یک توزیع برنولی است.

- تابع MSE برای طبقه بندی باینری غیر محدب است. به عبارت ساده، اگر یک مدل طبقه بندی باینری با تابع هزینه MSE آموزش داده شود، تضمینی برای به حداقل رساندن تابع هزینه نیست. این به این دلیل است که تابع MSE ورودی های با ارزش واقعی را در محدوده (∞-، ∞) انتظار دارد، در حالی که طبقه‌بندی باینری احتمالات را در محدوده (0،1) از طریق تابع سیگموئید/لجستیک مدل می‌کند.

به طور شهودی، MSE برای اندازه‌گیری کیفیت مدل بر اساس پیش‌بینی‌های انجام شده در کل مجموعه داده آموزشی در مقابل مقدار برچسب/خروجی واقعی استفاده می‌شود. به عبارت دیگر، می توان از آن برای نشان دادن هزینه های مرتبط با پیش بینی ها یا زیان های متحمل شده در پیش بینی ها استفاده کرد. هنگامی که ما رگرسیون خطی را انجام می دهیم، MSE انتخاب خوبی می‌باشد. در غیاب هیچ گونه دانشی از نحوه توزیع داده ها با فرض اینکه توزیع نرمال/گاوسی کاملاً منطقی است.

قسمت چهارم) تاثیر regularization:

<u>Regularization L2</u>: برای سادگی به آن منظم سازی نیز می گویند. اگر پیچیدگی مدل را تابعی از وزن ها در نظر بگیریم، پیچیدگی یک ویژگی با قدر مطلق وزن آن متناسب است. منظم‌سازی L2 وزن‌ها را به سمت صفر می‌آورد، اما آنها را دقیقاً صفر نمی‌کند. L2 Regularization تمام وزن ها را به مقادیر کوچک

کوچک می کند و از یادگیری هر مفهوم پیچیده ای در مدل جلوگیری می کند. هر گره/ویژگی خاص، در نتیجه از overfitting جلوگیری می کند.

همانطور که مشاهده می‌شود train و test نزدیک به هم هستند و overfitting رخ نداده است، همچنین دقت در این دو بسیار آرام در مراحل آخر تغییر می‌کند که به دلیل همین روش می‌شود. در این روش وزن‌ها رفته رفته کاهش یافته و به همین دلیل تغییرات کم می‌شوند.

Dropout: dropout تکنیکی است که در آن نورون‌های انتخابی تصادفی در طول تمرین نادیده گرفته می‌شوند. آنها به طور تصادفی "dropout" می شوند. این بدان معنی است که سهم آنها در فعال سازی نورون های پایین دست به طور موقت در گذر رو به جلو حذف می شود و هر گونه به روز رسانی وزنی برای نورون در گذر به عقب اعمال نمی شود. در این روش با دستکاری شبکه، باعث کاهش overfitting می‌شویم. از آنجایی که خروجی های یک لایه در حال dropout به صورت تصادفی زیر نمونه برداری می شوند، این اثر باعث کاهش ظرفیت یا نازک شدن شبکه در طول آموزش می شود. به این ترتیب، یک شبکه گسترده تر، به عنوان مثال. گره های بیشتری ممکن است هنگام استفاده از dropout مورد نیاز باشد.

همانطور که مشاهده می‌شود، dropout در مدل ما به طور مناسبی کار نمی‌کند. این عملکرد چند دلیل می‌تواند داشته باشد:

درست قبل از آخرین لایه این به طور کلی مکان بدی برای اعمال dropout است، زیرا شبکه توانایی اصلاح خطاهای ناشی از dropout را قبل از انجام طبقه بندی ندارد.

زمانی که زمان آموزش محدود است. معمولاً ترک تحصیل در شروع تمرین به عملکرد آسیب می رساند، اما منجر به کاهش خطای نهایی همگرا می شود.

هنگامی که شبکه نسبت به مجموعه داده کوچک است، منظم سازی معمولاً غیر ضروری است. اگر ظرفیت مدل در حال حاضر کم است، کاهش بیشتر آن با افزودن منظم سازی به عملکرد آسیب می رساند.

در مدل ما نیز به دلیل کم بودن تعداد نورون‌های هر لایه و همچنین تعداد لایه‌ها احتمال این وجود دارد که dropout به خوبی کار نکند.

نتیجه‌گیری کلی:

در این پروژه با استفاده از کتابخانه‌های ذکر شده، network neural را پیاده‌سازی کردیم و عکس‌ها را طبقه‌بندی کردیم. با تاثیر انواع پارامترها نیز آشنا شدیم و توانستیم در مجموع مدل‌های با دقت بالا و f1 بالا را پیدا کنیم. همچنین روش‌هایی برای جلوگیری از overfitting را نیز امتحان کرده و نتایج آن‌را نیز مشاهده کردیم.

منابع استفاده شده:

https://stackoverflow.com

https://www.geeksforgeeks.org

https://scikit-learn.org/

https://vidyasheela.com/

https://medium.com/

https://www.deeplearning.ai/

https://machinelearningmastery.com/

https://towardsdatascience.com/

https://stats.stackexchange.com/