پاسخ تمرین دوازدهم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

سوال اول)منبع: هفته ی اول کورس دوم **Deep Learning Specialization** در کورسرا

http://cafetadris.com/blog/%D8%A8%DB%8C%D8%B4-%D8%A8%D8%B1%D8%A7%D8%B2%D8%B4-overfitting/

http://cafetadris.com/blog/%da%a9%d9%85-%d8%a8%d8%b1%d8%a7%d8%b2%d8%b4-underfitting/

Overfitting:

تشخیص:

وقتی یک مدل عملکرد خوبی هنگام اجرا روی train set داشته باشد و عملکرد ضعیفی روی test set داشته باشد میگوییم مدل ما دچار overfitting شده است چون خوب تعمیم داده نشده است و به عبارتی شبکه به جای یادگیری مدلی که ما میخواستیم فقط داده های آموزشی را حفظ کرده است و آنها را از روی فیچر هایی از تصویر که خیلی هم ربطی به دسته بندی مدنظر ما نداشته به خروجی درست map میکند. این حالت به معنی داشتن واریانس بالا میباشد و تشخیص آن با مقایسه ی خطای مجموعه ی آموزشی و آزمایشی انجام میشود و به نوعی یکی از علت های آن بیش از اندازه آموزش دیدن شبکه روی دیتای آموزشی است. معمولا در شرایطی این اتفاق می افتد که برای حل سوال ساده از شبکه های خیلی عمیق با تعداد نورون خیلی زیاد استفاده میکنیم. این کار باعث میشود مدل داده ها را حفظ کند.

علت ها:

- مدل بیش‌ازحد پیچیده است و ویژگی‌های هم‌خط (Collinear) را دربرمی‌گیرد که واریانس داده‌های ما را افزایش می‌دهد.
- تعداد ویژگی‌های داده‌های ما بیشتر یا برابر با تعداد داده است.
- حجم داده بسیار کم است.
- داده پیش‌پردازش نشده تمیز نیست و نویز دارد.

راه حل ها:

۱) Holding Out Data :

از همه ی داده های موجود برای آموزش مدل استفاده نمیکنیم و آن را به دو دسته یکی برای آموزش و یکی برای آزمایش تقسیم میکنیم:
نسبت معمول میان این دو 80 درصد برای آموزش و 20 درصد برای تست است. مدل خود را آموزش می‌دهیم تا زمانی که نه‌تنها در مجموعه‌ی آموزش، در مجموعه‌ی تست نیز عملکرد خوبی داشته باشد. این امر نشان‌دهنده‌ی قابلیت تعمیم خوب مدل است؛ زیرا مجموعه‌ی تست نشان‌دهنده‌ی داده‌های دیده‌نشده‌ای است که برای آموزش استفاده نشده‌اند؛ البته این رویکرد به مجموعه‌داده‌ای نسبتاً بزرگ نیاز دارد.
برای داده های کمتر از ۱۰۰۰۰ این کار خوب است و از نسبت های ۷۰ به ۳۰ یا ۸۰ به ۲۰ استفاده میشود ما در کلان داده ها مثلا ۱ میلیون داده معمولا مجموعه ی dev و test درصد کمی از داده ها یعنی حدود ۱۰ هزار داده را تشکیل میدهند. چرا که میخواهیم ارزیابی مدل سریعتر انجام شود.
زیرا هدف مجموعه ی devاین است که الگوریتم های مختلف را روی آن آزمایش کنید و ببینید کدام بهتر عمل میکند بنابراین فقط باید به اندازه ای بزرگ باشد که دو یا نهایتا ۱۰ انتخاب الگوریتم را ارزیابی کند و به سرعت تصمیم بگیرد که کدام بهتر عمل میکند و برای اینکار به ۲۰ درصد کل داده ها نیاز نیست. مثلا برای یک میلیون داده ی کل ۱۰ هزار داده ی dev برای ارزیابی و evaluate کافی است.

۲) اعتبارسنجی متقابل داده‌ها(Cross Validation) :

می‌توانیم مجموعه‌داده‌مان را به k گروه تقسیم کنیم (K-fold Cross Validation). Cross Validation یا Hold-out یا dev بخشی از داده ها هستند که برای بهبود مدل هنگام آموزش شبکه استفاده میشود و تفاوتش با مجموعه ی تست این است که این مجموعه به انتخاب مدل بهتر کمک میکند اما مجموعه ی تست مدل نهایی انتخاب شده را میسنجد.در این روش اجازه می‌دهیم یکی از گروه‌ها مجموعه‌ی تست و گروه دیگر به‌عنوان مجموعه‌ی آموزش باشد. این روند را تا زمانی که هر گروه جداگانه یک بار به‌عنوان مجموعه‌ی تست استفاده شود تکرار

می‌کنیم (برای مثال، k بار تکرار می‌کنیم). برخلاف روش نگه‌داشتن داده، اعتبارسنجی متقابل اجازه می‌دهد تا تمامی داده‌ها یک بار برای آموزش استفاده شوند، اما از طرفی از نظر محاسباتی گران‌تر هستند.

۳)افزایش داده‌ها(Data Augmentation) :

یک مجموعه داده بزرگتر از overfitting جلوگیری می‌کند. اگر نتوانیم داده‌های بیشتری جمع کنیم و به داده‌هایی که در مجموعه‌ی داده فعلی خود داریم محدود باشیم، می‌توانیم برای افزایش اندازه‌ی مجموعه‌داده‌مان از روش افزایش داده استفاده کنیم؛ برای مثال، اگر کارمان طبقه‌بندی تصاویر است، می‌توانیم تغییرات مختلف تصویر را در مجموعه‌داده‌های تصویر خود اعمال کنیم؛ برای مثال، معکوس‌کردن (Flipping)، چرخش(Rotating) ، تغییر اندازه (Rescaling) و شیفت‌دادن.(Shifting)

۴)انتخاب ویژگی(Feature Selection) :

اگر فقط تعداد محدودی نمونه‌ی آموزشی داشته باشیم که هر یک از آن‌ها تعداد زیادی ویژگی داشته باشند، فقط باید مهم‌ترین ویژگی‌ها را برای آموزش انتخاب کنیم تا مدل ما به یادگیری تمامی ویژگی‌ها مجبور نباشد که سرانجام Overfitting اتفاق بیفتد. ما می‌توانیم به‌سادگی ویژگی‌های مختلف را آزمایش کنیم، مدل‌های جداگانه‌ای را برای این ویژگی‌ها آموزش دهیم و قابلیت تعمیم مدل‌ها را ارزیابی کنیم یا از یکی از روش‌های مختلف انتخاب ویژگی استفاده کنیم.

۵)منظم سازی L1 / L2 (Regularization) :

منظم سازی تکنیکی است که شبکه‌ی ما را در یادگیری مدلی که بیش‌ازحد پیچیده است و ممکن است به overfitting بینجامد، محدود می‌کند.. در منظم سازی L1 یا L2 می‌توانیم برای تابع Loss را یک مجازات در نظر بگیریم تا ضرایب برآوردشده (Coefficients) را به‌سمت صفر سوق دهیم. نظم‌دهی L2 اجازه می‌دهد وزن‌ها به‌سمت صفر پیش بروند، اما به صفر نرسند، درحالی‌که تنظیم L1 اجازه می‌دهد وزن‌ها به صفر برسند.

۶)انتخاب معماری ساده تر برای شبکه ی عصبی:

همان‌طور که در تنظیم L1 یا L2 ذکر شد، یک مدل بیش‌ازحد پیچیده به‌احتمال زیاد به بیش برازش (Overfitting) خواهد انجامید. بنابراین، ما می‌توانیم با حذف لایه‌ها به‌طور مستقیم از پیچیدگی مدل بکاهیم و اندازه مدل خود را کاهش دهیم.

۷)Dropout:

با استفاده از این تکنیک که نوعی منظم سازی محسوب می‌شود، کار drop outاین است که از هر یک از لایه های شبکه ی عصبی عبور کرده و احتمال حذف یک Node در شبکه را تعیین میکند و هر بار با حذف یک نود شبکه ی کوچکتر و کاهش یافته تر میشود چون با حذف هر گره تمام خروجی های آن هم حذف میشود. با وجود این، برای اجرای این تکنیک به ایپاک‌های (Epoch) بیشتری برای هم‌گراشدن مدل خود نیاز داریم.

۸)Early Stopping:

ابتدا می‌توانیم مدل خود را برای تعداد دلخواه زیادی از ایپاک‌ها آموزش دهیم و نمودار خطای اعتبارسنجی (Validation) را رسم کنیم. هنگامی که مقدار خطای اعتبارسنجی شروع به افزایش کرد، آموزش را متوقف می‌کنیم و مدل فعلی را ذخیره می‌کنیم. مدل ذخیره‌شده بهترین مدل برای تعمیم در میان مقادیر مختلف ایپاک برای آموزش آن خواهد بود.

پاسخ تمرین دوازدهم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

Underfitting:

تشخیص:

وقتی یک مدل هم روی داده های train و هم روی داده های developement درصد خطایی قابل توجهی(بالاتر از خطای بیز) دارد میگوییم مدل ما دچار underfitting شده است چون نتیجه اش حتی با داده های آموزشی هم مطابقت نداشته است و به معنی عدم تناسب خروجی با داده هاست. این حالت را میتوان با دیدن بایاس بالا و واریانس پایین الگوریتم تشخیص داد چون مدل نهایی شده ساده تر از خواسته ی مسئله است و بسیاری از ویژگی‌های داده‌های آموزشی را را نادیده می‌گیرد و نمی‌تواند رابطه‌ی میان ورودی و خروجی را یاد بگیرد.

علت ها:

- پارامتر های کم مدل
- تعداد فیچر ها خیلی کمتر از تعداد داده هاست.
- انتخاب مدل خیلی ساده
- تعداد ایپاک کم

۱)Decreasing Regularization :

چندین روش مختلف مانند تنظیمL1 ، دراپ‌اوت (Drop out) و غیره وجود دارد که به کاهش خطا در مدل کمک می‌کند؛ بااین‌حال اگر ویژگی‌های داده بیش‌ازحد یکنواخت شوند، مدل قادر به شناسایی روند غالب داده‌ها نیست و این موضوع به کم برازش (Underfitting) می‌انجامد. با کاهش میزان تنظیم (Regularization) پیچیدگی و تنوع مدل بیشتر می‌شود و امکان آموزش موفقیت‌آمیز مدل فراهم می‌آید.

۲)فزایش مدت‌زمان آموزش مدل:

توقف زودهنگام آموزش نیز می‌تواند به مدل کم‌برازش (Underfitted Model) بینجامد؛ بنابراین با افزایش مدت‌زمان آموزش می‌توان از بروز این مشکل جلوگیری کرد. لازم است در نظر بگیریم که آموزش بیش‌ازحد مدل نیز می‌تواند به مشکل بیش برازش (Overfitting) بینجامد و تعادل کم برازش (Underfitting) را بر هم بزند؛ پس باید تعادلی میان این دو ایجاد کنیم.

۳)Feature Selection :

در هر مدلی از ویژگی‌های خاصی برای تعیین نتیجه‌ی مشخص استفاده می‌شود. اگر ویژگی‌های کافی وجود نداشته باشد، باید ویژگی‌های بیشتر یا ویژگی‌هایی با اهمیت بیشتر به مدل اضافه شوند؛ برای مثال، در یک شبکه‌ی عصبی (ANN) ممکن است نودهای پنهان بیشتر یا در یک جنگل تصادفی (Random Forest) درختان بیشتری اضافه کنیم. این فرایند پیچیدگی بیشتری را به مدل تزریق و به‌این شکل نتایج بهتری را ارائه می‌کند.

۴)انتخاب معماری پیچده تر برای مدل:

میتوانیم از شبکه های عمیق تر با لایه های بیشتر و نود های بیشتر در هر لایه استفاده کنیم و یا به سراغ شبکه های کانولوشنی و باز گشتی با هایپرپارامتر های بیشتر برویم تا مدل بتواند ویژگی های اصلی داده ها را به درستی یاد بگیرد.