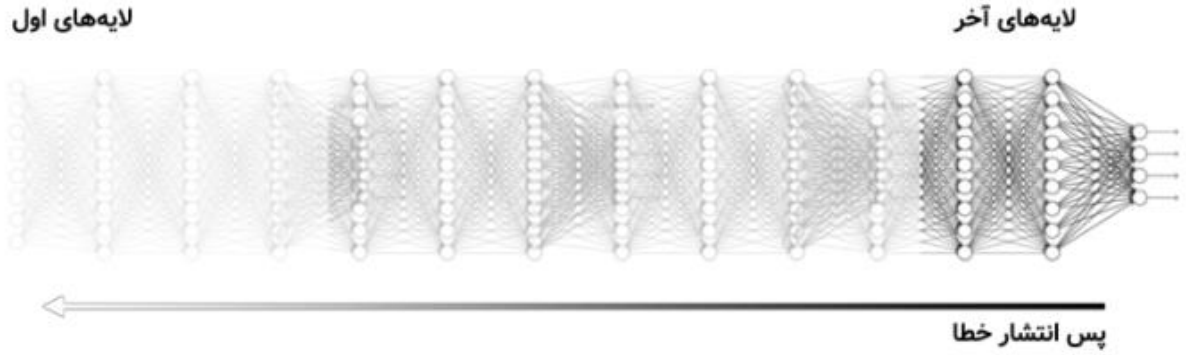
مشکل محوشدگی و انفجار گرادیان

مشکل محوشدگی گرادیان (Gradient Vanishing) در شبکه‌های عصبی عمیق

مشکل محو شدگی گرادیان ، مشکلی است که حین شبکه های عصبی مصنوعی با استفاده از روش های یادگیری مبتنی بر گرادیان رخ میدهد.هنگامی که تعداد لایه‌ها در شبکه‌های عصبی زیاد شده و به اصطلاح شبکه‌های عصبی عمیق یا همان یادگیری عمیق تشکیل شود، پس انتشار خطا نمی‌تواند وزن‌های لایه‌های اول را به خوبی اصطلاح کند. در این نوع روشها به منظور بروزرسانی پارامترهای شبکه عصبی از گرادیان استفاده میشود. هر پارامتر با توجه به میزان اثری که در نتیجه نهایی شبکه داشته است مورد تغییر قرار میگیرد. این مهم با استفاده از مشتق جزئی تابع خطا نسبت به هر پارامتر در هر تکرار فرایند آموزش صورت میپذیرد. مشکل محو شدگی اشاره به این مساله دارد مقادیر گرادیان ها با حرکت به سمت ابتدای شبکه رفته رفته به حدی کوچک میشوند که تغییرات وزن بصورت ناچیزی صورت میگیرد و به این علت فرایند آموزش بشدت کند میشود و در حالات شدیدتر این مساله باعث متوقف شدن فرایند آموزش میگردد. این مساله عموما بواسطه عمق زیاد شبکه رخ میدهد. عمق زیاد در شبکه های عصبی پیش خور یا feed forward neural networks همانند (MLP و CNN و..) به لایه های زیاد اشاره دارد در حالی که این مساله در شبکه های عصبی بازگشتی به گام های زمانی زیاد اشاره دارد چرا که در این نوع شبکه ها هر گام زمانی به منزله یک لایه در شبکه های پیش خور است.



در شکل بالا مشاهده می‌شود که الگوریتمِ شبکه‌ی عصبی عمیق می‌خواهد پس‌انتشار (back propagation) را از لایه‌ی آخر به اول اجرا کند ولی هر چه در لایه‌ها به سمت لایه‌ی اول پیش می‌رود قدرت آپدیت وزن کم و کم‌تر می‌شود و این همان محوشدگی گرادیان یا gradient vanishing است. محوشدگی گردایان باعث می‌شود که الگوریتمِ یادگیری عمیق در بسیاری از مواقع نتواند به وزن‌های مناسب برسد و در اصطلاح همگرا (converge) نمی‌شود .محوشدگی گرادیان (Vanishing Gradient)موقعیتی را توصیف می‌کند که در آن یک شبکه‌ی عصبی پیش‌خور عمیق (Deep Feed Forward Neural Network)یا یک شبکه‌ی عصبی بازگشتی (RNN) تواند اطلاعات مفید گرادیان را از سمت لایه‌ی خروجی به‌سمت لایه‌ی ورودی عبور دهد.

چرا مشکل محوشدگی گرادیان (Vanishing Gradient) اتفاق می‌افتد؟

برخی توابع فعال‌ساز، مانند تابع سیگموید(Sigmoid) ، مقادیر ورودی با مقیاس بزرگ را در یک بازه‌ی کوچک میان صفر و ۱ قرار می‌دهند؛ بنابراین زمانی‌که یک تغییر بسیار بزرگ در مقدار ورودی تابع اتفاق می‌افتد، خروجی تابع تنها مقدار کمی تغییر می‌کند؛ این یعنی مقدار مشتق آن خیلی کوچک می‌شود. تا اینجا متوجه شدیم محوشدگی گرادیان زمانی اتفاق می‌افتد که مقدار مشتق ورودی‌ها به صفر نزدیک شود، اما باید ببینیم این در شبکه چه اهمیتی دارد؟

دلیل محو شدگی گرادیان در شبکه عصبی بازگشتی چیست ؟

فرض کنید دنباله ورودی ما به شبکه جمله ای حاوی ۲۰ کلمه باشد " **دانش آموزان** نخبه کشور ایران به مرحله نهایی المپیاد ریاضی رسیده ... و به مقام نخست دست **یافتند**"

همانطور که در مثال بالا مشاهده میکنید شبکه عصبی بازگشتی ما برای اینکه بتواند کلمه "یافتند" را که در انتهای جمله آمده است بدرستی پیش بینی کند نیازمند اطلاعاتی از ابتدای جمله (کلمه دانش آموزان) است. به این نوع وابستگی ها وابستگی های بلند-مدت گفته میشود چرا که فاصله نسبتا قابل توجهی بین اطلاعات مرتبط (در اینجا دانش آموزان) و نقطه ای که این اطلاعات جهت انجام یک پیش بینی (یافتند) مورد استفاده است وجود دارد. متاسفانه در عمل هرچه این فاصله بیشتر شود شبکه های عصبی RNN با مشکل بیشتری در یادگیری این وابستگی ها مواجه میشوند چرا که یا با مشکل محو شدگی گرادیان یا انفجار گرادیان برخورد میکنند.

این مشکلات در حین آموزش یک شبکه عمیق در زمانی که گرادیان ها در فرایند پس انتشار از انتهای شبکه به سمت ابتدای شبکه منتقل میگردند رخ میدهد. گرادیان هایی که از لایه های انتهایی می آیند باید ضرب های متعددی را پشت سرگذاشته تا به لایه های ابتدایی برسند (بخاطر قائده زنجیره ای حسابان) و اینطور رفته رفته مقادیر آنها شروع به کوچک شدن میکند (کمتر از ۱). این فرایند انقدر ادامه میابد و این مقادیر انقدر کوچک میشوند که (در شبکه های بسیار عمیق) فرآیند آموزش مختل شده و عملا متوقف میگردد چرا که گرادیان ها مقادیر بسیار ناچیزی داشته (محو شدگی گرادیان) که تاثیری در تغییر وزنها صورت نمیدهند. به این مشکل، مشکل محو شدگی گرادیان گفته میشود. به همین شکل امکان انفجار گرادیان نیز وجود دارد و مقادیر گرادیان رفته رفته انقدر بزرگ میشوند تا نهایتا مدل دچار خطا گردد (سرریز در محاسبات رخ دهد.(دریافت !nan در پایتون بعنوان مثال یکی از علائم این مساله است)

مقابله با مشکل محو شدگی گرادیان:

۱- یک راه عدم استفاده از الگوریتم های یادگیری مبتنی بر گرادیان کاهشی است! روش **Hessian Free Optimizer With Structural Dumping**یکی از این روشهاست که از ماتریس Hessian بهره میبرد.

۲- استفاده از معماری شبکه عصبی بازگشتی همانی (Identity RNN)است . در این شبکه وزنها همگی بر اساس ماتریس همانی مقداردهی اولیه شده و بجای tanh از تابع فعالسازی ReLU استفاده میشود. این عمل سبب میشود تا محاسبات شبکه نزدیک به تابع همانی بوده و به همین دلیل نیز مشتق خطا که در زمان پس انتشار منتقل میشود همیشه مقداری برابر با ۰ یا ۱ خواهد داشت بنابر این با مشکل محو شدگی گرادیان مواجه نخواهیم بود.

۳- استفاده از توابع فعال‌سازی است که به مشتق با مقدار کوچک نمی‌انجامند؛ برای مثال، تابع واحد یک‌سوشده‌ی خطی (ReLU)جایگزین بسیار خوبی است.

۴- راه حل دیگری که عموما به بخش جدایی ناپذیری از شبکه های عصبی عمیق (چه شبکه های پیش خور همانند CNNو MLP و چه شبکه های عصبی بازگشتی همانند RNN تبدیل شده است استفاده از مقداردهی مناسب وزنهاست بگونه ای که پتانسیل رخداد محو شدگی گرادیان کمینه شود. به این منظور امروزه از الگوریتم های مقداردهی اولیه ای نظیر Xavier و یا MSRA که به He initialization یا Kaiming initialization هم معروف است برای این کار استفاده میشود. راه حل دیگری که بطور گسترده مورد استفاده قرار میگیرد استفاده از معماری LSTM است LSTM گونه دیگری از شبکه عصبی بازگشتی است که بطور ویژه بگونه ای طراحی شده است تا وابستگی های زمانی بلند مدت در دنباله ورودی را دریافت و حفظ کند. این شبکه عصبی بازگشتی بگونه ای عمل میکند که مقادیر حالت مخفی توسط سایر مقادیر فعالسازی محلی که نزدیک به آنهاست تحت تاثیر قرار میگیرند که این قابلیت نقش حافظه کوتاه مدت را بازی میکند درحالی که وزن های شبکه توسط محاسباتی که بر روی تمام دنباله رخ میدهد تاثیر میپذیرند که این به منزله حافظه بلند مدت است. بنابر این نوع از شبکه ها بگونه ای طراحی شده اند که در آن مقادیر

فعالسازی حالت مخفی آن بتوانند همانند وزن ها عمل کرده و اطلاعات را در فاصله های زیاد حفظ کنند. به همین علت این نوع از شبکه ها به شبکه های حافظه کوتاه و بلند مدت معروف شده اند.

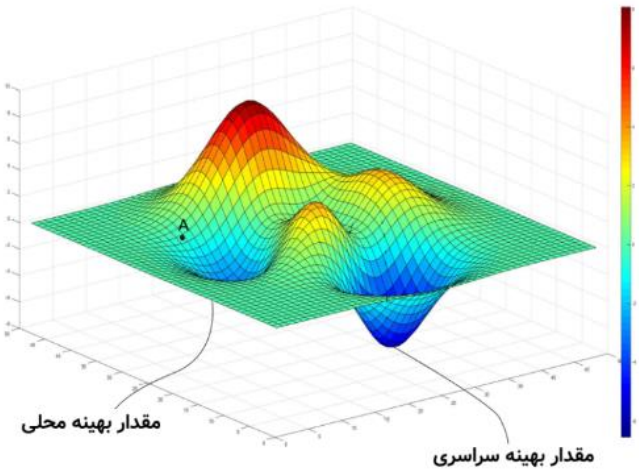
۵- راه‌حل دیگر استفاده از شبکه‌های باقیمانده (Residual networks) است. این شبکه‌ها مجموعه‌ای از اتصالات اضافی (Residual Connections) دارند که مستقیماً به لایه‌های اول وصل می‌شوند. روی این اتصال اضافی دیگر تابع فعال‌سازی اعمال نمی‌شود که مقدار مشتق را خیلی کوچک کند و به صفر نزدیک کند.

۶- راه‌حل آخر استفاده از لایه‌های عادی‌سازی دسته‌ای (Batch Normalization Layers) است. همان‌طور که توضیح دادیم، مشکل زمانی ایجاد می‌شود که یک ورودی با مقیاس بزرگ در یک بازه‌ی کوچک قرار می‌گیرد و به این می‌انجامد که مشتق به‌قدری کوچک شود که کم‌کم به صفر برسد. عادی‌سازی دسته‌ای، با عادی‌کردن مقدار ورودی، اجازه نمی‌دهد که مقدار ورودی به گوشه‌های تابع سیگموید برسد (در آنجا مشتق به صفر نزدیک می‌شود و از این راه به کاهش این مشکل کمک می‌کند).

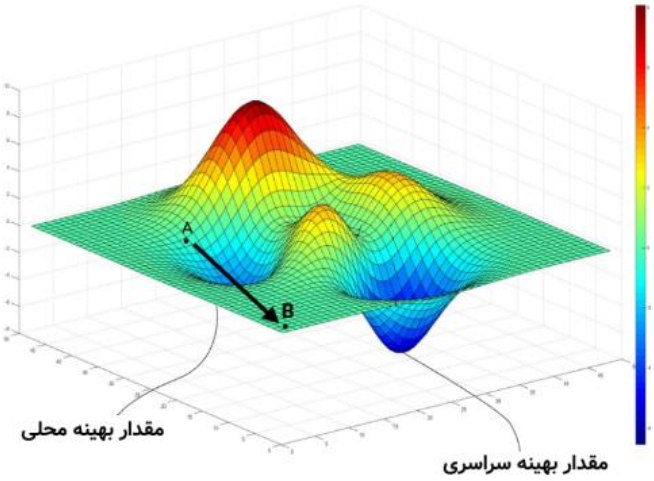
مشکل انفجار گرادیان (Exploding Gradients) در شبکه‌های عصبی عمیق

نقطه‌ی مقابل محوشدگی گرادیان، مشکل **انفجار گرادیان یا همان exploding gradients** است که به جای اضمحلال و محوشدن گرادیان، ممکن است آن را بیش از اندازه بزرگ نماید و به خاطر همین الگوریتم نتواند به یک همگرایی (converge)در میان وزن‌ها دست پیدا کند**.** در شبکه‌های عصبی عمیق با زیاد شدن تعداد لایه‌ها، بایستی هر کدام از وزن‌ها در لایه‌های مختلف با دقت بیشتری آپدیت شوند. اگر این گونه نشد، ممکن است وزن‌ها بیش از مقدار مورد انتظار، آپدیت شده و در اصطلاح الگوریتم به جای همگرایی به یک جواب بهینه (ترکیب وزن‌های مناسب(**، واگرا (diverge)** شده و نتواند به جواب‌های بهینه دست پیدا کند**.**

در شبکه عصبی آموختیم که وزن‌ها در ترکیب با یکدیگر خطایی ایجاد می‌کنند و با بالا و پایین کردن این اوزان، می‌توان خطا را تغییر داد. فضای خطا (error space) در یک شبکه با دو وزن به صورت زیر است**:**



در شکل بالا، الگوریتم دو وزن را انتخاب کرده و با توجه به این اوزانِ انتخاب شده در نقطه‌ی A قرار گرفته است. حال الگوریتم در فرآیند یادگیری بایستی آرام آرام وزن‌ها را تغییر دهد تا بتواند به یک بهینه‌ی محلی که مقدار خطای کمتری نسبت به اطراف دارد برسد. ولی در برخی از مواقع به دلیل محاسبات ریاضی و به تبع آن انفجار گرادیان، این آپدیت بیش از حد انجام شده الگوریتم با تغییر وزن‌ها به مقدار زیاد به نقطه‌ی) B در شکل زیر) پرش می‌کند**:**



همان‌طور که در شکل بالا ملاحظه می‌شود، وزن‌ها به میزان نامعقولی در هنگام آپدیت، تغییر کرده‌اند و این تغییرِ نامناسب منجر به عدم همگرایی الگوریتم شده است. الگوریتم با این واگرایی‌ها نمی‌تواند در یک فضای پیچیده که تعداد اوزان خیلی بیشتر از ۲ عدد است، به یک نقطه‌ی بهینه (محلی یا سراسری) برسد و عملاً یادگیری انجام نمی‌شود.

چگونگی برخورد با انفجار گرادیان

۱- تکنیک‌های مختلفی مانند **برش گرادیان (gradient clipping)** برای مقابله با انفجار گرادیان موجود است. در روشِ بُرش گرادیان، اگر مقدار گرادیان از یک حدی بالاتر یا پایین‌تر برود، می‌توان آن را به یک بازه‌ی مشخص (مثلاً بین ۵/۰- تا ۵/۰+) محدود کرد تا گرادیان تغییر غیر منطقی و زیادی نکند.

۲- همچنین روش **نرمال‌سازی (normalization)** برای مقادیر گرادیان نیز می‌تواند در این مورد استفاده قرار گیرد. در نرمال‌سازی گرادیان کاری همانند روش نرمال‌سازی که در پیش‌پردازش داده‌ها انجام می‌دهیم. یعنی بعد از هر بار آپدیت وزن‌ها می‌توانیم مقادیر گرادیان را به یک بازه‌ی کوچکتر نرمال یا در اصطلاح rescale کنیم.

۳- استفاده از توابع فعال‌سازی که از ایجاد مقادیر غیرنرمال جلوگیری می‌کنند (مانند تابع tanh) می‌توان این مشکل را تا حد خوبی کنترل کرد.

۴- عملیات پس انتشار را تا یک گام زمانی معین انجام دهیم (میتوان از truncated back-propagation استفاده کرد و تمام دنباله را پوشش داد)

۵- گرادیان را بصورت مصنوعی کاهش داد یا اصطلاحا مجازات کرد استفاده از(L1/L2 weight decay)

۶- یک حد بالایی بر روی گرادیان قرار داد (استفاده از Gradient Clipping)

Vanishing/Exploding Gradient اساسا چه تفاوتی باهم دارند و چه تاثیری بر رفتار شبکه دارند؟

انفجار گرادیان Exploding Gradient:

انفجار گرادیان زمانی اتفاقی می‌افتد که وزن‌ها مقادیر بالایی بگیرند و در نتیجه ضرب آنها به عدد بسیار بزرگی تبدیل شود.

• مدل اطلاعات زیادی را در طول فرآیند آموزش یاد نمی گیره، در نتیجه loss ضعیفی دارد.

• به علت پایدار نبودن مدل، تغییرات زیادی در بروز رسانی وزنها دیده میشه و وزنها در زمان آموزش به صورت نمایی رشد می کنند.

• در طول آموزش تابع هزینه مقدار NaN بگیره

محو شدگی گرادیان Vanishing Gradient :

محوشدن گردایان هم زمانی اتفاق می‌افتد که مقادیر گرادیان خیلی کوچک شوند و در نتیجه یادگیری مدل متوقف یا بسیار کند شود.

وزن های مدل در حین آموزش به صفر میل می کنه.

وزن های مدل به صورت نمایی کم میشه.

در طول آموزش بهبود مدل بسیار کنده، و ممکنه آموزش خیلی زود متوقف شه .

تغییرات در وزن های نزدیک به لایه خروجی بیشتر از لایه های نزدیک به لایه ورودی است.

برای جلوگیری از این دو پدیده می تونیم به صورت زیر عمل کنیم:

استفاده از توابع فعال سازی (یکسوساز خطی)

تغییر در روش وزندهی اولیه دیگر

بریدگی گرادیان :این متدکه برای انفجار گرادیان مناسبه ، اندازه گرادیان را با یک حد آستانه محدود می کنه. اینکار باعث میشه ، گرادیان هایی که حد آستانه های بالاتر از نرم تعیین شده دارند قطع شده تا با نرم مطابقت پیدا کنند.