جلسه دوم:

جلسه سوم:

اگر تابع فعالیت یکنوا افزایشی باشد به این صورت کاری میکنیم که خروجی ما بیشتر بشود یعنی اگر خروجی درست ما 1 بوده و خروجی ما کمتر بوده مثلا -1 کمک میکند بیشتر شود و همگرا میشود و بدین ترتیب تغییر وزن ها باعث کاهش خطا میشود. فرض جدایی پذیر خطی مهم هست در استفاده از پرسپترون. اگر وزن بهینه را در بیاوریم یعنی معادله خط یا مرز تصمیم گیری را بدست آورده ایم حالا بستگی به بعد هم دارد.

نحوه محاسبه مثال:

وزن را که ابتدا صفر قرار دادیم نظیر به نظیر با ورودی ضرب میکنیم از تابع فعالیت عبور میدهیم اگر خروجی با خروجی واقعی یکی بود که هیچی اگر نبود اپدیت میکنیم، چطوری؟ ورودی را در خروجی واقعی ضرب میکنیم بعلاوه یا منهای وزن فعلی میکنیم.

قضیه همگرا اگر مسئله جدایی پذیر خطی باشد در صورت وجود وزن ها مسئله به همگرایی میرسد و پاسخ همگرا خواهد بود.

Adaline تابع خروجی آن تابع خطی است چرا؟ در LMS میگفت میزان مقدار خطا مهم هست و اگر خطا بالا بود اپدیت بیشتر باشد بر خلاف پرسپترون که میزان خطا تاثیری نداشت و در نتیجه در LMS همگرایی سریع تر هست. Adaline تابع خطی دارد دیگر -1 و +1 نیست چرا؟ وقتی میخواهد تاثیر خطا در نظر بگیرد دیگر تابع نباید دو سطحی یا این یا اون باشد باید میزان خطا را در نظر بگیرد پس خروجی خطی هست. در Adaline از یک تابع فعالیت خطی رد میشود با وجود اینکه خروجی آن دو سطحی هست تنها نکته اینجا هست که بر خلاف پرسپترون اینجا مهم میشود که چه قدر خطا داریم و فقط داشتن یا نداشتن خطا مهم نیست.

جلسه چهارم:

LMS برخلاف پرسپترون میزان خطا در آن مهم هست در صورتی که در پرسپترون صرفا اگر خطا داشت میگفتی کم کن یا زیاد کن. در نتیجه همگرایی زودتر است. واسه همین هست که ما از تابع خطی استفاده میکنیم و feedback میدهیم. بعد حالا به صورت دو سطحی ها غیر خطی خروجی میدهیم. در واقع اولین تابع فعالیت ما خطی هست بر خلاف پرسپترون تا بتوانیم فیدبک بدهیم. نرخ آموزش بزرگی تغییرات وزن ها را مشخص میکند اگر زیاد باشد یعنی میزان تغییر وزن باید زیاد باشد.

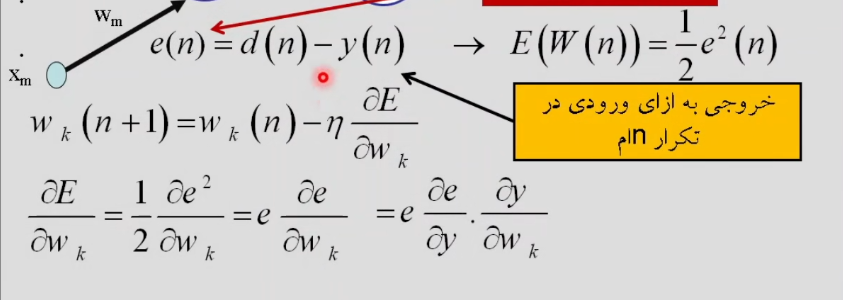
Epoch: یکبار همه ورودی ها را ببینی و خطا را حساب بکنی و MSE آن را بدست بیاوری بعد وزن ها را اپدیت بکنی، میشود یک دوره. البته batch گفته میشود به این سیستم که همه ورودی ها را ببینی و خطا را حساب بکنی و اپدیت وزن انجام بدهی که در مخالف sequence mode هست.

وزن ها پارامتر های آزاد و قابل تغییر ما هستند برای تابع خطا چون با تغییر وزن میتوانیم کاهش خطا بدهیم.

یک وزنی در نظر میگیریم تابع خطا را حساب میکنیم در عکس گرادیان یا همان مشتق وزن را بروز رسانی میکنیم یعنی اگر شیب منفی بود یا گرادیان منفی بود وزن را زیاد میکنیم و اگر شیب مثبت بود و گرادیان مثبت بود وزن را کم میکنیم. به این میگوییم steepest descent. اگر بیشتر از یک متغیر بود بردار گرادیان داریم.

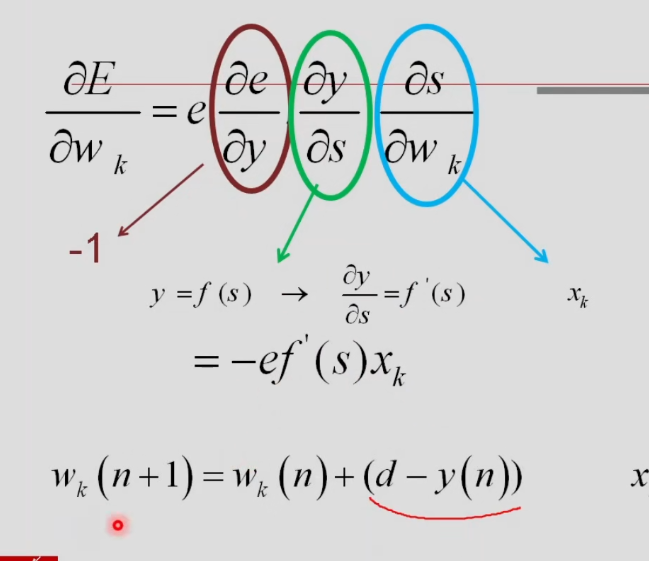
در نرخ یادگیری کم همگرایی کند و نرخ یادگیری زیاد ناپایدار میشود سیستم.

Sequential یعنی ورودی ها دونه به دونه میان و خطا حساب میشود و وزن اپدیت میشود. روند اپدیت شدن تابع خطا روند نرم تری در batch mode داریم ولی در sequential تابع ما زیاد نرم نیست. چون batch دارد میانگین خطا میگیرد دیگه.



در اینجا از مشتق زنجیره ای استفاده میکنیم بخاطر اینکه ما مشتق ارور را به صورت جزئی میخواهیم از وزن بگیریم اما مستقیم به آن دسترسی نداریم بلکه در y قرار دارد پس باید زنجیره ای بریم تا به w برسیم.

دقت کن d ثابت هست در حین مشتق گیری صفر میشود.



طبق عکس بالا واسه همین هست که مشتق تابع سیگموید در حین محاسبه خطا نیاز داریم.

پرسپترون 3 ورودی مرز جدا سازش صفحه میشود.

جلسه پنجم:

مسئله XOR جدایی پذیر خطی نیست پس با یک نورون نمیتوانیم این تابع را جدا کنیم و با یک خط نمیتوانیم پس به 2 خط حداقل نیاز داریم. میتوانیم یک نگاشت کنیم به یک فضای دیگه که جدای پذیر خطی میشود در نتیجه با یک نورون میتوانیم جدا کنیم آنها را. لایه میانی آن استخراج کننده فیچر هست که توانسته هست مسئله را ببرد به یک فضای دیگر که جدایی پذیر خطی هست. fully connected یعنی هر واحد قبلی به تمام واحدهای لایه بعدی متصل هست. تعداد واحد ها و لایه های مخفی مشخص است و تمامی اتصالات رو به جلو است یا feed forward. تابع فعالیت لایه های مخفی باید غیر خطی باشد. ترکیب یک سری لایه خطی و تابع خطی، خطی هست و دیگر عملا فایده ندارد اضافه کردن لایه های جدید و دیگر ورودی میشود یک ماتریس دیگر ماتریس بیشتر نداریم.

تک نورون: مسائل جدایی پذیر خطی را حل میکردیم که تک لایه هم گفته میشود.

دو لایه: یعنی یک لایه مخفی داشته باشیم به ازای هر نورون میتوانیم فضا را به 2 قسمت ترکیب بکنیم و نورون لایه آخر فقط خط ها را ترکیب میکند بسته به تعداد نود لایه مخفی خط ها زیاد یا کم میشوند.

سه لایه: اینجا ترکیبی از نواحی محدب داریم نسبت به لایه قبلی. و انگار یک سری ناحیه محدب میتوانیم داشته باشیم.

مشتق یک تابع نسبت به یک پارامتر یعنی تغییرات اون تابع نسبت به اون پارامتر.

پس انتشار خطا یا backpropagation:

اول forward:

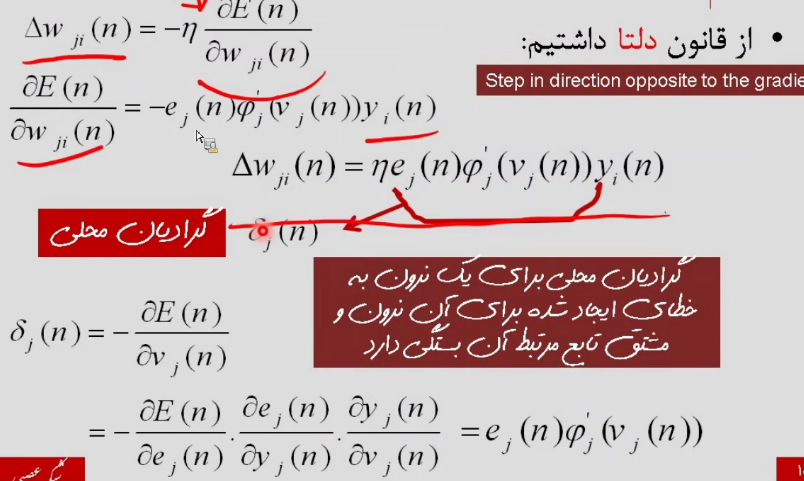
بردار ورودی به شبکه اعمال شده و خروجی واقعی محاسبه میشود.

Backward:

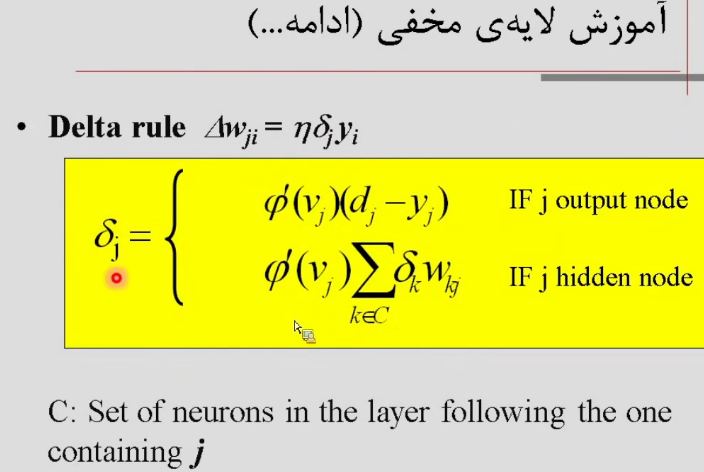
یک خروجی مطلوب داریم و یک خروجی واقعی که توسط شبکه ما تولید شده است که اختلاف این 2 تا را محاسبه میکنیم بر حسب تابع معیار سیگنالی متناسب با خطا تولید و لایه به لایه حرکت میکند و وزن را اصلاح میکند. پس از اصلاح وزن ها میگوییم iteration.

دقت کن در اسلاید میزان خطا نورون j ام نورون خروجی ما هست. خطا را برای اون نورون داریم حساب میکنیم. بعد که جمع میکنیم مجموع خطا های همه لایه های آخر هست یا خروجی به ازای هر ورودی و این لحظه ای است. حالا همین رو سیگما بگیریم تقسیم بر n کنیم میانگین خطا لحظه کل ورودی ها میشود نه فقط یک ورودی. دقت کن این با فرمول batch یکسان هست و چون میانگین میگیریم تابع خیلی نرم تر خواهد بود.   
خلاف جهت گرادیان یعنی به سمت مینیموم حرکت میکنی. به فرمول دلتا این شکلی نگاه کنی که با مشتق خطا با در نظر گرفتن وزن جهت مینیموم را پیدا میکنیم و با نرخ آموزش طول گام را مشخص میکنیم حالا به سمت منهای این باید برویم تا در جهت مینیموم باشد یا سمت خلاف آن.

گرادیان محلی را فقط یکبار محاسبه میکنی چون ثابت و مشترک هست.



نورون لایه قبلی به همه نورون های لایه بعد وصل هست و خطا آن نسبت به همه نورون های لایه بعد تاثیر گذار هست. چرا فرمول سیگما داریم روی ek؟ چون نورون لایه قبل به همه نورون های لایه بعد وصل هست پس باید مجموعه e ها را جمع کنیم برای نورون فعلی در لایه مخفی.



جلسه ششم:

Epoch: یک دوره کامل همه مجموعه آموزشی را دیده باشیم. iteration: در روش ترتیبی هست که به ازای هر ورودی که اصلاح وزن صورت میگیرد. روش ترتیبی و روش دسته ای تقریبا فرمول مشابه ای در اپدیت دارند.

شیوه ترتیبی به حافظه کمتری نیاز دارد چون برای دسته ای نیاز هست کل داده های آموزشی لود کنیم. در حالت دسته ای رسیدن به همگرایی مطمئن هست. روش ترتیبی نسبت به داده های تکراری عملکرد بهتری دارد چرا؟ چون دسته ای میانگین میگیرد و میانگین یک سری تکراری یک چیز است و فایده ای ندارد.

پردازش موازی در شیوه دسته ای مناسب تر هست. پس در کل بهتر هست یک ترکیبی از این 2 استفاده کنیم به نام mini batch که از یک سری دسته های کوچک تر تشکیل شده است. پیاده سازی ترتیبی ساده تر هست.

SGD: همان خلاف جهت گرادیان برای رسیدن مینیموم و این stochastic یعنی به صورت تصادفی داده ها وارد شوند و به جای استفاده از همه داده از یک زیر مجموعه به صورت تصادفی استفاده میکنیم.

Early stopping: اگر نسبت به مجموعه validation خطا نسبت به دفعه قبلی بیشتر شود یعنی مدل دارد به سمت overfit شدن حرکت میکند بنابراین فرآیند آموزش را متوقف میکنیم. به overfit شدن واریانس بالا هم گفته میشود و مدل مستعد overfit شدن است. high bias یعنی سوگیری مدل بالا هست یا underfit مستعد شدن هست و انگار مدل متعصب هست و خودش را تغییر نمیدهد و سوگیری و تعصب آن بالا هست.

جلسه هفتم:

3 فاکتور تصمیم پذیری: حجم مجموعه آموزشی 2. ساختار شبکه عصبی 3. پیچیدگی مساله که فقط مورد آخر دست ما هست و اگر پیچیدگی بالا باشد شبکه overfit میشود و به نویز هم حساس میشود.

Cross validation: کل مجموعه را آموزش نمیدهیم دو بخش میکنیم یک بخش آموزشی یک بخش validation ضمن اینکه برای توزیع داده ها نباید بهم بریزد و توزیع آنها باید حفظ شود و رندوم نباید باشد.

k-fold cross validation: در این کار مجموعه به k تا شکسته میشود و هر بار یک زیر مجموعه k مورد validation قرار میگیرد و k-1 دیگر برای آموزش میروند. برای محاسبه خطا کلی هم میانگین میگیریم.

n-fold cross validation: یا leave – one – out یعنی یکی رو بردار رو برا test بعد بقیه رو برای آموزش برمیدارد.

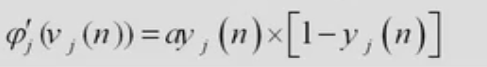
باید در cross validation تعداد افراز بالا باشد تا به یک جواب خاص بستگی نداشته باشد و median آن را گزارش میدهیم یا میانه آن ها را.

توابع فعال سازی باید مشتق پذیر و غیر خطی باشند.

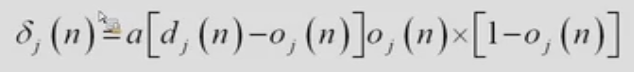
برای regression از توابع فعالیت خطی استفاده میکنیم. مقدار خروجی مطلوب باید توی برد تابع فعال سازی باشند و گرنه پارامتر های آزاد به سمت بی نهایت میروند چرا چون اگر بیشتر از برد باشد مشتق صفر میشود و اصلا آموزش صورت نمیگیرد و از طرف دیگر خروجی حالا از تابع فعال سازی بیشتر باشد تابع هعی میخواد بهش برسد پس باید وزن را بیشتر کند و نمیرسد ولی وزن به بینهایت میرود.

Vanishing gradient: در عمق شبکه که جلو میرویم و شبکه عمیق میشود گرادیان از یک محلی خیلی کوچک میشود چرا چون مشتق اون خیلی کوچک میشود و لایه های اولی خیلی خوب آموزش نمیبیند چون الگوریتم back propagation دارد خطا را بر میگرداند دیگه حالا فرض کن مشتق خیلی کوچک شود خوب مشخص است لایه های اول خوب آموزش نمیبیند.

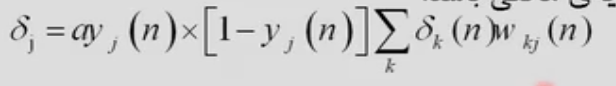
تابع سیگموید: مشتق این خیلی ساده است.

 مشتق تابع سیگموید. برای محاسبه گرادیان محلی.

فرمول گرادیان محلی با تابع سیگموید برای لایه آخر:



فرمول گرادیان محلی با تابع سیگموید برای لایه میانی:



در نتیجه اصلا دیگر نیازی به مشتق تابع سیگموید نداریم.

در تابع فعالیت tanh اگر به مقدار اشباع برویم یعنی خیلی نزدیک به 1 یا صفر آموزش کند میشود چون مشتق نزدیک به صفر میشود و vanishing gradient رخ میدهد.

Relu: بر خلاف tanh دیگر اشباع ندارد و عدم اشباع دارد. محاسبات ساده ای ام دارد. منتها منفی ها را اثر نمیدهد و نورون آموزش نمیدهد و نورون مرده است یعنی نورون هیچگاه فعال نمیشود و آموزش نمیبیند.

تاثیر نرخ آموزش:

پایین باشد دیر همگرا میشود. میتوان نرخ آموزش را متغیر در نظر گرفت یا وفقی یعنی مثلا در نزدیکی نقطه بهینه نرخ آموزش را کوچک کنیم. بردار گرادیان میخواهد بیشترین تغییرات را داشته باشد و ما باید خلاف آن حرکت کنیم پس باید بر روی کانتور ها بردار گرادیان را عمود کنیم تا بیشترین تغییر رخ بدهد . و دوست داریم این مسیر زودتر طی شود. اگر در حرکت ما به سمت بهینه تاریخچه کار را هم در نظر بگیریم و تاریخچه حرکت را همگرایی ما سریعتر میشود که به این momentum یا نرخ یادگیری وفقی گفته میشود.

Momentum:

اگر ثابت صفر باشد بروز رسانی عادی است اگر به یک باشد یعنی بیشتر از الگوی قبلی تبعیت کند. در صورتی که تغییرات هم جهت باشد در نظر گرفتن این باعث تسریع آموزش میشود که به accelerating effect گفته میشود. اگر تغییرات وزن در مرحله قبلی با مرحله فعلی هم جهت نباشد یعنی از مینیموم عبور کرده ایم و باید برگردیم و با در نظر گرفتن آن موجب پایداری میشود و این حرکت جهت مخالف صورت نمیگیرد. دقت کن هر چه قدر به ضریب آلفا توان بدهیم در هر مرحله باعث میشود تاثیرات اخیر را بهتر ببینیم و تاثیرات دورتر کمرنگ شوند یعنی یک پنجره ای از گذشته را استفاده میکنیم نه همه گذشته را. این ابزار برای تسریع همگرایی صورت میگیرد.

نرخ یادگیری خیلی پایین با ضریب مومنتوم خیلی بالا مناسب هست و باعث افزایش سرعت همگرایی میشود. در طرف مقابل ثبات یادگیری خیلی پایین یا صفر با نرخ یادگیری خیلی بالا مثل 1 مناسب هست و باعث ثبات یادگیری میشود. باید یک تناسبی بین این 2 باشد و گرنه اگر هر دو زیاد باشند باعث ناپایداری شبکه میشود و میزان خطا به صورت نوسانی تغییر کند.

اگر جهت حرکت نامناسب باشد ضریب مومنتوم را صفر بزار تا بدانی مسیر درست نیست و نرخ یادگیری را تغییر نده. اگر هم جهت بود یک مقداری نرخ یادگیری را زیاد تر بکن اما بصورت کلی مومنتوم را به مقدار اصلی برگردان. این شیوه برای آموزش دسته ای مفید است. در صورت استفاده در ترتیبی منجر به واگرایی میشود.

Annealing: به جای نرخ یادگیری وفقی نرخ آموزش به صورت نزولی در نظر گرفته میشود. بدین ترتیب در شروع نرخ یادگیری ثابت است بعد از رسیدن به محدوده مینیمم نرخ آموزش کاهش میباید. K هر چه قدر بزرگتر شود نرخ یادگیری کمتر میشود چون کسر بزرگتر میشود.

روش بهبود کارایی:

روش ترتیبی برای جاهایی که مجموعه آموزشی بزرگ داریم و افزونگی داده ها بالاست روش ترتیبی مناسب هست یعنی تکراری دارند این داده ها چون در دسته ای میانگین میگیریم برای تکراری ها این میانگین تاثیر خاصی ندارد. در مواردی که مجموعه آموزشی ثابت نیست و داده های جدید به مجموعه اضافه میشود هم روش ترتیبی مناسب است.

در آموزش شبکه بهتر هست از نمونه های آموزشی استفاده کنیم که بیشترین خطا را تولید میکنند. نمونه هایی که با هم متفاوت هستند استفاده کنیم تا طیف وسیعی از وزن ها مورد استفاده قرار بگیرد. نمونه های دشوارتر را به شبکه بیشتر اعمال کنیم. این روش ها در جهت بهبود کارایی است. 2 مشکل دارد استفاده از نمونه های دشوار تر : 1. ترتیب اعمال نمونه های آموزشی بهم میریزد. 2. Outlier ها اگر باشند باعث مشکل در تعمیم پذیری میشوند.

پیش پردازش ورودی:

داده های ورودی باید پیش پردازش بشوند و میانگین ورودی ها صفر شود و گرنه تابع خطا به سمت مینیموم زیگزاگی میشوند. بعد از، صفر کردن میانگین ورودی، ارتباط داده ها را از بین میبریم تا داده ها بهم ربطی نداشته باشند. در مرحله بعدی این داده را به شبکه میدهیم. چون اگر ورودی فقط از یک کلاس باشد قانون دلتا یا یهو بالا میره یا یهو پایین میره ولی اگه میانگین صفر باشد دلتا وزن معقول تر هست و زیگزاگی نمیشود و حرکت معقول میشود.

Covariance: داده ها را در هر جهت به واریانس تقسیم بکنی سرعت آموزش بهتر میشود چون پراکندگی داده ها در جهت یکسان میشود و اپدیت شدن تمام وزن ها به یک شیوه خواهد بود.

یکی دیگر از روش های بهبود کارایی مقدار دهی اولیه های وزن ها هست، چون از روی نقطه اولیه شروع میکنیم به مینیموم پس نقطه اولیه مهم هست اگر مقادیر بزرگ باشد میریم به ناحیه اشباع و آموزش کند میشود. اگر مقادیر کوچک باشد نزدیکی به مبدا داریم و مبدا به صورت نقطه زینی است این نقطه از هر دو جهت از یک جهت مینیموم داریم از یک جهت ماکسیموم و ما دنبال این نیستیم باید یک چیزی بین این 2 نقطه باشد.