به نام خدا

هانیه اسعدی ۹۹۵۲۱۰۵۵

۱-

۲- الف) در تابع فعال ساز خطی، خروجی توابع بین هیچ محدوده‌ای محدود نخواهد شد. در این تابع، فعالسازی متناسب با ورودی است و این را میتوان روی نورون های مختلف اعمال کرد و چند نورون را همزمان فعال کرد. وقتی چند کلاس داشته باشیم میتوانیم کلاسِ با مقدار بیشتر را انتخاب کنیم. اما مشتق این تابع ثابت است و به مقدار ورودی بستگی ندارد. وقتی هم که back propagation داشته باشیم، گرادیان یکسان خواهد بود و این یعنی خطا بهبود نمی‌یابد و مهم نیست چند لایه داشته باشیم، خروجی نهایی چیزی جز تبدیل خطی ورودی نیست.

در توابع فعال ساز غیر خطی، محدودیت های فعال ساز خطی رفع شده است و دیگر مشکل back propagation را نداریم. این توابع، در مرحله back propagation میتوانند مشخص کنند کدام وزن گره ورودی به تشخیص نهایی مدل میتواند کمک بهتری کند. خروجی توابع غیر خطی، ترکیبی غیر خطی از ورودی هاست که از لایه های متعدد رد شده‌اند. توبع غیر خطی تعمیم و تطبیق مدل با انواع داده ها و تمایز بین خروجی را آسان کرده‌اند.

[۱](https://abhigoku10.medium.com/activation-functions-and-its-types-in-artifical-neural-network-14511f3080a8#:~:text=The%20activation%20function%20does%20the,able%20to%20perform%20such%20tasks.) [۲](https://blog.faradars.org/%D8%AA%D8%A7%D8%A8%D8%B9-%D9%81%D8%B9%D8%A7%D9%84%D8%B3%D8%A7%D8%B2%DB%8C-%D8%AF%D8%B1-%D8%B4%D8%A8%DA%A9%D9%87-%D9%87%D8%A7%DB%8C-%D8%B9%D8%B5%D8%A8%DB%8C/)

ب) اگر وزن ها را صفر بدهیم، امکان گیر کردن در مینیمم محلی وجود دارد. ممکن است زمان زیادی برای آموزش نیاز داشته باشد و در مسائل پیچیده مشکل مشهود میشود. همچنین اگر همه نورون ها وزن یکسان داشته باشند، همه نورون ها از یک گرادیان برای تغییر استفاده میکنند و همه بطور یکسان تغییر خواهند کرد. پس بهتر است وزن ها رندوم و متفاوت داده شوند.

اگر بایاس صفر داده شود، در ابتدا نورون ها هیچ تاثیری در یادگیری ندارند. مدل تلاش میکند وزن ها را به گونه ای تغییر دهد که با داده های ورودی بهترین عملکرد را نشان دهد. البته تضمینی نیست که یک بهبود خواهیم داشت یا خیر.

بنابراین بهتر است هم بایاس و هم وزن ها بصورت رندوم در ابتدا مقدار دهی شوند تا مدل یادگیری بهتری داشته باشد.[۱](https://stats.stackexchange.com/questions/27112/danger-of-setting-all-initial-weights-to-zero-in-backpropagation) [۲](https://stackoverflow.com/questions/44883861/initial-bias-values-for-a-neural-network)

ج)پرسپترون: یک classifier خطی است، همگرا شدن آن تضمین نمیشود، اگر مسئله تفکیک پذیر خطی باشد آنرا حل میکند، نشانه ای از کیفیت راه حل پیدا شده نمی دهد. بنابراین مخصوصا برای مسائلی که تفکیک پذیر خطی نیستند، قابلیت تعمیم کمتری دارد.

آدالاین: یک مدل بهبود یافته از پرسپترون است که این هم فقط برای مسائل تفکیک پذیر خطی است. از یک تابع فعال ساز خطی استفاده میکند و وزن ها را براساس یکی ارور پیوسته آپدیت میکند. قابلیت تعمیم بیشتری نسبت به پرسپترون دارد ولی همچنان محدود به مسائل تفکیک پذیر خطی است.

مادالاین: مدل پیشرفته تر آدالاین است که از چند آدالاین استفاده میکند(چندین نورون خطی)، قابلیت تعمیم بیشتری دارد و میتواند مسائل پیچیده تر را حل کند. مخصوصا وقتی با شبکه ادغام میشود.

MLP: یک شبکه عصبی مصنوعی است که از چند نورون غیر خطی تشکیل شده است، روی مسائل با الگوهای پیچیده خوب عمل میکند و این باعث افزایش قابلیت تعمیم پذیری آن میشود.

د)مزایا: با استفاده از مشتق دوم سرعت آموزش مدل افزایش می‌یابد. روش های نزول گرادیان دو گانه به مدل کمک می کند تا به بهترین مینیمم برسند.

معایب: از پیچیدگی بیشتری برخوردارند و زمان بیشتری برای انجام محسابات لازم دارند. برای محاسبه مشتق دوم، نیاز به حافظه بیشتر داریم که برای داده های بزرگ مشکل ساز میشود. اگر داده ها نویز داشته باشند، مشتق دوم به درستی تخمین زده نمی شود و به مشکلاتی منجر میشود.

۳-

۴- یک مدل sequential می سازیم سپس لایه ها را به آن اضافه میکنیم. لایه اول، لایه ورودی است با 64 نورون، ورودی 25 بعدی و تابع فعال سازی ReL.

لایه دومی که اضافه میکنیم لایه hidden میانی است با 64 نورون و تابع فعال سازی ReLU. لایه سوم که همان لایه خروجی است، با 10 نورون(چون میخواهیم در 10 کلاس 0 تا 9 دسته بندی شوند) و تابع فعال سازی softmax. از تابع softmax استفاده میکنیم چون وقتی دسته بندی چند کلاسه داریم این تابع تضمین میکند جمع احتمالات کلاسها برابر یک شود و همچنین برای هر کلاس یک احتمال تعریف میکند و کلاسی که بیشترین احتمال داشته باشد، به عنوان خروجی بیان میشود.

Loss plot:

نمودار loss بر اساس هر epoch را نشان میدهد که:

خط آبی نشان دهنده مقدار loss روی داده آموزش است و نشان میدهد مدل چقدر با داده های آموزشی تطابق دارد و کاهش یافتن بعد ازepoch های متوالی یعنی مدل در حال یادگیری می باشد.

خط نارنجی مقدارloss روی داده تست را نشان میدهد و نشان میدهد مدل چقدر با داده های تست تطابق دارد و میتواند روی دیتاهای تست تعمیم پذیر باشد. اگر مدل دچار overfitting شود، در ابتدا شروع به افزایش می کند. فاصله بین این دو خط نشان دهنده overfitting مدل است.

Accuracy plot:

همانند نمودار قبلی، نمودار accuracy بر اساس هر epoch را نشان میدهد که:

خط آبی میزان دقت روی داده آموزش است و به مرور زمان به دلیل یادگیری مدل در حال افزایش است.

خط نارنجی میزان دقت روی داده تست است و قابلیت تعمیم آن روی داده تست را نشان میدهد. همانند نمودارloss، فاصله زیاد بین این دو خط نشان دهنده overfitting خواهد بود.