هوش محاسباتی – تمرین اول

سوال اول

الف) بدون استفاده از bias خروجی یک نورون برابر $y = wx$ میشود که معادل خطی است که با وجود هر ورودی دلخواه از مرکز محور اعداد عبور میکند. این روش انعطاف پذیری مدل را کاهش میدهد چون خط همواره از نقطه (0,0) عبور میکند. اما اگر bias داشته باشیم، خروجی نورون برابر $y = mx + b$ که بر میتوان انواع خطوط را رسم کرد.

از طرفی بودن 0 در مرکزیت خروجی های نورون میتواند باعث کوچک شدن بسیاری از وزن ها و خروجی ها شده که به مرور به صفر میل خواهند کرد. با افزودن bias مانع کوچک شدن بسیار این پارامتر ها میشویم.

ب) بدون استفاده از activation function ها، خروجی هر نورون لزوما خطی میشود و هر چقدر تعداد نورون ها، لایه ها و پیچیدگی مدل را افزایش دهیم، همچنان خروجی نهایی نیز خطی خواهد شد که باعث کاهش چشم گیر انعطاف پذیری مدل میشود. برای اثبات این موضوع به محاسبات زیر توجه کنید:

$$Hidden\ Layer\ 1: X.W1 + B1$$

$$Hidden\ Layer\ 2: (X.W1 + B1).W2 + B2$$

$$Output\ Layer: \big((X.W1 + B1).W2 + B2\big).W3 + B3$$

$$Symplified: (X.W1.W2.W3) + (B1.W2.W3 + B2.W3 + B3)$$

$$Simplified\ Further: (X.W) + B$$

پس با توجه به محاسبات بالا، جدا از تعداد لایه ها و پیچیدگی شبکه، خروجی همیشه خطی خواهد شد.

پ) تابع Sigmoid ورودی را بین 0 و 1 تبدیل میکند که در آن هیچ عددی 0 و 1 نمیشوند، بلکه اعداد بزرگ به سمت این دو انتها پیش میروند. در تابع ReLU مقادیر مثبت را عینا خروجی داده و مقادیر منفی را 0 میکند.

تابع Sigmoid به دلیل ذات فرمول، برای لایه های انتهایی شبکه های Binary Classification استفاده میشود، در صورتی که تابع ReLU معمولا در لایه های پنهان شبکه های عصبی مورد استفاده قرار میگیرند.

یکی از مشکات بزرگ تابع Sigmoid، مفهومی به نام Vanishing Gradient Decent است که در طول زمان وزن های شبکه به مرور کوچک شده و به صفر میل میکنند. از طرفی این تابع صرفا برای شبکه های Binary Classification مورد استفاده قرار گرفته و در شبکه های Multilabel Classification استفاده ای ندارند.

از مشکل مطرح تابع ReLU میتوان به صفر کردن مقادیر منفی اشاره کرد که باعث میشود شبکه نسبت به مقادیر منفی واکنش یکسان و خنثی داشته باشد که از میزان یادگیری مدل کم میکند که اصطلاحا به آن Dying ReLU گفته میشود. برای حل این مشکل از Leaky ReLU استفاده میشود تا برای مقادیر منفی، شیب منفی کمی در نظر گرفته شود.

ت) مفهوم Regularization و Dropout هر دو برای جلوگیری مدل شبکه عصبی از overfit شدن استفاده میشوند. در Regularization، همانطور که از اسم آن پیداست، به وزن های شبکه پنالتی وارد کرده که بر حسب میزان بزرگ بودن آن وزن ها افزایش میابد (بزرگ شدن برخی وزن ها نشانه overfit شدن شبکه است). این پنالتی بر روی Loss Function اعمال میشود و میتواند انواع مختلفی داشته باشد از جمله L1، L2 و L1+L2 که در آن L1 مقدار مجموع قدر طلق وزن ها، و L2 مقدار مجموع به توان ۲ رسیده شده وزن ها را به Loss Function اضافه میکند که مانع افزایش غیرعادی برخی وزن ها میشود.

هوش محاسباتی – تمرین اول

در مفهوم Dropout، برخی وزن ها را در مرحله Feedforward را در Epoch ها نادیده گرفته یا اصطلاحا آموزش نمیدهم (نمیگذاریم دیتا جدید را ببینند). با این کار اگر بخشی از دیتا ورودی قابلیت overfit کردن مدل را داشته باشد، تعدادی نورون هایی هستند که این دیتا bias شده را ندیده اند و در نتیجه از overfit شدن مدل می کاهد.

سوال دوم

الف) با استفاده از رابطه کلی زیر، محاسبه میکنیم:

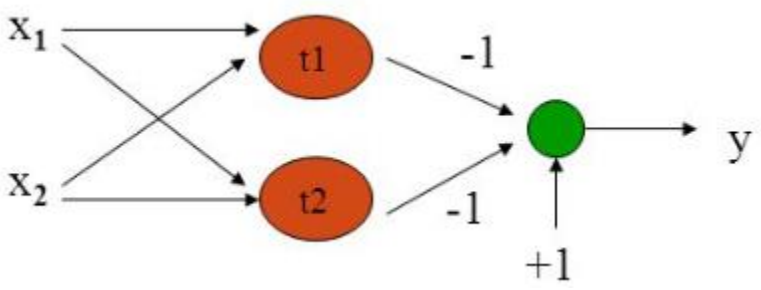
$$Output\ Size = \frac{(N + 2.P - F)}{Stride} + 1$$

حال به جایگذاری داریم:

$$Layers: \frac{128 + 4 - 7}{3} + 1 = 42$$

$$Parameters: (7.7.3 + 1).10 = 1480$$

ب) برای مدل کردن XOR نیاز به دو RBF داریم. از طرفی میدانیم $\mu = e^{-\gamma|x-V|^2}$ است و برای کوچک کردن شعاع باید y را مقدار بزرگی بگذاریم (من 100 فرض کردم و وزن ها را برابر 1 گذاشتم تا محاسبات کمی ساده تر شود). همچنین دو دایره به مراکز (0,0) و (1,1) داریم که در شکل مقابل نشان داده میشوند:



حال برای هر جفت ورودی، طبق فرمول کلی زیر، خروجی ها را محاسبه میکنیم:

$$Y = W1.e^{-\gamma|x-V1|^2} + W2.e^{-\gamma|x-V2|^2} + 1$$

$$Simplified: Y = e^{-\gamma|x-V1|^2} + e^{-\gamma|x-V2|^2} + 1$$

$$Y(0,0) = 0$$

$$Y(0,1) = 1$$

$$Y(1,0) = 1$$

$$Y(1,1) = 0$$

سوال سوم

$$Cost = (O1 - Yt)\ \hat{}\ 2$$

هوش محاسباتی – تمرین اول

$$Derivative\ of\ "Cost"\ =\ 2(O1-Yt)*\frac{\partial_{O1}}{\partial U0}$$

$$Simplified\ =\ 2(O1-Yt)*\frac{\partial_{O1}}{\partial U(Bt+b3)}$$

$$y=\frac{\partial\ Cost}{\partial\ U0}=2(O-Yt).\sigma(Bt+b3).\big(1-\sigma(Bt+b3)\big).t0.\sigma(AV+b2).\big(1-\sigma(Av+b2)\big).v1\sigma(UX+b1).\big(1-\sigma(UX+b1)\big).X0$$

سوال چهارم

الف) یکی از دلایل نوسانات بالا در Loss Function، قرار دادن batch size بسیار کوچک است. به طوری که در هر Epoch فقط تعداد خیلی کمی از دیتا را مشاهده کنیم که باعث میشود تغییرات کوچک تاثیر بسزایی در مدل بگذارد. به عبارتی این batch های کوچک میتوانند بسیار متفاوت از یکدیگر باشند که باعث اتفاق افتادن این نوسانات میشود. طبق تئوری معروفی هرچه مقدار نمونه بیشتر باشد توزیع به توزیع نرمال مشابه تر شده و در نتیجه این نوسانات کاهش میابد، پس میتوانیم batch size را افزایش دهیم.

از طرفی مدل میتواند بسیار پیچیده تر از دیتا موجود باشد، به طوری که بالا بودن بیش از نیاز ظرفیت مدل برای یادگیری الگو ها در دیتا های نسبتا کوچک تر از مورد نیاز مدل، باعث میشود مدل الگو های جدیدی در هر Epoch توسط مدل یاد گرفته شود. برای حل این مشکل میتوان تعداد پارامتر های شبکه را کاهش داد (تعداد لایه ها یا نرون ها)

ب) میدانیم اگر Learning Rate خیلی کم باشد، مدل همگرا خواهد بود اما زمان زیادی برای همگرا شدن نیاز خواهد داشت. هرچه Learning Rate را افزایش دهیم سرعت همگرایی بیشتر خواهد شد و اگر Learning Rate از حدی بیشتر شود، مدل می تواند واگرا شود. با این فرض مشاهده میشود که خط زیگزاگ مشکی مربوط به LR زیاد است که کمی واگرایی در آن مشاهده میشود. خط قرمز به سرعت کاهش یافته و ثابت شده که نشان دهنده همگرا بودن سریع مدل است اما با LR کمتر نسبت به خط زیگزاگ مشکی. در آخر خط آبی که با سرعت کمی به همگرایی میل میکند، کمترین LR را دارد. به عبارتی،

$$Learning\ Rates: Z > X > Y$$

پ) زمانی مدل overfit میشود که بیش از حد پیچیده باشد و الگو های ناخواسته دیتا را یاد بگیرد. این پیچیدگی ناشی از تعداد زیاد نرون ها و لایه ها بوده، پس با کاهش تعداد لایه ها و نرون ها میتوانیم به کمتر/دیرتر شدن overfit کمک کنیم.

از طرفی زمانی overfit مشاهده میشود که مدل variance بالا و bias کم نشان دهد. برای توضیح بیشتر، افزایش variance در دیتا باعث خوب بودن دقت مدل بر روی training data و بد بودن این دقت بر روی test data میشود که نشانه واضح overfit است. از آنجایی که variance و bias مخالف یکدیگر هستند، پس در صورت افزایش variance لزوما bias کاهش میابد.

ت) پیشبینی قیمت کالا از جنس Regression است، پس مدل Regression نیاز داشته و به طبع activation function های مرتبط با این موضوع باید استفاده شوند. بین توابع معرفی شده، دو تابع Sigmoid و Tanh توابع فعالیت مسئله Classification بوده و مناسب مسئله ما نیستند. از طرفی تابع ReLU معمولا در لایه های پنهان (میانی) استفاده میشود، پس تنها گزینه مانده Linear است.