دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

عنوان

نگارش
بردیا اردکانیان

استاد
دکتر محمدمهدی عبادزاده

آبان ۱۴۰۱

سوال ۱.

الف) با ذکر دلیل بیان کنید چرا افزودن بایاس به یک نورون عملکرد آن را بهبون می‌بخشد؟

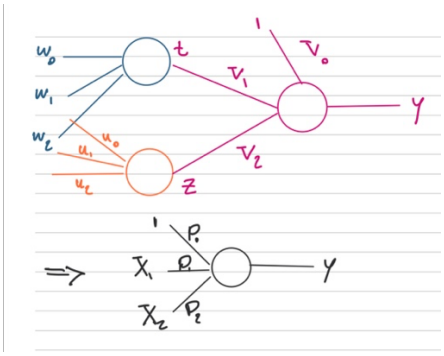
بایاس همانند وزن‌ها، یادگیرنده است. به این معنی که در طول اجرای شبکه یادگرفته می‌شود و مقدار آن تغییر می‌کند. بدون استفاده از بایاس خروجی یک نورون برابر $y = wx$ می‌شود. در نتیجه مدل رفتار خطی از خود نشان می‌دهد و انعطاف پذیری مدل را کاهش می‌دهد (همواره از نقطه $(0, 0)$ عبود می‌کند. اگر بایاس داشته باشیم خروجی نرون به شکل $y = mx + b$ می‌شود که انعطاف پذیری مدل را افزایش می‌دهد. در واقع بایاس انعطاف شبکه را برای فیت شدن به داده افزایش می‌دهد. به این معنی که در ابتدا بگوییم اگر مقدار خروجی نرون صفر بود، نورون غیر فعال است ولی بایاس می‌تواند این نتیجه را تغییر دهد. از طرفی بودن ۰ در مرکزیت خروجی‌های نورون می‌تواند باعث کوچک شدن بسیاری از وزن‌ها و خروجی‌ها شده که به مرور به صفر میل خواهند کرد. با افزودن بایاس مانع کوچک شدن بسیار این پارامترها می‌شویم.

ب) با ذکر مثال و انجام محاسبات توضیح دهید که در صورت عدم استفاده از توابع فعالیت و یا استفاده از توابع فعالیت خطی برای

همه‌ی لایه‌ها در یک شبکه‌ی پرسپترون چند لایه، چه اتفاقی می‌افتد.

در صورتی که تابع فعالیت نداشته باشیم و یا برای تمامی لایه‌ها از توابع فعالیت خطی استفاده کنیم، ترکیب خطیِ ترکیب‌های خطی حاصل، در نهایت برابر با یک ترکیب خطی جدید می‌شود و این به این معناست که انگار در عمل، تنها یک پرسپترون وجود دارد و نمی‌توان به معنای واقعی یک شبکه از پرسپترون‌ها داشت.

$$t = w_0 + x_1 w_1 + x_2 w_2, \quad z = u_0 + x_1 u_1 + x_2 u_2$$
$$y = v_0 + t v_1 + z v_2$$
$$y = v_0 + w_0 v_1 + u_0 v_2 + (w_1 v_1 + u_1 v_2) x_1 + (w_2 v_1 + u_2 v_2) x_2$$
$$y = p_0 + x_1 p_1 + x_2 p_2$$



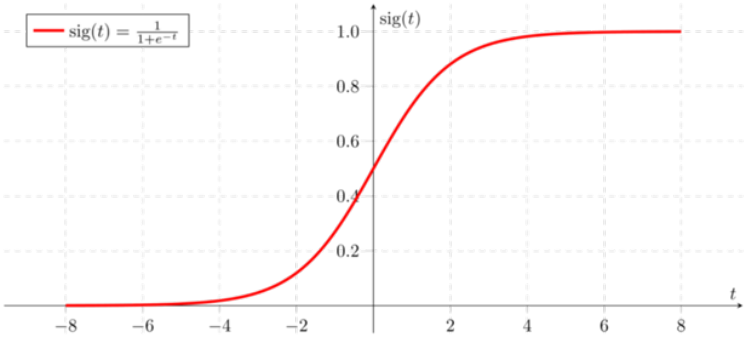
بدون استفاده از تابع فعالیت، نورون فقط یک تبدیل خطی را روی ورودی‌ها انجام می‌دهد این امر سبب موارد زیر می‌شود.

۱. لایه‌های پنهان بی استفاده می‌شوند. چون مجموع تمام تبدیلات خطی در لایه‌های مختلف معادل در نهایت منجر به یک تبدیل خطی خواهد شد.
۲. دیگر رابطه‌های غیرخطی را نمی‌تواند پیدا کند در نتیجه قدرت شبکه پایین می‌آید.

پ) از توابع فعالیت معروف می‌توان به سیگموید و رلو اشاره کرد. این دو تابع را با هم مقایسه کنید و نقاط ضعف هریک را بیان کنید.

تابع سیگموید

تابع Sigmoid به عنوان تابع لجستیک شناخته می‌شود که به عادی سازی خروجی هر ورودی در محدوده بین ۰ تا ۱ کمک می‌کند. هدف اصلی تابع فعال سازی حفظ خروجی یا مقدار پیش بینی شده در محدوده خاص است که باعث بازدهی خوب و دقت مدل می‌شود.



معایب: تابع سیگموید مقادیر بین ۰ و ۱ را اختیار می‌کند. همچنین می‌دانیم که گرادیان این تابع به ازای مقادیر نزدیک به مثبت و منفی بینهایت به صفر میل می‌کند. مشکل عمده این تابع به خصوص در شبکه‌های عمیق، gradient vanishing است؛ به این معنا که در هنگام backpropagation، هنگامی که به تدریج از لایه‌های پنهان نزدیک به خروجی به سمت لایه‌های پنهان نزدیک به ورودی حرکت می‌کنیم، در اثر ضرب گرادیان‌های بین صفر و یک، مقادیر گرادیان بسیار کوچک و نزدیک به صفر می‌شود و این امر باعث میشود که وزن‌ها به سختی تغییر کنند و بنابراین یادگیری بسیار بسیار کند شود.

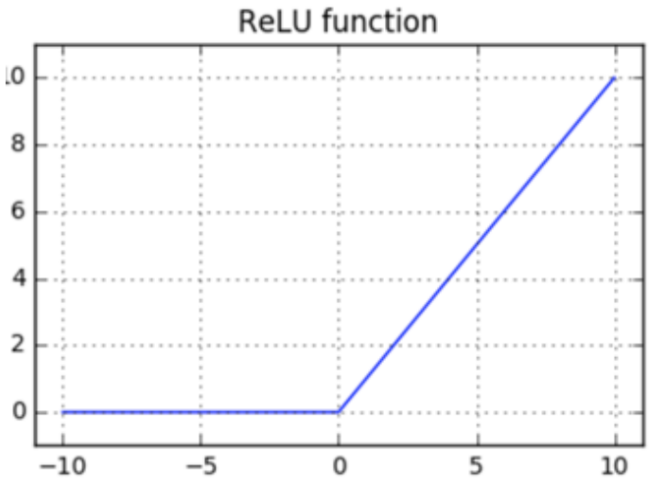
اشکال دیگر سیگموید که نسبت به مورد قبلی اهمیت کمتری دارد، این است که خروجی‌های این تابع اصطلاحا non-zero centered هستند که این مطلب روی نحوهی به روز رسانی شدن وزن‌ها و در نتیجه حرکت به سمت جواب تاثیر می‌گذارد. هنگامی که داده‌ای که به یک نورون وارد می‌شود مثبت است، همه‌ی گرادیان‌ها نسبت به وزن‌ها با یکدیگر هم‌علامت می‌شوند. اگر فرض کنیم که دو وزن داریم، در این حالت گرادیان نسبت به این دو یا هر دو مثبت است و یا هر دو منفی و بنابراین در صفحه یا می‌توانیم به سمت شمال شرق حرکت کنیم یا جنوب غرب. حال اگر فرض کنیم که نقطه بهینه در شمال غرب باشد، حرکت ما به سمت این نقطه زیگ زاگی می‌شود. این حرکات زیگ زاگ معمولا بهینه‌سازی را دشوار می‌کنند. استفاده کردن از optimizerهایی مانند Adam به این امر بهبود می‌بخشند.

همچنین همانطور که از نمودار تابع سیگموید واضح است این تابع در مرزهایش اشباع می‌شود به این معنا که کاملا افقی می‌شود. در آن نواحی گرادیان تقریبا صفر است و شبکه یاد نمی‌گیرد.

مزایا: ۱) پیوسته است و گرادیان پیوسته می‌دهد. که در نتیجه آن تمام نرخ‌های اکتیو شدن یک نورون از هیچی اکتیو نشدن تا کاملا اکتیو شدن را دارد. ۲) از بهترین تابع‌های نرمال شده. ۳) یک پیش بینی (کلاس بندی) دقیق می‌دهد. ۴) به خوبی، غیرخطی بودن در فضای ورودی را نشان می‌دهد.

تابع رلو

ReLu بهترین و پیشرفته ترین تابع فعال سازی در حال حاضر در مقایسه با سیگموئید و TanH است زیرا تمام ایراداتی مانند مشکل گرادیان ناپدید شده به طور کامل در این تابع فعال سازی حذف شده است که این عملکرد فعال سازی را در مقایسه با سایر عملکردهای فعال سازی پیشرفته تر می‌کند.



معایب: 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 ReLU Leaky استفاده می‌شود تا برای مقادیر منفی، شیب منفی کمی در نظر گرفته شود.

وقتی یک تابع فعالیت قسمت بزرگی از ورودی را مجبور می‌کند تا صفر یا تقریبا کاملا صفر شوند، نورون‌های مربوط در شرکت در خروجی غیرفعال یا مرده می‌شوند. هنگام اپدیت وزن، این امکان وجود دارد که وزن‌ها، به صورتی اپدیت شوند که جمع وزن دار قسمت بزرگی از شبکه مجبور به مقدار صفر شود. یک شبکه، به سختی می‌تواند، از چنین شرایطی مجددا بازگشت کند و قسمت بزرگی از ورودی از شرکت در شبکه ناتوان می‌ماند. این باعث یک مشکل بزرگ می‌شود به خاطر اینکه قسمت بزرگی از ورودی، هنگام اجرای شبکه، کاملا غیرفعال باقی می‌ماند. به این مشکل، نورون‌های مرده می‌گویند.

مزایا: ۱) در اینجا تمام مقادیر منفی به ۰ تبدیل می‌شوند، بنابراین هیچ مقدار منفی در دسترس نیست. ۲) مقادیر حداکثر آستانه Infinity هستند، بنابراین مشکلی در مورد گرادیان ناپدید وجود ندارد، بنابراین دقت پیش‌بینی خروجی و بازده حداکثر است. ۳) سرعت در مقایسه با سایر عملکردهای فعال سازی سریع است.

ت) مفاهیم dropout و regularization وکاربرد آن‌ها را در شبکه عصبی را توضیح دهید.

مفهوم Regularization و Dropout هر دو برای جلوگیری مدل شبکه عصبی از overfit شدن استفاده می‌شوند.

Regularization، روش Regularization ، راهی برای پیدا کردن یک tradeoff variance-bias خوب هست که با تنظیم پیچیدگی مدل اتفاق میفتد. رایج ترین شکل regularization ، به اصطلاح L۲ Regularization نامیده می‌شود اما نسخه L۱ آن نیز موجود است که فرمول‌های آن به شرح زیر است:

L1

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) + \frac{\lambda}{m}\sum_{j=1}^{n} |\theta_j|$$

L2

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

L1 Regularization	L2 Regularization
1. L1 penalizes sum of absolute values of weights.	1. L2 penalizes sum of square values of weights.
2. L1 generates model that is simple and interpretable.	2. L2 regularization is able to learn complex data patterns.
3. L1 is robust to outliers.	3. L2 is not robust to outliers.

لاندا پارامتری است که وابسته به شرایط می‌تواند تنظیم شود یعنی مقدار بالای وزن‌ها با درنظر گرفتن مقدار بالایی برای لاندا قابل کنترل خواهد بود و بطور مشابه مقدار کم برای لاندا به منظور تنظیم مقدار کم وزن‌ها درنظر گرفته می‌شود.

Dropout، در این روش برای همه نورون‌ها به غیر از نورون‌های آخر یک عدد تصادفی تولید می‌کنیم. آن نورون‌هایی که عدد تصادفی آنها کمتر از ۰.۵ است را علامت گذاری کرده و بعد تمام وزن‌های ورودی و خروجی به آنها را حذف می‌کنیم. با این کار نقش نورون‌های بلا استفاده را حذف کرده و شبکه را سبکتر می‌کنیم. در نتیجه منحنی تولید شده پیچیده نیست و بیش برازش رخ نمی‌دهد.

سوال ۲.

الف)

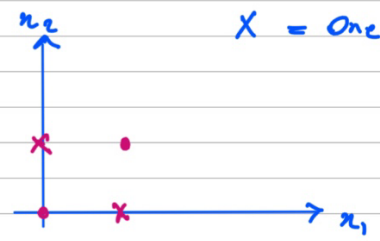
we have Output Size $= \frac{(N + 2.P - F)}{Stride} + 1$

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

Params $= (7 \times 7 \times 3 + 1) \times 10 = 1480$

ب)

x_1	x_2	y
1	1	0
1	0	1
0	1	1
0	0	0



X = One , O = Zero

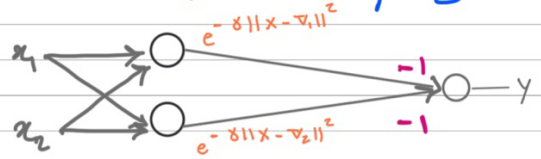
- با یک پرسپترون نمی‌توان مسئله را حل کرد. به RBF با توابع Base function نیاز داریم.

$\mu = e^{-\gamma \|x - v\|^2}$

باید برای کدام γ بزرگی انتخاب کرد.

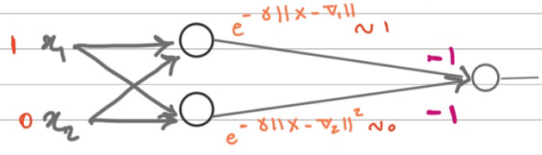
$v_1 = \begin{vmatrix} 1 \\ 1 \end{vmatrix}$

$v_2 = \begin{vmatrix} 0 \\ 1 \end{vmatrix}$



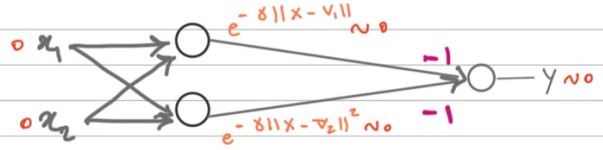
$\gamma = 4$

با این صل در نواحی که X داریم خروجی ۱ و غیر آن ۰ است.



$\gamma = 4$

و برای ۰،۰ داریم



سوال ۳.

الف)

الف)

$$\frac{\partial O_1}{\partial o_1} = \sigma(o_1)(1-\sigma(o_1)) \qquad C = (O_1 - Y_t)^2$$

$$C' = 2(O_1 - Y_t)$$

$$\frac{\partial o_1}{\partial B_0} = t_0$$

$$\frac{\partial B_0}{\partial b_0} = \sigma(b_0)(1-\sigma(b_0))$$

$$\frac{\partial b_0}{\partial A_1} = V_1$$

$$\frac{\partial A_1}{\partial a_1} = \sigma(a_1)(1-\sigma(a_1))$$

$$\frac{\partial a_1}{\partial u_0} = X_0$$

$$\frac{\partial L}{\partial u_0} = C' \times \frac{\partial O_1}{\partial o_1} \times \frac{\partial o_1}{\partial B_0} \times \frac{\partial B_0}{\partial b_0} \times \frac{\partial b_0}{\partial A_1} \times \frac{\partial A_1}{\partial a_1} \times \frac{\partial a_1}{\partial u_0}$$

$$= 2(O_1 - Y_t) \cdot \sigma(o_1)(1-\sigma(o_1)) \cdot t_0 \cdot \sigma(b_0)(1-\sigma(b_0)) \cdot V_1 \cdot \sigma(a_1)(1-\sigma(a_1)) \cdot X_0$$

ب)

$$a = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} b_0 & w_0 & w_1 & w_2 \\ b_1 & u_0 & u_1 & u_2 \end{bmatrix} \begin{bmatrix} 1 \\ X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} .6 & .4 & .3 & .2 \\ .2 & .3 & .4 & .6 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1.1 \end{bmatrix}$$ (ب)

$$A = G(a) = G\left(\begin{bmatrix} 1 \\ 1.1 \end{bmatrix}\right) = \begin{bmatrix} .73 \\ .75 \end{bmatrix}$$

$$b = [b_0] = \begin{bmatrix} b_2 & V_0 & V_1 \end{bmatrix} \begin{bmatrix} 1 \\ A_0 \\ A_1 \end{bmatrix} = \begin{bmatrix} .5 & .6 & .7 \end{bmatrix} \begin{bmatrix} 1 \\ .73 \\ .75 \end{bmatrix} = \begin{bmatrix} 1.463 \end{bmatrix}$$

$$B = G(b) = G\left(\begin{bmatrix} 1.463 \end{bmatrix}\right) = \begin{bmatrix} .75 \end{bmatrix}$$

سوال ۴.

الف) دلیل اولی که به ذهن می‌آید پیچیده تر بودن مدل داده موجود می‌باشد. به طوری که بالا بودن بیش از نیاز ظرفیت مدل برای یادگیری الگوها در داده‌های کوچک‌تر، باعث می‌شود مدل، الگوهای جدیدی در هر Epoch یاد بگیرد. برای حل این مشکل میتوان تعداد پارامترهای شبکه را کاهش داد. به عنوان مثال اگر مدل پیچیدگی طبقه بندی عکس‌های ۱۰۲۴*۱۰۲۴ را داشته باشد و در ورودی عکسی با فیچرهای کمتر داده شود، مدل در هر تغییر داده الگوری جدیدی را یاد می‌گیرد.

یکی از دلایل نوسانات در Loss Function، قرار دادن batch size بسیار کوچک است. به طوری که در هر Epoch فقط تعداد خیلی کمی از داده را مشاهده کنیم که باعث می‌شود تغییرات کوچک تاثیر بسزایی در مدل بگذارد. هرچه مقدار نمونه بیشتر باشد توزیع به توزیع نرمال مشابه تر شده و در نتیجه این نوسانات کاهش میابد.

ب) اگر نرخ یادگیری خیلی کم باشد مدل همگرا خواهد بود و این همگرایی زمان زیادی را صرف می‌کند. اگر نرخ یادگیری را افزایش دهیم زمان صرف شده برای همگرایی کم و کمتر خواهد شد. در صورتی که نرخ همگرایی از ترشهولدی بیشتر شود مدل می‌تواند واگرا شود. در نتیجه خط زیگزاگ مشکی مربوط به نرخ یادگیری زیاد است که واگرایی در آن مشاهده می‌شود. ممکن است با صرف کردن زمان بیشتر به همگرایی برسد.خط قرمز به سرعت همگرا شده است که نشان دهنده نرخ یادگیری زیاد ولی کماکان کمتر از خط مشکی می‌باشد. خط آبی نیز با سرعت کمتری به همگرایی می‌رسد از همه نرخ یادگیری کمتری دارد.

نرخ یادگیری: مشکی > قرمز > آبی

پ) زمانی که مدل بیش از حد پیچیده باشد و نسبت به کوچک ترین نوسانات عکس‌العمل تند نشان دهد (به عبارت دیگر از الگوهای ناخواسته داده بگیرد) بیش برازش رخ داده است. این پیچیدگی ناشی از تعداد زیاد نرون‌ها و لایه‌ها می‌باشد. پس با کاهش تعداد لایه‌ها و نرون‌ها می‌توانیم بیش‌برازش را به **تعبیق** بی‌اندازیم.

زمانی بیش برازش مشاهده می‌شود که مدل واریانس بالا و بایاس کم نشان دهد. افزایش واریانس در داده باعث خوب بودن دقت مدل بر روی داده آموزشی و بد بودن این دقت بر روی داده تست می‌شود که نشانه بیش برازش می‌باشد. همچنین واریان و بایاس با هم در تضادند پس افزایش واریانس لزوما بایاس را کاهش می‌دهد.

ت) پیش‌بینی قیمت از جنس مسائل رگریسون می‌باشد. پس به تبع مسئله می‌بایست توابع فعال‌سازی مربوط به رگریسون استفاده شوند. بین توابع معرفی شده دو تابع سیگموید و Tanh توابع مسئله طبقه‌بندی هستند و مناسب مسئله ما نیستند. از طرفی تابع رلو معمولا در لایه پنهان استفاده می‌شود. در نتیجه تنها گزینه تابع فعالیت **خطی** می‌باشد.