Assignment NO.1 Solutions

Neural Networks | Fall 1400 | Dr.Mozayani

Teacher Assistants:

Mohammad Hossein Khojaste

Amir Hossein Aminimehr

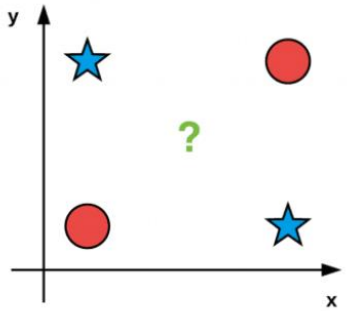
Student name : **Amin Fathi**

Student id : **400722102**

Problem 1.a

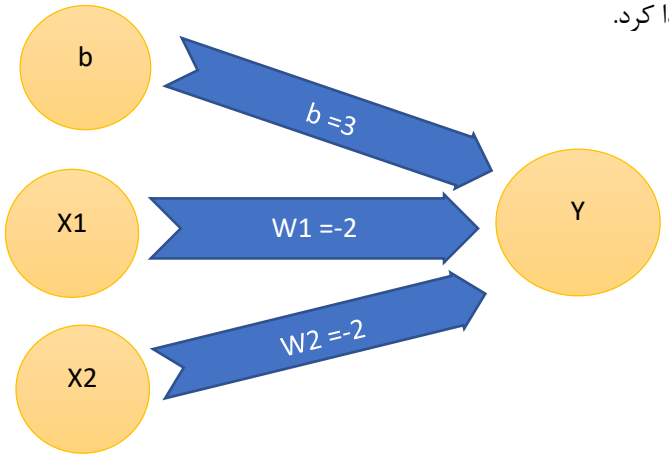
این تابع منطقی درواقع یک تابع XNOR می باشد ، که آن را نمیتوان با یک نورون پیاده سازی کرد ، چرا که برای پیاده سازی آن می بایست نقاط (۰,۰) و (۱،۱) را در یک کلاس قرار داد و نقاط (۱, ۰) و (۱،۰) را در یک کلاس قرار داد که این عمل با کشیدن یک خط در صفحه مختصات ممکن نیست.

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



Problem 1.b

این تابع منطقی در واقع یک NAND می باشد ، و میتوان آن را با یک نورون ساده پیاده سازی کرد ، چرا که برای پیاده سازی آن فقط میبایست نقاط (۰,۰) و (۱,۰) و (۰,۱) را توسط یک خط از نقطه (۱,۱) جدا کرد.



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

W1 =-2 و W2 =-2 در نظر میگیریم و مقدار Threshold را هم برابر با ۳ در نظر میگیرم که در آن صورت به ازای هر ٤ حالت معادله برقرار است . (کافیست شرایط زیر برقرار باشد) (مقدار threshold برابر منفی مقدار b است) در واقع باید ضرایبی انتخاب شوند برای w1 و w2 که وقتی w1 *x1 + w2 *x2 شد به ازای جفت x1 ها وx2 های (۰ و ۰) و (۰ و ۱) و (۱و۰) مقدارش از threshold بیشتر شود ، و برای (۱و۱) از آن کمتر شودو

$W1\times0 + W2\times0 - b \geq 0 \quad \Rightarrow \quad b \geq 0$

$W1\times0 + W2\times1 - b \geq 0 \quad \Rightarrow \quad b \geq -W2$

$W1\times1 + W2\times0 - b \geq 0 \quad \Rightarrow \quad b \geq -W1$

$W1\times1 + W2\times1 - b \leq 0 \quad \Rightarrow \quad b \leq -(W1+W2)$

Problem 2

X1	X2	bias	target
+1	+1	+1	+1
+1	-1	+1	+1
-1	+1	+1	+1
-1	-1	+1	-1

$$\mathbf{Y = b + \sum xi*wi}$$

$$\mathbf{b(new) = b(old) + \eta(d - y)}$$

$$\mathbf{w_i(new) = w_i(old) + \eta(d - y)x_i}$$

$$w1 = w2 = b = 0.2$$

$$\eta = 0.1$$

step1:

y = 0.2 + 1*0.2 + 1*0.2 = 0.6

b(new) = 0.2 + 0.1*(1-0.6) = 0.24

w1(new) = 0.2 + 0.1*(1-0.6)*1= 0.24

w2(new) = 0.2 + 0.1*(1-0.6)*1 = 0.24

y = 0.24 + 1*0.24 -1*0.24 = 0.24

b(new) = 0.24 + 0.1*(1-0.24) = 0.316

w1(new) = 0.24 + 0.1*(1-0.24)*1 = 0.316

w2(new) = 0.24 + 0.1*(1-0.24)*-1 = 0.164

y = 0.316 -1*0.316 +1*0.164 = 0.164

b(new) = 0.316 + 0.1*(1-0.164) = 0.3996

w1(new) = 0.316 + 0.1*(1-0.164)*-1 = 0.2324

w2(new) = 0.164 + 0.1*(1-0.164)*1 = 0.2476

y = 0.3996 -1*0.2324 -1*0.2476 = -0.0804

b(new) = 0.3443 + 0.1*(-1+0.0804) = 0.30764

w1(new) = 0.2324 + 0.1*(-1+0.0804)*-1 = 0.32436

w2(new) = 0.2476 + 0.1*(-1+0.0804)*-1 = 0.33956

Problem 3.a

اولا، مقادیر خروجی یک پرسپترون به دلیل تابع انتقال (transformation) ، می‌توانند تنها یکی از دو مقدار (۰ یا ۱) را خروجی دهد.

دوما ، پرستپترون فقط برای مجموعه تفکیک پذیری خطی (linear seperable) ها همگراست و می تواند کلاس بندی را انجام دهد

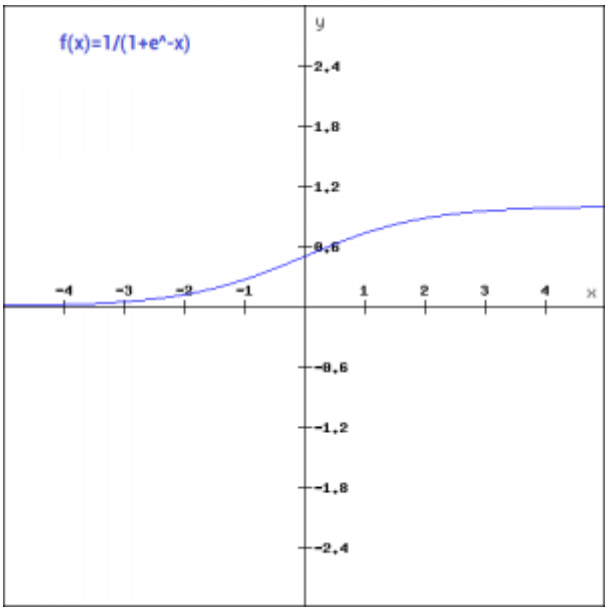
سوما ، پرسپترون توانایی تضمین کیفیت جواب را ندارد .

Problem 3.b

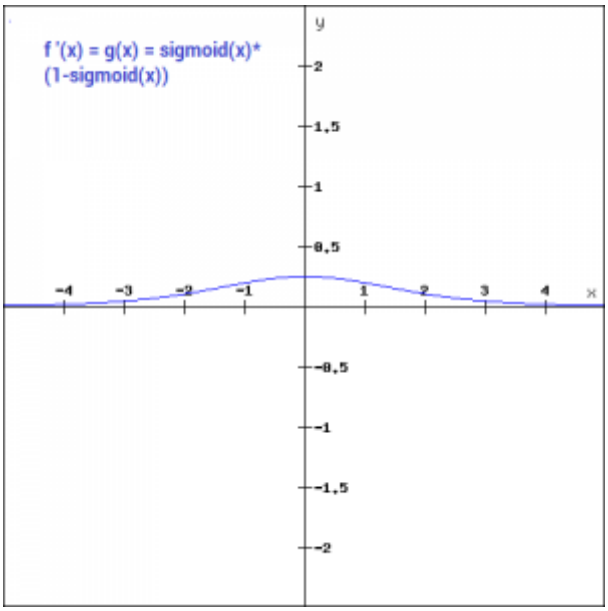
Sigmoid

تابع سیگموید (Sigmoid function) یک تابع ریاضی است که شکلی مشابه با حرف S در زبان انگلیسی دارد. به طور کلی به توابعی که شکل آن‌ها مشابه با حرف S است تابع‌های سیگموید یا دارای خمیدگی سیگموید می‌گویند.

یک تابع غیر خطی است و بنابراین خروجی غیر خطی می دهد ، خروجی در حد فاصل ۰ و ۱ است. (مطابق شکل) ، برای کلاس بندی باینری مناسب است .



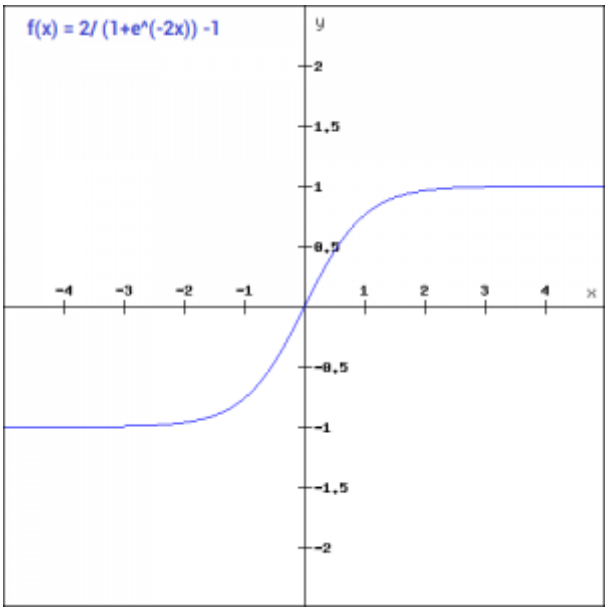
مشتق تابع **sigmoid** برابر است با شکل زیر :



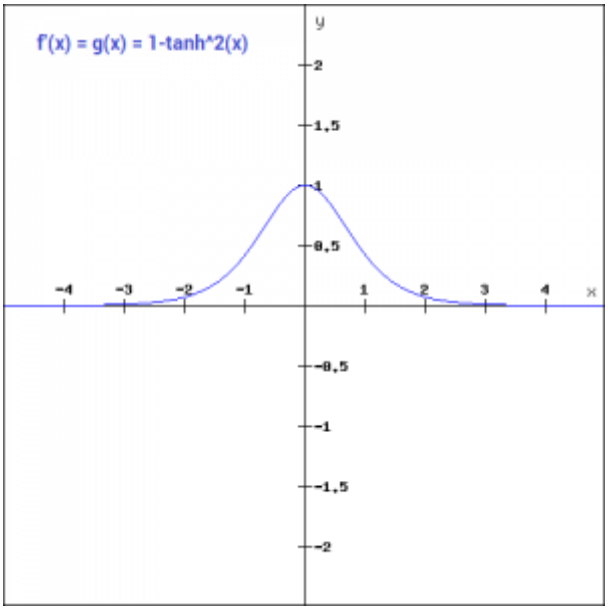
همانطور که در شکل مشاهده می شود مقادیر گرادیان(**مشتق**) برای محدوده -۳ و ۳ قابل توجه است، اما نمودار در مناطق دیگر بسیار صاف تر می شود و دارای گرادیان های بسیار کوچک خواهد بود.

Tanh

بسیار شبیه تابع سیگموئید است. تنها تفاوت این است که در اطراف مبدا مختصات متقارن است و محدوده مقادیر در آن از ۱- تا ۱ است. بنابراین ورودی های لایه های بعدی همیشه علامت یکسانی ندارند. تابع **tanh** به این صورت تعریف می شود:



همانند تابع **sigmoid** این تابع هم پیوسته و در تمام نقاط مشتق پذیر است ، ولی نسبت به سیگوئید از شیب بالاتری برخوردار است.



softmax

اغلب به عنوان ترکیبی از چند سیگموئید توصیف می شود. و می تواند برای مسائل طبقه بندی چند کلاسه استفاده شود. این تابع احتمال تعلق یک نقطه داده به هر کدام از کلاس ها را برمی‌گرداند. د

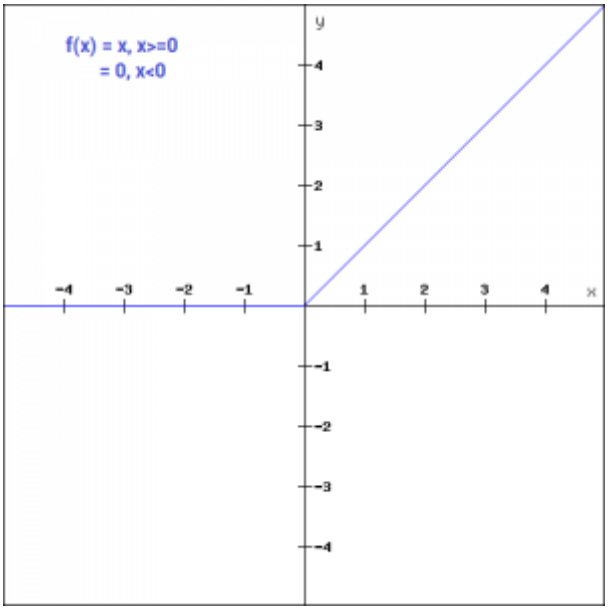
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

در حین ساختن یک شبکه برای یک مسئله چند کلاسه، لایه خروجی به تعداد کلاس های هدف نورون خواهد داشت. به عنوان مثال اگر ما سه کلاس داریم، سه نورون در لایه خروجی وجود خواهد داشت. فرض کنید خروجی نورون ها را به صورت [۱,۲، ۰,۹، ۰,۷۵] دریافت کرده اید.

با اعمال تابع softmax بر روی این مقادیر، نتیجه زیر را دریافت خواهیم کرد - [۰,۴۲، ۰,۳۱، ۰,۲۷]. اینها نشان دهنده احتمال تعلق نقاط داده(دیتا) به هر کدام از کلاس هاست است که همه جمع همه آنها برابر ۱ است .

RELU

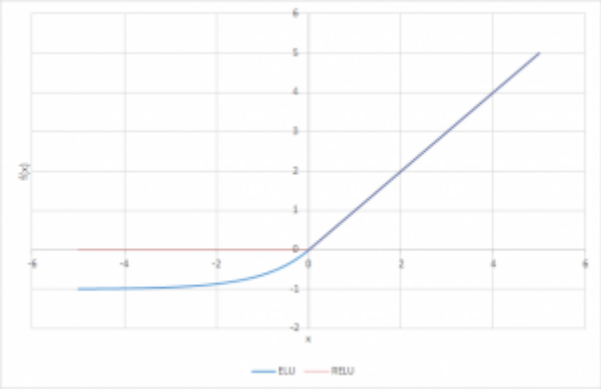
یک تابع فعال سازی غیر خطی است (پر کاربرد ترین) و مزیت آن در این است که همه نورون ها را به طور همزمان فعال نمی کند ، در صورتی که ورودی این تابع کوچکتر از صفر باشد ، فعال نخواهد بود .



همانطور که در تصویر مشتق RELU مشاهده میشود، مقدار گرادیان این تابع به ازای ورودی های منفی صفر است که این اتفاق در backpropogation میتواند منجر به به روز نشدن تعدادی از وزن ها و در نتیجه غیر فعال شدن تعدادی از نورون ها شود.

ELU

نوعی از ReLU (Rectiufied Linear Unit) است که شیب قسمت منفی را تغیر می دهد (لگارتیمی می کند)



Problem3.b

معایب SGD

در روش کاهش آماری گرادیان(SGD) برای هر نمونه بروزرسانی پارامترها انجام می شود. تابع هزینه در این روش بسیار ناهموار است و نوسانات شدیدی خواهد داشت زیرا داده ها به طور مکرر به روز می شوند و ممکن است به حداقل های محلی بهتر بپرند. به همین دلیل برای توابع غیر محدب مناسب نیست

SGD with momentum

مومنتوم اینرسی یک شی را در هنگام حرکت شبیه سازی می کند، یعنی مسیر به روز رسانی قبلی تا حدودی در طول به روز رسانی حفظ می شود، و گرادیان به روز رسانی فعلی برای بهینه سازی جهت به روز رسانی نهایی استفاده می شود. ثبات تا حد معینی افزایش می یابد، بنابراین یادگیری سریعتر است، و همچنین توانایی خلاص شدن از کمینه محلی را دارد.

در فرمول پایین می‌بینیم که می‌توانیم تصمیم بگیریم به کدام داده‌ها وزن بدهیم، که در زمان خاصی بسته به مقدار بتا می‌آیند.

بنابراین یادگیری از نقاط داده قبلی مفید خواهد بود و جهت خود را حفظ می کند و در نتیجه نویز را حذف می کند.

به گفته محققان، بتای بهینه ۰/۹۵ در نظر گرفته شده است.

Normal Weight Updation Formula in Gradient Descent

$W_t = W_{t-1} - \eta \, dL/dW_{t-1}$

Exponential Weighted Average for momentum

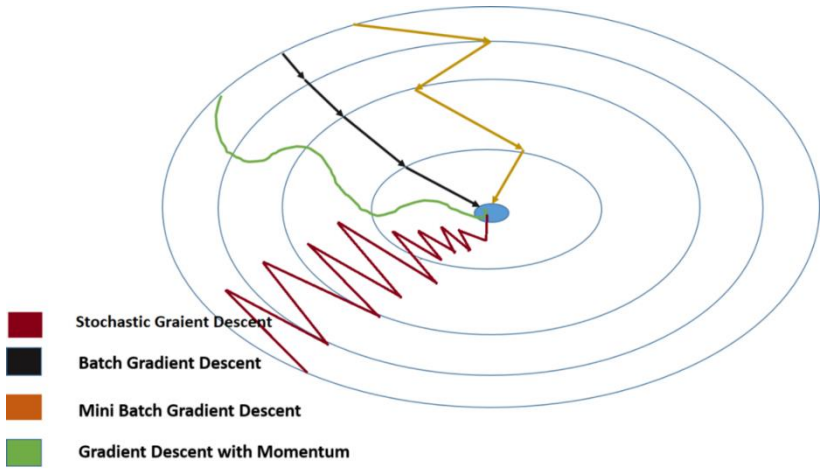
$W_t = W_{t-1} - \eta \, Vdw$ η =learning rate

$b_t = b_{t-1} - \eta \, Vdb$ b_t =bias

$Vdw_t = ß * Vdw_{t-1} + (1-ß) * dL/dW_{t-1}$

$Vdb_t = ß * Vdb_{t-1} + (1-ß) * dL/db_{t-1}$

If ß =0 formula will be same as used in gradient descent.



در تصویر بالا مقایسه نحوه رفتار SGD , BGD , SGD with Momentum را مشاهده میکنید

RMS-Prop

یا همان Root Mean Square Prop یک تکنیک بهینه سازی مبتنی بر گرادیان است که در آموزش شبکه های عصبی استفاده می شود جفری هینتون پیشنهاد شد.

با استفاده از میانگین متحرک گرادیان مجذور(moving average of squared gradients) به نرمال کردن گرادیان می پردازد، این نرمال سازی اندازه ی طول گام را متعادل می کند به معنی که گام را برای جاهایی که گرادیان در آن زیاد است برای جلوگیری از انفجار کاهش داده و در جایی که گرادیان در آن کم است برای جلوگیری از ناپدید شدن طول گام را افزایش می دهد. در واقع RMS-Prop از نرخ یادگیری انطباقی adaptive learning rate به جای نرخ یادگیری ثابت به عنوان یک هایپرپارامتر استفاده می کند. این بدان معناست که نرخ یادگیری در این روش در طول زمان در حال تغییر است. این بروزرسانی از روابط زیر به دست می آید:

$$v_{dw} = \beta \cdot v_{dw} + (1 - \beta) \cdot dw^2$$

$$v_{db} = \beta \cdot v_{dw} + (1 - \beta) \cdot db^2$$

$$W = W - \alpha \cdot \frac{dw}{\sqrt{v_{dw}} + \epsilon}$$

$$b = b - \alpha \cdot \frac{db}{\sqrt{v_{db}} + \epsilon}$$

منابع :

https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-be٨٣d٤٥٠٢f٧٥

https://towardsdatascience.com/optimizers-c٨٨٦٩٤٥٠٩٣١١