گزارش پیاده‌سازی پرسپترون چندلایه و بررسی و آزمایش توابع فعالیت مختلف

سارا حسینی ۴۰۰۲۲۲۰۲۶

۱ مقدمه

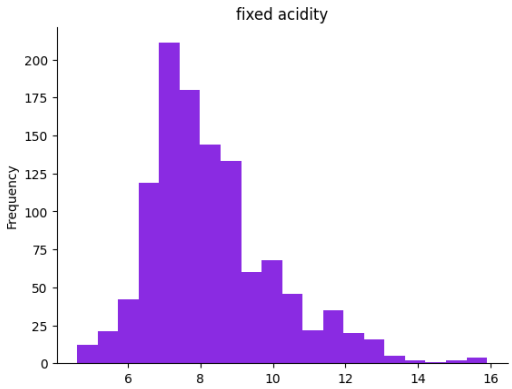
هدف ما، پیاده‌سازی یک شبکه عصبی ساده با چند لایه پرسپترون بود تا بتوانیم با گرفتن ورودی‌ای از ویژگی‌های نوشیدنی، کیفیت آن را که عددی بین ۰ و ۱۰ بود، پیش‌بینی نماییم. ما برای این کار، ۲ روش پیاده‌سازی داشتیم که هر هر دو، بدون استفاده از کتابخانه‌ی خاص و تنها با کمک numpy این شبکه را ساختیم. در این فرایند تنها با روابط ریاضی و مشتقگیری زنجیره‌ای، می‌توانیم backpropagation را انجام دهیم. در بخش دوم این گزارش نیز ده تابع فعالیت مختلف را بررسی کردیم و نمودار توزیع داده‌ای تصادفی و گرادیان آن را را قبل و بعد از اعمال توابع رسم کردیم.

۲ پیش‌پردازش داده‌ها

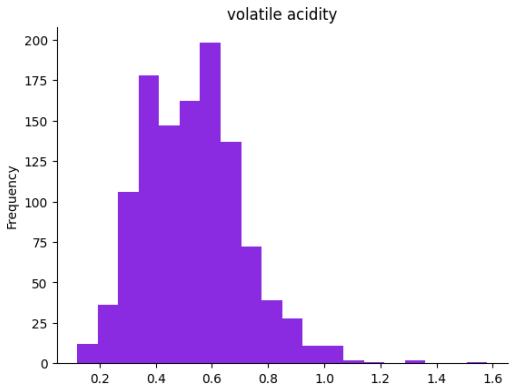
ابتدا نگاهی کلی به داده داشتیم. داده‌ها شامل ۱۱۴۳ سطر و ۱۳ ستون بودند. هیچ سطری تکراری نبود و همچنین هیچ مقدار null وجود نداشت. ستون‌های ما، شامل موارد زیر بودند:

'fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', 'quality', 'Id'

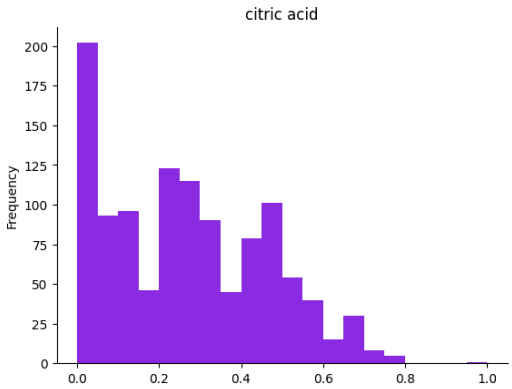
که از این بین، فقط quality مقداری گسسته و categorical داشت. مقادیر آن ۳، ۴، ۵، ۶، ۷، ۸ بودند.



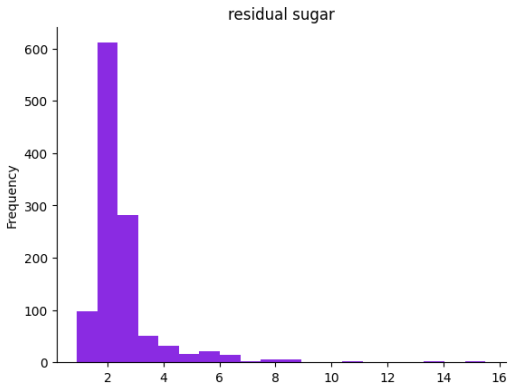
شکل ۱: توزیع اسیدی بودن ثابت



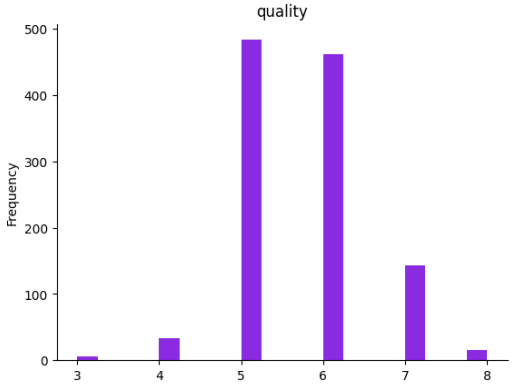
شکل ۲: توزیع اسیدی بودن فرار



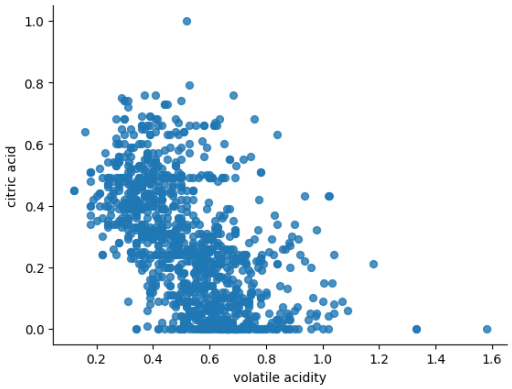
شکل ۳: توزیع سیتریک اسید



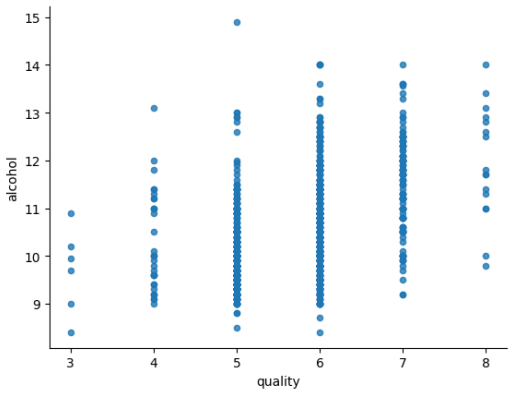
شکل ۴: توزیع شکر



شکل ۵: توزیع کیفیت



شکل ۶: رابطه بین سیتریک اسید و اسیدی بودن فرار

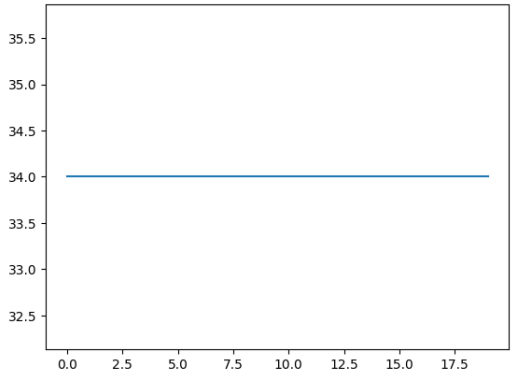


شکل ۷: رابطه الکل و کیفیت

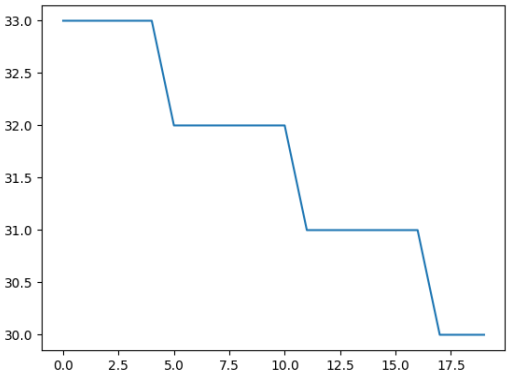
سپس، با نسبت ۸۰ به ۲۰، داده را تقسیم کردیم. از آنجا که توزیع اکثر متغیرها نرمال نبود، فهمیدیم که نرمال‌سازی یا استانداردسازی، لازم هستند. پس تمام ستون‌ها بجز پی‌اچ و کیفیت، جداگانه در داده‌ی آموزشی و داده‌ی آزمایشی، استاندارد شدند. ستون شناسه آی‌دی نیز حذف شد زیرا حاوی اطلاعات مهمی نبود. در مرحله‌ی بعد، مدلی طراحی می‌کنیم که بتواند یکی از feature ها، یعنی quality را پیش‌بینی نماید.

۳ طراحی مدل

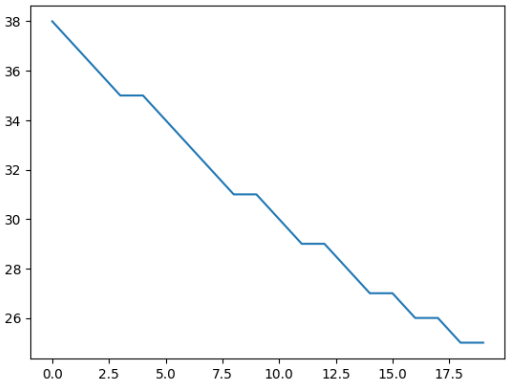
برای پیاده‌سازی این مدل، رویکردی مشابه pyTorch می‌توان داشت. به این صورت که یک کلاس برای "لایه" داشته باشیم و بعد یک کلاس برای شبکه که خود شامل یک سری لایه می‌شود. هر لایه باید مقادیر ورودی، گرادیان تابع هزینه نسبت به ماتریس وزن‌ها، گرادیان تابع هزینه به bias، گرادیان تابع هزینه نسبت به ورودی‌های لایه را در خود ذخیره نگه دارد. آنگاه به راحتی می‌توان سهم هر مسیر در خطای خروجی را اندازه گرفت و به عقب انتشار داد.
مدل ما شامل ۴ لایه است که به صورت "خطی، خطی، ReLU ، خطی" قرار گرفته‌اند. . لایه‌ی خروجی تنها یک نورون دارد. لایه‌های درونی هر کدام ۲۰ نورون دارند و اولین لایه ۱۱ نورون (تعداد feature های داده) دارد. ما مدل را با ۱۵ نرخ یادگیری مختلف بین 10^{-8} تا 10 آموزش دادیم و برای هر کدام ده epoch این کار را ادامه دادیم و نتایج مقادیر هزینه را رسم کردیم. تابع هزینه ما نیز MSE بوده است.
برای ۷ نرخ آموزشی اول، نمودار هزینه به صورت یک خط صاف بود.



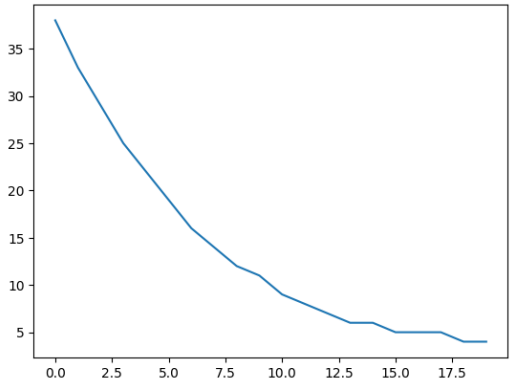
برای مقادیر نرخ یادگیری $0.0003, 0.0013, 0.0061, 0.0268$ نمودار به شکل مناسبی درآمد:



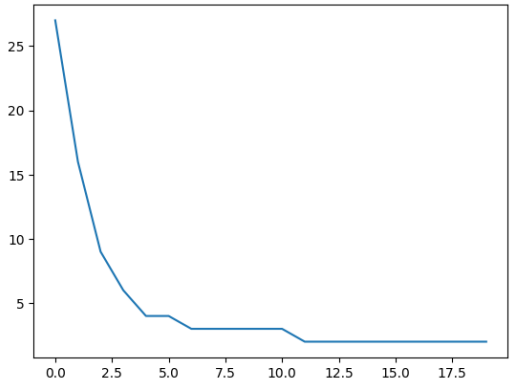
شکل ۸: LR=0.0003



شکل ۹: LR=0.013

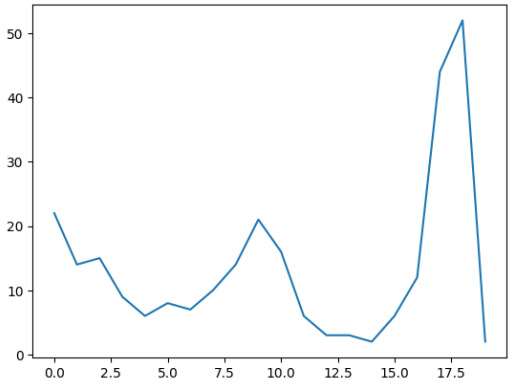


شکل ۱۰: LR=0.0061



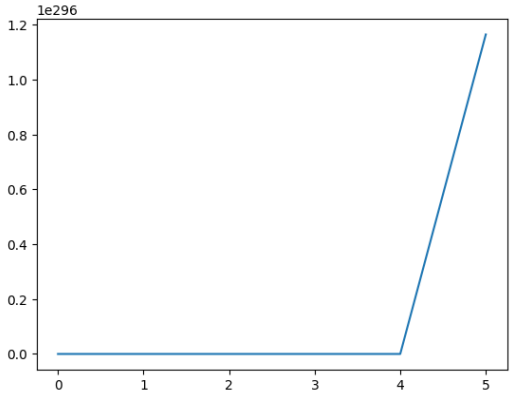
شکل ۱۱: LR=0.0268

برای 0.1178 نمودار به شکل غیرپایدار زیر درآمد:



شکل ۱۲: LR=0.1178

و برای مقادیر بزرگتر زیر تابع ب همچین فرمی درآمد:



شکل ۱۳: LR=0.5179

در نهایت ما مقدار 0.2682 را به عنوان لرنینگ ریت مناسب انتخاب نمودیم و مدل جدیدی ساختیم و آن را با این نرخ دوباره آموزش دادیم. سپس، داده‌ی آزمایشی به مدل دادیم. MSE برای داده‌ی تست برابر با یک شد. همچنین، مقادیر $Precision$ ، $Recall$ و $F1$ برای ۶ کلاس quality به صورت زیر بودند:

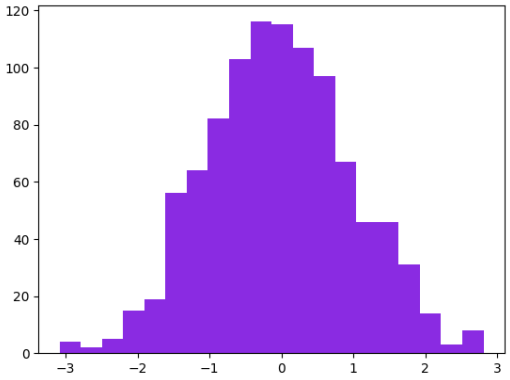
Precision : $[0.0, 0.0, 0.5104166666666666, 0.391304347826087, 0.35714285714285715, 0.0]$
Recall : $[0.0, 0.0, 0.5833333333333334, 0.2872340425531915, 0.35714285714285715, 0.0]$
f1 : $[0, 0, 0.5444444444444445, 0.3312883435582822, 0.35714285714285715, 0]$

مقدار $accuracy$ نیز برابر با 0.39 شد. این یعنی شبکه ما احتمالا بیش از حد ساده بوده است یا اینکه تعداد داده‌ی آموزشی کم بوده پس نتوانسته مسئله را یاد بگیرد.

۴ بررسی ۱۰ تابع فعالیت مختلف

تابع فعالیت باعث میشود که شبکه، قسمت بدردبخور دیتا را از قسمت دیگر جدا کند. تابع فعالیت تصمیم میگیرد که یک نورون فعال (fired) بشود یا نه، یعنی خروجی نورون برای شبکه مهم باشد یا نه. معمولا همه لایه‌های پنهان تابع فعالیت یکسانی دارند اما لایه‌ی خروجی بر اساس هدف مدل متفاوت است. ما به تابع فعالیت نیاز داریم چون اگر نباشد، عملا میتوانیم تمام لایه‌ها را با یک لایه نشان بدهیم و شبکه به یک رگرسیون خطی تبدیل میشود. یعنی بدون اکتیویشن فانکشن، افزایش عمق شبکه غیرممکن است چون همه چیز خطی جلو میرود. با بودن تابع فعالیت، غیرخطی بودن وارد شبکه میشود.

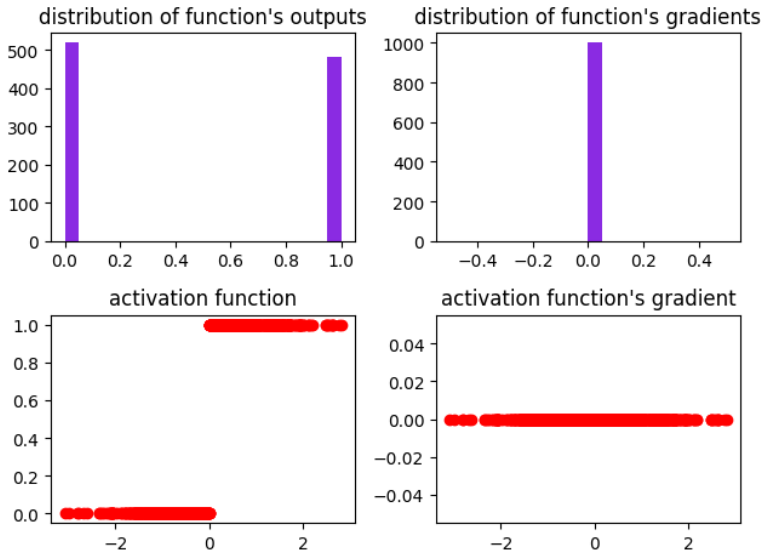
در این بخش ما ده تابع فعالیت مختلف را بررسی کردیم. یک ماتریس عدد با توزیع رندوم نرمال مثال زدیم و توابع و گرادیان آنان را روی این ماتریس امتحان کردیم و رسم نمودیم.



شکل ۱۴: توزیع اولیه داده مورد آزمایش

الف) binary step function

برای مقادیر منفی، ۰ و برای صفر و مثبت، ۱ برمی‌گرداند.

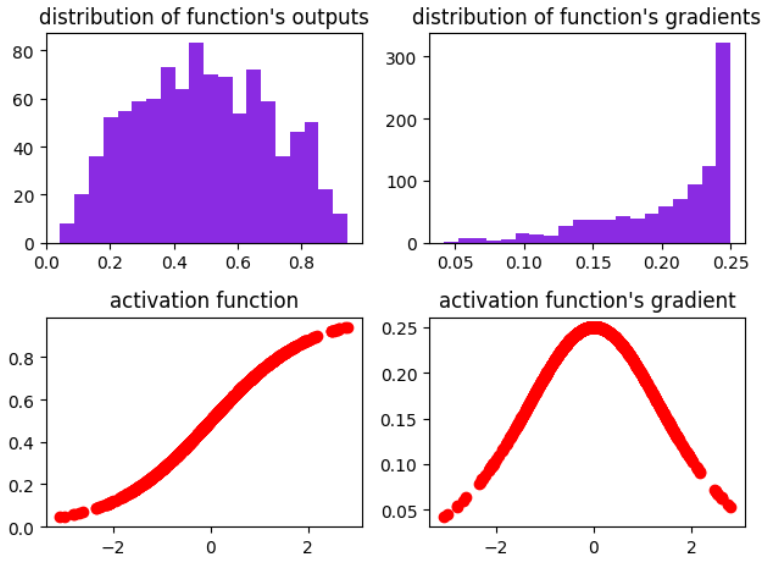


شکل ۱۵: تابع باینری استپ

مشکلات: نمیتواند بیش از دو کلاس را کلسیفای کند. گرادیانش صفر است که برای بکپروپ مشکلساز میشود چون عملا وزنها اپدیت نمیشوند. کاربرد: کلسیفایر دودویی در لایه آخر.

ب) Indentity function/ linear ativation function
تابعی که خود ورودی را برگرداند یا اسکالری را در ورودی ضرب کرده و برگرداند. مشکلات: مشتق همواره یک عدد ثابت است و هیچ ارتباطی با ورودی ندارد. در واقع وزن‌ها و بایس اپدیت میشوند اما همیشه به یک اندازه ثابت. ممکن است فقط مناسب تسک‌های ساده باشد که می‌خواهیم تفسیرپذیری بالا باشد. برای تسک رگرسیون هم در لایه‌ی آخر مناسب است.

پ) sigmoid/logistic
مناسب وقتی که خروجی، احتمال باشد چون مقادیر خروجی سیگموید بین ۰ و یک است. همه جا گرادیان دارد و همه جا مشتق‌پذیر است. اینکه همه خروجی‌ها مثبت هستند و تابع اطراف صفر متقارن نیست، در یادگیری تاثیر منفی میگذارد. چون علامت گرادیان همیشه مثبت است هم مشکل ونیشینگ گرادیان پیش می‌آید.



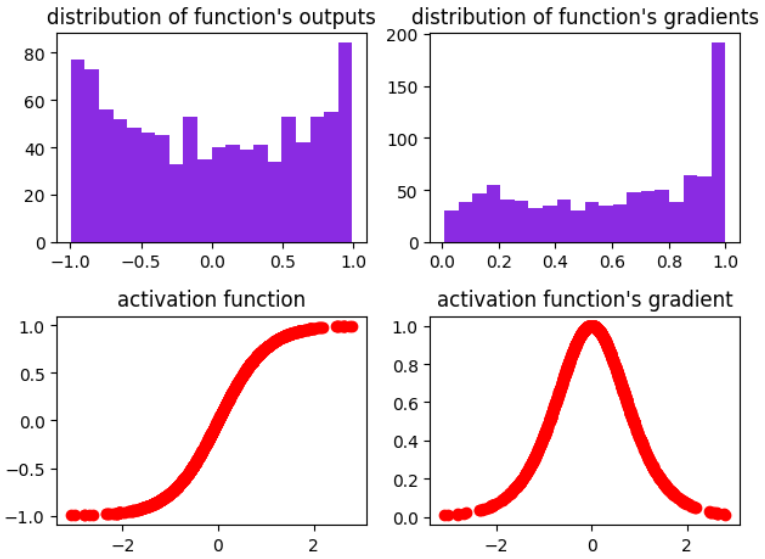
شکل ۱۶: سیگموید

چون گرادیان خارج از بازه ۳- تا ۳ تقریبا صفر است، شبکه دیگر چیزی یاد نمیگیرد و ونیشینگ گرادیان پدید می‌آید.

ت)tanh function
مثل سیگموید، ولی بین ۱- و ۱. حول مبدا متقارن است. به نوعی scale شده‌ی سیگموید است.

.tanh=2sigmoid(2x)-1

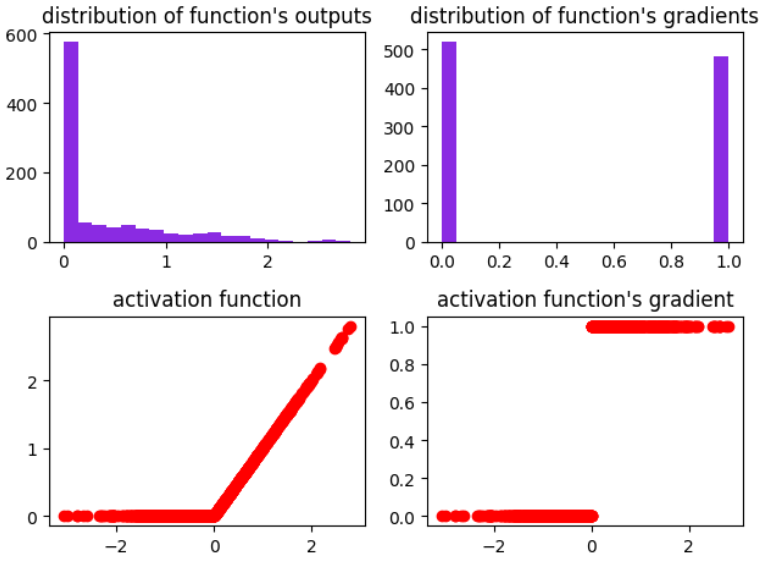
مناسب کلسیفایر باینری است.



شکل ۱۷: تانژانت هیپربولیک

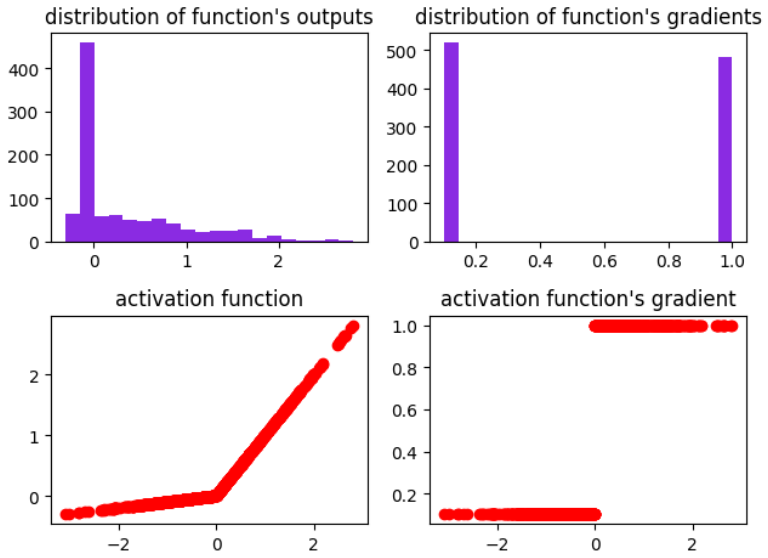
ث)ReLU function

نکته این است که همه نورون‌ها را فایر نمیکند. مقادیر منفی، فایر نمیشوند. همین باعث میشود از نظر محاسباتی بهتر عمل کند. همگرایی به سوی گلوبال مینیمم سریعتر است. اما چون مشتق مقادیر منفی هم صفر است، نورون ها میمیرند و کلا دیگر اپدیت نمیشوند.



شکل ۱۸: رلو

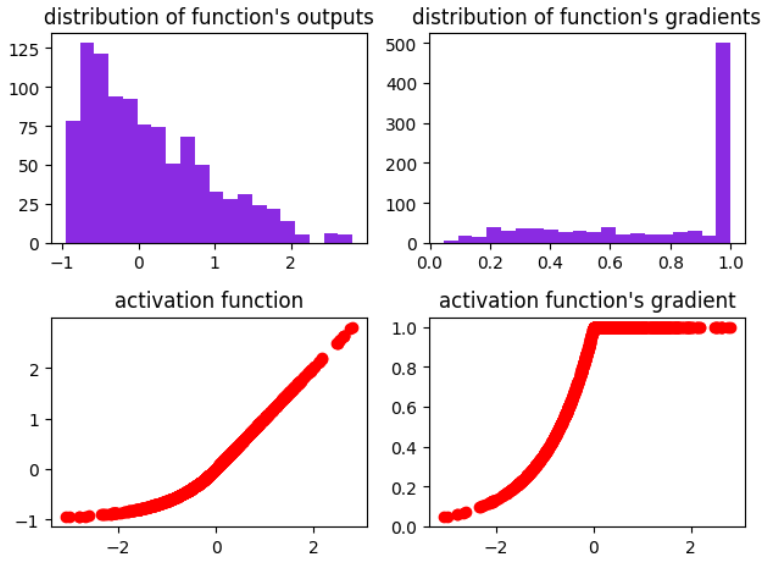
ج) leaky ReLU
مشکل مردن نورون‌ها حل میشود چون برای مقادیر منفی هم گرادیان صفر نیست، در عوض مقداری ناچیز است و امکان ریکاوری از مرگ نورون میدهد.



شکل ۱۹: leaky relu

چ) parametric ReLU
مانند leaky ReLU است با این فرق که مقدار شیب خط در ناحیه منفی، از پیش تعیین‌شده نیست و لرن میشود.

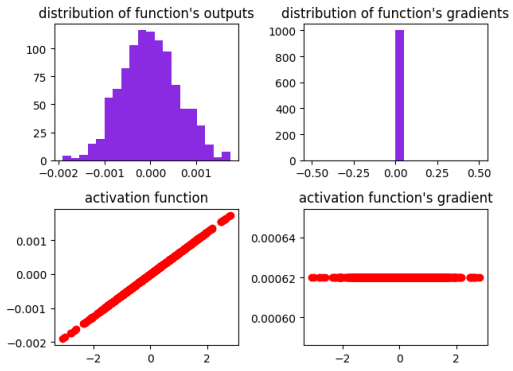
ح) ELU
مثل رلو ولی در مقادیر منفی، خط نداریم بلکه یک تابع لگاریتمی داریم. چون مقادیر منفی هم میدهد، دیتا را کمی نرمال میکند و میانگین را به صفر نزدیک میکند.



شکل ۲۰: ELU

خ)softmax

ترکیب چندین سیگموید. معمولا در لایه اخر استفاده میشود برای کلسیفای به چندین کلاس. البته این تابع از آنجایی که چندین متغیر دارد، نمی‌تواند به خوبی پلات شود (فقط به ورودی وابسته نیست و به بقیه‌ی داده‌ها در فضای نمونه هم بستگی دارد).

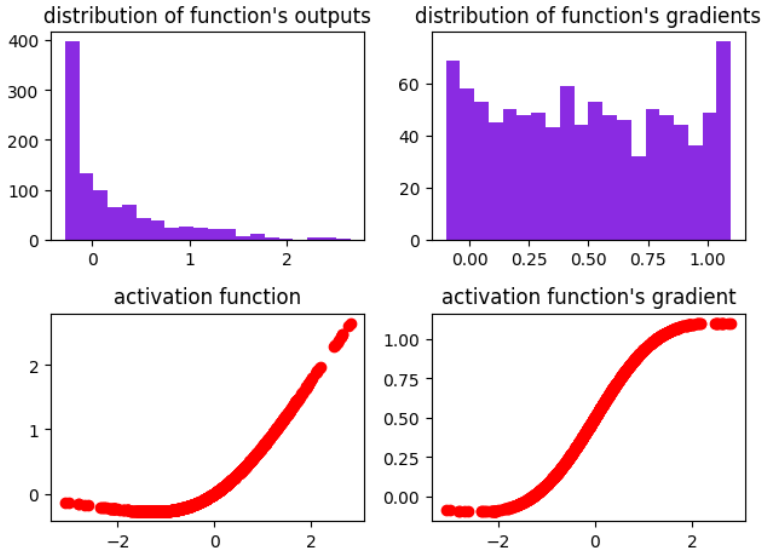


شکل ۲۱: سافتمکس

د)swish

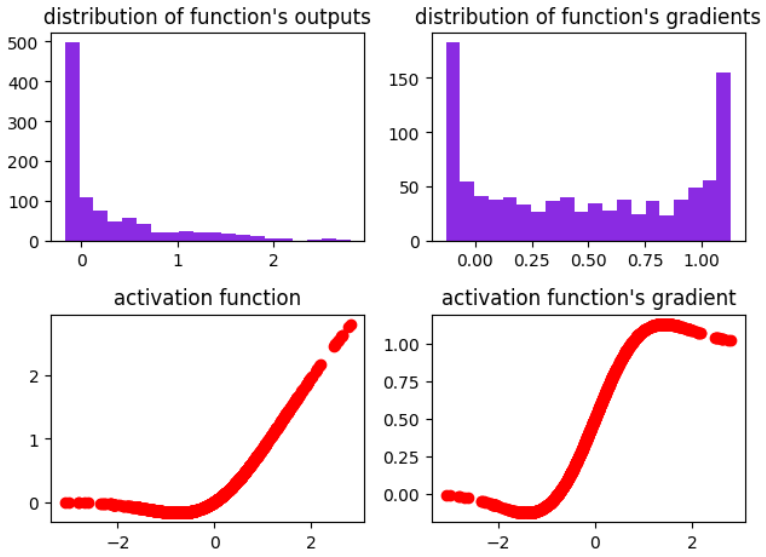
برای ایمیج کلسیفیکیشن، ترجمه ماشینی و... بسیار بهتر از رلو عمل کرده است. در رلو، مقادیر منفی کلا صفر شدند ولی اینجا مقادیر کوچک منفی خنثی نشدند و تاثیرگذارند فقط مقادیر بزرگ منفی صفر شدند. این تابع یکنوا نیست و همچنین کران‌هایش از دو طرف بی‌نهایت

است.



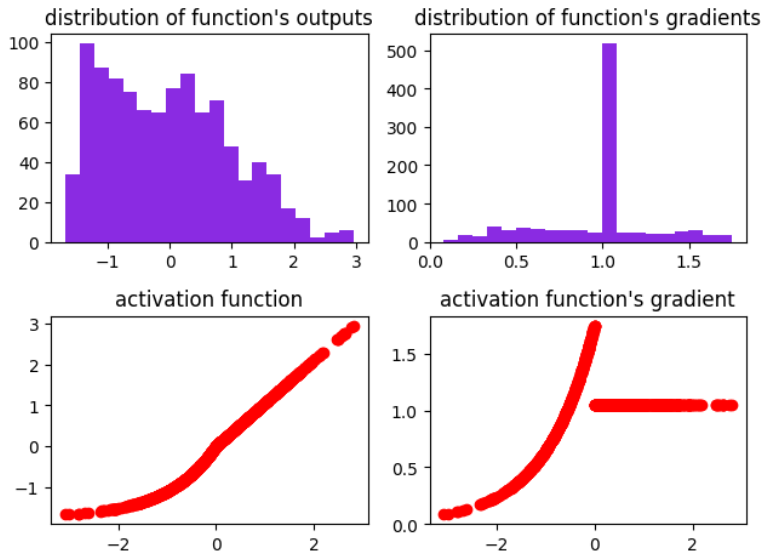
شکل ۲۲: swish

GeLU: Gaussian Error Linear Unit(ذ
مناسب برای برت، روبرتا و مدلهای ان ال پی. ترکیب رلو و تکنیک دراپ اوت است.
$$\Phi(x) = P(X \leq x), \quad X \sim N(0,1)$$
تابع توزیع تجمعی از توزیع نورمال است. چون نورونها از توزیع نورمال پیروی میکنند (مخصوصا بعد از بچ نورمالیزیشن)، از توزیع نورمال استفاده شده است. کاربرد در:
computer vision, natural language processing, and speech recognition.



شکل ۲۳: GELU

SELU: Scaled Exponential Linear Unit(ر
مزیت آن این است که یانگین و واریانس لایه قبل را حفظ میکند.



شکل ۲۴: SELU

در کل بهتر است اول در لایه های هیدن، رلو بذاریم و اگر جواب نداد، برویم سراغ بقیه.
سیگموید و تانژانت هیپربولیک بخاطر ونیشینگ گرادیان، بهتر است در هیدن استفاده نشود.

در شبکه‌های با عمق بالای ۴۰ هیدن لیر، از سوییش استفاده کنید.
برای لایه خروجی:
Regression -> Linear Activation Function
Binary Classification—>Sigmoid/Logistic Activation Function
Multiclass Classification—>Softmax
Multilabel Classification—>Sigmoid
Multilabel یعنی وقتی که یک ورودی، به دو تا کلاس خروجی یا بیشتر تعلق داشته باشد.
برای لایه‌های هیدن:
Convolutional Neural Network: ReLU activation function.
Recurrent Neural Network: Tanh and/or Sigmoid activation function.