به نام خدا

هانیه اسعدی ۹۹۵۲۱۰۵۵

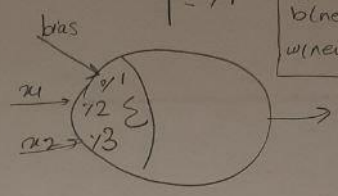
x_1	x_2	bias	target
1	1	1	-1
1	-1	1	1
-1	1	1	1
-1	-1	1	1

$\eta = 0.1$

$y = bias + \sum x_i w_i$

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

$w(new) = w(old) + \eta(d - y)x_i$



1 step: input (1, 1), output = -1

$y = 0.1 + 0.2 \times 1 + 0.3 \times 1 = 0.6$

$b(new) = 0.1 + 0.1(-1 - 0.6) = -0.06$

$w_1(new) = 0.2 + 0.1(-1 - 0.6) \times 1 = 0.04$

$w_2(new) = 0.3 + 0.1(-1 - 0.6) \times 1 = 0.14$

largest weight change is 0.16

---

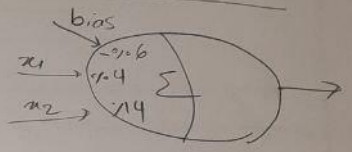
2 step: input (1, -1), output = 1

$y = -0.06 + 0.04 \times 1 + 0.14 \times -1 = -0.16$

$b(new) = -0.06 + 0.1(1 + 0.16) = 0.056$

$w_1(new) = 0.04 + 0.1(1 + 0.16) \times 1 = 0.156$

$w_2(new) = 0.14 + 0.1(1 + 0.16) \times -1 = 0.024$



largest weight change is 0.116

---

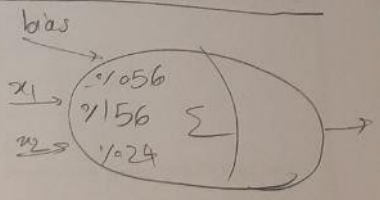
3 step: input (-1, 1), output = 1

$y = 0.056 + (-1) \times 0.156 + 1 \times 0.24 = -0.076$

$b(new) = 0.056 + 0.1(1 + 0.076) = 0.1636$

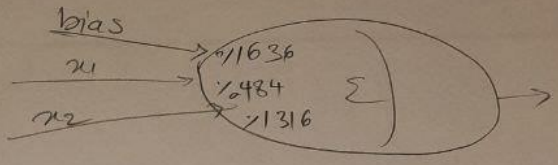
$w_1(new) = 0.156 + 0.1(1 + 0.076) \times -1 = 0.484$

$w_2(new) = 0.24 + 0.1(1 + 0.076) \times 1 = 0.1316$



largest weight change is 0.076

4 row: input $(-1,-1)$, output $= 1$



$y = 0.1636 + (-1)\times 0.484 + (-1)\times 0.1316 = -0.164$

$b(\text{new}) = 0.1636 + 0.1(1 + 0.164) = 0.26524$

$w_1(\text{new}) = 0.484 + 0.1(1 + 0.164)\times -1 = -0.5324$

$w_2(\text{new}) = 0.1316 + 0.1(1 + 0.164)\times -1 = 0.2996$

largest weight change is 0.164

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 ۲

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 ۲

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- دیتاست Circle:

Tanh: تابع فعالساز غیر خطی است که دیتا را روی محدوده -1 تا 1 محدود میکند. روی دیتاهای پیچیده مخصوصا دیتاهایی که به شکل دایره تفکیک پذیرند به خوبی اعمال میشوند.

ReLU: یک تابع ساده است که برای داده های مثبت، خطی و برای داده های منفی، صفر است و برای داده های غیر خطی مناسب است. البته نسبت به تابع تانژانت و روی داده های دایره ای شکل کمتر مناسب است.

Sigmoid: برای دیتاهایی که دو کلاس به خوبی از هم تفکیک شده اند مناسب است.

Linear: همانند سیگموید، برای دیتاهای دایره ای مناسب نیست و منجر به underfittingمیشود.

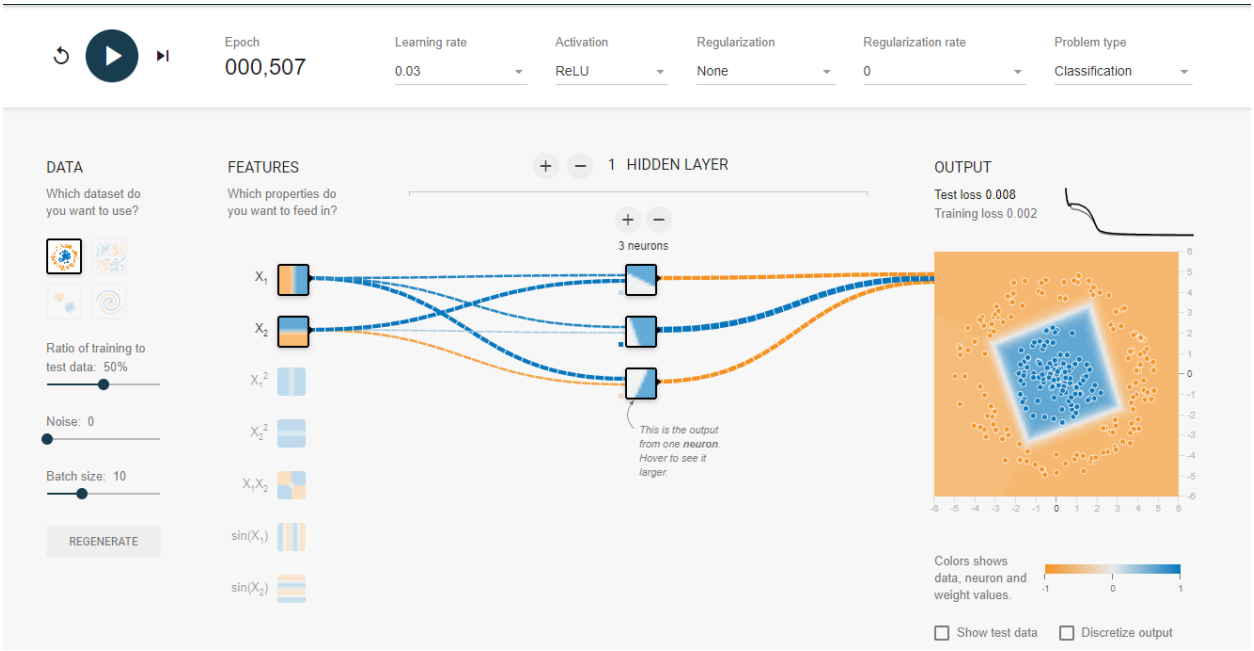


Figure 1 ReLU

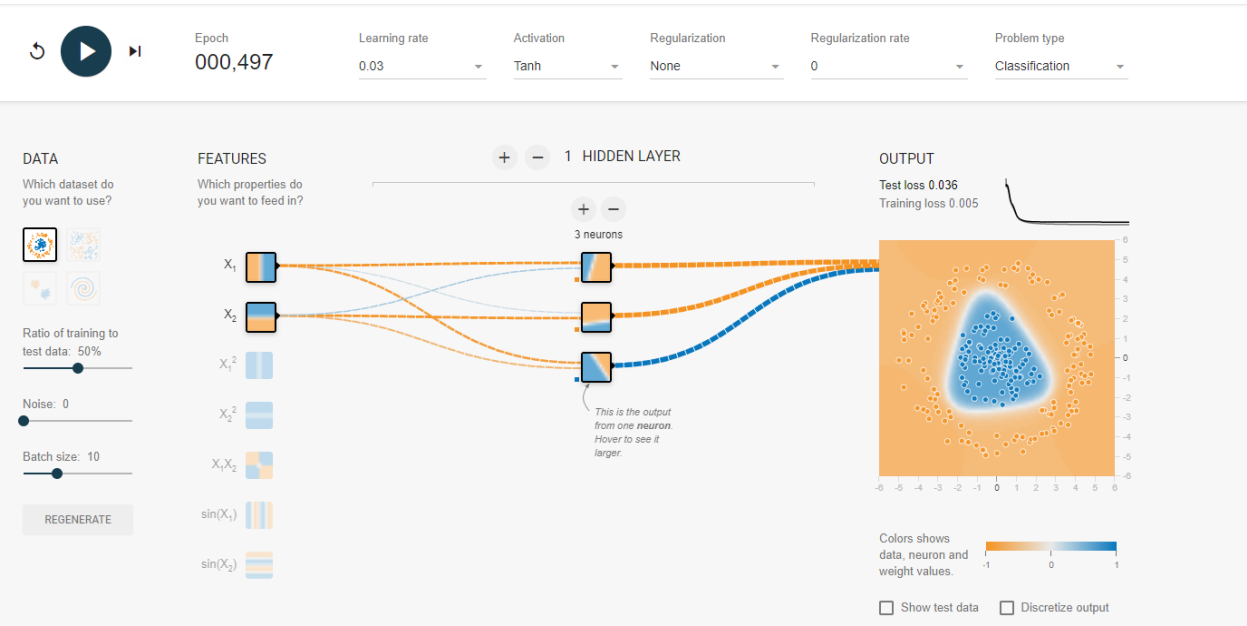


Figure 2 Tanh

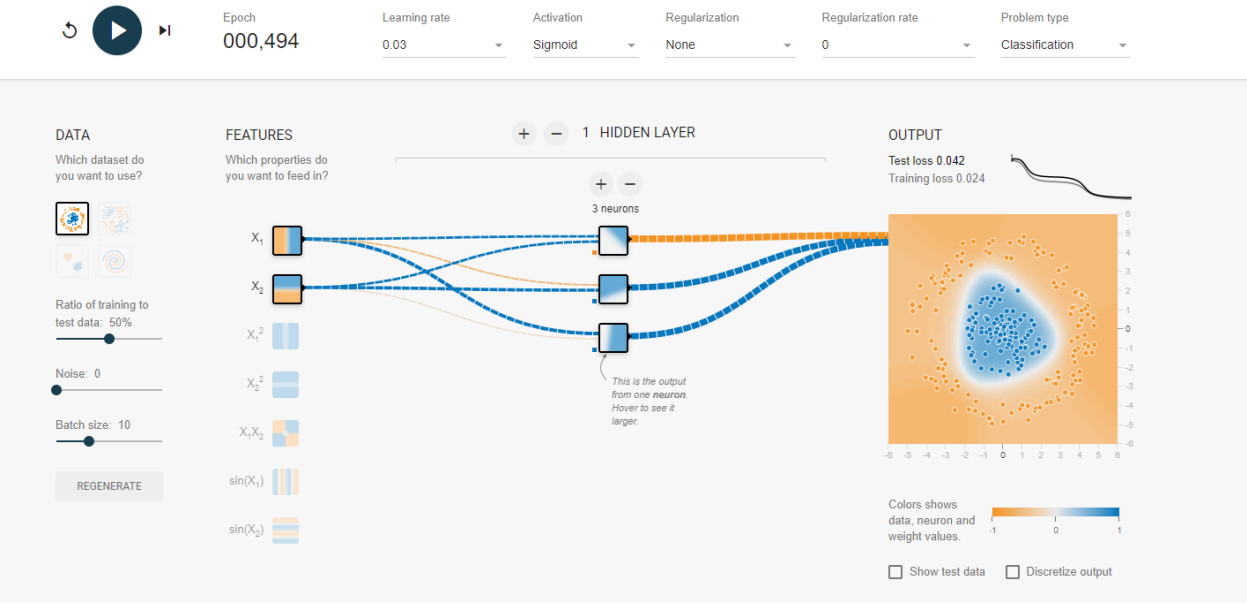


Figure 3sigmoid

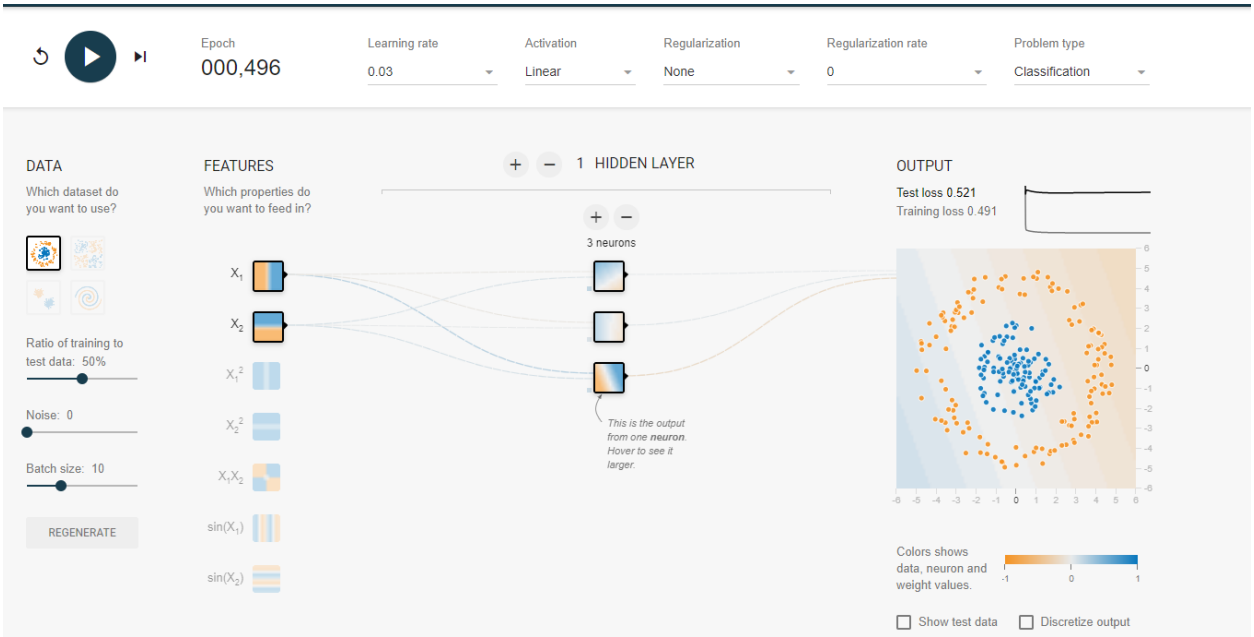


Figure 4 Linear

همانطور که دیده میشود مقدار loss چه در داده آموزش چه در داده تست در توابع تانژانت و ReLU به نسبت دیگر توابع کمتر است و مرز تفکیک خوبی اعمال کرده اند.

دیتاست exclusive or :

Tanh: روی داده های پیچیده هم به خوبی عمل میکند و قادر به تفکیک مسئله xor هست.

ReLU: وقتی با MLP ترکیب میشود میتواند مسائلی که با چند خط تفکیک میشوند را به خوبی تفکیک کند.

Sigmoid: این تابع هم وقتی با MLP ترکیب میشود مرز تفکیک خوبی برای مسئله xor ایجاد میکند.

Linear: به خوبی روی داده xor عمل نمیکند چون مسئله تفکیک پذیر خطی نیست. دیده میشود که مقدار loss روی هر دو داده تست و لرن زیاد است.

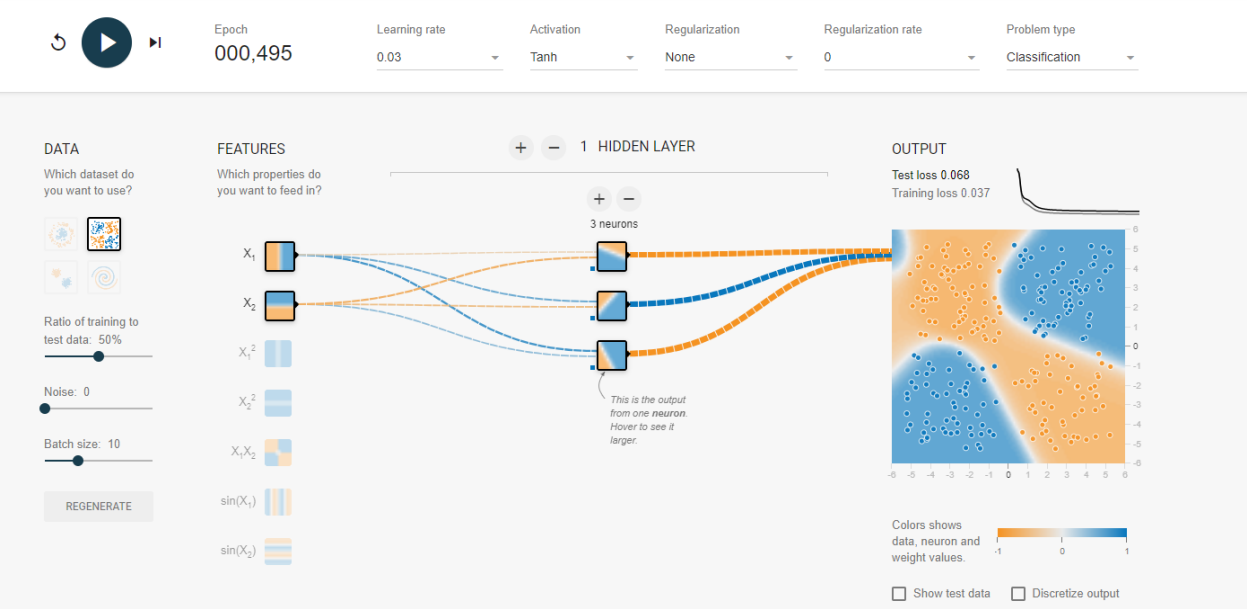


Figure 5 tanh

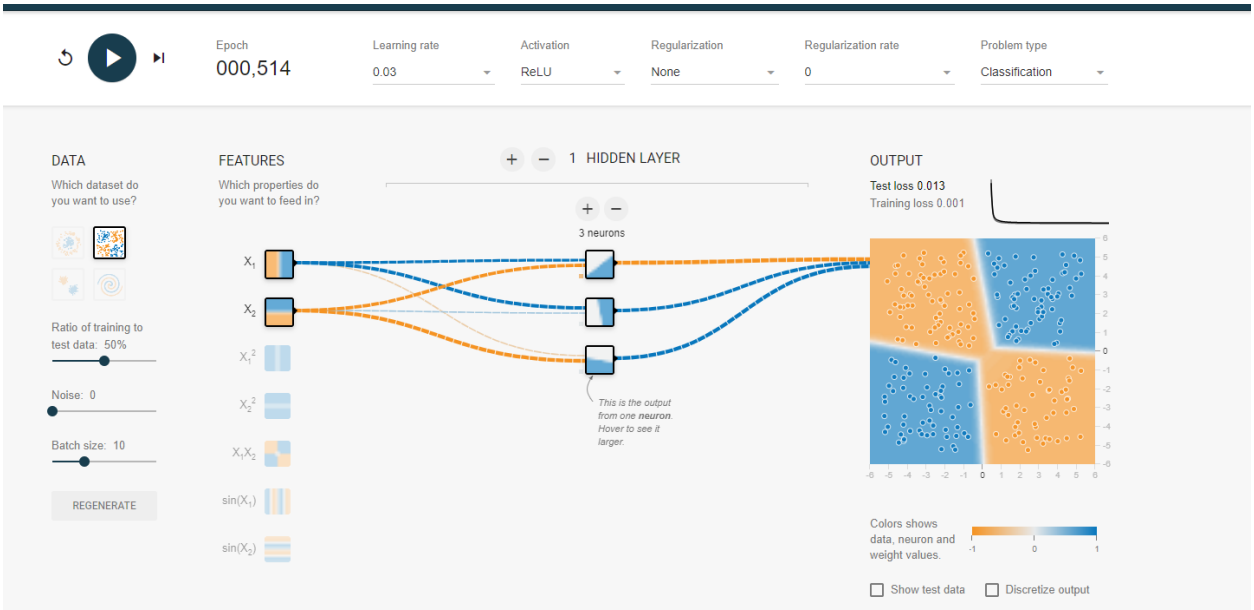


Figure 6 ReLU

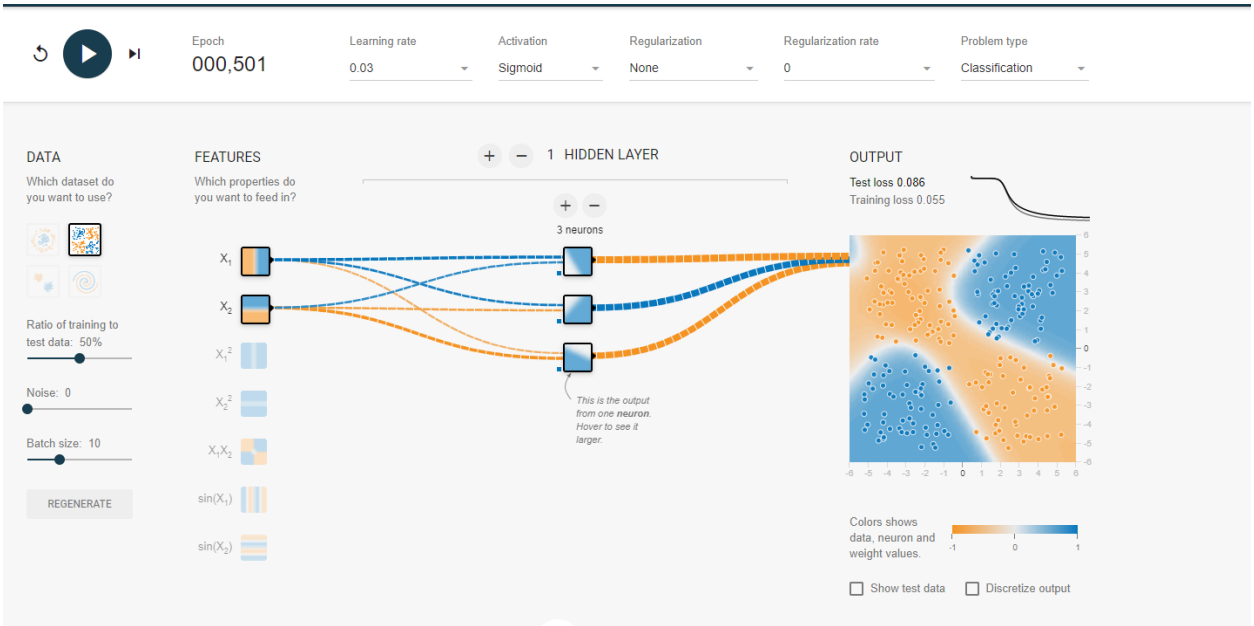


Figure 7 sigmoid

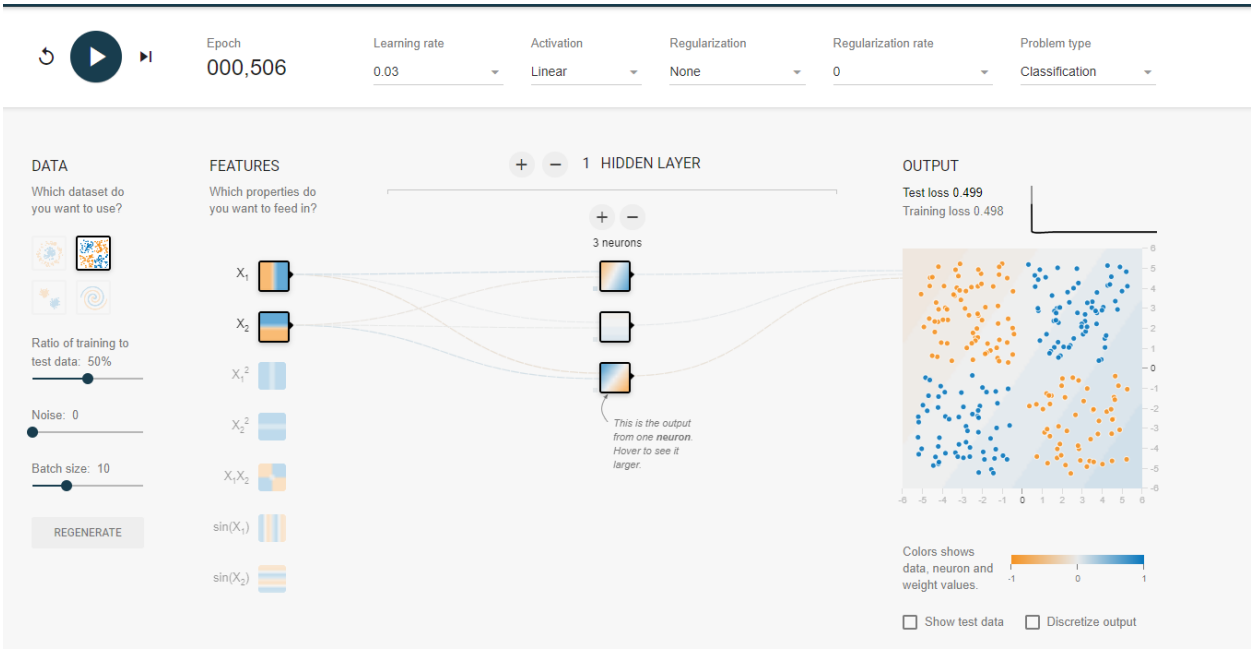


Figure 8 Linear

دیتاست Gaussian :

چون داده ها به شکل تفکیک پذیر خطی هستند، همه توابع روی آنها به خوبی عمل کرده و با دقت خوبی از هم تفکیک میکنند و به همگرایی بسیار خوبی

میرسند و مقدار Loss روی هر دو داده به شدت کم و یا حتی صفر است. تعمیم خوبی هم روی داده تست دارند.

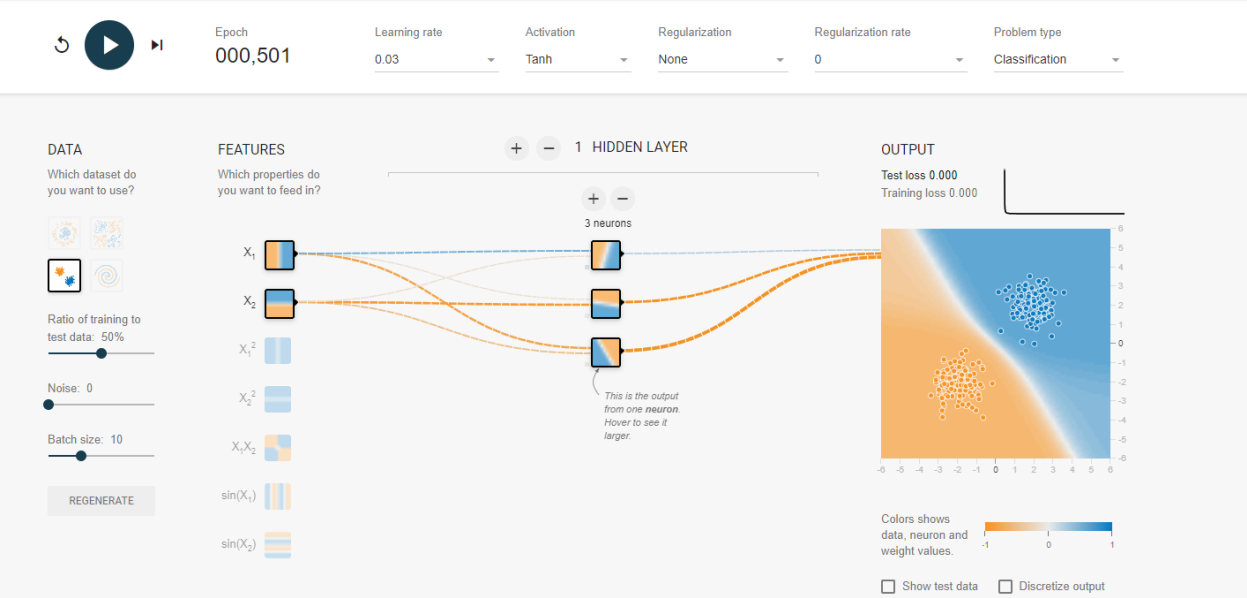


Figure9 Tanh

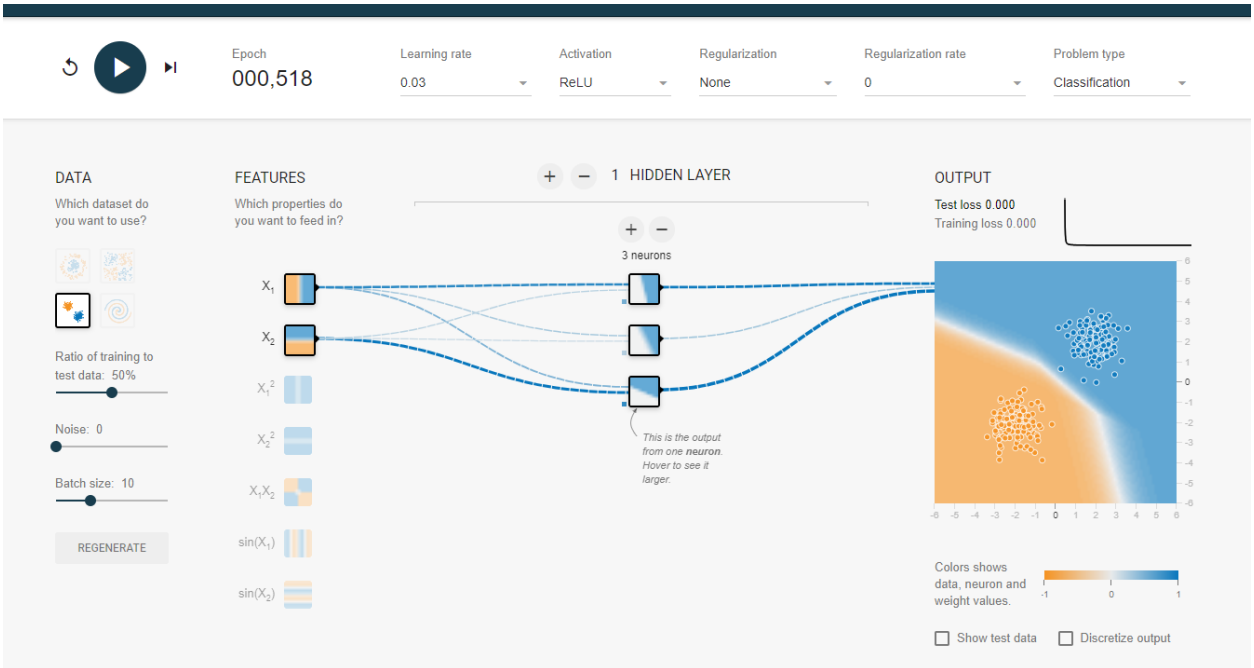


Figure 10 ReLU

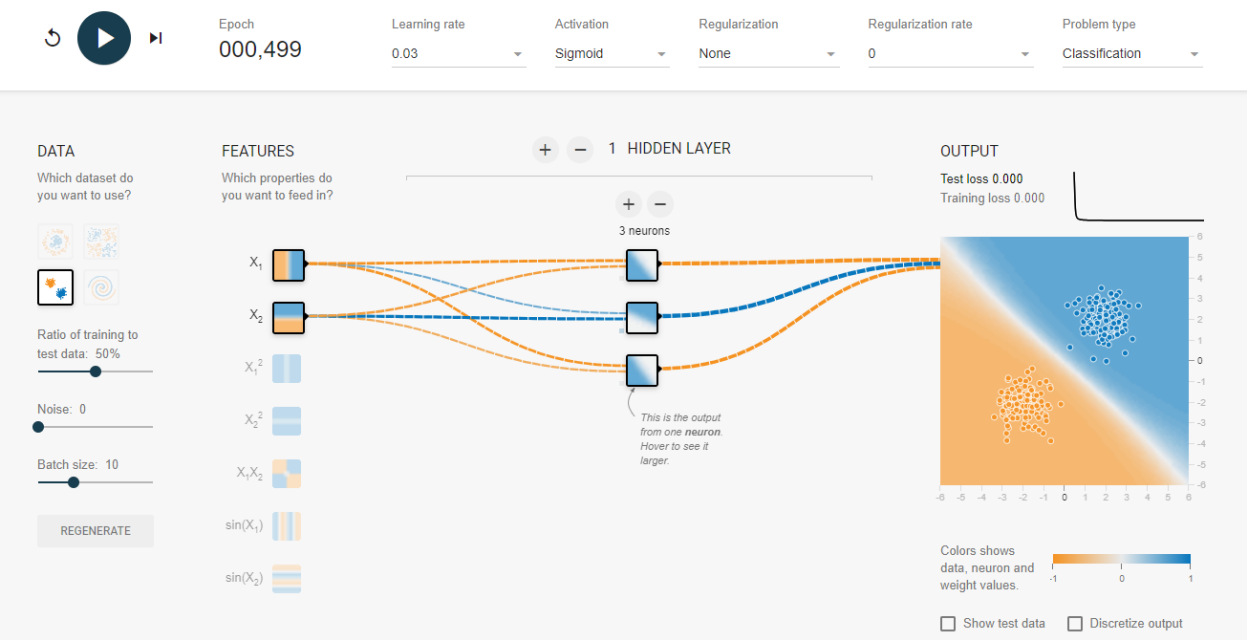


Figure 11 Sigmoid

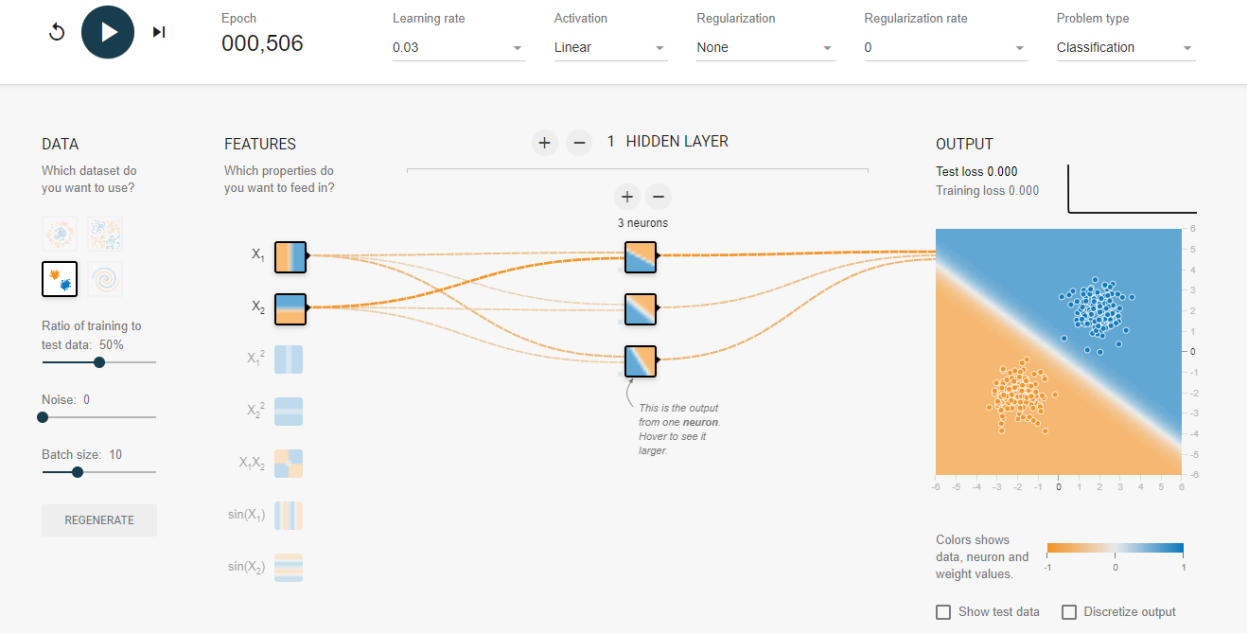


Figure 12 Linear

دیتاست spiral :

این داده پیچیدگی بیشتری نسبت به سایر داده ها دارد و همه توابع با عملکرد تقریبا یکسانی روی این داده لرن میکنند. و مقدار Loss روی هر دو داده تست و

لرن تقریبا بهم نزدیک هستند. این نشان میدهد همه توابع برای رسیدن به همگرایی تلاش میکنند ولی به دلیل پیچیدگی داده، به همگرایی زیادی نمیرسند.

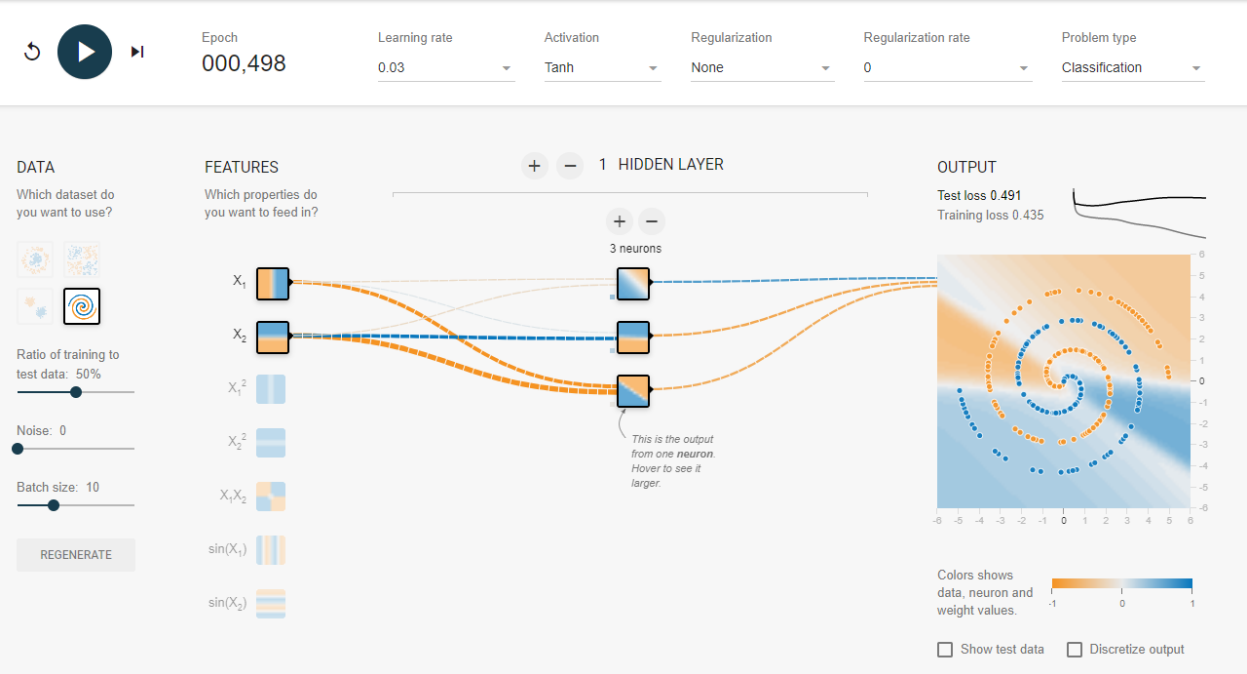


Figure 13 Tanh

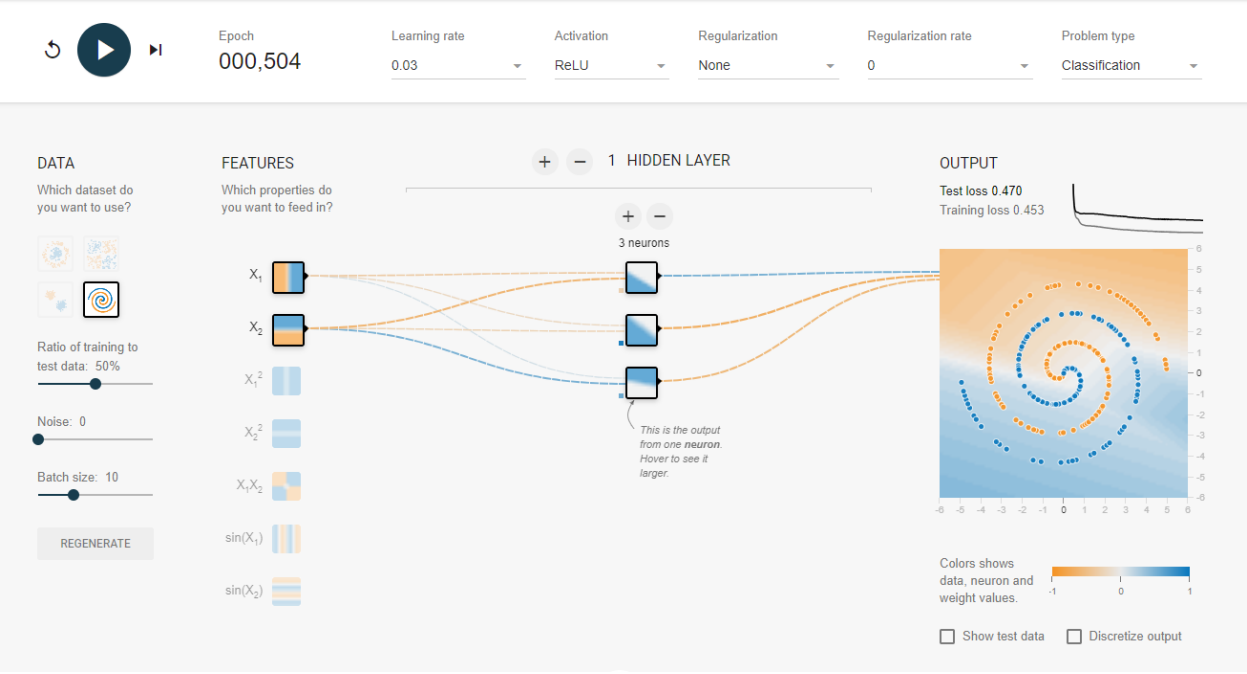


Figure 14 ReLU

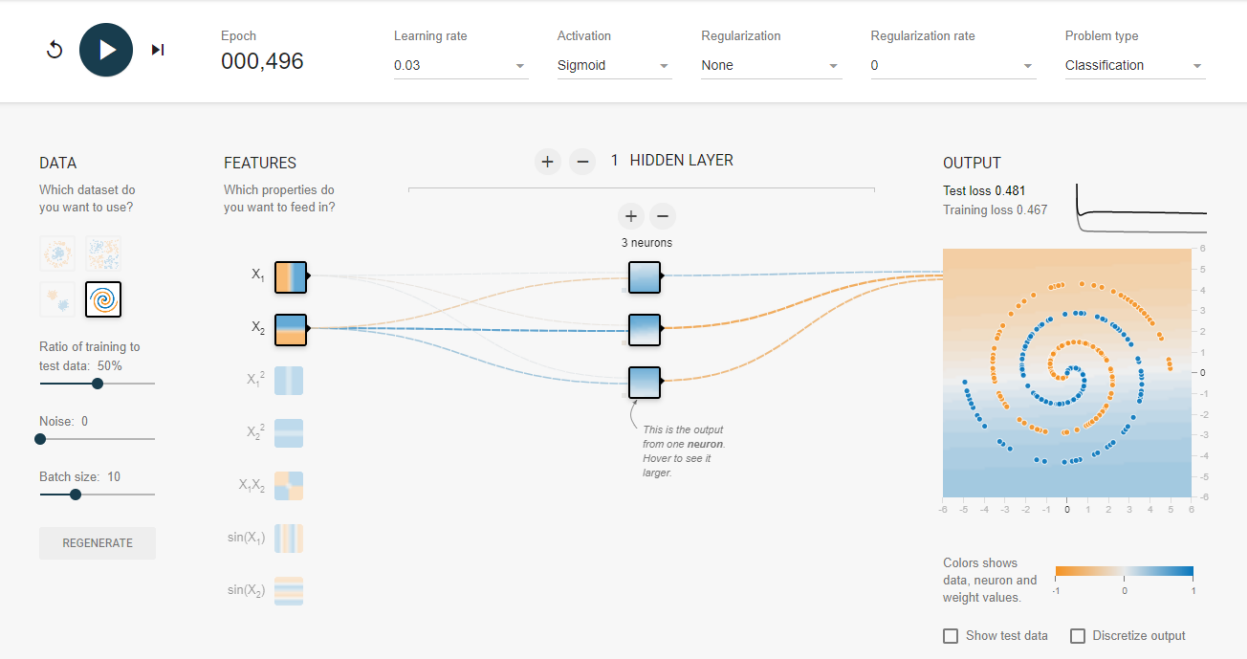


Figure 15 Sigmoid

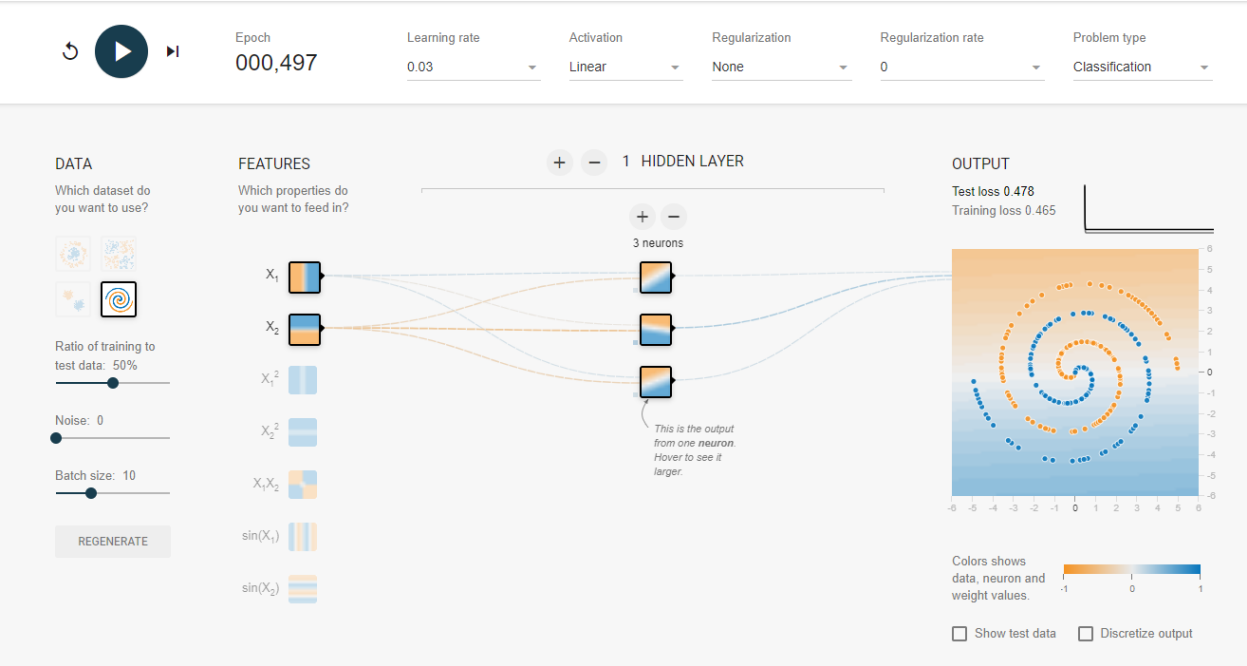


Figure 16 Linear

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 سازیReL .

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. مدل را کامپایل میکنیم با پارامترهای: تابع بهینه سازی adam(یک الگوریتم بهینه سازی پرکاربرد)، تابع خطای sparse_categorical_crossentropy(برای دسته بندی چند کلاسه) و معیار ارزیابی accuracy. مدل با ده epoch یادگیری فیت میکنیم.

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.

۵- تابع فعالسازی را تابع سیگموید در نظر میگیریم. مشتق تابع را طبق لینک گذاشته شده قرار میدهیم. تعداد نورون های لایه میانی را 4 در نظر میگیریم.

وزنهای لایه های میانی و لایه آخر که منجر به خروجی میشود طبق ابعاد ورودی و خروجی تابع، بصورت رندوم ایجاد میکنیم. نرخ یادگیری را 0.1 و تعداد epochهای یادگیری را 10000 در نظر میگیریم.

ورودی ها و خروجی ها را نیز طبق جدول xnor، initialize میکنیم.

در هر epoch یادگیری، آرایه ورودی را در وزنهای لایه میانی ضرب میکنیم. سپس مقدار سیگموید ماتریس را بدست می آوریم. دوباره خروجی سیگموید را در وزنهای لایه خروجی ضرب میکنیم. سپس مقدار ارور را در این مرحله حساب میکنیم. مقدار ارور برابر است با مابه التفاوت مقدار بدست آمده و خروجی target. در مرحله backpropagation، مقدار گرادیانت لایه میانی و خروجی را با اسفتاده از مشتق تابع سیگموید بدست می آوریم. پس از این مرحله، وزن ها را با استفاده از گرادیان دیسنت آپدیت میکنیم.

هر 1000 مرحله از این پروسه، مقدار ارور را پرینت میگیریم که دیده میشود این مقدار به مرور یادگیری کاهش می یابد که نشان میدهد مدل در حال یادگیری است.

۶- داده ها را لود میکنیم. داده ها به دو بخش تست و لرن تقسیم میشوند. یک مدل Sequential ایجاد میکنیم. لایه ورودی از ورودی های 28*28 تشکیل میشوند(اندازه هر تصویر دیتاست). لایه میانی از 128 نورون(این عدد به صورت تجربی نتیجه بهتری میدهد) و تابع فعال ساز ReLU(چون پیاده سازی آن پیچیدگی کمتری دارد و ساده تر محاسبات آن انجام میشود) تشکیل شده است. از dropout برای جلوگیری از overfitting داده ها استفاده میکنیم. لایه خروجی

هم 10 کلاس دارد(اعداد 0 تا 9) و تابع فعال ساز softmax. همانند سوال 4 مدل را کامپایل میکنیم. مدل را با ده epoch یادگیری fit میکنیم و نمودارهای loss و accuracy را نیز مانند سوال 4 رسم میکنیم.