به نام خدا

نگین اسماعیل زاده ۹۷۱۰۴۰۳۴

پروژه نهایی درس هوش مصنوعی

دکتر عبدی

بخش اول

در این بخش با استفاده از تابع MLPRegressor در کتابخانه ی آماده ی sklearn در پایتون به یادگیری نمونه هایی از توابع پرداخته شده (مسئله ی رگرسیون) و سپس اثر پارامتر های متعدد از جمله تغییر تعداد لایه های شبکه، تغییر تعداد نورون های هر لایه از شبکه، تغییر دامنه ی ورودی، تغییر تابع فعالسازی نورون ها، تغییر ماکزیمم تکرار فرایند یادگیری، تغییر مقدار داده های آموزش و پیچیدگی های متفاوت روی تابع تخمینی خروجی برسی شد. لازمبه ذکر است برای طبیعی تر بودن همواره دامنه ی داده های تست از داده های آموزش وسیع تر در نظر گرفته شده است.

۶ تابع متفاوت از جهت پیچیدگی برای این مطالعه استفاده شده اند که به ترتیب به صورت زیر تعریف شده اند:

1) $ax + b$ $\qquad\qquad$ $a\ and\ b\ could\ be\ random\ constants$

2) $x + |x|$

3) $0.5x^3$

4) $x^4 + x^3 + x^2 + x + 5$

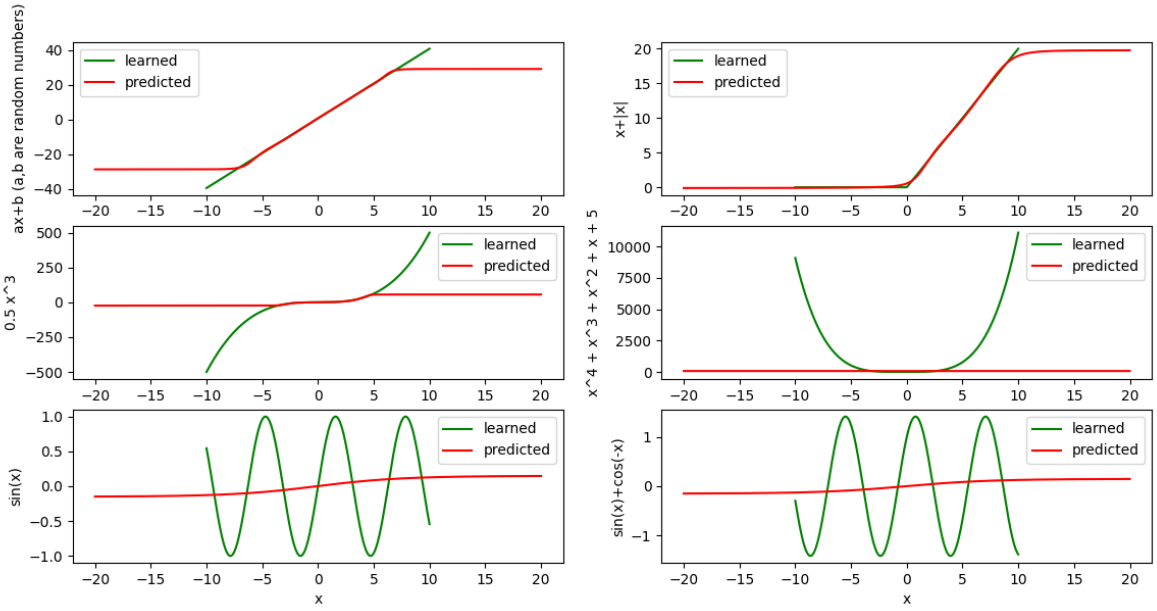
5) $\sin(x)$

6) $\sin(x) + \cos(-x)$

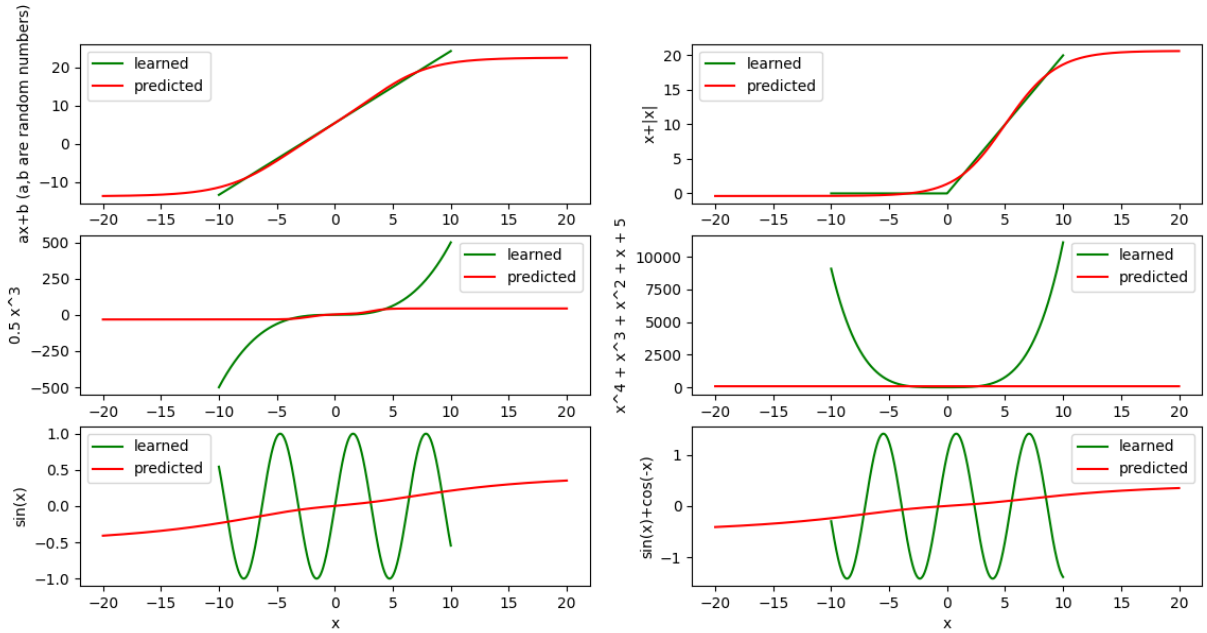
توجه : در هر شکل نمودار تابع داده های آموزشی با رنگ سبز و قلم ضخیم تر و تابع تخمینی توسط داده های تست با رنگ قرمز وقلم نازک تر نمایش داده شده اند.

ابتدا خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰۰، تابع فعالسازی logistic برای نورون ها، دامنه ی ورودی (10,10-) و تعداد ۱۰۰۰ داده ی ورودی برسی میکنیم و

فرض میکنیم این حالت اولیه هست (باقی تغییرات هرکدام برای اعتبار نتیجه نسبت به این حالت ثابت برسی میشوند).

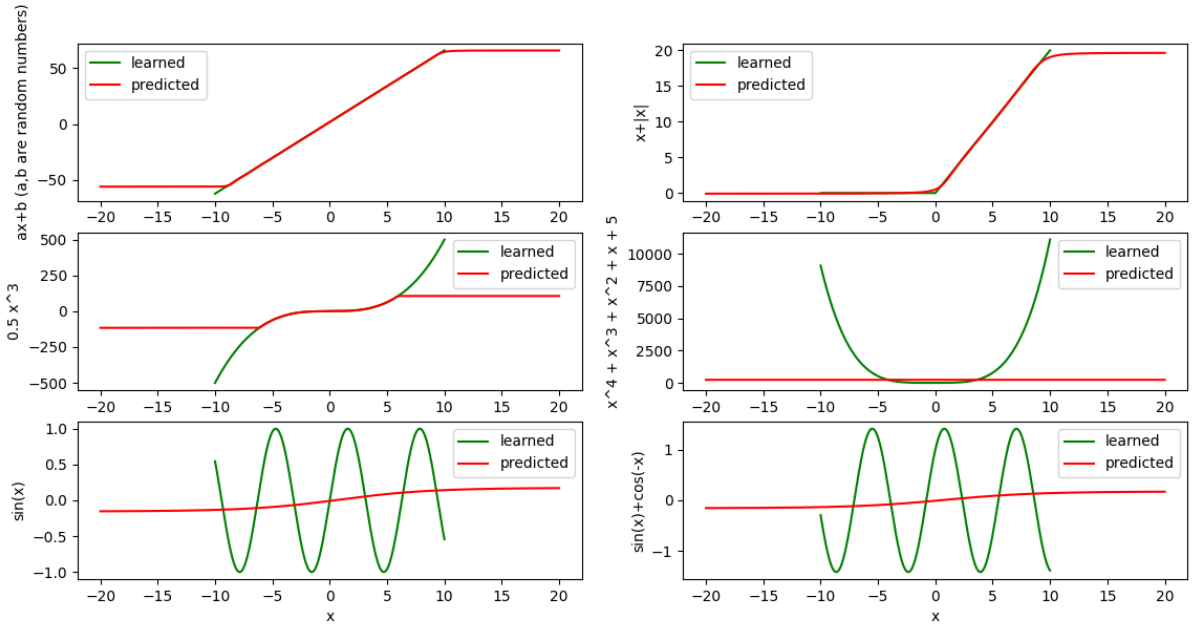


حال خروجی را برای شبکه ای شامل ۱ لایه نورون با ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰۰، تابع فعالسازی logistic برای نورون ها، دامنه ی ورودی (-10,10) و تعداد ۱۰۰۰ داده ی ورودی برسی میکنیم.



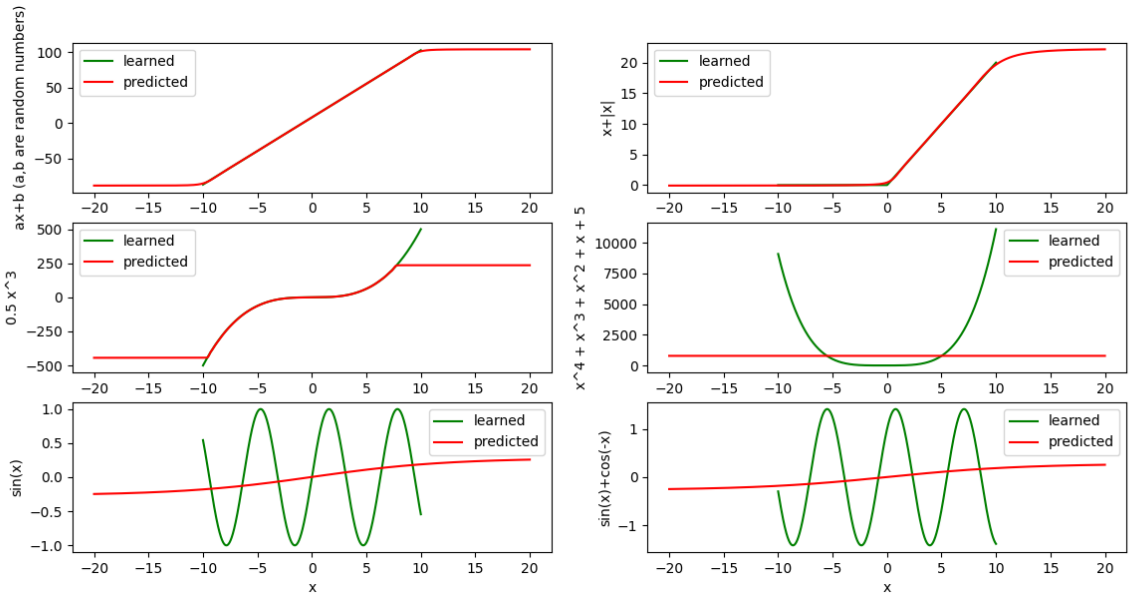
همانطور که مشاهده می شود افزایش تعداد لایه ها (مقایسه ی حالت قبلی با این حالت) میتواند به یادگیری بهتر کمک کند مخصوصا در مواردی که شکل از چند زیرمجموعه تشکیل شده است (گوشه ی تیز یا تغییر انحنا دارد). اما باید دقت کنیم برای توابع خیلی خیلی پیچیده ممکن است اینموضوع به تنهایی تغییر آن چنانی در بهبود یادگیری ایجاد نکند.

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۳۰۰۰، تابع فعالسازی logistic برای نورون ها، دامنه ی ورودی (-10,10) و تعداد ۱۰۰۰ داده ی ورودی برسی میکنیم.



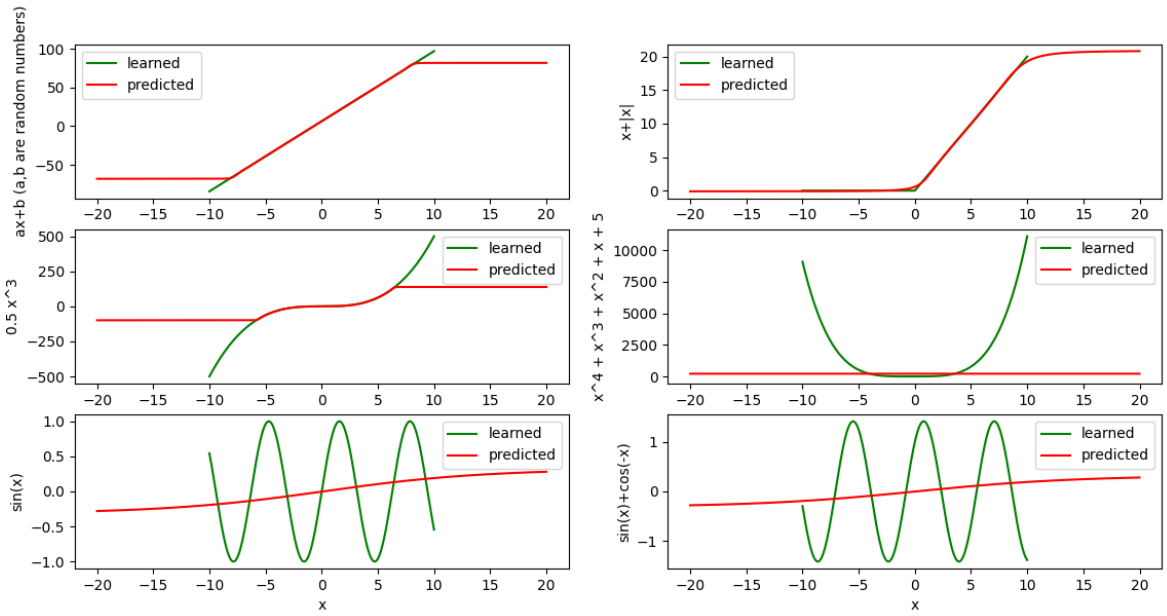
مشاهده می کنیم که افزایش تعداد چرخه ها برای یادگیری میتواند به یادگیری بهتر بیشتر توابع کمک کند. به طور کلی عموما هر چقدر زمان و حجم محاسبات به ما اجازه دهد طبیعتا افزایش تعداد چرخه ها برای یادگیری به کمتر شدن خطای یادگیری کمک میکند. اما باید در نظر داشت که از یک حدی به بعد امکان بهبود توسط این پارامتر وجود ندارد(به حد همگرایی میرسیم)، همچنین در برخی مسائل مانند دو تابع پیچیده ی آخر ممکن است این موضوع به ما کمک محسوسی نکند چون مقدار خود خطا کم است و در واقع تغییرات ریز و سریع یک تابع باید دنبال شوند.

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰۰، تابع فعالسازی logistic برای نورون ها، دامنه ی ورودی (-10,10) و تعداد ۱۰۰۰۰ داده ی ورودی (ده برابر شدن تعداد نقاط ورودی برای آموزش) برسی میکنیم.



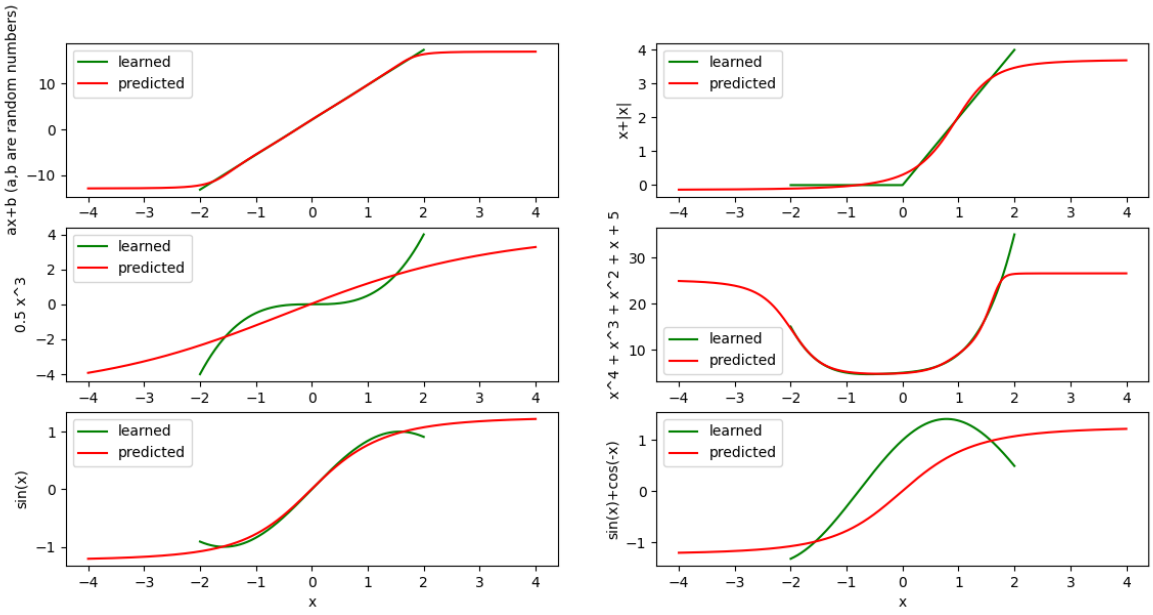
مشاهده میکنیم که در خیلی از توابع بیشتر کردن تعداد داده های آموزش میتواند عملا باعث دقیق شدن تخمین شود که انتظار این موضوع را هم داشتیم که هرچه داده های آموزشی بیشتر باشد تخمین بهتری از خروجی یاد گرفته شود. مخصوصا در تابع به نسبت پیچیده ی سوم این تغییر بسیار محسوس است.

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۳۰ عدد نورون، تعداد چرخه ی ۱۰۰۰، تابع فعالسازی logistic برای نورون ها، دامنه ی ورودی (-10,10) و تعداد ۱۰۰۰ داده ی ورودی برسی میکنیم.



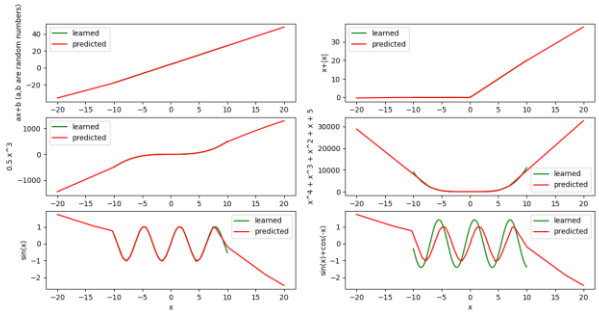
مشاهده میکنیم که افزایش تعداد نورون های هر لایه نیز میتواند به افزایش دقت دنبال کردن تغییرات تابع و آموزش بهتر و نتیجتا تخمین بهتر کمک کند.

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰۰، تابع فعالسازی logistic برای نورون ها، دامنه ی ورودی (-2,2) و تعداد ۱۰۰۰ داده ی ورودی برسی میکنیم.

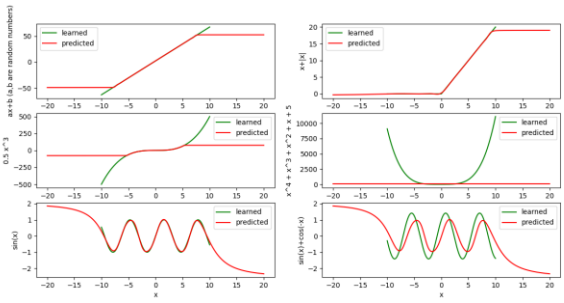


مشاهده میشود که تغییر دامنه ی وروی هم میتواند تاثیر گذار باشد اما تاثیر آن برای توابع مختلف متفاوت است. برای مثال در توابع ساده تر (با انحنا یا شکستگی کمتر) ، وسعت دامنه ی خروجی به افزایش گستره ی اطمینان تخمین و یادگیری بهتر کمک میکند در حالی که در توابع پیچیده تر که انحنا و تغیرات بیشتر و سریع تر دارند کاهش وسعت دامنه میتواند به بهتر دنبال کردن تغییرات و تخمین بهتر تابع در بازه ی کوچک کمک بیشتری کند.

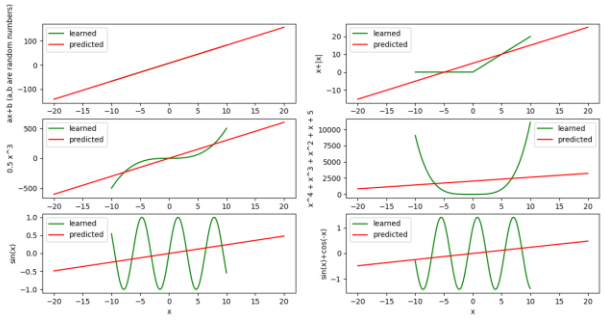
حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰۰، تابع فعالسازی های متفاوت tanh,relu,identity,logistic برای نورون ها، دامنه ی ورودی (-10,10) و تعداد ۱۰۰۰ داده ی ورودی برسی میکنیم.



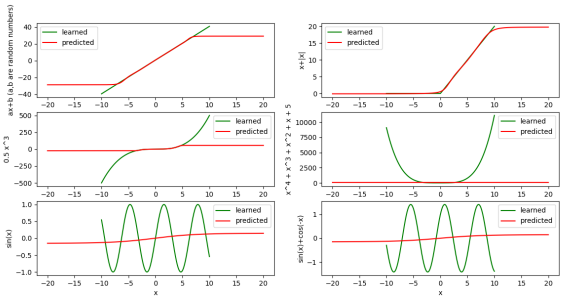
activation function = relu



activation function = tanh



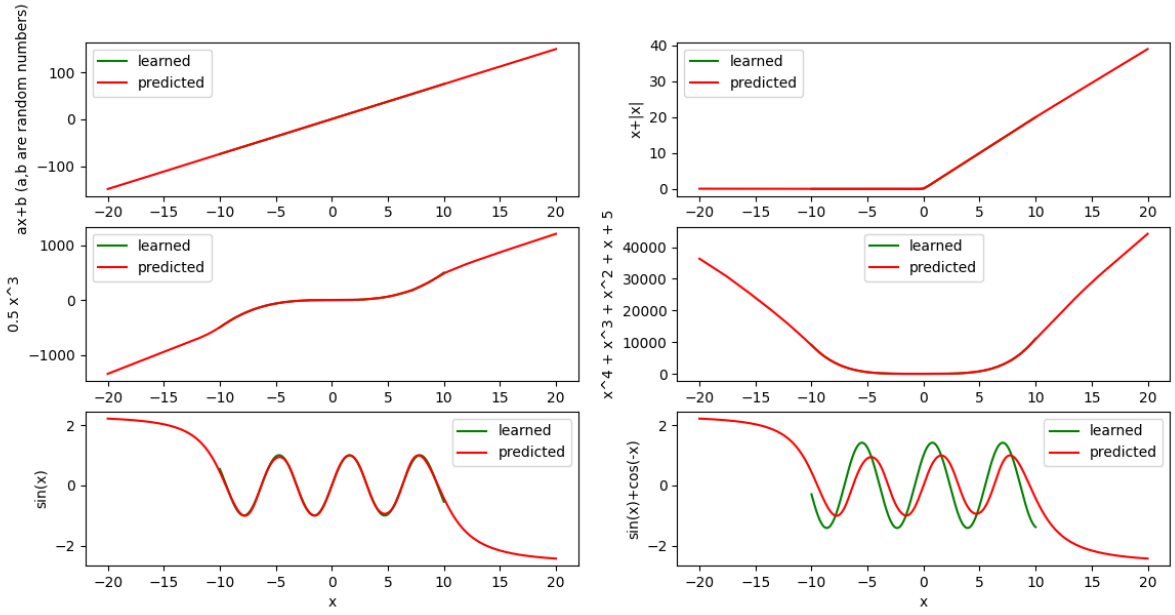
activation function = identity



activation function = logistic

علاوه بر تاثیرات پارامتر های ذکر شده در سوال تاثیر تابع فعالسازی نورون در مسئله ی رگرسیون از نظر من خیلی مهم آمد و آن را برسی کردم. همانطور که مشاهده میکنیم این پارامتر بسیار بسیار تاثیر گذار و تا حدی تعیین کننده ی اصلی است. علت این امر هم این هست که با انتخاب توابع فعالسازی مناسب ممکن است یادگیری یک مسئله ی پیچیده ی غیر خطی به یک مسئله ی خط تبدیل شود (برای مثال توابع پریودیک مثلثاتی همانطور که مشاهده میکنیم در دو حالت بسیار متفاوت از حالت های دگیر تخمین زده میشوند، همینطور توابع خطی ..). بنابراین اگر از جنس تابع تا حدودی آگاه باشیم با تغییر مناسب این پارامتر و استفاده از پارامتر های موثر دیگر که بالاتر ذکر شد میتوانیم دقت یادگیری خود را افزایش دهیم. همچنین در این صورت زمان لازم برای یادگیری شبکه نیاز کاهش میابد به علت شده شدنی که توضیح داده شد.

در شکل زیر با در نظر گرفتن توابع فعالسازی مناسب برای هر کدام از توابع (identity برای تابع اول، relu برای توابع ۲ تا ۴ و tanh برای دو تابع ۵ و ۶) و تعداد نورون های ۳۰ عدد در ۳ لایه و با ۲۰۰۰ چرخه تکرار، دامنه ی ورودی (-10,10) و تعداد ۲۰۰۰ داده ی ورودی نتیجه را مشاهده میکنیم.

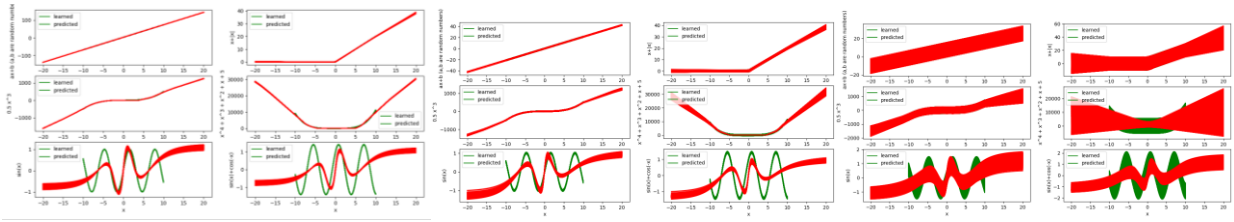


همانطور که مشاهده میکنیم در اثر ترکیب مجموعه ی این پارامتر ها به دقت بسیار بالاتری در تخمین توابع میتوانیم برسیم.

بخش دوم

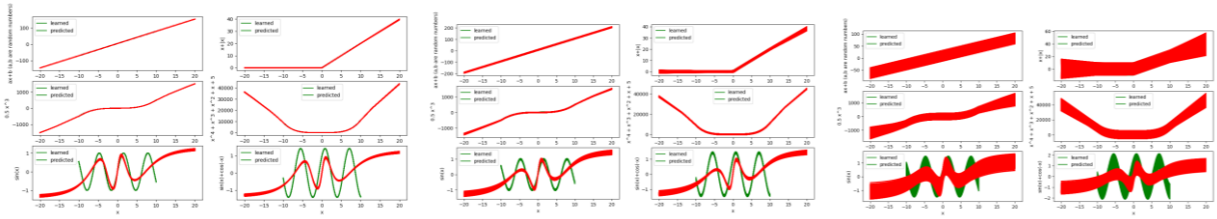
در این قسمت در دو حالت خروجی را برای ۳ مقدار نویز کم، متوسط و زیاد برسی میکنیم. حالت اول زمانی که شبکه ی ما شبکه ای با سه لایه نورون هر لایه ۱۵ نورون ، ماکزیمم تعداد چرخه ی ۱۰۰۰ ، تعداد نقاط ورودی ۱۰۰۰ و تابع فعالسازی متناسب با هر تابع باشد وحالت دوم شبکه ی خیلی قوی تری با ۳۰ نورون در هر لایه، ماکزیمم تعداد چرخه ی ۲۰۰۰، تعداد نقاط ورودی ۲۰۰۰ و تابع فعالسازی متناسب با هر تابع، اثر نویز را در این چند حالت مورد مطالعه قرار میدهیم.

شبکه ی اول



شدت نویز زیاد (متناسب با دامنه) شدت نویز متوسط (متناسب با ۱۰ درصد دامنه) شدت نویز کم (متناسب با ۱ درصد دامنه)

شبکه ی دوم



شدت نویز زیاد (متناسب با دامنه) شدت نویز متوسط (متناسب با ۱۰ درصد دامنه) شدت نویز کم (متناسب با ۱ درصد دامنه)

همانطور که مشاهده میشود چون هر دو شبکه ی فوق به نسبت قوی هستند در برابر مقدار کم نویز تقریبا مقاوم اند اما با این حال تفاوت بین حالت بدون نویز در توابع کمی پیچیده تر حتی در حالت نویز بسیار کم نیز مشهود هست. اما با افزایش مقدار نویز از یک حدی بیشتر شبکه قادر به یادگیری تابع اصلی نمی باشد (حتی با وجود

تمام تمهیداتی ک در بخش قبل برای بهبود شبکه گفتیم). علت این موضوع هم این هست ک مقدار نویز باعث ایجاد خروجی های تصادفی و به تبع آن ایجاد عدم اطمینان و تغییرات متعدد شبکه، طولانی شدن فرایند آموزش و خطای بسیار بالا در توابع پیچیده تر میشود. همانطور که میبینم روی توابع ساده ممکن است نیز اثر زیادی ایجاد نکند یا حداقل میتوانیم بگوییم **اثرات تخریبی حاصل از نویز** را در **توابع ساده** میتوانیم تا حد خوبی با **تمهیدات تقویت شبکه** که در بخش قبل گفته شد **جبران کنیم** اما در **توابع پیچیده تر** به دلیل اینکه خود تابع به خودی خود تغییرات سریع و شدید دارد و نویز هم خاصیت تصادفی دارد و بالتبع تغییرات شیدید و سریع مستقل خود را دارد **تقریبا نمیتوان** تخریب بیش از اندازه ی سیگنال اصلی را جبران کرد و باعث میشود خروجی نیز بسیار خطا دار و بسیار نویزی شود.

بخش سوم

مطالعه ی بخش اول را اینبار روی توابعی از مراتب بالاتر نیز انجام میدهیم تا صحت گزاره ها مجددا برسی شود. در این بخش به دلیل بالا بودن مرتبه ی تابع و ماسب نبودن معیار گرافیکی صرفا از معیار عددی مقدار loss یا همان خطا برای برسی و مقایسه ها استفاده می کنیم. برای این بخش ۴ تابع دو بعدی و سه بعدی زیر با درجه سختی های متفاوت در نظر گرفته شده اند :

1) $ax_1 + bx_2 + c \qquad a, b \; and \; c \; could \; be \; random \; constants$

2) $x_1 + |x_1| + \; x_2 + |x_2| + x_3 + |x_3|$

3) ${x_1}^4 + {x_2}^3 + {x_3}^2 + x_2 + 5$

4) $\sin(x_1) + \cos(-x_2)$

ابتدا خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰۰، تابع فعالسازی logistic برای نورون ها، دامنه ی ورودی (10,10-) و تعداد ۱۰۰۰ داده ی ورودی (به ازای هر کدام از ابعاد ورودی) برسی میکنیم و فرض میکنیم این حالت اولیه هست (باقی تغییرات هرکدام برای اعتبار نتیجه نسبت به این حالت ثابت برسی میشوند).

```
Iteration 999, loss = 4.11973662
Iteration 1000, loss = 4.10624151
```

تابع اول

```
Iteration 999, loss = 2.81999981
Iteration 1000, loss = 2.81127127
```

تابع دوم

```
Iteration 999, loss = 5601671.88708062
Iteration 1000, loss = 5601518.15273197
```

تابع سوم

```
Iteration 74, loss = 0.50019564
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع چهارم

حال خروجی را برای شبکه ای شامل ۱ لایه نورون با ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰۰، تابع فعالسازی logistic برای نورون ها، دامنه ی ورودی (10,10-) و تعداد ۱۰۰۰ داده ی ورودی(به ازای هر کدام از ابعاد ورودی) برسی میکنیم.

```
Iteration 999, loss = 453.70607209
Iteration 1000, loss = 453.17776650
```

تابع اول

```
Iteration 999, loss = 14.61075368
Iteration 1000, loss = 14.58237432
```

تابع دوم

```
Iteration 999, loss = 6083215.03445416
Iteration 1000, loss = 6083044.35793109
```

تابع سوم

```
Iteration 39, loss = 0.45398414
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع چهارم

همانطور که مشاهده می شود افزایش تعداد لایه ها (مقایسه ی حالت قبلی با این حالت) باعث شده بود که loss عموما کمتر شود، به خصوص در ۳ تابع ساده تر به وضوح مشخص است که کاهش لایه خطا را بسیار بالا برده و توابعی که در مرحله ی قبل تقریبا به طور قابل قبولی یاد گرفته شده بودند با کاهش لایه ها یاد گرفته نشدند. البته تابع آخر که مثلثاتی است و مرتبه خود تابع از اردر زیر ۱ هست در هیچکدام از دو حالت loss

قابل قبولی نداشته و این نشان میدهد برای توابع پیچیده تر می تواند افزایش لایه ها به تنهایی تاثیر چندانی نداشته باشید.

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۳۰۰۰، تابع فعالسازی logistic برای نورون ها، دامنه ی ورودی (10,10-) و تعداد ۱۰۰۰ داده ی ورودی(به ازای هر کدام از ابعاد ورودی) برسی میکنیم.

```
Iteration 2999, loss = 82.72776100
Iteration 3000, loss = 82.61795352
```

تابع اول

```
Iteration 2548, loss = 0.18611566
Iteration 2549, loss = 0.18581793
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع دوم

```
Iteration 2999, loss = 5189737.47717202
Iteration 3000, loss = 5189594.92290068
```

تابع سوم

```
Iteration 90, loss = 0.47468941
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع چهارم

مشاهده می کنیم که در بیشتر توابع (به ویژه توابع ساده تر) افزایش تعداد چرخه ها باعث کمتر شدن مقدار خطا شده است و پس میتواند به همگرایی الگوریتم آموزش پرسپترون و در نتیجه یادگیری بهتر و بیشتر توابع کمک کند. به طور کلی عموما هر چقدر زمان و حجم محاسبات به ما اجازه دهد طبیعتا افزایش تعداد چرخه ها برای یادگیری به کمتر شدن خطای یادگیری کمک میکند. اما باید در نظر داشته افزایش این پارامتر تا یک میزانی به بهبود پاسخ کمک میکند اما از آن میزان به بعد چون به حد همگرایی رسیده ایم افزایش تعداد چرخه ها کمکی به تفاوت پاسخ نمیکند. در برخی مسائل مانند دو تابع پیچیده ی آخر ممکن است این موضوع به ما کمک محسوسی نکند چون برای مثال در تابع سوم مشخص است که مقدار خطا با این که کاهش داشته اما بسیار زیاد است و شاید برای همگرایی لازم باشد تعداد چرخه حتی تا ۴ یا ۵ برابر بیشتر هم شود. در تابع چهارم که تابع بسیار پیچیده با تغییرات زیاد در حدود دامنه ی کمتر از ۱ هست افزایش تعداد چرخه ها کمکی به تغییر تابعیاد گرفتهشده نمیکند چون اصلا مشکل همگرایی در یادگیری این تابع نداشته ایم.

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰۰، تابع فعالسازی logistic برای نورون ها، دامنه ی ورودی (10,10-) و تعداد ۱۰۰۰۰ داده ی ورودی (ده برابر شدن تعداد نقاط ورودی به ازای هر کدام از ابعاد ورودی توابع برای آموزش) برسی میکنیم.

```
Iteration 367, loss = 0.07648806
Iteration 368, loss = 0.07638705
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع اول

```
Iteration 694, loss = 0.14473525
Iteration 695, loss = 0.14486079
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع دوم

```
Iteration 999, loss = 4436816.67635951
Iteration 1000, loss = 4435832.69706570
```

تابع سوم

```
Iteration 130, loss = 0.48380925
Iteration 131, loss = 0.48408720
```

تابع چهارم

مشاهده میکنیم که در خیلی از توابع بیشتر کردن تعداد داده های آموزش میتواند عملا باعث دقیق شدن یادگیری می شود که انتظار این موضوع را هم داشتیم که هرچه داده های آموزشی بیشتر باشد تخمین بهتری از خروجی یاد گرفته شود و میزان loss شبکه و همچنین تعداد چرخه ی لازم برای همگرایی فرایند آموزش کاهش قابل توجی داشته باشد. مخصوصا در تابع به نسبت پیچیده ی سوم این تغییر بسیار محسوس است.

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۳۰ عدد نورون، تعداد چرخه ی ۱۰۰۰، تابع فعالسازی logistic برای نورون ها، دامنه ی ورودی (10,10-) و تعداد ۱۰۰۰ داده ی ورودی (به ازای هر کدام از ابعاد ورودی) برسی میکنیم.

```
Iteration 999, loss = 219.36964757
Iteration 1000, loss = 218.88475676
```

تابع اول

```
Iteration 999, loss = 0.60325791
Iteration 1000, loss = 0.60139362
```

تابع دوم

```
Iteration 999, loss = 5372042.70081118
Iteration 1000, loss = 5371728.19854272
```

تابع سوم

```
Iteration 76, loss = 0.46370222
Iteration 77, loss = 0.46365902
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع چهارم

مشاهده میکنیم که افزایش تعداد نورون های هر لایه نیز میتواند به loss و در نتیجه آموزش بهتر و نتیجتا تخمین بهتر کمک کند. البته در تابع خیلی پیچیده شاید تغییر کم تعداد نورون ها به تنهایی خیلی کار ساز نباشد مثل تابع آخر که البته خواهیم دید با انتخاب مناسب تابع فعالسازی میشود این اثر گذاری را بسیار بیشتر کرد.

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰۰، تابع فعالسازی logistic برای نورون ها، دامنه ی ورودی (-2,2) و تعداد ۱۰۰۰ داده ی ورودی (به ازای هر کدام از ابعاد ورودی) برسی میکنیم.

```
Iteration 999, loss = 0.08998378
Iteration 1000, loss = 0.08989066
```

تابع اول

```
Iteration 822, loss = 0.06979048
Iteration 823, loss = 0.06970611
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع دوم

```
Iteration 999, loss = 1.03678190
Iteration 1000, loss = 1.03300453
```

تابع سوم

```
Iteration 164, loss = 0.11729260
Iteration 165, loss = 0.11724421
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع چهارم

مشاهده میشود که تغییر دامنه ی وروی هم میتواند تاثیر گذار باشد. به نظر میرسد هر چه قدر بازه را کوچکتر کنیم تغییرات در بازه ی کوپک بسیار دقیق تر یاد گرفته میشوند و loss شبکه پایین می آید. به خصوص در توابع پیچیده تر تاثیر این موضوع بسیار محسوس است (مانند تابع ۳ و۴)، به این علت که توابع پیچیده تر تغییرات انحنی بیشتر وسریع تری دارند ک در یک دامنه وسیع یادگیری و دنبال آن سخت تر میشود.

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰۰، تابع فعالسازی های متفاوت tanh,relu,identity,logistic برای نورون ها، دامنه ی ورودی (10,10-) و تعداد ۱۰۰۰ داده ی ورودی (به ازای هر کدام از ابعاد ورودی) برسی میکنیم.

تابع فعالسازی logistic (حالت اولیه) :

```
Iteration 999, loss = 4.11973662
Iteration 1000, loss = 4.10624151
```

تابع اول

```
Iteration 999, loss = 2.81999981
Iteration 1000, loss = 2.81127127
```

تابع دوم

```
Iteration 999, loss = 5601671.88708062
Iteration 1000, loss = 5601518.15273197
```

تابع سوم

```
Iteration 74, loss = 0.50019564
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع چهارم

تابع فعالسازی relu :

```
Iteration 550, loss = 0.03240129
Iteration 551, loss = 0.03232414
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع اول

```
Iteration 583, loss = 0.03120597
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع دوم

```
Iteration 999, loss = 58057.57398964
Iteration 1000, loss = 57775.19683059
```

تابع سوم

```
Iteration 624, loss = 0.06484855
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع چهارم

تابع فعالسازی identity :

```
Iteration 91, loss = 0.00988830
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع اول

```
Iteration 47, loss = 21.19692248
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع دوم

```
Iteration 143, loss = 3584426.34792915
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع سوم

```
 Iteration 32, loss = 0.49046601
 Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع چهارم

تابع فعالسازی tanh :

```
Iteration 999, loss = 210.00116748
Iteration 1000, loss = 209.49869778
```

تابع اول

```
Iteration 999, loss = 0.73133632
Iteration 1000, loss = 0.72772001
```

تابع دوم

```
Iteration 999, loss = 4972690.51151533
Iteration 1000, loss = 4972543.08266570
```

تابع سوم

```
Iteration 495, loss = 0.19923606
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع چهارم

مجددا مشاهده میکنیم در توابع چندی هم پارامتر تابع فعالسازی بسته به جنس مسئله بسیار تعیین کننده است که علت این امر هم در بخش اول توضیح داده شد. از مقایسه ی مقادیر loss متوجه میشویم که برای مثال در تابع پیچیده ی مثلثاتی چهارم، تابع فعالسازی tanh بسیار به کاهش خطا و همگرایی آموزش کمک کرده است. استفاده از توابع فعلسازی مناسب بسته به جنس تابع مسئله در کنار مقادیر مناسبی از پارامتر های قبلی ذکر ده میتواند نتیجه بهینه را به ما بدهد. حالت زیر نتیجه انتخاب مناسب تمام پارامتر هاست :

(توابع فعالسازی مناسب برای هر کدام از توابع (identity برای تابع اول، relu برای توابع ۲و۳ و tanh برای تابع ۴) و تعداد نورون های ۳۰ عدد در ۳ لایه و با ۲۰۰۰ چرخه تکرار، دامنه ی ورودی (10,10-) و تعداد ۲۰۰۰ داده ی ورودی به ازای هر کدام از ابعاد ورودی)

```
Iteration 34, loss = 0.01811155
Iteration 35, loss = 0.01811143
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع اول

```
Iteration 372, loss = 0.02285243
Iteration 373, loss = 0.02300590
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع دوم

```
Iteration 1003, loss = 1331.10412255
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

تابع سوم

```
Iteration 361, loss = 0.06849643
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

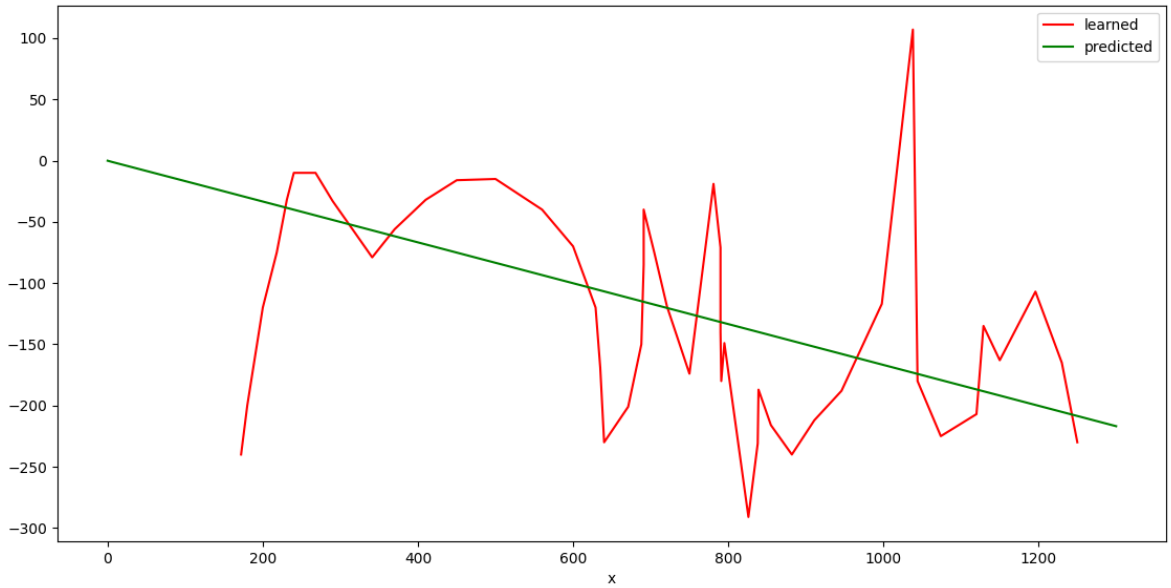
تابع چهارم

همانطور که مشاهده میکنیم در اثر ترکیب مجموعه ی این پارامتر ها به خطای بسیار کمتری در یادگیری توابع میتوانیم برسیم.

بخش چهارم

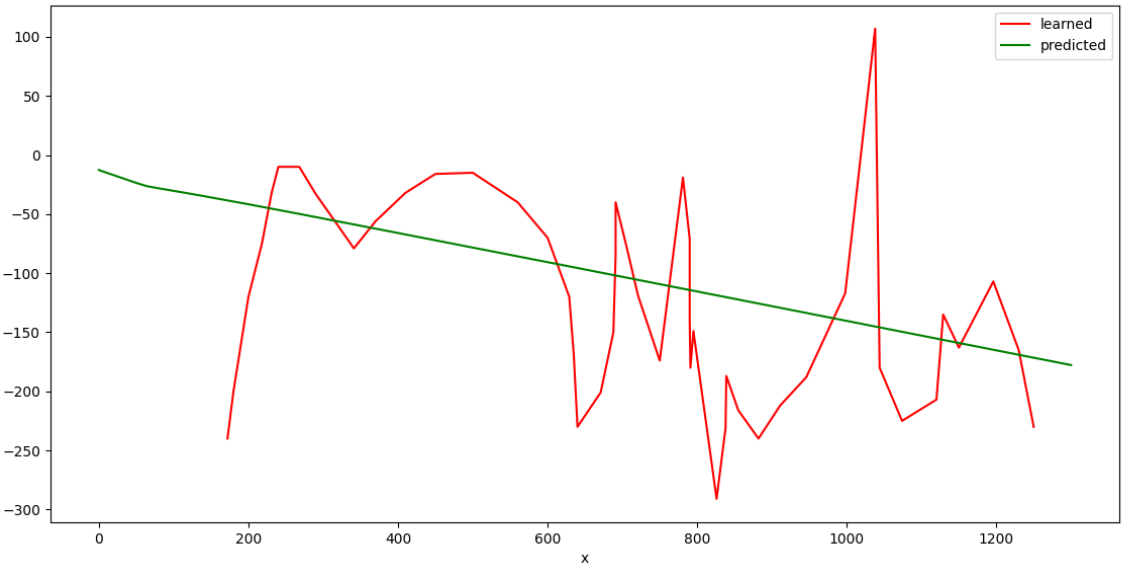
در این بخش آزمایش های انجام شده میزان سختی (از نظر واقعی بودن تخمین و همچنین پیچیدگی شبکه ی مورد نیاز برای یک تخمین خوب) برای آموزش یک شبک با داده های پراکنده ی دلخواه (خط خطی) را برسی میکنند.

شکل زیر نمونه ی اول خط خطی هست، قست قرمز رنگ بخشی است که با نمونه های نشان داده شده(نقاط آبی در تصویر) به یک شبکه داده شده است.



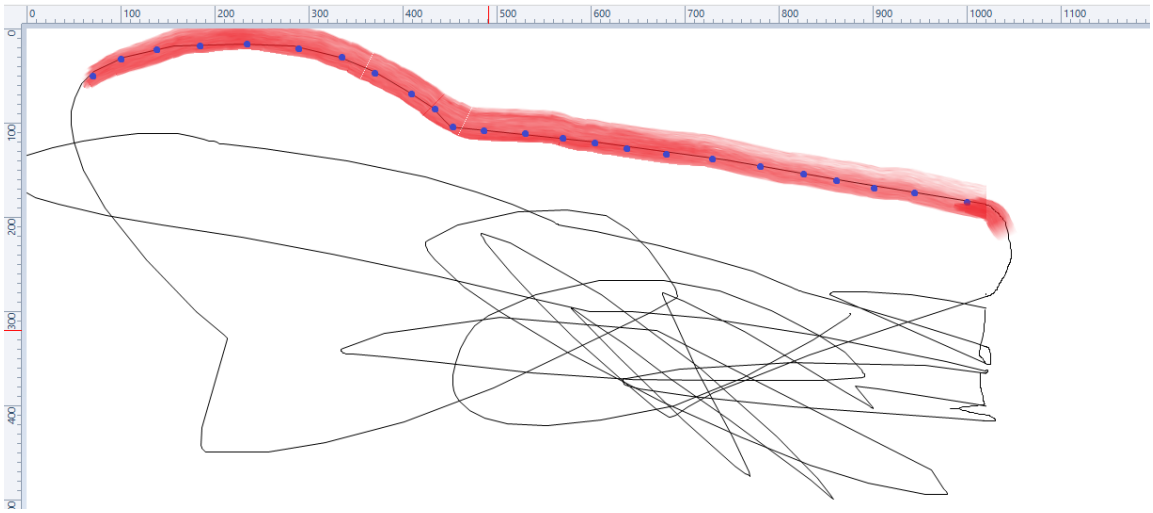
شکل بالا نمونه ی های واقعی و خروجی شبکه آموزش دیده با ۱ لایه نورون که شامل ۱۵ عدد نورون است را نشان میدهد (باقی پارامترها به بهترین نحو تعیین شده اند تا کوچکترین شبکه ی ممکن را داشته باشیم، در نیجه تابع فعالسازی متناسب و تعداد ۱۰۰۰۰چرخه در نظر گرفته شده است). اما با این حال مشاهده میکنیم

که یادگیری خوب انجامنشده بنابراین تعداد لایه ها را به ۵ لایه هر کدام با ۱۰۰ نورون افزایش دادیم!!!، تیه به صورت زیر است :

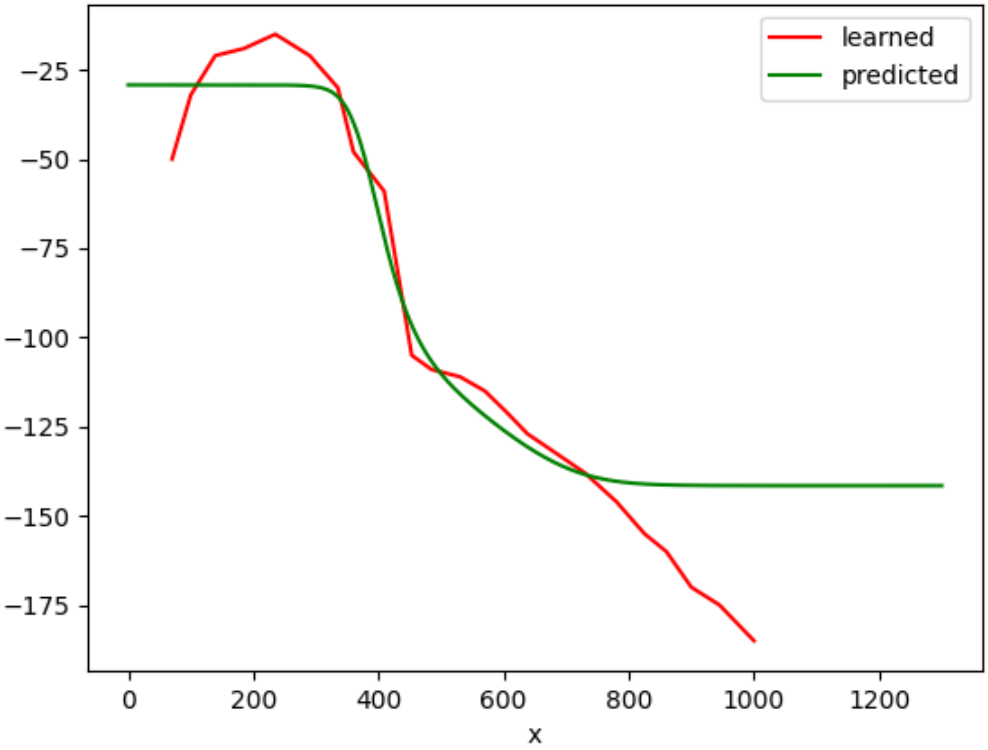


نشاهده میکنیم که این خط خطی بسیار خشمگین 😊 است. زیرا با شکبه ای به اندازه ی ۵۰۰ نورون هم نمیتوان تخمین خوبی از آن ارائه داد و عملا با حالت قبل تفاوتی نکرد. البته این مشکل بیشتر به این خاطر است که تعداد نمونه ها برای تابعی با این تغییرات شدید و سریع بسیار کم است که بتوانیم شبکه ای به این پیچیدگیرا آموزش دهیم. تعداد پرش های ناگهانی همانطور که میبینم بالا است و در این حالت شبکه های بسیار پیچیده هم با تعداد داده ها محدود قادر به یادگیری تغییرات نیستند.

یک نمونه ی دیگر را برسی میکنیم :

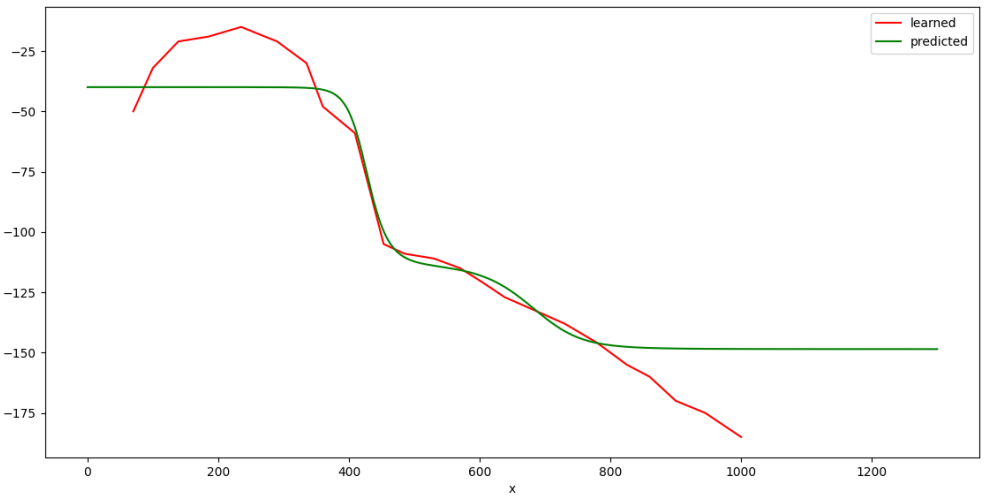


ابتدا خروجی شبکه ای تک لایه با ۳۰ نورون و پارامتر های دیگر در بهینه ترین حالت را برسی میکنیم:



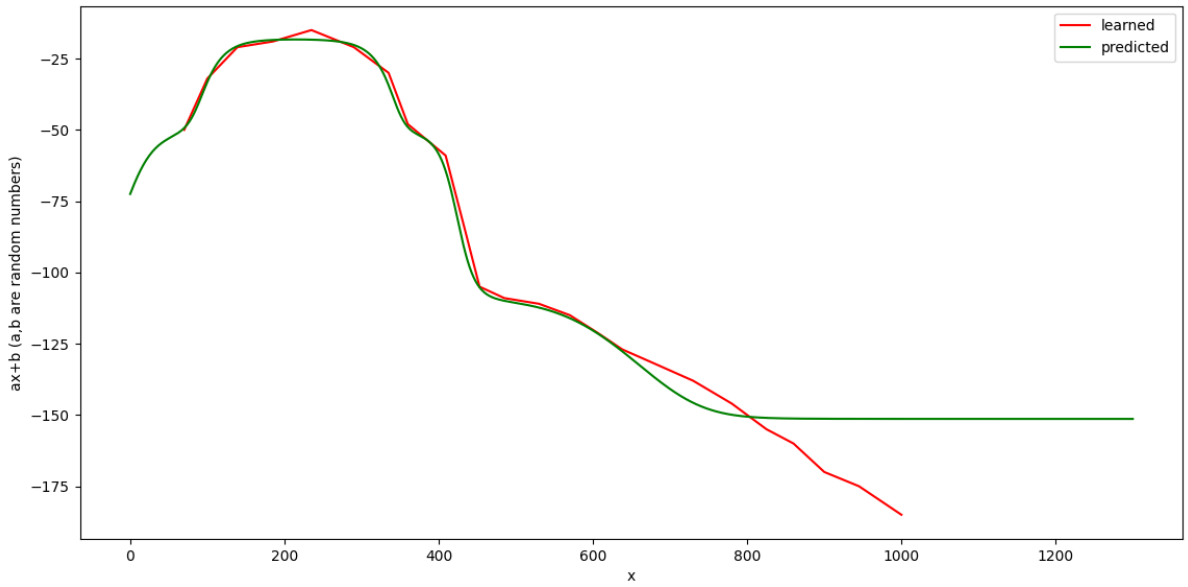

```
Iteration 9999, loss = 122.67664066
Iteration 10000, loss = 121.83410885
```

اینبار اندازه ی شبکه را به ۵ لایه، لایه ی اول و دوم هر کدام ۳۵ نورون، لایه ی سوم ۸۰ نورون، لایه ی چهارم ۵ و لایه ی آخر ۱۰۰ نورون (مجموعا ۲۵۵ نورون) گسترش میدهیم نتیجه به این شکل می باشد:




```
Iteration 3014, loss = 106.04981626
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

همانطور که میبینیم انحنای تابع تخمینی نسبت به حالت قبل یک مقدار بهبود داشت، بنابراین این بار تعداد نورون ها را ۵ لایه هر کدام با ۱۰۰ نورون در نظر گرفتیم و نتیجه به این صورت شد:




```
Iteration 3762, loss = 56.93761005
Iteration 3763, loss = 54.89508661
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

مشاهده میکنیم ک تخمین تقریبا دقیق شده است اما تعداد نورون های لازم در اینحالت هم بسیار زیاد بود. اما در مقایسه با نمونه ی قبل که پرش های ناگهانی بسیاری داشت آموزش شبکه به مراتب بهتر انجام شده است.

نتایج :

۱) افزایش تعداد چرخه ها تا یک جایی باعث بهبود نتیجه ی آموزش میشود اما از یک جایی تاثیری ندارد هر چقدر تعداد چرخه ها را بیشتر کنیم.

۲) کم بودن تعداد داده های آموزشی به خصوص در توابع پیچیده تر (دارای تغییرات بیشتر) بسیار نکته مهمی است که اگر از یک حدی دیتا ها به نسبت پیچیدگی تابع کم باشند نمیتوان با پیچیده ترین شبکه ها هم تخمین خوبی از روند کلی ارائه داد.

۳) در توابع بسیار پیچیده که فراز و نشیب های متعدد و مخصوصا پرش های ناگهانی دارند شبکه ی بسیار پیچیده تری (تعداد لایه ها و نورون های بیشتر) لازم است تا تغییرات بسیار سریع یاد گرفته شوند.

بخش پنجم

در این بخش با استفاده از تابع MLPClassifier در کتابخانه ی آماده ی sklearn در پایتون به یادگیری مسئله ی طبقه بندی دیتاست آماده ی اعداد انگلیسی دست نویس پرداخته ایم. ابتدا به کمک توابع آماده ی پایتون تصاویر را به آرایه های فلت نامپای تبدیل کردیم و برای هر تصویر با توجه به اسم تصویر لیبل را در آرایه ی دیگری ذخیره کردیم. سپس به صورت تصادفی بخشی از داده ها را برای تست و بخش دیگری را برای آموزش جدا کردیم. در نهایت با تغییر پارامتر های آموزش مانند تعداد لایه های شبکه، تعداد نورون های هر لایه شبکه، تعداد چرخه های الگورتیم آموزش، تابع فعالسازی نورون ها و نسبت داده های آموزش به تست، اثر هر کدام از این پارامتر ها را روی دقت طبقه بندی داده های تست برسی کردیم. دقت بیشتر به معنای تعداد لیبل های بیشتر درست حدس زده شده است و بنابراین معیار خوبی برای بهتر آموزش دیدن شبکه میباشد. توجه میکنیم که در تمام حال ها دقت به روش 5 fold cross validation محاسبه شده است.

ابتدا خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰، تابع فعالسازی logistic برای نورون ها و در نظر گرفتن ۷۰ درصد داده ها برای آموزش و ۳۰درصد برای تست برسی میکنیم و فرض میکنیم این حالت اولیه هست (باقی تغییرات هرکدام برای اعتبار نتیجه نسبت به این حالت ثابت برسی میشوند). در این حالت داریم :

```
Accuracy =  0.7340501792114695 with std =  0.0131008364749083
```

تاثیر تعداد نورون ها ولایه ها :

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۳۰ عدد نورون، تعداد چرخه ی ۱۰۰، تابع فعالسازی logistic برای نورون ها و در نظر گرفتن ۷۰ درصد داده ها برای آموزش و ۳۰درصد برای تست برسی میکنیم. داریم :

```
Accuracy =  0.9157706093189963 with std =  0.0054357530064885595
```

حال خروجی را برای شبکه ای شامل ۱ لایه نورون با ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰، تابع فعالسازی logistic برای نورون ها و در نظر گرفتن ۷۰ درصد داده ها برای آموزش و ۳۰درصد برای تست برسی میکنیم. داریم :

```
Accuracy =  0.917921146953405 with std =  0.009095754509282811
```

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با ۱۵ نورون در لایه ی اول ، ۱۵ نورون در لایه ی دوم، ۱۰ نورون در لایه ی سوم ، تعداد چرخه ی ۱۰۰، تابع فعالسازی logistic برای نورون ها و در نظر گرفتن ۷۰ درصد داده ها برای آموزش و ۳۰درصد برای تست برسی میکنیم. داریم :

```
Accuracy =  0.6684587813620071 with std =  0.025420248353400912
```

از مجموعه ی این مشاهدات به نظر میرسد تعداد نورونهای لایه ها نقش بسیار کلیدی تری از عمق شبکه حداقل در این مسئله خاص دارد. با افزایش نورون ها به وضوح مشخص است که دقت طبقه بندی تغییر قابل توجهی میکند. اما نکته مهم برایتوجه این هست که اگر سایز شبکه از یک حدی بیشتر هم شود مجددا دقت خروجی افت خواهد کرد. این به این علت است که در یادگیری داده های آموزشی بیش برازش رخ میدهد و دقت روی داده های تست پایین می آید. بنابراین همواره باید این موضوع را در نظر بگیریم که با اینکه میتوان شبکه ی بسیار پیچیده ای ساخت که تمام داده های آموزشی را ب خوبی یاد بگیرد، معمولا نباید این کار را کرد چون پیچیدگی در مقابل قابلیت تعمیم پذیری می باشد و باعث میشود دقت در آزمایش محصول پایین بیاید. بنابراین ایده آل این است که به یک حدی از پیچیدگی شبکه اکتفا کنیم.

تاثیر تعداد چرخه های الگوریتم آموزش شبکه :

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۲۰۰، تابع فعالسازی logistic برای نورون ها و در نظر گرفتن ۷۰ درصد داده ها برای آموزش و ۳۰درصد برای تست برسی میکنیم. داریم :

```
Accuracy =  0.8028673835125447 with std =  0.032158348467609996
```

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۵۰۰، تابع فعالسازی logistic برای نورون ها و در نظر گرفتن ۷۰ درصد داده ها برای آموزش و ۳۰درصد برای تست برسی میکنیم. داریم :

```
Accuracy =  0.8591397849462366 with std =  0.009919965952041625
```

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۷۰۰، تابع فعالسازی logistic برای نورون ها و در نظر گرفتن ۷۰ درصد داده ها برای آموزش و ۳۰درصد برای تست برسی میکنیم. داریم :

```
Accuracy =  0.8620071684587816 with std =  0.010930430596399235
```

آزمایش های بالا نشان میدهد افزایش تعداد چرخه ها برای الگوریتم یادگیری تا یک جایی بسیار به آموزشبهتر شبکه و درنتیجه طبقه بندی دقیقتر کمک میکند اما مشخص است که از یک میزانی به بعد (که همان حد همگرایی الگوریتم است) افزایش این پارامتر تغییر بخصوصی در میزاندقت طبقه بندی ایجاد نمیکند.

تاثیر تابع فعالسازی نورون ها :

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰، تابع فعالسازی tanh برای نورون ها و در نظر گرفتن ۷۰ درصد داده ها برای آموزش و ۳۰درصد برای تست برسی میکنیم. داریم :

```
Accuracy =  0.9193548387096774 with std =  0.009346526745810239
```

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰، تابع فعالسازی relu برای نورون ها و در نظر گرفتن ۷۰ درصد داده ها برای آموزش و ۳۰درصد برای تست برسی میکنیم. داریم :

```
Accuracy =  0.9168458781362008 with std =  0.008141804080000373
```

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰، تابع فعالسازیidentity برای نورون ها و در نظر گرفتن ۷۰ درصد داده ها برای آموزش و ۳۰درصد برای تست برسی میکنیم. داریم :

```
Accuracy =  0.9232974910394265 with std =  0.016440637908501365
```

مجموعه ی آزمایش های بالا نشان میدهد که تغییر تابع فعالسازی بسیار میتواند در دقت خروجی تاثیر گزار باشد. علت هم به این دلیل است که دیتا های متفاوت میتوانند در یک شبکه با تابع فعالسازی خاصی رابطه ی

خطی داشته باشند در حالی که با تابع فعالسازی دیگری اینطور نباشد که در بخش اول مفصلا به آن پرداخته شد. همانطور که در این بخش هم دیدیم دقت طبقه بندی در حالت logistic حدود ۰.۷ بود اما برای مثال با ثابت ماندن پارامتر های دیگر در حالت tanh به ۰.۹ رسید که بسیار تغییر محسوسی میباشد و نشان میدهد برای این داده ها شبکه ی مناسب بهتر است تابع فعالسازی logistic نداشته باشد.

تاثیر تعداد داده های آموزشی (یا نسبت داده های آموزش به تست) :

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰، تابع فعالسازی logistic برای نورون ها و در نظر گرفتن ۹۰ درصد داده ها برای آموزش و ۱۰درصد برای تست برسی میکنیم. داریم :

```
Accuracy =  0.8088709677419355 with std =  0.0163603488445917154
```

حال خروجی را برای شبکه ای شامل ۳ لایه نورون با هر لایه ۱۵ عدد نورون، تعداد چرخه ی ۱۰۰، تابع فعالسازی logistic برای نورون ها و در نظر گرفتن ۶۰ درصد داده ها برای آموزش و ۴۰درصد برای تست برسی میکنیم. داریم :

```
Accuracy =  0.6684587813620071 with std =  0.025420248353400912
```

مجموعه ی آزمایش های بالا نشان میدهد که هر چه درصد بیشتری از داده ها را صرف آموزش شبکه کنیم یادگیری بهتر و در نتیجه دقت طبقه بندی روی داده های تست بیشتر می شود و برعکس که مشابه این موضوع را هم در بخش های قبل برسی کردیم. این تناسب نشان میدهد که برای تخمین مناسب روی تعداد وسیعی از داده ها تست لازم است به اندازه ی کافی داده ی آموزش نیز داشته باشیم.

با در نظر گرفتن مجموعه ای از پارامتر های مناسب (شبکه تک لایه با ۲۰ نورون با تابع فعاسازی tanh ، نسبت داده های آموزش به تست ۷۰ به ۳۰ و تعداد چرخه ی ۲۰۰) به نتیجه ی بهتری خواهیم رسید :

```
Accuracy =  0.9351254480286737 with std =  0.008881370389508174
```

بخش ششم

در این بخش مجددا با استفاده از تابع MLPRegressor به ساخت تابعی خطی برای رفع نویز تصاویر می پردازیم. در ابتدا به تصاویر load شده در بخش قبل نویز اضافه میکنیم. نحوه ی اضافه کردن نویز به اینصورت است که بصورت تصادفی برخی پیکسل های تصویر شدت نور متفاوت با شدت نور اصلی خود میگیرند و هر چه درصد این پیکسل ها بیشتر باشد نویز بیشتر است. بنابراین تابع noise_adder یک noise_ratio در ورودی خود میگیرد که پارامتری است که ما با آن تعیین میکنیم که چند درصد تصویر نویزی شود و نتایج را تحلیل میکینم. در این بخش برسی ها برای noise_ratio = 0.1,0.2,0.5 انجام شده است که یک طیفی ازنویزکم تا خیلی زیاد هست. در هر آزمایش مقدار accuracy به صورت میزان شباهت پیکسل های تصویر خروجی شبکه و تصویر original محاسبه شده است. همچنین به صورت تصادفی ۱۰ عدد از تصاویر خروجی در کنار تصویر original و تصویر نویزی ورودی به صورت گرافیکی نمایش داده شده اند. برای هر آزمایش هر دو معیار عددی و گرافیکی برای هر دو دسته ی داده ی های آموزشی و آزمایشی برای مقایسه نمایش داده شده اند.

ابتدا برای نویز ۳۰ درصد و نسبت داده های آموزشی به آزمایشی ۷۰ به ۳۰ و شبکه ای به فرم ۳ لایه و در هر لایه ۵۰۰ نورون با ۲۰۰۰ چرخه آزمایش را انجام دادیم. دقت در داده های آموزشی و آزمایشی به این صورت شد :

```
Accuracy =  0.9405050645537052
```

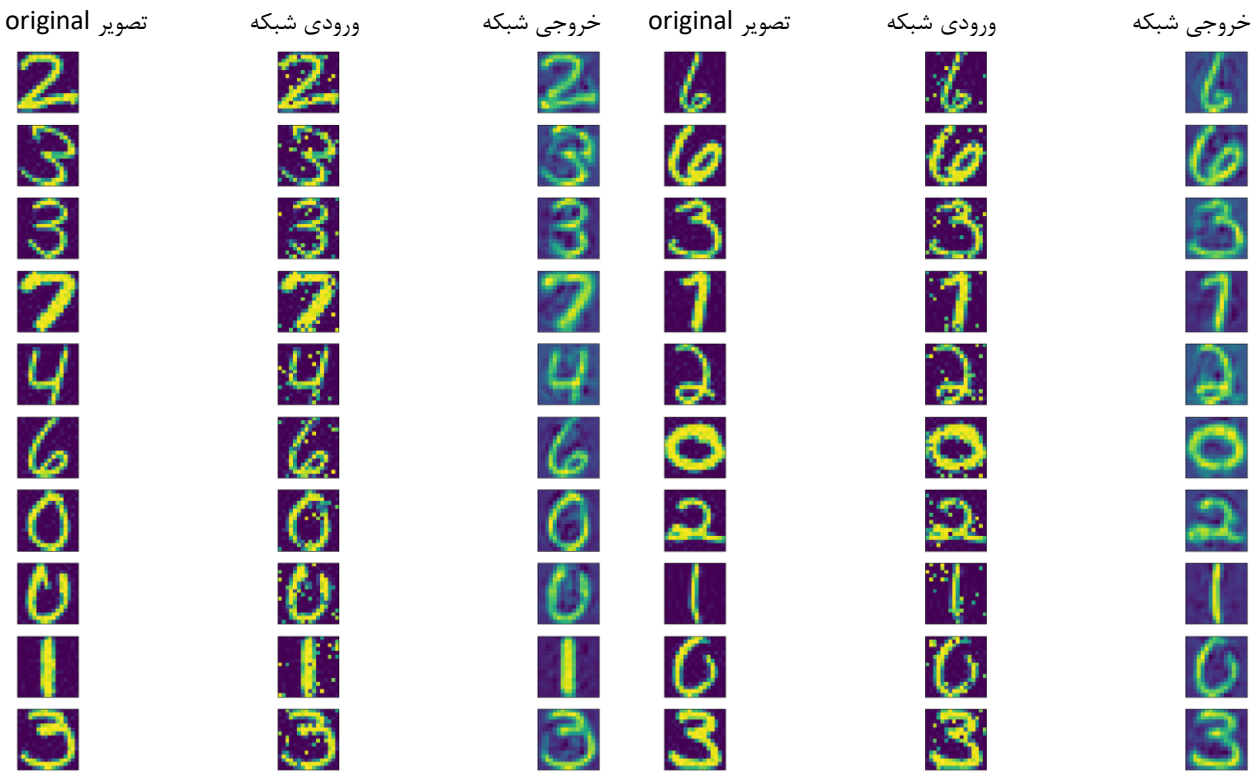
آموزشی

```
Accuracy =  0.536183551702551
```

آزمایشی

همانطور که مشاهده میکنیم با همین معیار عددی مشخص است که تفاوت زیادی بین نتایج آموزشی و آزمایشی ایجاد شده است و گویای این است که بیش برازش رخ داده است. بنابراین در مراحل بعد آزمایش ها را با این ساختار انجام داده ایم : شبکه ی سه لایه در هر لایه ۱۰۰ نورون و با ۱۰۰۰ چرخه و همچنین نسبت داده های آموزشی به آزمایشی ۹۰ به ۱۰. در این حالت اگر چه ممکن است دقت در داده های آموزشی پایین تر بیاید اما مجموعا دقت در داده های تست بالا تر میرود و دقت داد های تست و آموزش به هم نزدیک تر میشود که نشان دهنده ی اعتبار بیشتر است.

حالت اول : نویز ۱۰ درصد



داده های آزمایشی

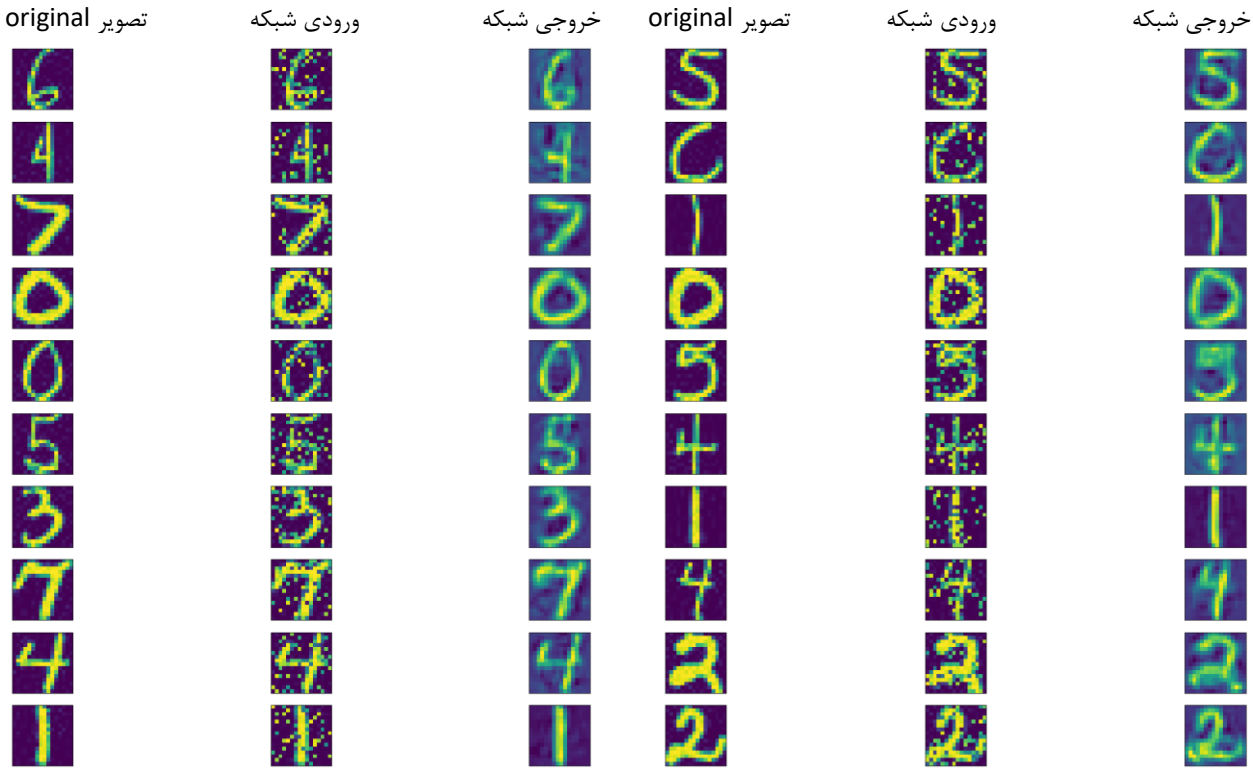
Accuracy = 0.7709183260261527

داده های آموزشی

Accuracy = 0.8176414121483886

مشاهده میکنیم که اگر نویز کم باشد شبکه به خوبی میتواند با فیت کردن تابعی که تصاویر نویزی را به تصاویر اصلی می نگارد رفع نویز را یاد بگیرد. همچنین در این حالت از مقایسه متوجه میشویم آزمایش روی داده های تست و آموزش تفاوت چندانی نمیکند که اختلاف کم بین accuracy ها هم گویای همین موضوع است و هر دو نشان میدهند شبکه واقعا یادگیری رفع نویز را فراگرفته و قابلیت تعمیم دارد و نتایج معتبر اند.

حالت دوم : نویز ۲۰ درصد

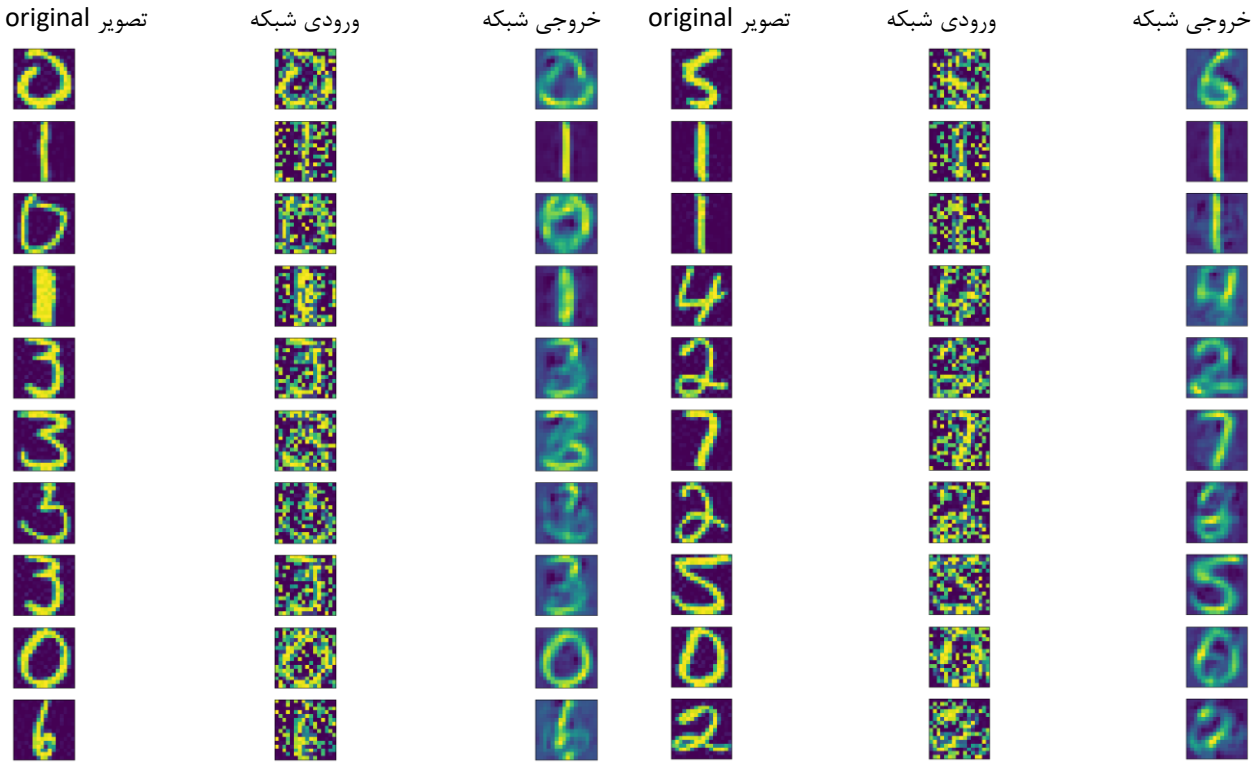


Accuracy = 0.6966934062028363

Accuracy = 0.7849867515308818

مشاهده میکنیم که اگر نویز بیشتر شود شبکه به نسبت حالت قبل نادقیق تر میشود و نویز کمتری را میتواند رفع کند اما همچنان یادگیری به خوبی انجام شده و از مقایسه متوجه میشویم آزمایش روی داده های تست و آموزش تفاوت چندانی نمیکند که اختلاف کم بین accuracy ها هم گویای همین موضوع است و هر دو نشان میدهند شبکه واقعا یادگیری رفع نویز را فراگرفته و قابلیت تعمیم دارد و نتایج معتبر اند.

حالت سوم : نویز ۵۰ درصد



مشاهده میکنیم که اگر نویز خیلی بیشتر شود (در این حالت تقریبا نصف پیکسل ها آسیب جدی دیده اند، یعنی از رنگ روشن به رنگ تیره تغییر کرده اند که این نویز بسیار زیادی است و ملاحظه میکنیم که با چشم هم خودمان نمیتوانیم بعضا درست تشخیص دهیم) شبکه به نسبت حالت های قبل نادقیق تر میشود و نویز کمتری را میتواند رفع کند و طبیعتا خروجی آن محو شده و نادقیق بنظر می رسد. بعضا حتی ملاحظه میکنیم اشتباه هم رخ داده! برای مثال در نمونه ی اول اعداد آموزشی یک عدد ۵ دست نویس داشتیم که پس از افزودن نویز به آن و رفع نویز با شبکه خروجی عدد ۶ تشخیص داده شده است!! بنابراین توانایی یادگیری شبکه در نویز بسیار بالا بسیار کاهش میابد. همچنین تفاوت بسیار زیاد معیار عددی دقت در داده های آموزشی و آزمایشی نشان مجددا نشان میدهد که در این حالت شبکه خوب نتوانسته است رفع نویز را یاد بگیرد و تابعی که توسط شبکه فیت شده خیلی وابسته به دیتای آموزشی است و قابلیت تعمیم ندارد. این مشکل در نویزکم وجود نداشت

اما در نویز بالا چون تصاویر عملا عنصر شباهت را از دست میدهند تقریبا مانند این است که انگار تعداد داده های مشابه کم شده باشد! بنابراین شبکه ی که در دو حالت قبل خوب پاسخ میداد در این حالت تقریبا میتوان گفت قادر به یادگیری ارتباط بین تصاویر اصلی ونویز ودر نتیجه رفع نویز نیست.