به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین اول

پرسش‌های ۱ و ۲	نام دانشجو	محسن ایزدی
	شماره دانشجویی	۸۱۰۱۰۱۳۱۷
پرسش‌های ۳ و ۴	نام دانشجو	سید محمد مهدی رضوی
	شماره دانشجویی	۸۱۰۱۰۲۱۵۵
	مهلت ارسال پاسخ	**۱۴۰۳.۰۱.۲۰**

فهرست

شکل‌ها

جدول‌ها

قوانین

قبل از پاسخ دادن به پرسش‌ها، موارد زیر را با دقت مطالعه نمایید:

- از پاسخ‌های خود یک گزارش در قالبی که در صفحه‌ی درس در سامانه‌ی Elearn با نام ***REPORTS_TEMPLATE.docx*** قرار داده شده تهیه نمایید.
- پیشنهاد می‌شود تمرین‌ها را در قالب گروه‌های دو نفره انجام دهید. (بیش از دو نفر مجاز نیست و تحویل تک نفره نیز نمره‌ی اضافی ندارد) توجه نمایید الزامی در یکسان ماندن اعضای گروه تا انتهای ترم وجود ندارد. (یعنی، می‌توانید تمرین اول را با شخص A و تمرین دوم را با شخص B و ... انجام دهید)
- **کیفیت گزارش شما در فرآیند تصحیح از اهمیت ویژه‌ای برخوردار است**؛ بنابراین، لطفا تمامی نکات و فرض‌هایی را که در پیاده‌سازی‌ها و محاسبات خود در نظر می‌گیرید در گزارش ذکر کنید.
- در گزارش خود مطابق با آنچه در قالب نمونه قرار داده شده، برای شکل‌ها زیرنویس و برای جدول‌ها بالانویس در نظر بگیرید.
- الزامی به ارائه توضیح جزئیات کد در گزارش نیست، اما باید نتایج بدست آمده از آن را گزارش و تحلیل کنید.
- **تحلیل نتایج الزامی می‌باشد، حتی اگر در صورت پرسش اشاره‌ای به آن نشده باشد.**
- **دستیاران آموزشی ملزم به اجرا کردن کدهای شما نیستند**؛ بنابراین، هرگونه نتیجه و یا تحلیلی که در صورت پرسش از شما خواسته شده را به طور واضح و کامل در گزارش بیاورید. در صورت عدم رعایت این مورد، بدیهی است که از نمره تمرین کسر می‌شود.
- **کدها حتما باید در قالب نوت‌بوک با پسوند .ipynb تهیه شوند، در پایان کار، تمامی کد اجرا شود و خروجی هر سلول حتما در این فایل ارسالی شما ذخیره شده باشد.** بنابراین برای مثال اگر خروجی سلولی یک نمودار است که در گزارش آورده‌اید، این نمودار باید هم در گزارش هم در نوت‌بوک کدها وجود داشته باشد.
- **در صورت مشاهده‌ی تقلب امتیاز تمامی افراد شرکت‌کننده در آن، 100- لحاظ می‌شود.**
- تنها زبان برنامه نویسی مجاز **Python** است.
- **استفاده از کدهای آماده برای تمرین‌ها به هیچ وجه مجاز نیست. در صورتی که دو گروه از یک منبع مشترک استفاده کنند و کدهای مشابه تحویل دهند، تقلب محسوب می‌شود.**

- نحوه محاسبه تاخیر به این شکل است: پس از پایان رسیدن مهلت ارسال گزارش، حداکثر تا یک هفته امکان ارسال با تاخیر وجود دارد، پس از این یک هفته نمره آن تکلیف برای شما صفر خواهد شد.
  - سه روز اول: بدون جریمه
  - روز چهارم: ۵ درصد
  - روز پنجم: ۱۰ درصد
  - روز ششم: ۱۵ درصد
  - روز هفتم: ۲۰ درصد
- حداکثر نمره‌ای که برای هر سوال می‌توان اخد کرد ۱۰۰ بوده و اگر مجموع بارم یک **سوال** بیشتر از ۱۰۰ باشد، در صورت اخد نمره بیشتر از ۱۰۰، اعمال نخواهد شد.
  - برای مثال: اگر نمره اخذ شده از سوال ۱ برابر ۱۰۵ و نمره سوال ۲ برابر ۹۵ باشد، نمره نهایی تمرین ۹۷.۵ خواهد بود و نه ۱۰۰.
- لطفا گزارش، کدها و سایر ضمایم را به در یک پوشه با نام زیر قرار داده و آن را فشرده سازید، سپس در سامانه‌ی Elearn بارگذاری نمایید:

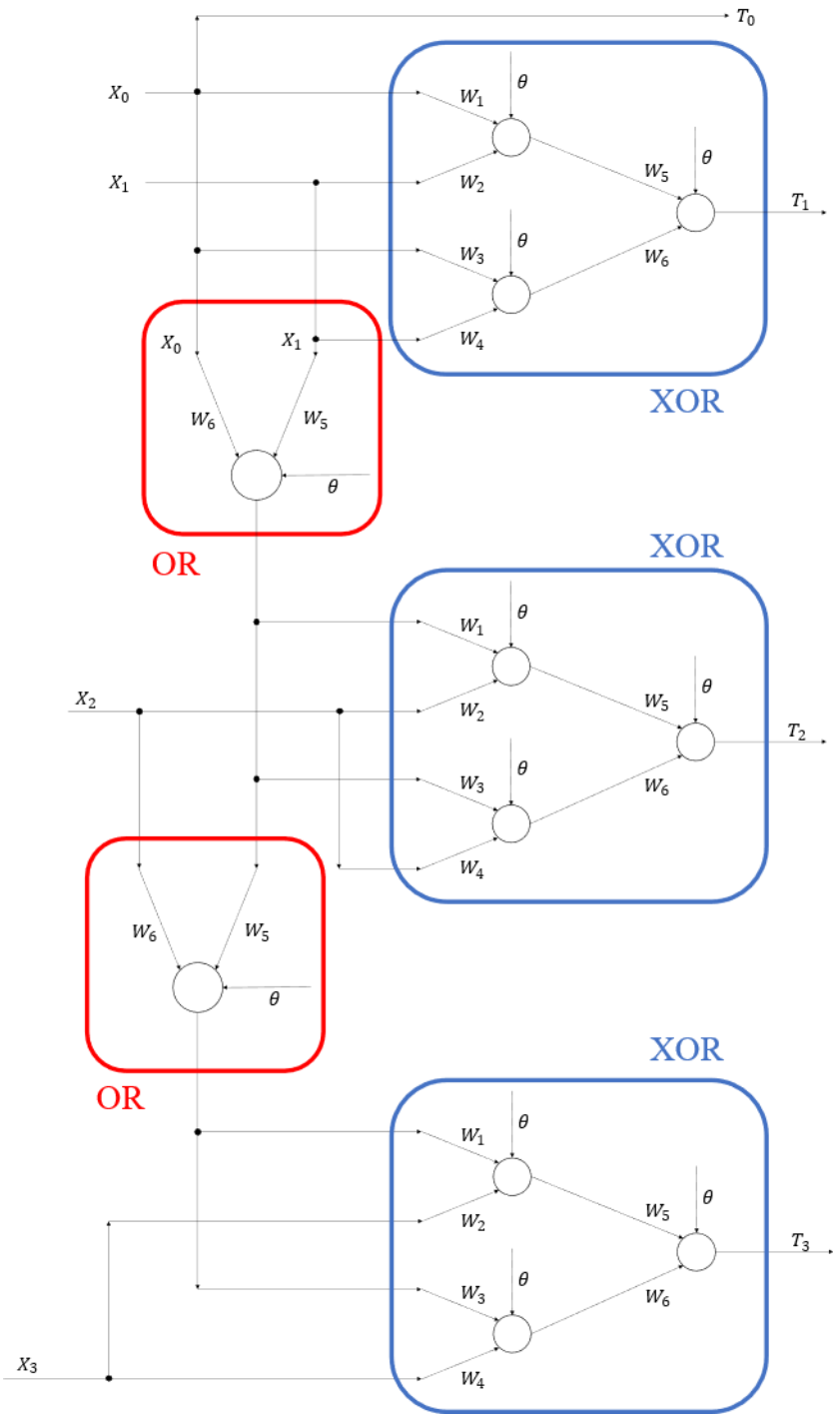
HW[Number] _[Lastname]_[StudentNumber]_[Lastname]_[StudentNumber].zip

(مثال: HW1_Ahmadi_810199101_Bagheri_810199102.zip)

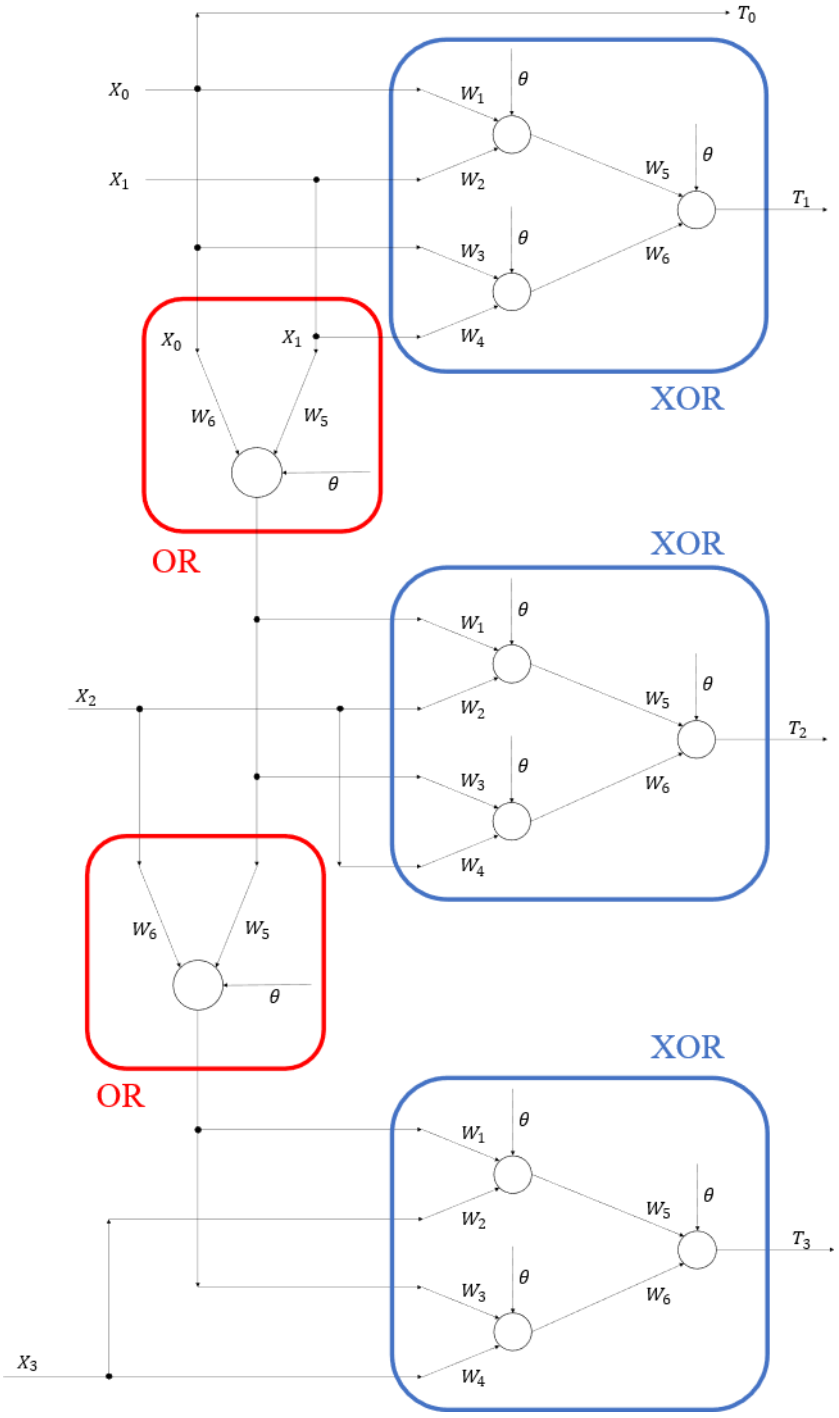
- برای گروه‌های دو نفره، بارگذاری تمرین از جانب یکی از اعضا کافی است ولی پیشنهاد می‌شود هر دو نفر بارگذاری نمایند.

پرسش ۱. شبکه محاسبه مکمل ۲

۱-۱. ساختار شبکه



شکل ۱. ساختار شبکه با نورون‌های **Mcculloch Pitts**



شکل ۱ دیده می‌شود در این شبکه از ۱۱ نورون Mcculloch Pitts استفاده و ساختار شبکه مطابق طراحی سطح گیت انجام شده است. (در شکل معادل عصبی هر گیت با رنگ‌های آبی و قرمز به ترتیب برای گیت XOR و OR مشخص شده است.)

جدول ۱. وزن‌ها در شبکه

	وزن
W1	+۱
W2	-۱
W3	-۱
W4	+۱
W5	+۱
W6	+۱

مقدار آستانه برای تمام نورون‌ها ۱+ و تابع فعال ساز مطابق زیر در نظر گرفته شده است:

$$f(x) = \begin{cases} 1 & if\ x \geq \theta \\ 0 & o.w \end{cases}$$

۲-۱. پیاده‌سازی در پایتون

برای پیاده‌سازی از کتابخانه خارجی استفاده نشده است. یک کلاس به نام Neuron ایجاد شده که اطلاعات وزن‌ها و آستانه را در constructor خود می‌گیرد. هر نورون با استفاده از تابع Connect می‌تواند به نورون‌های دیگر وصل شود یا با تابع AsFix ورودی‌های ثابت داشته باشد.

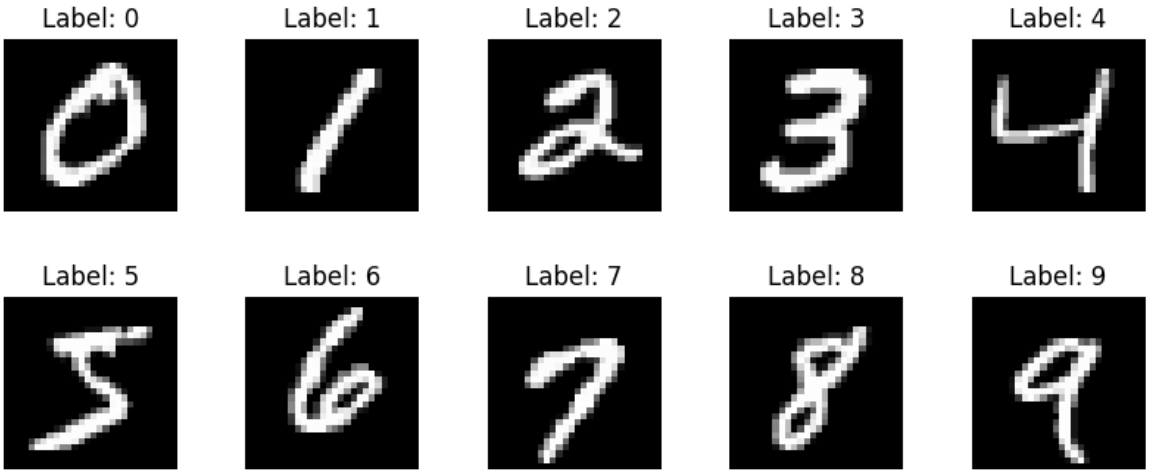
در ابتدای ساخت شبکه هر بیت در ورودی باینری به یک نورون ثابت (AsFix) تبدیل شده و سپس شبکه‌ای متشکل از نورون‌های به هم متصل با تابع Connect متصل می‌شود. برای آزمودن کد تمام ۱۶ حالت ممکن به شبکه داده شده که خروجی‌ها در شکل مشاهده می‌شود.

[55]:
```
TEST_SET = [
    "0000",
    "0001",
    "0010",
    "0011",
    "0100",
    "0101",
    "0110",
    "0111",
    "1000",
    "1001",
    "1010",
    "1011",
    "1100",
    "1101",
    "1110",
    "1111"
]

for x in TEST_SET:
    neurons = get_neurons(x)
    run(neurons)
```

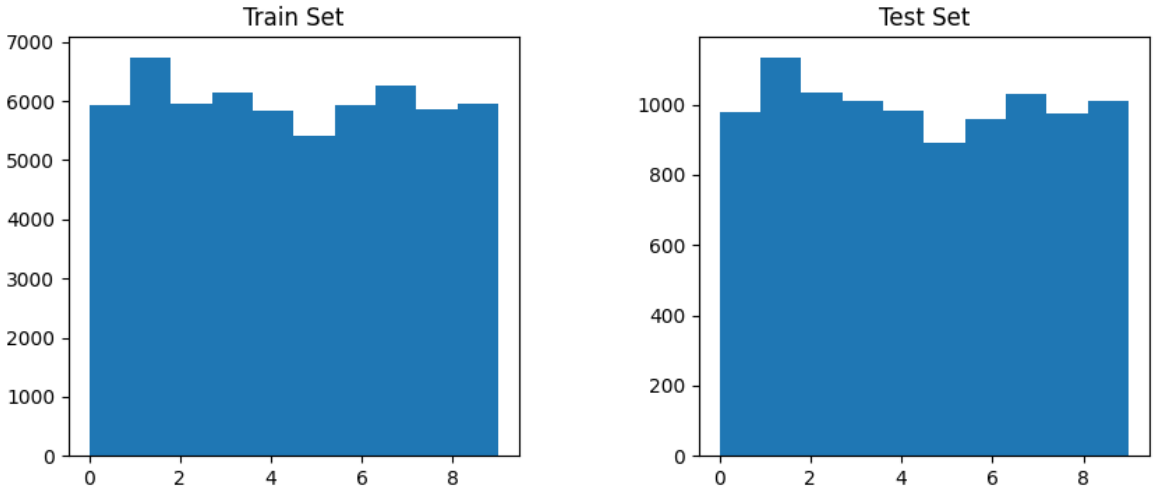
```
0000
1111
1110
1101
1100
1011
1010
1001
1000
0111
0110
0101
0100
0011
0010
0001
```

شکل ۲. اعمال تمام حالات ممکن ورودی و خروجی متناظر آن‌ها

پرسش ۲ – حملات خصمانه در شبکه‌های عصبی



شکل ۳. نمونه‌ی هر کلاس از مجموعه دادگان **MNIST**



شکل ۴. هیستوگرام داده‌های آموزش و آزمون

۲-۱. آشنایی با مجموعه دادگان

- مجموعه دادگان MNIST مجموعه‌ای از تصاویر ۲۸*۲۸ است که در مجموعه آموزش ۶۰ هزار و در مجموعه آزمون ۱۰ هزار داده وجود دارد.
- در شکل ۳ نمونه‌ای از هر کلاس دیده می‌شود.

- هیستوگرام مربوط در شکل ۴ دیده می‌شود. داده‌ها تاحدودی توزیع یکنواخت دارند. مجموعه دادگان MNIST تا حد خوبی بالانس است و نیاز به پردازش خاصی ندارد. تنها داده‌ها را قبل از آموزش و آزمون Reshape و نرمالایز و همچنین لیبل‌ها را انکود One-Hot می‌کنیم.
- برای بهبود همگرایی، جلوگیری از Vanishing Gradients، بهبود تعمیم پذیری، تضمین پایداری و...

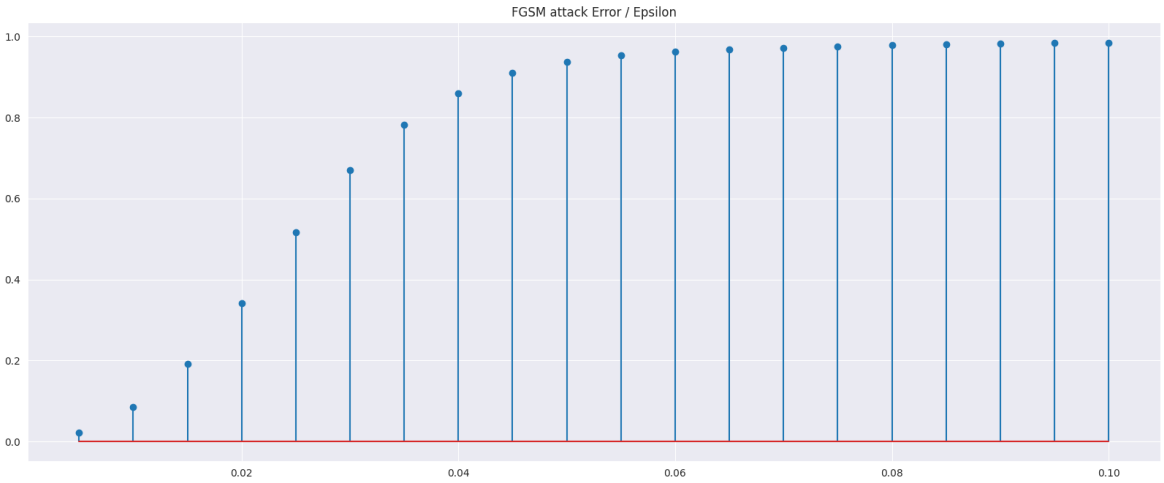
۲-۲. ایجاد و آموزش مدل

در این بخش از کتابخوانه Keras برای ایجاد و آموزش مدل استفاده می‌کنیم. مشخصات مدل پیاده شده در شکل ۵ مشاهده می‌شود.

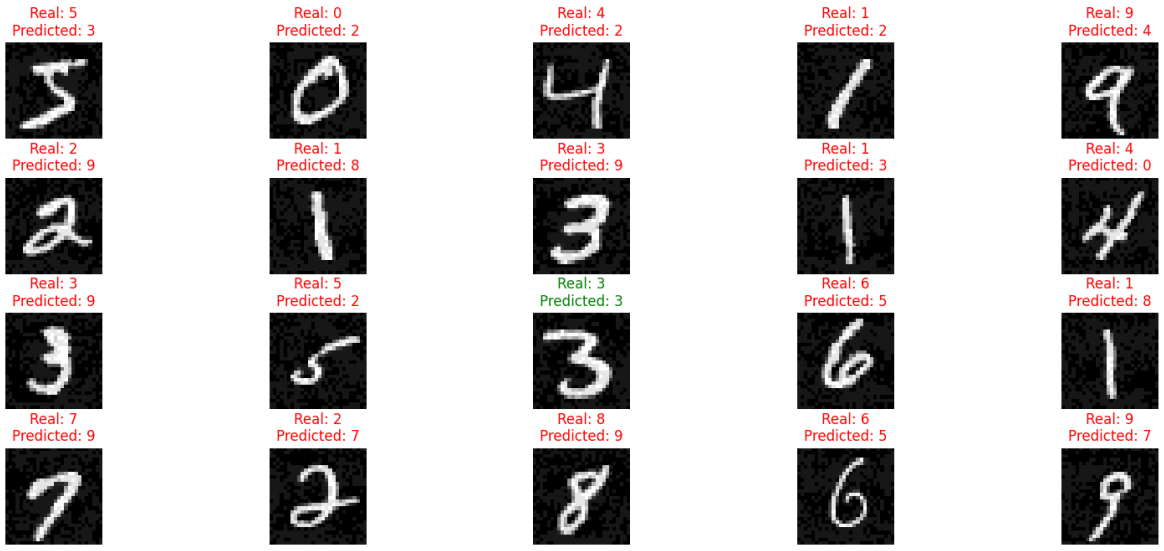
- همانطور که در بخش قبل گفته شد، داده‌ها را قبل از آموزش و آزمون Reshape می‌کنیم. به این دلیل هر داده ۲۸*۲۸ تبدیل به یک بردار ۱*۷۸۴ می‌شود.
- در مسائل کلاس‌بندی با بیش از دو کلاس معمولا از تابع Softmax استفاده می‌کنیم. این تابع خروجی‌های مدل (logits) را به یک توزیع احتمالاتی در کلاس‌های خروجی تبدیل می‌کند و در نتیجه به هر کلاس یک احتمال نسبت می‌دهد، خروجی را نرمالیزه می‌کند و همچنین تابعی مشتق پذیر است.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               401920

 dense_1 (Dense)             (None, 128)               65664

 dense_2 (Dense)             (None, 32)                4128

 dense_3 (Dense)             (None, 10)                330

=================================================================
Total params: 472042 (1.80 MB)
Trainable params: 472042 (1.80 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
Learning rate before first fit: 6e-05
```

شکل ۵. مشخصات مدل پیاده شده



شکل ۶. نمودار خطا بر اساس پارامتر **epsilon** در حمله **FGSM**



شکل ۷. نمونه داده‌ها و لیبل واقعی و خروجی مدل در حمله **FGSM**

۲-۳. پیاده‌سازی حمله FGSM

در پیاده‌سازی این حمله از کتابخانه tensorflow کمک گرفته‌ایم.

- کد پیاده‌سازی در فایل نوت‌بوک موجود است.
- نمودار خطای خروجی بر اساس تغییرات پارامتر epsilon با گام‌های ۰.۰۰۵ در شکل ۶ مشاهده می‌شود. واضح است که با افزایش مقدار epsilon خطا افزایش می‌یابد و در نهایت تقریبا به ۱۰۰ درصد نمونه‌ها خواهد رسید. بدیهی است هرچه مقدار بیشتر شود اثر آن به طور ظاهری توسط چشم نیز در نمونه‌ها دیده می‌شود.
- نمونه‌ای از داده‌ها با نویز و لیبل اصلی و پیشبینی مدل از آنها در شکل ۷ رسم شده است.

جدول ۲. نتایج حمله **PGD**

	Epsilon	Alpha	Iteration	Error (%)
آزمایش ۱	۰.۰۱	۰.۰۰۱	۱۰	۸۶.۹
آزمایش ۲	۰.۰۱	۰.۰۰۱	۴۰	۹۰.۱۰
آزمایش ۳	۰.۱	۰.۰۰۱	۱۰	۹۰.۰۵
آزمایش ۴	۰.۱	۰.۰۰۵	۱۰	۹۸.۴۵۵
آزمایش ۵	۰.۱	۰.۰۰۱	۱۵	۹۰.۶۵۵
آزمایش ۶	۰.۱	۰.۰۰۱	۲۰	۹۲.۰۴
آزمایش ۷	۰.۱	۰.۰۰۱	۴۰	۹۷.۸۳
آزمایش ۸	۰.۱	۰.۰۰۵	۱۰	۹۸.۴۵۵
آزمایش ۹	۰.۱	۰.۰۰۱۵	۱۵	۹۲.۹۱

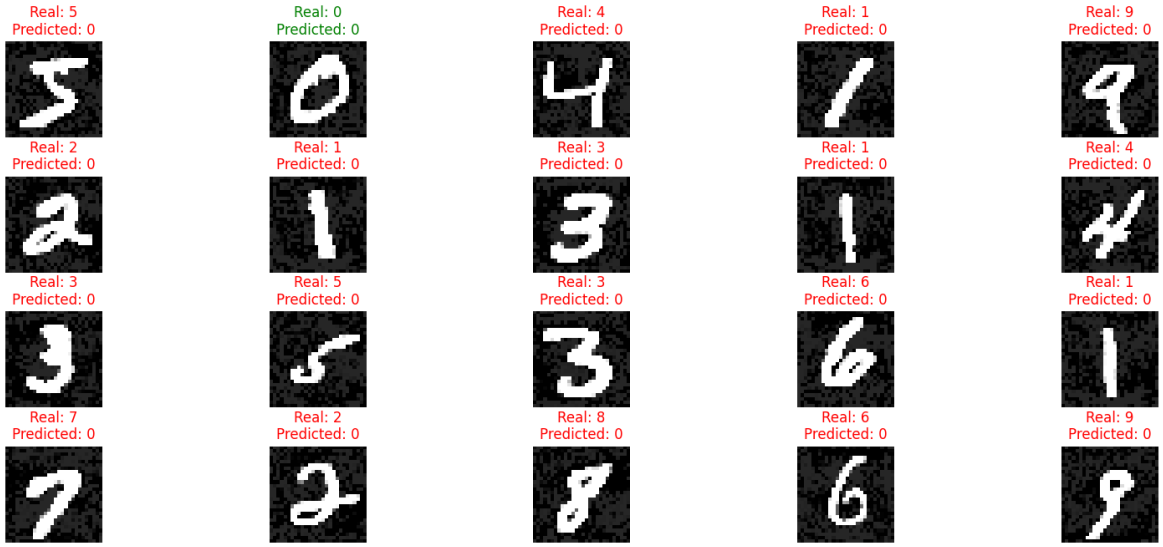
۲-۴. پیاده‌سازی حمله PGD

برای این حمله نیز از tensorflow بهره می‌گیریم. این حمله از حمله قبل سنگین‌تر است در نتیجه برای تحلیل پارامترها را در چند آزمایش تغییر داده و خطا را حساب می‌کنیم. نتیجه آزمایش‌ها در جدول ۱ آورده شده است. نتایج به شرح زیر است:

- با تغییر پارامتر epsilon و ثابت نگه داشتن بقیه پارامترها در این حمله در تصاویر از نظر ظاهری تغییر کمی دیده می‌شود اما همانطور که در آزمایش ۱ و ۳ مشاهده می‌شود خطا و اثر بخشی حمله را بیشتر می‌کند.
- تغییر پارامتر alpha با ثابت نگه داشتن پارامترهای دیگر، در آزمایش ۳ و ۴، علاوه بر اینکه خطا را بیشتر می‌کند در ظاهر نیز تصاویر را خراب می‌کند.
- پارامتر Iteration مشابه alpha با زیاد شدن خطا را زیاد و ظاهر تصاویر را خراب‌تر می‌کند. افزایش این پارامتر به طور قابل توجهی زمان حمله را افزایش می‌دهد.

پاسخ به سوالات:

- پیاده‌سازی در نوت‌بوک سوال دوم موجود است.



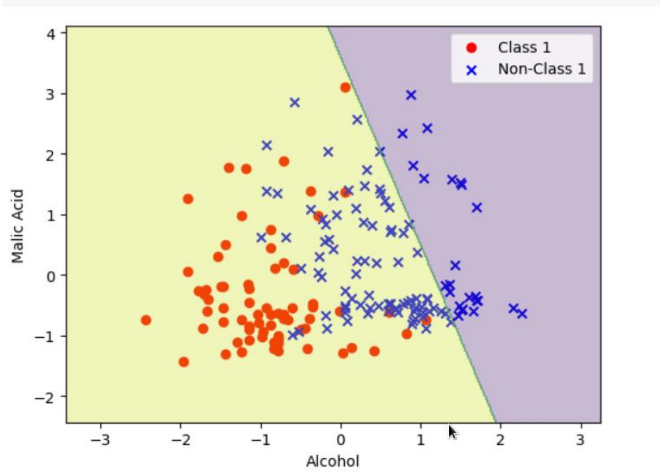
شکل ۸. نمونه‌های آزمایش ۵ و حمله **PGD**

- این روش بر اساس گرادیان تابع تلفات نسبت به ورودی و انجام مکرر (Iterations) این محاسبات روی داده ورودی انجام می‌شود و همچنین سعی می‌کند خطا را در بازه خاصی نگه دارد. در روش FGSM که تکرار نداریم و در یک مرحله انجام می‌شود، حمله بر اساس علامت تابع گرادیان تلفات روی داده ورودی خواهد بود. به همین دلیل چون در یک مرحله انجام می‌شود از نظر محاسباتی سبک‌تر است اما حمله PGD حمله بالقوه قوی‌تری است و شناسایی و مقابله با آن می‌تواند برای شبکه عصبی سخت‌تر باشد.
- همانطور که گفته شد حمله بالقوه قوی‌تری است، پارامترهای بیشتر و درجه آزادی بیشتری به حمله‌کننده می‌دهد. و شناسایی و مقابله با آن برای شبکه عصبی می‌تواند دشوار باشد. در این آزمایش مشاهده شد که با این حمله می‌توان با حفظ ظاهر نمونه‌ها به خطای بیشتری رسید.
- نمونه از هرکلاس، لیبل اصلی و لیبل پیشبینی شده توسط مدل در شکل ۸ ارائه شده است.

پرسش ۳ – Adaline و Madaline

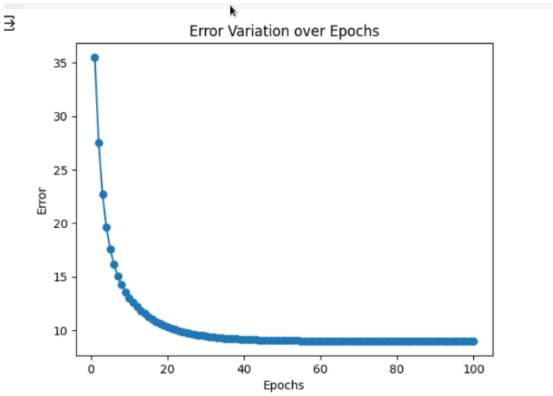
۳-۱. Adaline

تمرین ۳ -۱ - الف



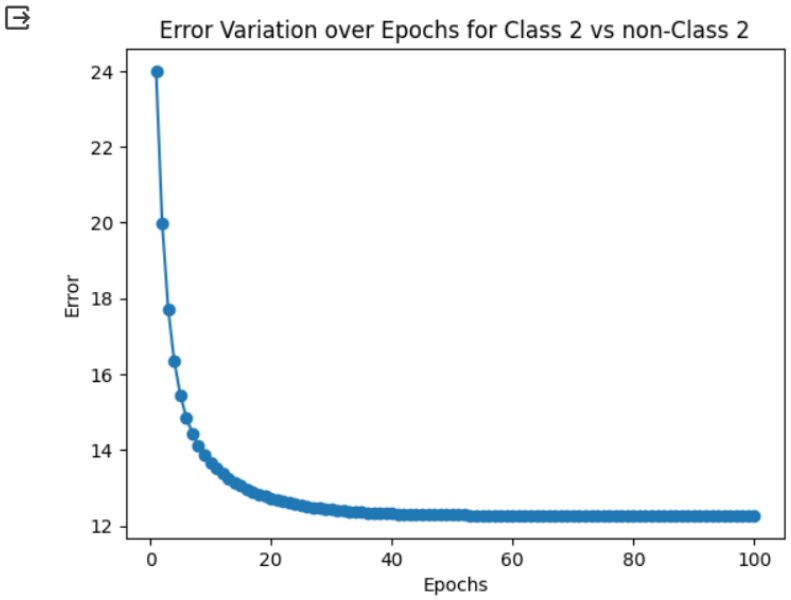
شکل ۹. مودار توزیع پراکندگی داده‌ها با برچسب شراب کلاس ۱

شبکه در حین آموزش دیدن در ابتدا میزان مجموع خطاها زیادتر است به نسبت شبکه‌ای که جلوتر برای کلاس ۲ آموزش دیده‌شده‌است. اما به هر حال با آموزش دیدن الگوی کاهش هر دوی این شبکه‌ها به یک شکل می‌باشد.

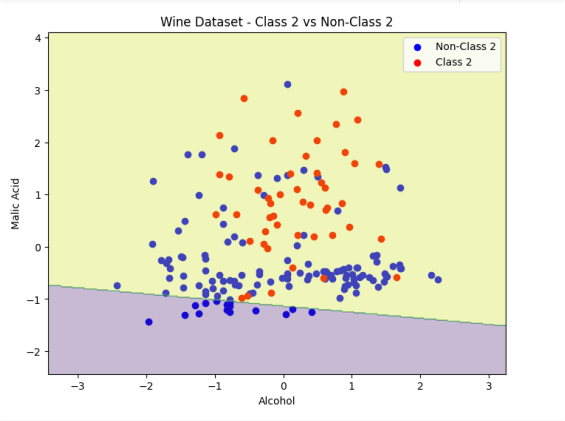


شکل ۱۰. نمودار خطا بر حسب ایپاک‌های آموزش دیده‌شده برای شراب کلاس ۱

تمرین ۳ – ۱ – ب



شکل ۱۱. نمودار خطا بر حسب تعداد ایپاک‌های آموزش شبکه



شکل ۱۲. توزیع پراکندگی داده‌ها با برچسب شراب کلاس ۲

در این دو حالت میزان تناسب ابعاد و تعداد ابعاد داده‌ها و همچنین فراپارامترها یکسان هستند.

عواملی که می‌توان از آن‌ها نام‌برد که باعث جداسازی بهتر شبکه آدالاین می‌شود در دو شکل بالا را می‌توان از حیث **توزیع‌داده‌ها ، سطح نویز و میزان جداشونده خطی بودن توزیع داده‌ها** بررسی کرد.

هرچه میزان پراکندگی داده‌ها دریک کلاس کمتر باشد و تجمع داده‌ها در یک موقعیت هندسی بیشتر باشد ، شبکه آدالاین بهتر می‌تواند عملیات طبقه‌بندی را انجام دهد.

همچنین هرچه جنس پراکندگی داده‌ها به این صورت باشد که اصطلاحا **جداپذیرخطی** باشند ، شبکه آدالاین دقت بیشتری از خود نشان‌خواهد داد.

تمرین ۳-۲

Madaline

MRI (MADALINE RULE I)

قانون ۱ مادالاین : بیشترین تعداد رای ، این واحد تمام خروجی‌ها را می‌گیرد و اگر تعداد برچسب‌های مثبت بیشتر بود ، برچسب مثبت را به خروجی می‌دهد . در غیر این صورت برچسب منفی را به خروجی می‌دهد.

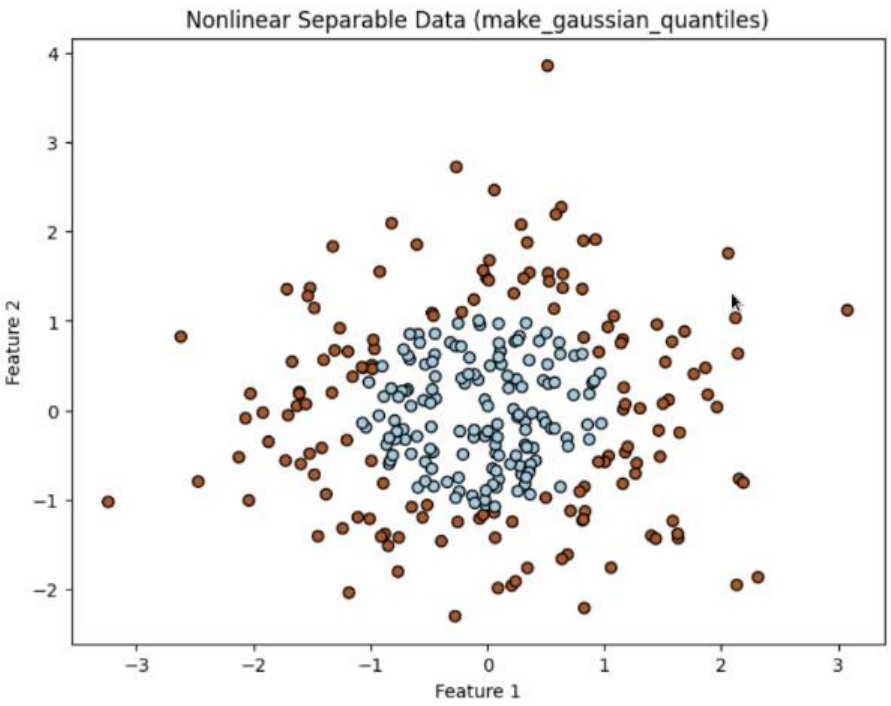
MRII (MADALINE RULE II)

قانون ۲ مادالاین : این قاعده مرسوم به minimal disturbance نیز می‌باشد . این رویکرد با تکرار فرآیندهای زیر تا زمانی که خطای شبکه تا حد قابل قبولی نزدیک به صفر و یا خود صفر شود ، ادامه می‌یابد :

۱. لایه پنهان با کمترین اطمینان در پیش‌بینی را پیدا می‌کند.
۲. به طور آزمایشی علامت واحد را برمی‌گرداند . ()
۳. پذیرفتن و یا عدم پذیرفتن بر اساس کاهش و یا افزایش خطای شبکه.

آزمایش‌های مدل مادالاین بر شبکه‌هایی با ۳ ، ۵ و ۸ نرون

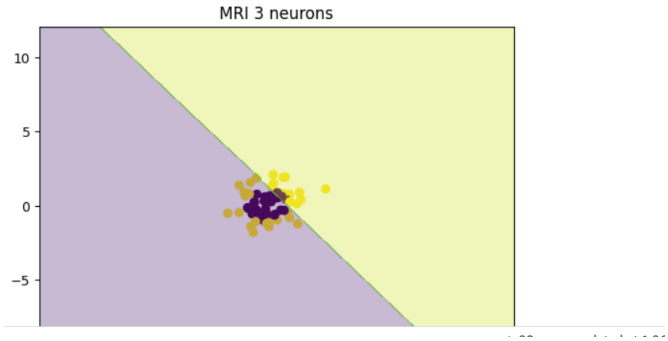
با توجه به این‌که شبکه های فوق دارای هایپرپارامترهای نرخ یادگیری و تعداد دفعات یادگیری (iterations) هایپرپارامترهای ما هستند ، با توجه به این هایپرپارامترها هر دفعه با نرخ دقت جدیدی مواجه خواهیم بود اما به صورت کلی دقت ها بین بازه ۵۵ درصد تا ۶۱ درصد نوسان خواهد کرد.



شکل ۱۳. نمودار توزیع داده‌های مصنوعی تولید شده

```
# Using MRI algorithm
y_pred_mri = madaline.predict(X_test)
accuracy_mri = np.mean(y_pred_mri == y_test) * 100
print(f"Accuracy with {n_neurons} neurons using MRI: {accuracy_mri:.2f}%")

# # Using MRII algorithm
# y_pred_mrii = madaline.predict_mrii(X_test)
# accuracy_mrii = np.mean(y_pred_mrii == y_test) * 100
# print(f"Accuracy with {n_neurons} neurons using MRII: {accuracy_mrii:.2f}%")
```

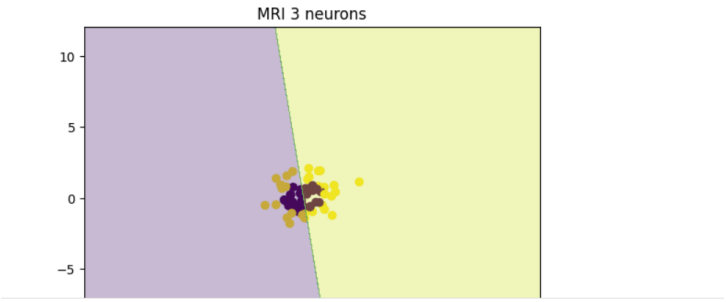
```
:curacy with 3 neurons using MRI: 61.67%
:curacy with 5 neurons using MRI: 56.67%
:curacy with 8 neurons using MRI: 58.33%
```



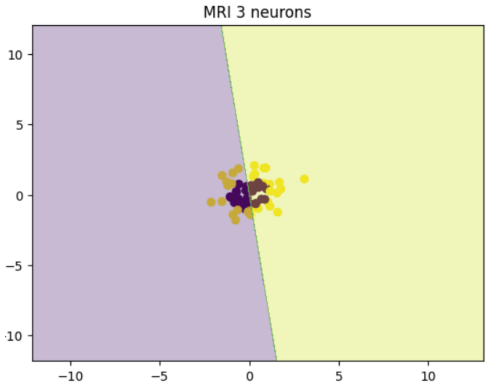
شکل ۱۴. دقت مدل های آدلاین با تعداد نرون های متفاوت

```
    # # Using MRII algorithm
    # y_pred_mrii = madaline.predict_mrii(X_test)
    # accuracy_mrii = np.mean(y_pred_mrii == y_test) * 100
    # print(f"Accuracy with {n_neurons} neurons using MRII: {accuracy_mrii:.2f}%")
```

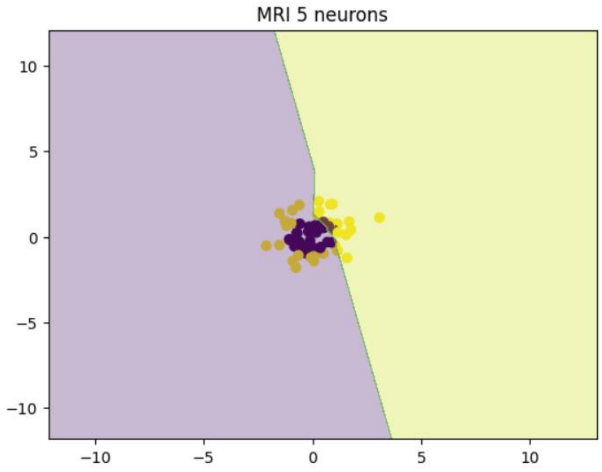
```
Accuracy with 3 neurons using MRI: 56.67%
Accuracy with 5 neurons using MRI: 66.67%
Accuracy with 8 neurons using MRI: 55.00%
Accuracy with 11 neurons using MRI: 58.33%
```



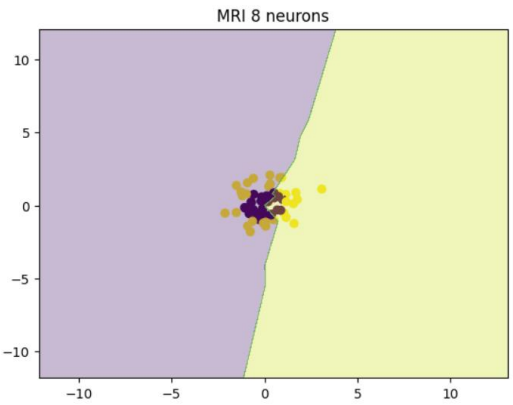
شکل ۱۵. دقت شبکه مادالاین با نرخ یادگیری ۰.۰۵ و تعداد ایپاک ۱۵۰۰



شکل ۱۶. شبکه عصبی با ۳ نرون ، شبکه ۳ مرز تصمیم‌گیری خواهد داشت.



شکل ۱۷. شبکه عصبی با ۵ نرون



شکل ۱۸. شبکه عصبی با ۸ نرون . همانطور که مشاهده می‌گردد مرزهای تصمیم‌گیری به نسبت حالت قبل بیشتر شده است.

با توجه به مشاهدات ما از شکل ۱۵ مشاهده می‌شود با افزایش تعداد نرون‌ها انتظار داریم که دقت مدل ما افزایش پیداکند اما می‌بینیم که این اتفاق برای حالت با ۸ نرون اتفاق نمی‌افتد .

به نظرمی‌رسد که وزن‌های شبکه عصبی برروی داده‌های آموزشی بیش از حد آموزش دیده‌است و شبکه به نسبت حالت‌های با نرون کمتر دچار overfit گردیده‌است.

پرسش ۴ – شبکه‌ی عصبی بهینه

۴-۱. عنوان بخش اول

تمرین ۴-۱- الف

دلایل بیش‌برازش overfitting :

۱.پیچیدگی مدل :

اگر شبکه عصبی به نسبت توزیع داده‌های آموزشی بسیار پیچیده باشد ، ممکن است مدل به جای الگوهای عمومی ، نویز را نیز مدل‌سازی کند.

۲.داده آموزشی به تعداد کافی موجود نباشد :

کافی نبودن داده‌های آموزشی باعث می‌شود تا مدل مثال‌های مختلف را نبیند و نتواند الگوهای مختلف را آموزش ببیند.

۳. کمبود تعمیم

۴. تعداد epochهای بسیار بالا برای آموزش‌دیدن مدل :

این امر باعث آموزش نویز در شبکه خواهدشد.

۵. عدم پیش‌پردازش مناسب داده‌ها :

اگر داده‌ها به خوبی نرمال‌سازی نشوند ، شبکه عصبی با چالش استخراج الگو معنادار روبرواست ، درنتیجه شبکه به آموزش نویز می‌پردازد.

تمرین ۴-۱- ب

جلوگیری از overfitting :

1.Regularization

2.DropOut

نرون‌های رندوم در زمان یادگیری به صورت موقت حذف شوند و این کار از وابستگی بیش‌ازحد به یک نرون جلوگیری می‌کند.

3.Early Stopping

هنگامی که تابع ضرر در مدت یادگیری شبکه شروع به افزایش کرد ، فرآیند آموزش را متوقف کنیم.

4.Data Augmentation

5.Cross-Validation

6.Simplifying the Model

7.Feature Selection

تمرین ۴-۱- پ

هایپرپارامتر متغیرهایی هستند که در شبکه برای تنظیم معماری و ساختار یادگیری شبکه استفاده می‌شود و معمولا به سرعت یادگیری و آموزش مدل مربوط می‌شود .

همچنین یک حالت استقلال از وزن‌های شبکه‌عصبی خواهند داشت .

هایپرپارامترهای معروف شبکه عصبی عبارتند از :

1. Learning Rate
2. Number of hidden layers
3. Number of neurons in each layer
4. Activation Functions
5. Batch Size
6. Dropout Rate

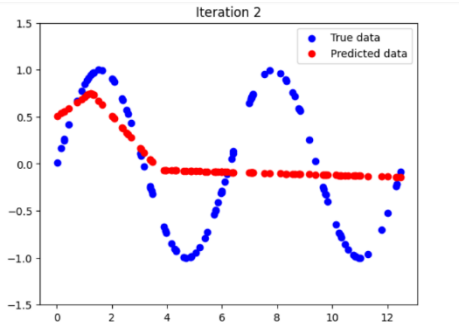
تمرین ۴-۱- ت

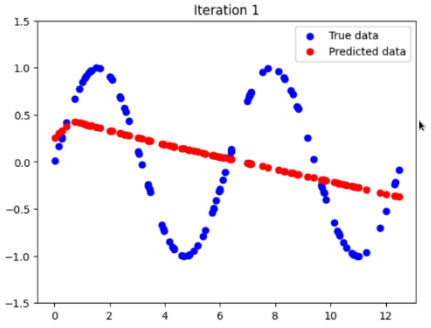
افزایش داده‌ها :

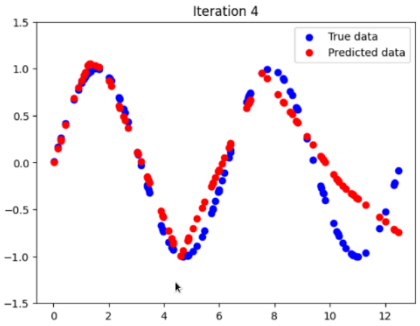
همانطور که از تصاویر زیر مشاهده خواهید کرد ، نقاط قرمز که توسط شبکه عصبی ما تولید می‌شود در حال همگرایی به تابع سینوسی خواهد شد .

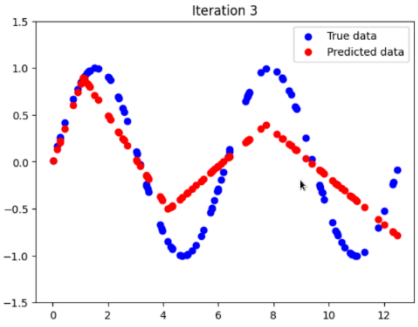
در هر گام یا iteration به سمت همگرایی بیشتر با افزایش میزان داده‌آموزشی خواهیم رفت .

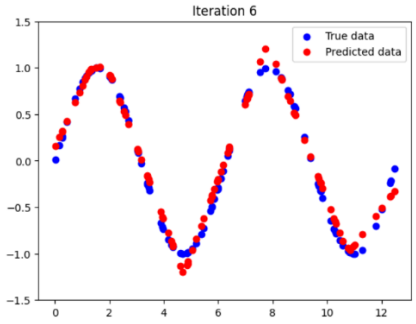
پس به صورت کلی می‌توان این نکته را در نظر گرفت که افزایش داده آموزشی در رگرسیون به افزایش دقت مدل کمک خواهد کرد.

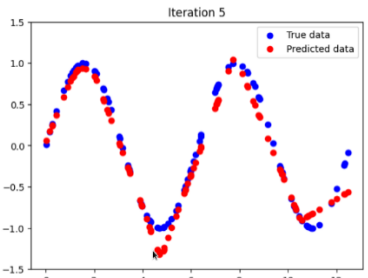

Iteration 2
True data
Predicted data

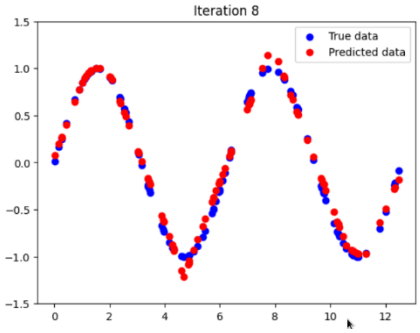

Iteration 1
True data
Predicted data

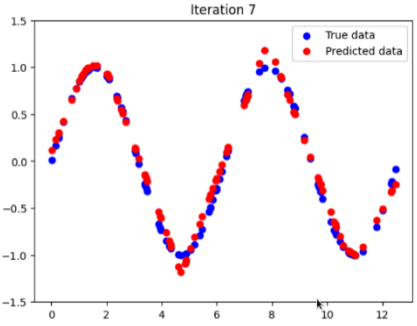

Iteration 4
True data
Predicted data

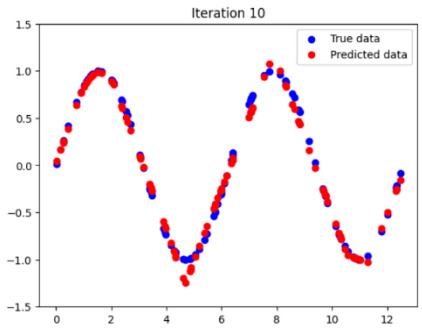

Iteration 3
True data
Predicted data

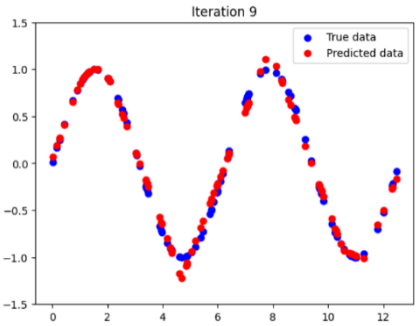

Iteration 6
True data
Predicted data


Iteration 5
True data
Predicted data


Iteration 8
True data
Predicted data


Iteration 7
True data
Predicted data


Iteration 10
True data
Predicted data

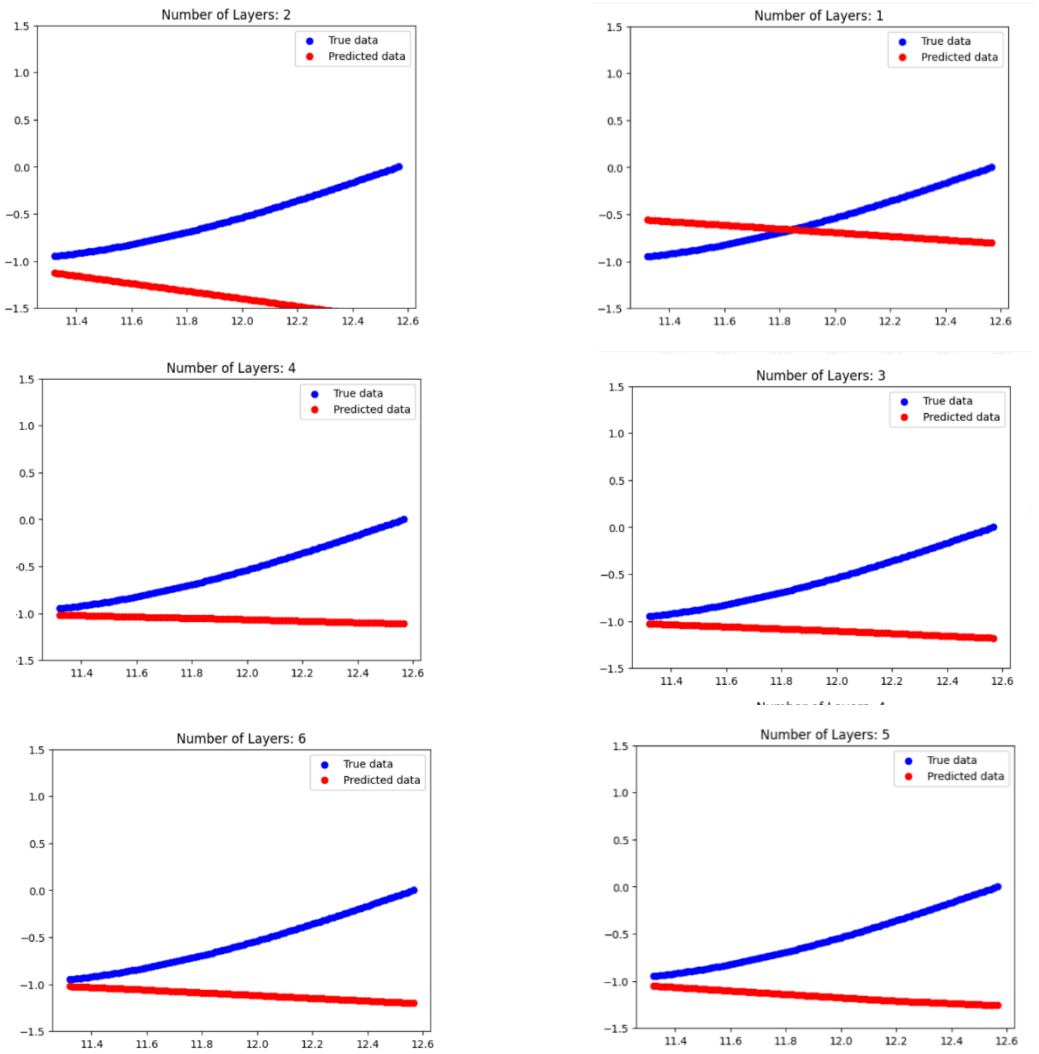

Iteration 9
True data
Predicted data

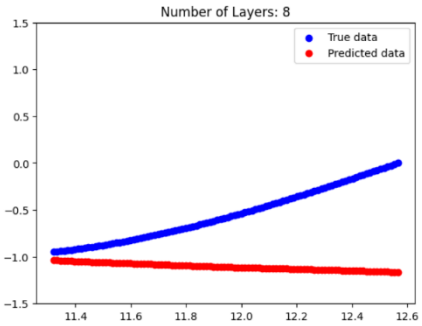
شکل ۱۹. افزایش‌داده‌های آموزشی برای رگرسیون تابع سینوسی

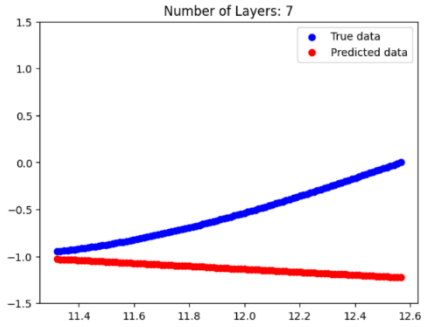
افزایش لایه‌ها :

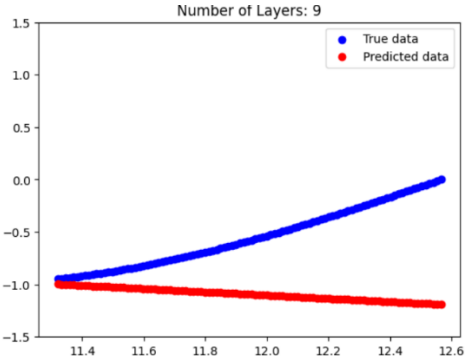
همانطور که از تصاویر نیز مشخص است ، افزایش تعداد لایه‌ها **از یک جایی به بعد** موجب افزایش پیچیدگی مدل ما خواهد شد و تاثیر مثبتی در روند رگرسیون نخواهد داشت.

درنتیجه همواره افزایش تعداد لایه‌ها و افزایش پیچیدگی مدل تاثیرمثبت در روند رگرسیون نخواهد داشت.



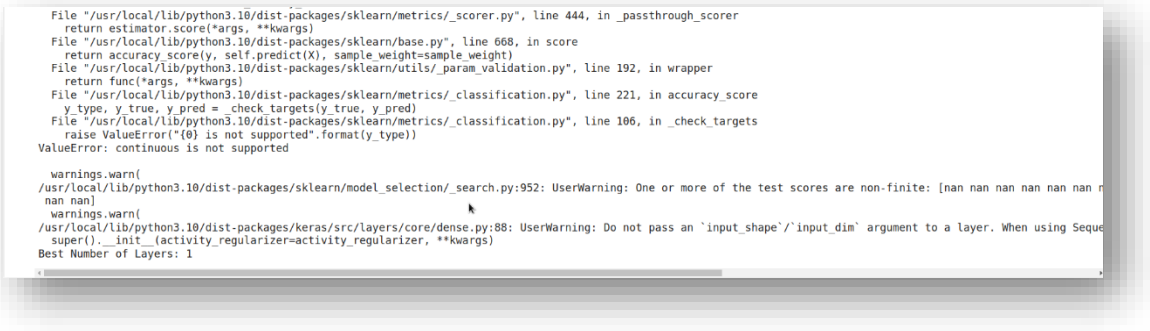

Number of Layers: 8
True data
Predicted data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
11.4
11.6
11.8
12.0
12.2
12.4
12.6


Number of Layers: 7
True data
Predicted data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
11.4
11.6
11.8
12.0
12.2
12.4
12.6


Number of Layers: 9
True data
Predicted data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
11.4
11.6
11.8
12.0
12.2
12.4
12.6

شکل ۲۰. افزایش تعداد لایه‌ها برای آموزش رگرسیون تابع سینوسی

تمرین ۴-۱- ث



شکل ۲۱. خروجی کتابخانه مدنظر **GridSearchCV**

طبق تصویر بالا ، کتابخانه مدنظر این سوال تعداد لایه مناسب برای این شبکه را ۱ پیش‌بینی کرده‌است.

شاید به این خاطر بوده‌است که پیچیدگی داده‌ها کم بوده‌است و با یک تبدیل بتوان داده‌ها را به فرم **جداپذیرخطی** تبدیل کرد.

تمرین ۴ – ۲ – طبقه‌بندی

```
(3, 0.07421066612005234, 0.9761999845504761)
(4, 0.12677186727523804, 0.9628000259399414)
(5, 0.0964893326163292, 0.9726999998092651)
(6, 0.09718208760023117, 0.9717000126838684)
(7, 0.11263754963874817, 0.9700999855995178)
(8, 0.11216064542531967, 0.9672999978065491)
(9, 0.13143523037433624, 0.9646000266075134)
(10, 0.11375528573989868, 0.9710000157356262)
(11, 0.11350701004266739, 0.9710999727249146)
(12, 0.14559513330459595, 0.9623000025749207)
(13, 0.12923473119735718, 0.9671000242233276)
(14, 0.14621324837207794, 0.9641000032424927)
(15, 0.1627344787120819, 0.9613000154495239)
(16, 0.178893581032753, 0.9550999999046326)
(17, 0.1541605144739151, 0.960099995136261)
(18, 0.17032869160175323, 0.9595999717712402)
(19, 0.16693784296512604, 0.9603999853134155)
(20, 0.24934343993663788, 0.9424999952316284)
```

شکل ۲۲. میزان تابع ضرر شبکه و همچنین دقت شبکه در ایپاک‌های مختلف

همانطور که در شکل بالا نیز مشاهده می‌گردد ، با افزایش تعداد لایه‌ها میزان تابع ضرر و میزان دقت مدل را می‌توانید مشاهده بفرمایید. به نوعی با افزایش میزان لایه‌ها دچار کمی بیش‌برازش شده‌ایم.

```
Code  + Text
Epoch 2/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0061 - accuracy: 0.9986
Epoch 3/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0053 - accuracy: 0.9987
Epoch 4/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0072 - accuracy: 0.9983
Epoch 5/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0036 - accuracy: 0.9989
313/313 [==============================] - 1s 3ms/step - loss: 0.2248 - accuracy: 0.9784
Training data size: 165000, Accuracy: 0.9783999919891357, Error rate: 0.021600008010864258
Epoch 1/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0065 - accuracy: 0.9985
Epoch 2/5
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0046 - accuracy: 0.9989
Epoch 3/5
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0064 - accuracy: 0.9985
Epoch 4/5
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0044 - accuracy: 0.9987
Epoch 5/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0041 - accuracy: 0.9990
313/313 [==============================] - 1s 4ms/step - loss: 0.2527 - accuracy: 0.9777
Training data size: 174000, Accuracy: 0.9776999950408936, Error rate: 0.022300004959106445
Epoch 1/5
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0051 - accuracy: 0.9986
Epoch 2/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0033 - accuracy: 0.9993
Epoch 3/5
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0048 - accuracy: 0.9988
Epoch 4/5
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0055 - accuracy: 0.9988
Epoch 5/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0054 - accuracy: 0.9988
313/313 [==============================] - 1s 2ms/step - loss: 0.2697 - accuracy: 0.9762
Training data size: 183000, Accuracy: 0.9761999845504761, Error rate: 0.023800015449523926
```

شکل ۲۳. خروجی مدل بر اساس حجم داده آموزش‌دیده‌شده و دقت و خطای شبکه

همانطور که از تمرین قبل به خاطر داریم ، با افزایش میزان داده آموزشی ، میزان دقت ما بالاتر خواهد رفت و در این مساله میزان دقت ما از ۹۲ درصد تا ۹۸ درصد پیشرفت خواهد کرد.

اما اگر داده آموزشی از یک حدی نیز بیشتر شود ، ما دچار بیش‌برازش یا همان overfit خواهیم شد که نمود آن در این مسال کاهش دقت ما تا ۹۷ درصد می‌باشد.

شکل ۲۴. میزان دقت شبکه کانولوشن بر روی مجموعه داده **MNIST**

طبق بررسی ما شبکه عصبی بهینه برای این مجموعه داده ، از نوع Convolutional Neural Network خواهد بود که در دفترچه کولب پیاده‌سازی شده‌است .

دقتی که این مدل به دست خواهدآورد حدود ۹۹ درصد خواهد بود.