پروژه ی هوش : شبکه ی عصبی

مدرس : دکتر عبدی

مریم جعفری -99521181

بخش اول:

در ابتدا ما یک تابع ساده (y=x+10) را به عنوان ورودی به تابع خود می دهیم و با پارامتر های ورودی بازی می کنیم تا تغییرات حاصل بر روی خروجی را تحلیل کنیم.

```
# This is our chosen function
def SimpleFunction(input_list):
    return [i+10 for i in input_list]
```

در اینجا تابعی تعریف میکنیم که در بازه ای که به عنوان ورودی به آن می دهیم تعداد مشخص شده ای نقطه و خروجی به ازای نقاط مشخص شده را برگرداند.

```
# Give us correct input and output in specified range
def correct_input_output(num_of_points,min,max):
    my_X = np.linspace(min, max, num=num_of_points).reshape(-1, 1)
    my_Y = np.array(SimpleFunction(my_X)).reshape(-1, 1)
    return my_X,my_Y
```

ما نیاز داریم که درخت خود را بر اساس ورودی و خروجی تست Train کنیم که این کار در تابع زیر انجام می شود:

```
def Train_Network(hidden_layer, num_of_iteration, num_of_points,min,max):
    x_input,y_input = correct_input_output(num_of_points=num_of_points,min=min,max=max)
    network = MLPRegressor(hidden_layer_sizes=hidden_layer,
                           max_iter=num_of_iteration,
                           random_state=0, shuffle=True).fit(x_input,y_input.ravel())
    return network
```

یک تابع نیز برای محاسبه ی میزان خطا نیاز داریم که در اینجا من از تابع زیر استفاده کردم

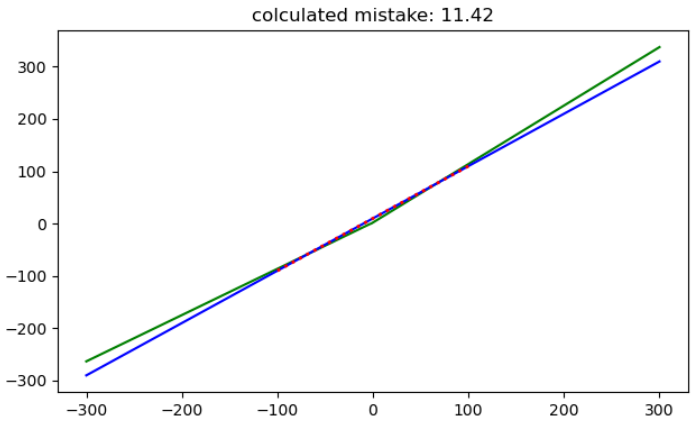
```
def colculate_mistakes(our_answer,correct_Answer):
    l = len(our_answer)
    sum = 0
    for i in range (0, l):
        sum = sum+ abs(our_answer[i]-correct_Answer[i])
    return sum/l
```

برای ساده تر شدن انجام آزمایش ها و تغییر دادن پارامتر های مختلف تابع زیر را نوشتم که در ابتدا درخت را Train می کند سپس داده ی تست را به عنوان ورودی به درخت می دهیم و خروجی میگیریم و خطایش را محاسبه می کنیم و در نهایت نمودار هر 3 را به عنوان خروجی به ما نشان می دهد.

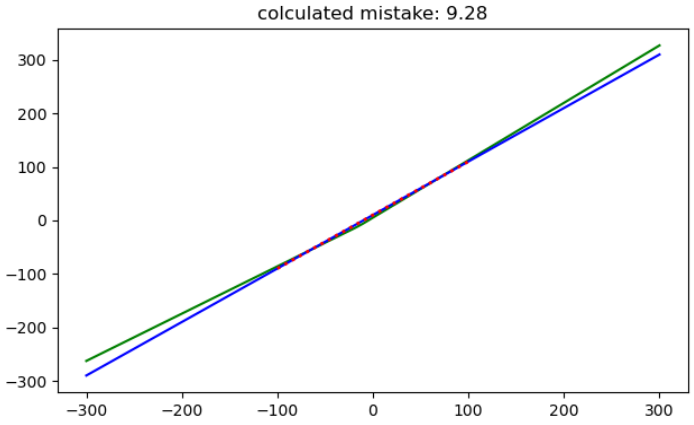
```
def Train_Test_Network(hidden_layer,num_of_iteration,train_info,test_info):
    x_train,y_train = correct_input_output(train_info[0],train_info[1],train_info[2])
    x_test ,y_test = correct_input_output(test_info[0],test_info[1],test_info[2])
    train_network  = Train_Network(hidden_layer=hidden_layer,
                                   num_of_iteration=num_of_iteration,
                                   num_of_points=train_info[0],
                                   min=train_info[1],max=train_info[2])

    y_tree_output = train_network.predict(x_test)

    fig, ax = plt.subplots()
    test_plt,  = plt.plot(x_test, y_tree_output,color = 'g', label='Test')
    expected_plt,  = plt.plot(x_test, y_test,color='b', label='Expected_result')
    train_plt, = plt.plot(x_train, y_train,color ='r', label='Train',
linestyle=':',linewidth=2)
    mistake = colculate_mistakes(y_test,y_tree_output)
    plt.title('colculated mistake: ' + str(round(mistake[0],2)))
    ax.set_xlabel('\n Train Info : number of points : '+str(train_info[0])+"    Train
Range : ("+ str(train_info[1])+ ","+str(train_info[2])+")"+
                  '\n Test  Info : number of points :
'+str(test_info[0])+"    Test  Range : ("+ str(test_info[1])+
","+str(test_info[2])+")"+
                  '\n Number of Hidden layer : '+str(hidden_layer[0])+
                  '\n Number of Iteration : '+str(num_of_iteration))

    # Adding legend, which helps us recognize the curve according to it's color
    # plt.legend()
    plt.tight_layout()
    plt.show()
```

حال این تابع را با ورودی های مختلف صدا می زنیم و نتایج را با یکدیگر مقایسه می کنیم.

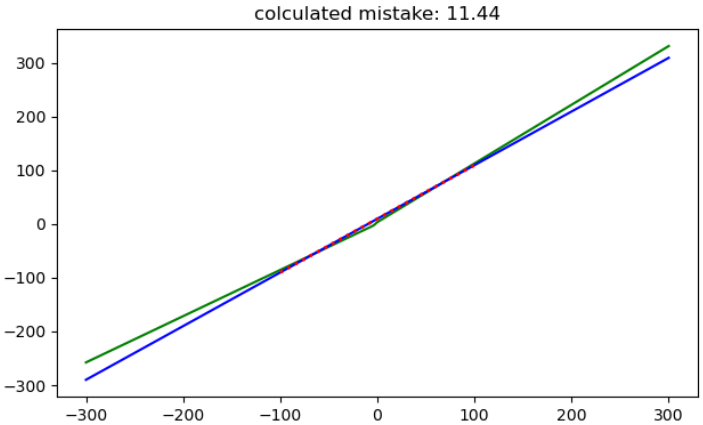
تعداد لایه ها:



Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 20
Number of Iteration : 2000

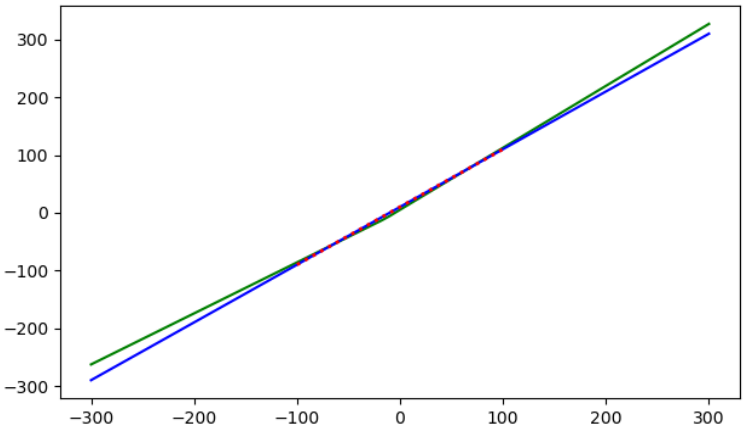


Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000



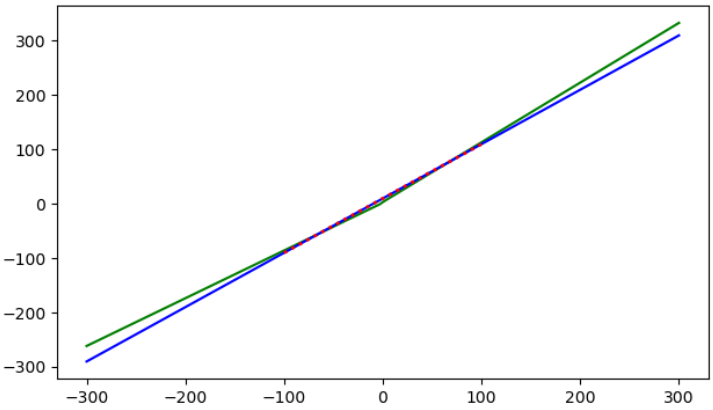
Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 5
Number of Iteration : 2000

colculated mistake: 9.28



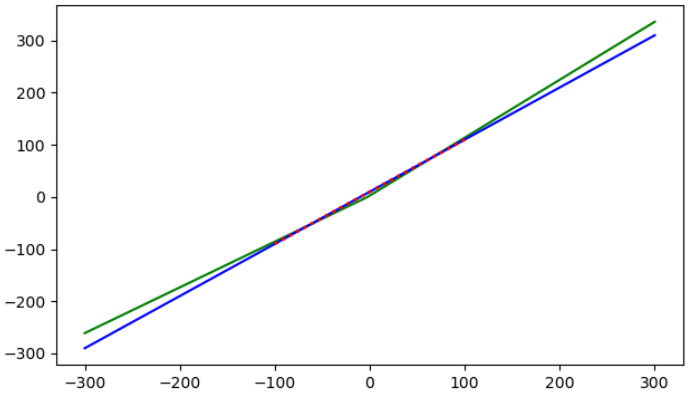
Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000

colculated mistake: 10.87



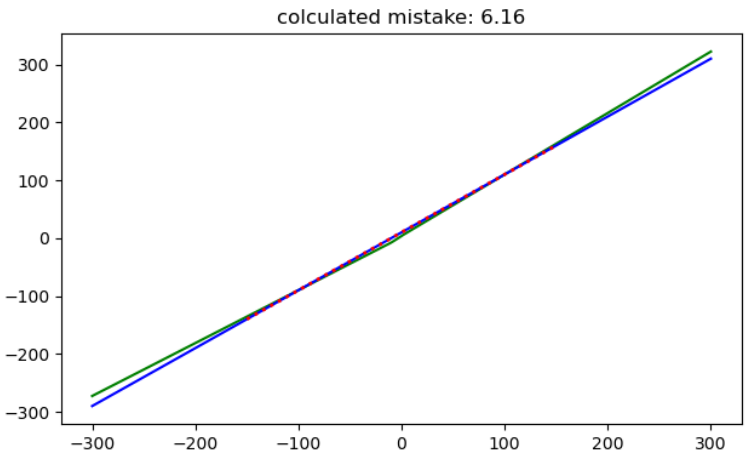
Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 9
Number of Iteration : 2000

colculated mistake: 11.55

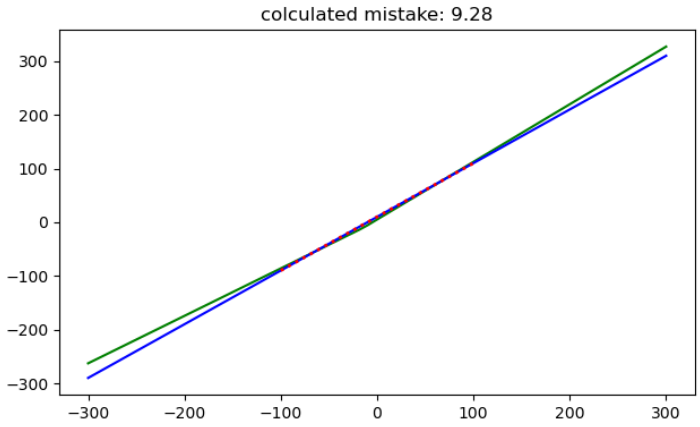


Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 11
Number of Iteration : 2000

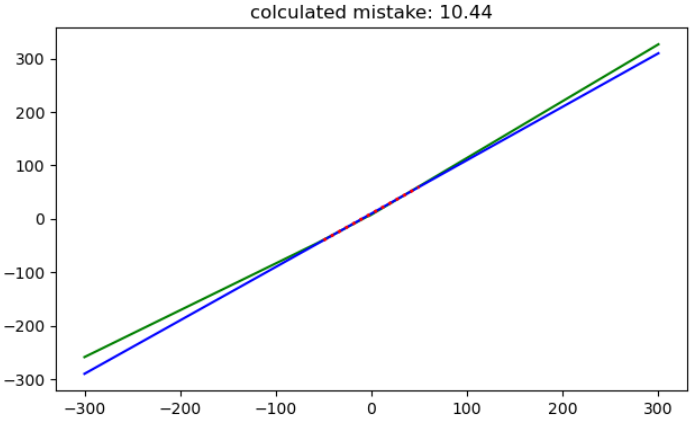
بازه آموزشی:



Train Info : number of points : 200 Train Range : (-150,150)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000

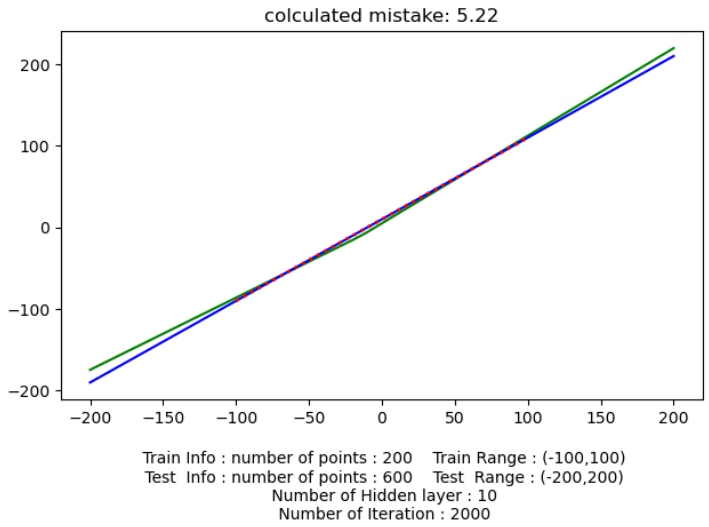


Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000

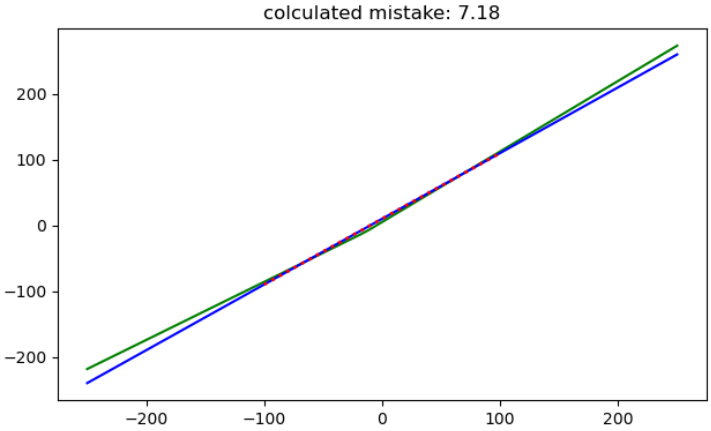


Train Info : number of points : 200 Train Range : (-50,50)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000

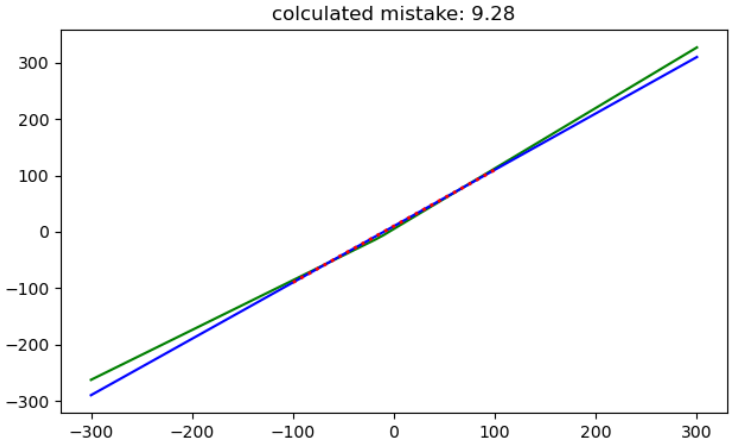
بازه تست:



Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-200,200)
Number of Hidden layer : 10
Number of Iteration : 2000

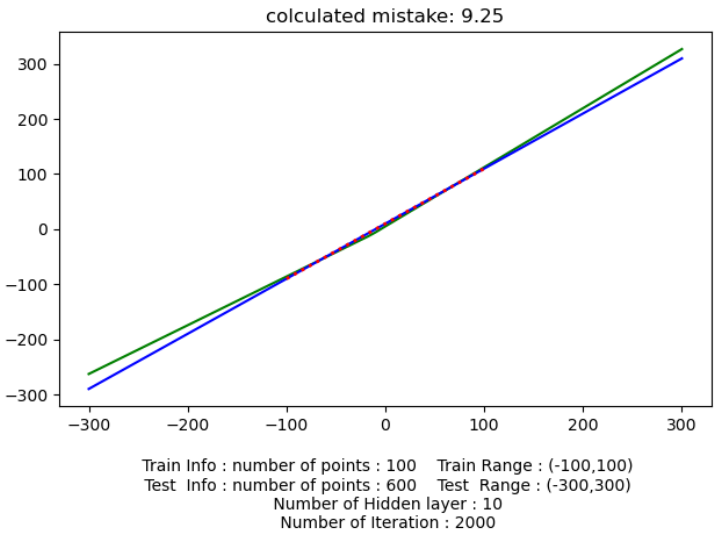


Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-250,250)
Number of Hidden layer : 10
Number of Iteration : 2000

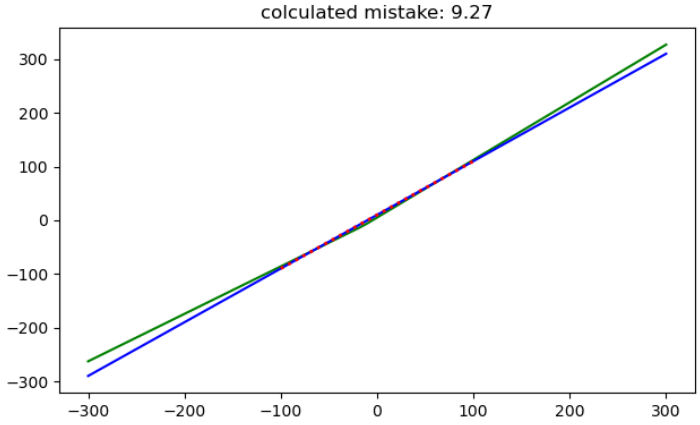


Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000

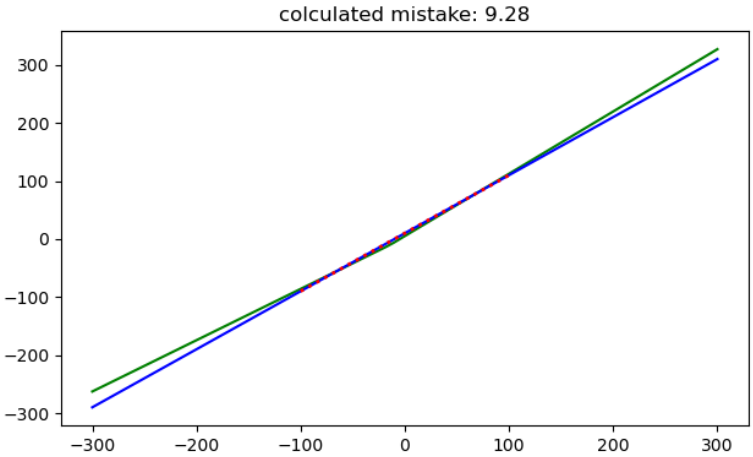
تعداد نقاط آموزشی:



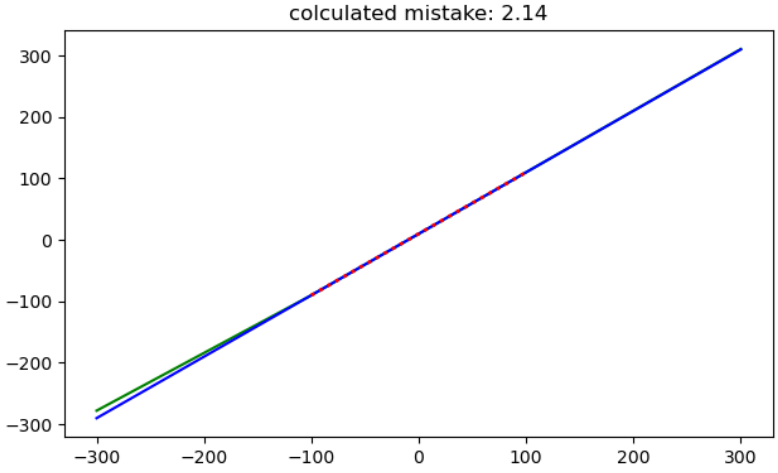
Train Info : number of points : 100 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000



Train Info : number of points : 150 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000

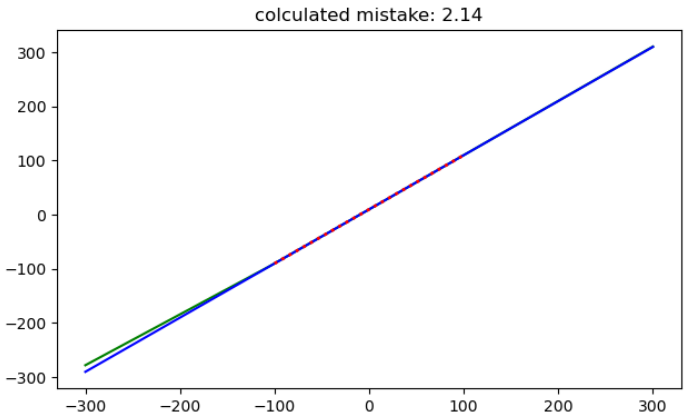


Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000

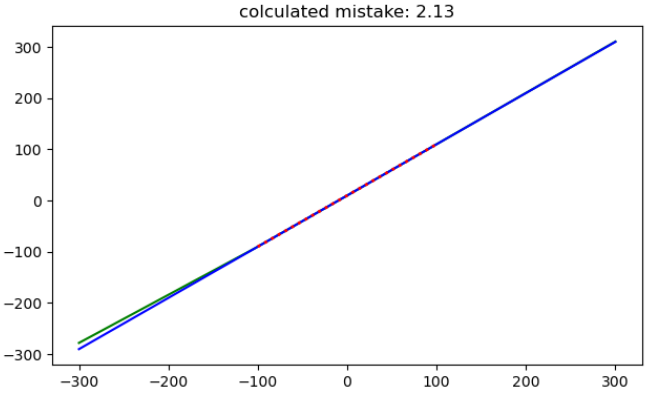


Train Info : number of points : 400 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000

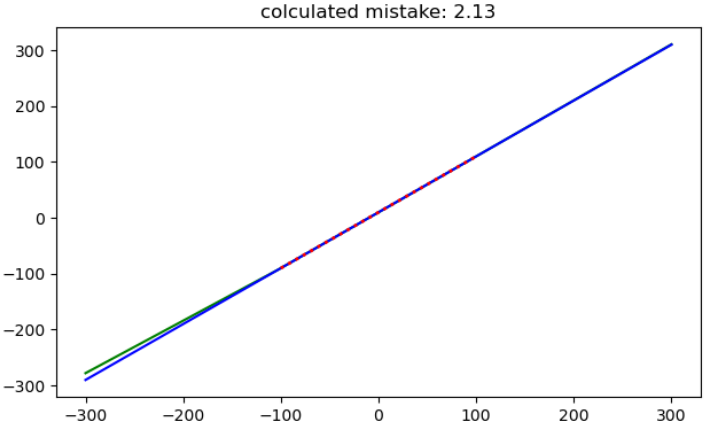
تعداد نقاط تست:



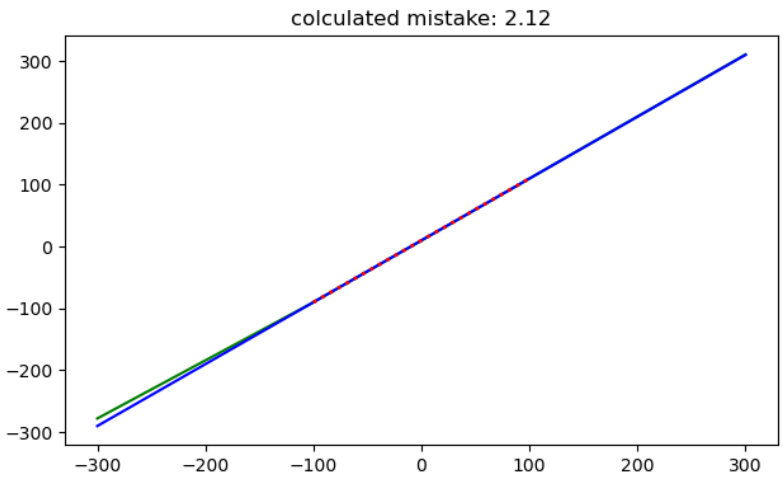
Train Info : number of points : 300 Train Range : (-100,100)
Test Info : number of points : 200 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000



Train Info : number of points : 300 Train Range : (-100,100)
Test Info : number of points : 300 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000

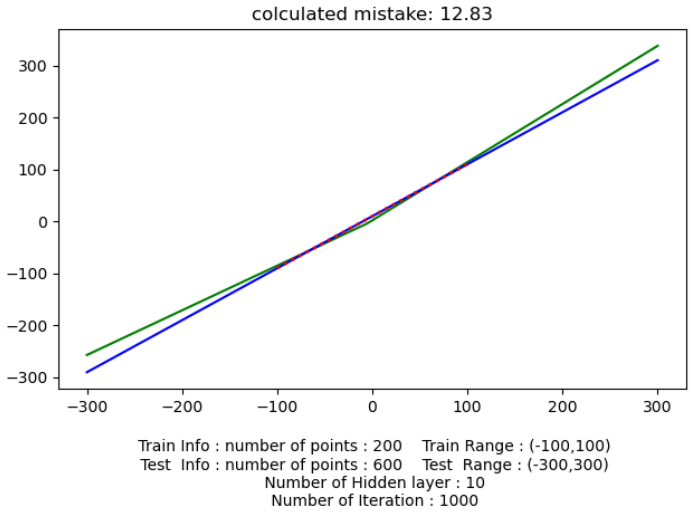


Train Info : number of points : 300 Train Range : (-100,100)
Test Info : number of points : 400 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000

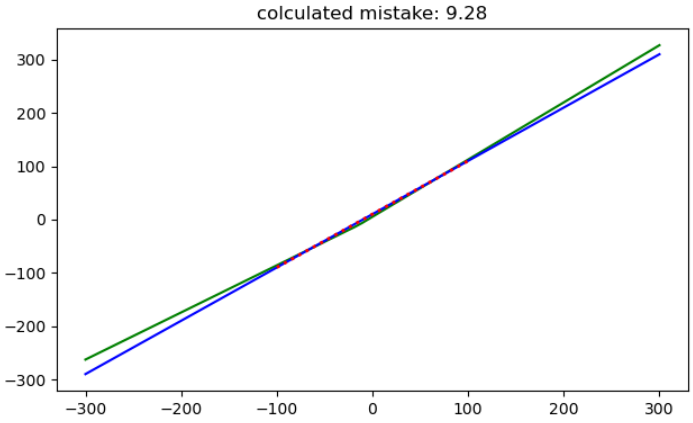


Train Info : number of points : 300 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000

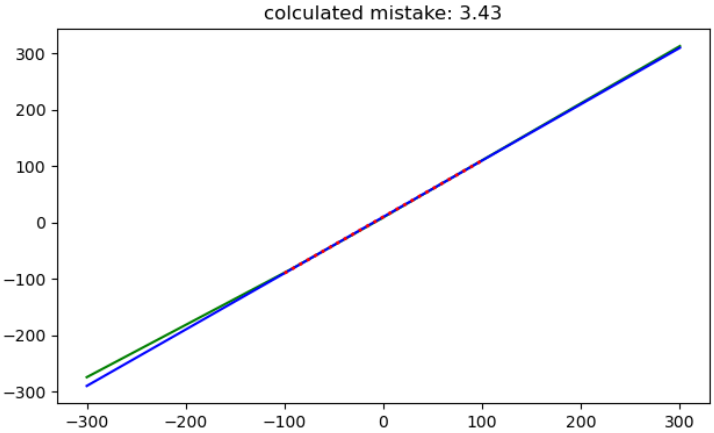
تعداد iterations :



Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 1000



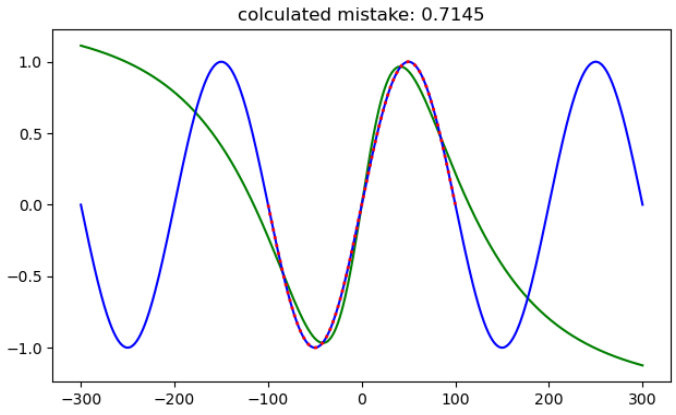
Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 2000



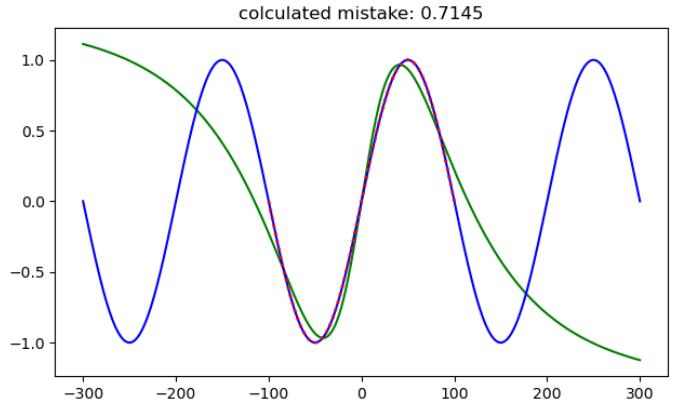
Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 10
Number of Iteration : 3000

حال تابع مد نظر خود را کمی سخت تر می کنیم و همین آزمایش ها را روی آن انجام می دهیم و نتایج را بررسی می کنیم.

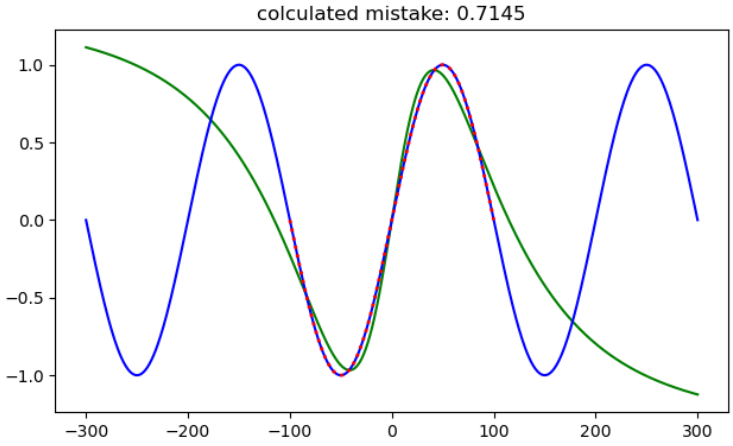
تعداد iterations :



Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 1000
Number of Iteration : 3000

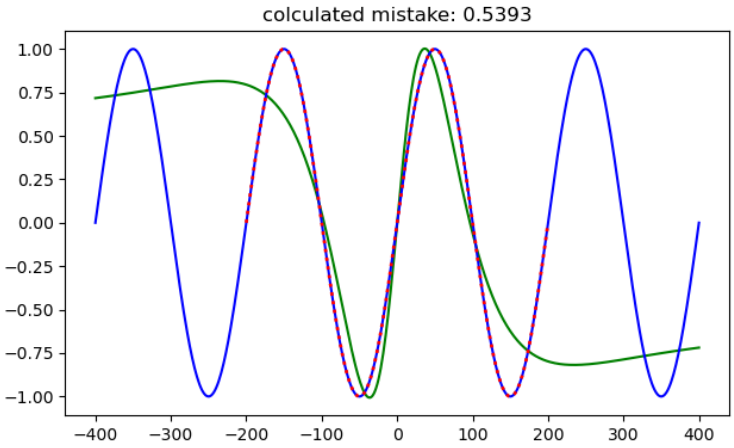


Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 1000
Number of Iteration : 1500

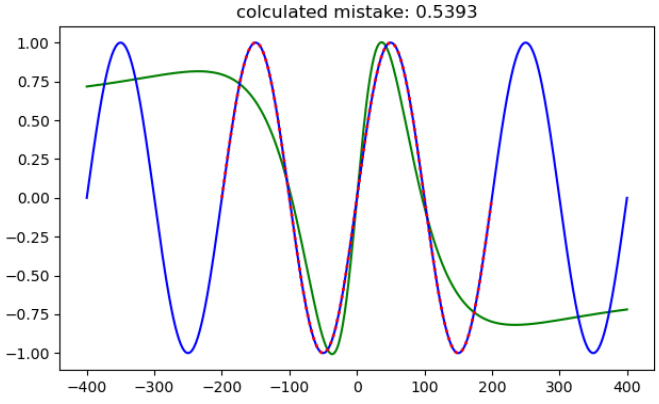


Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 1000
Number of Iteration : 2000

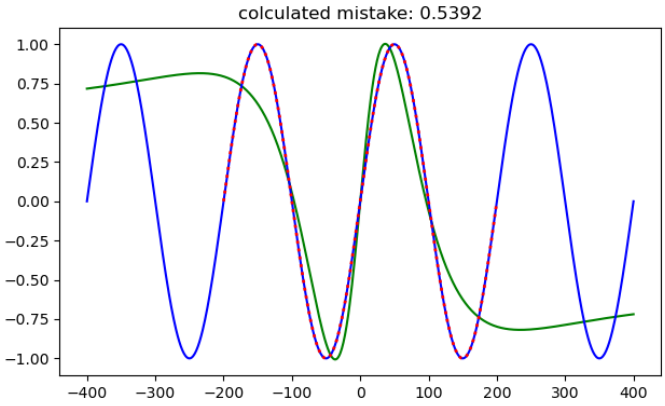
تعداد نقاط تست :



Train Info : number of points : 200 Train Range : (-200,200)
Test Info : number of points : 500 Test Range : (-400,400)
Number of Hidden layer : 1000
Number of Iteration : 2000

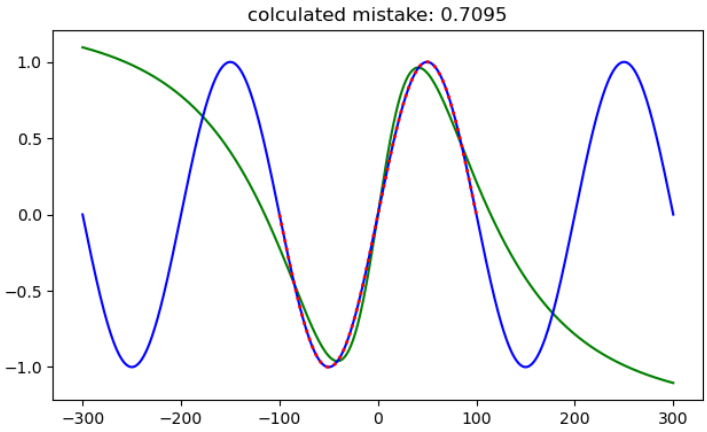


Train Info : number of points : 200 Train Range : (-200,200)
Test Info : number of points : 600 Test Range : (-400,400)
Number of Hidden layer : 1000
Number of Iteration : 2000

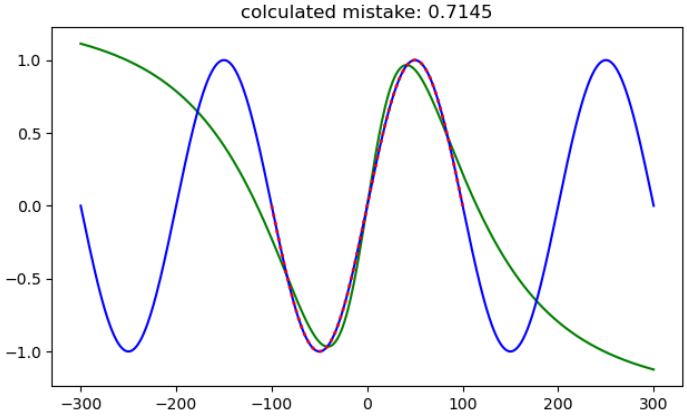


Train Info : number of points : 200 Train Range : (-200,200)
Test Info : number of points : 700 Test Range : (-400,400)
Number of Hidden layer : 1000
Number of Iteration : 2000

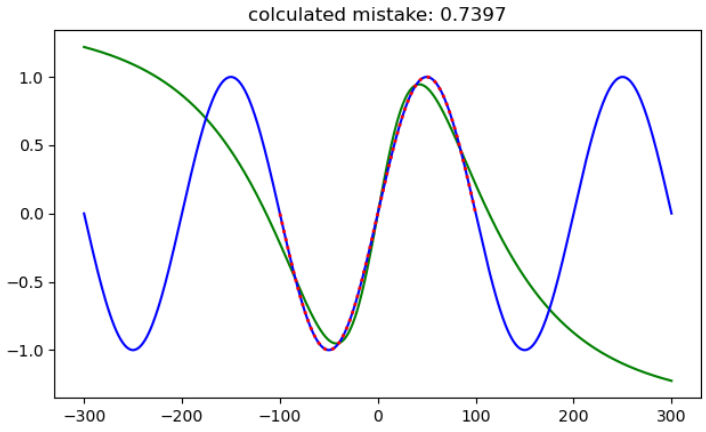
تعداد نقاط آموزشی



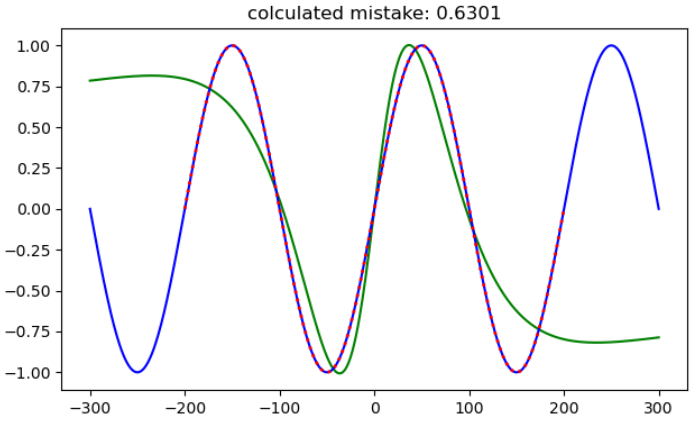
Train Info : number of points : 150 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 1000
Number of Iteration : 2000



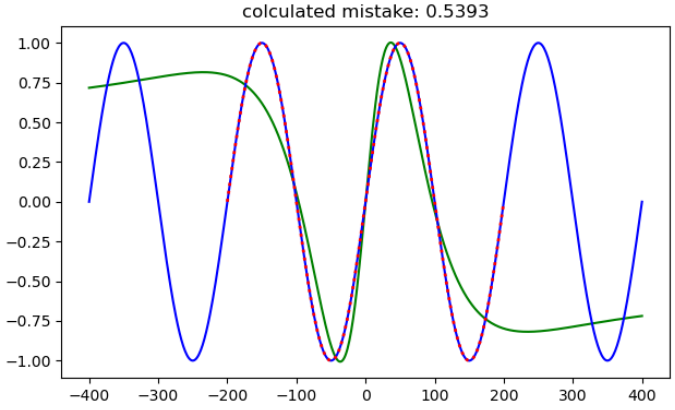
Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 1000
Number of Iteration : 2000



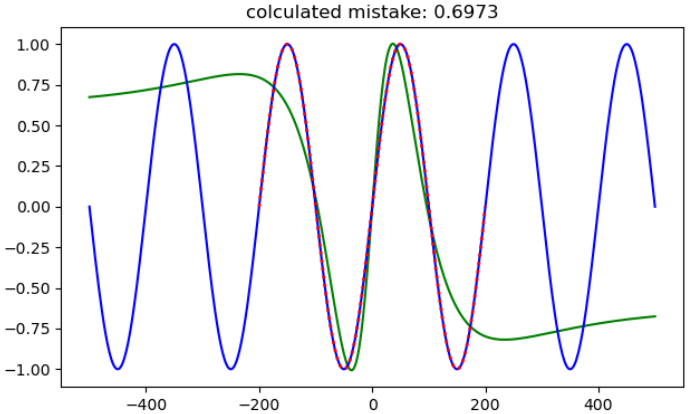
Train Info : number of points : 250 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 1000
Number of Iteration : 2000



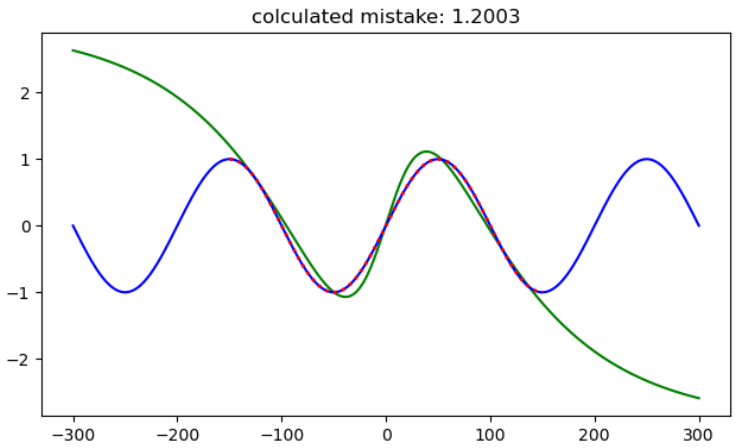
Train Info : number of points : 200 Train Range : (-200,200)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 1000
Number of Iteration : 2000



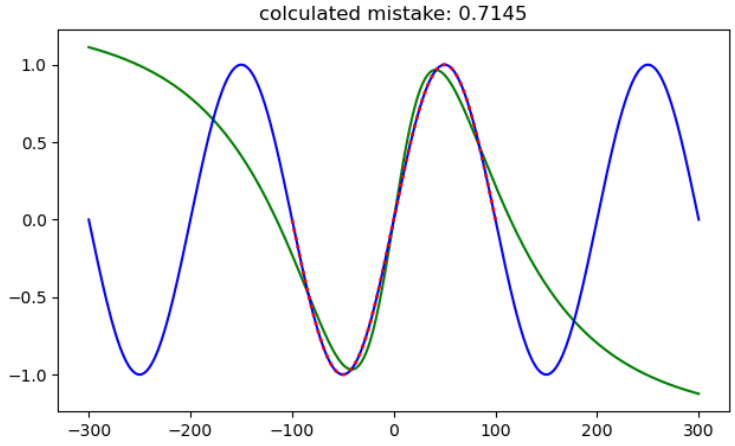
Train Info : number of points : 200 Train Range : (-200,200)
Test Info : number of points : 600 Test Range : (-400,400)
Number of Hidden layer : 1000
Number of Iteration : 2000



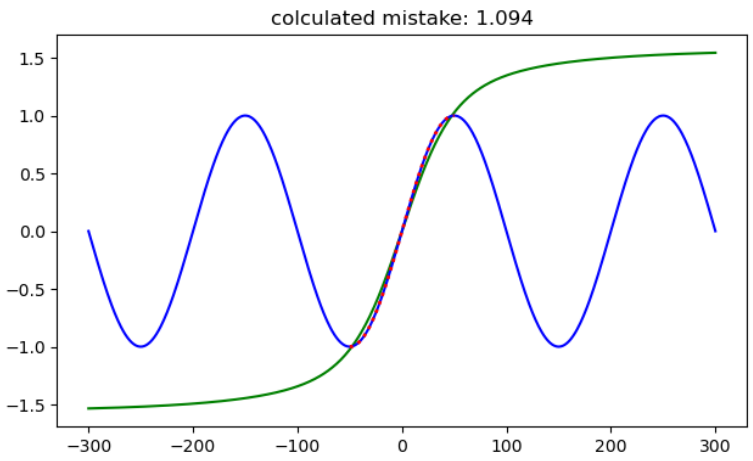
Train Info : number of points : 200 Train Range : (-200,200)
Test Info : number of points : 600 Test Range : (-500,500)
Number of Hidden layer : 1000
Number of Iteration : 2000



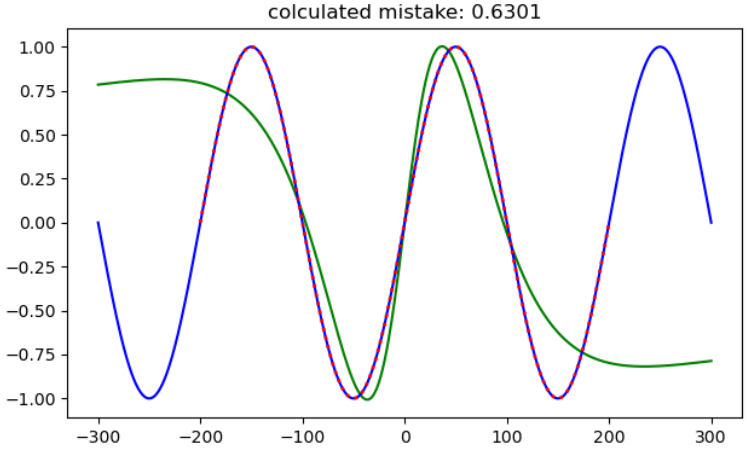
Train Info : number of points : 200 Train Range : (-150,150)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 1000
Number of Iteration : 2000



Train Info : number of points : 200 Train Range : (-100,100)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 1000
Number of Iteration : 2000



Train Info : number of points : 200 Train Range : (-50,50)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 1000
Number of Iteration : 2000



Train Info : number of points : 200 Train Range : (-200,200)
Test Info : number of points : 600 Test Range : (-300,300)
Number of Hidden layer : 1000
Number of Iteration : 2000

بخش دوم:

به توابع بخش اول مقدار رندوم اضافه می کنیم و باز هم مثل بخش اول دیتا برای train به شبکه می دهیم و بعد از آن دیتای تست را به عنوان ورودی به شبکه میدهیم تا ببینیم شبکه چگونه رفتار می کند

```
def myFuction(input_list):
    return [i+4+round(random.uniform(0,1),2) for i in input_list]
```

```
def myFuction(input_list):
    return [math.sin(i*math.pi/100)+round(random.uniform(0,0.1),4) for i in input_list]
```

```
def myFuction(input_list):
    return [math.sin(i*math.pi)+i**2+round(random.uniform(0,1),2) for i in input_list]
```

بخش سوم:

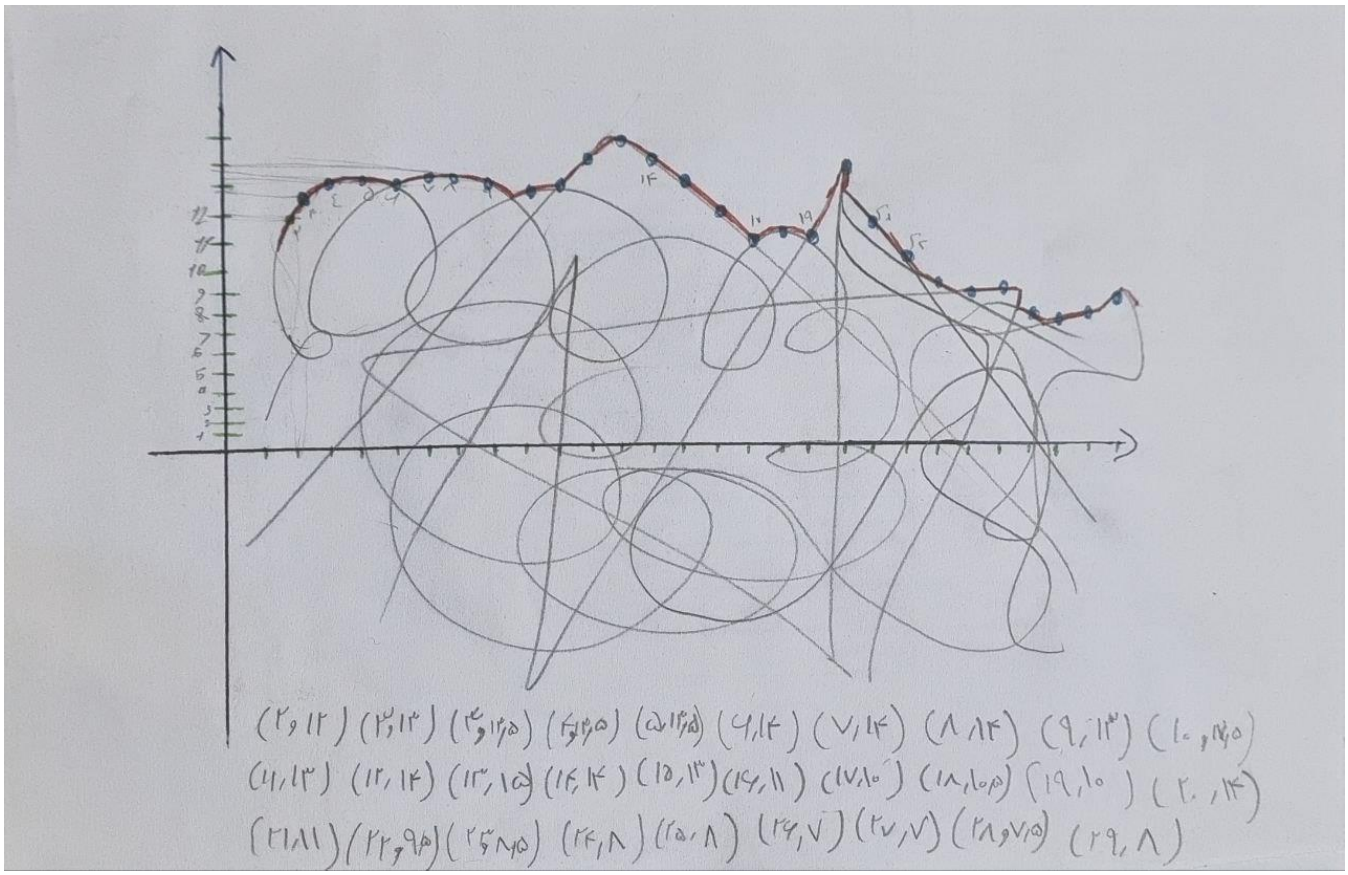
در این بخش من یک تابع با دو پارامتر را به عنوان ورودی می گیرم و در جواب یک خروجی میگیرم

```
def my_function(input):
    res = [ 2*i[0]+i[1] for i in input]
    return res
```

تنها تفاوتی که این بخش با بخش های قبلی دارد این است که ورودی ما دو بعدی است .

بخش چهارم:

در این بخش ابتدا بر روی یک برگه تمام خشم خود را خالی کرده



سپس نقاط مشخص شده را به عنوان ورودی و خروجی در نظر میگیریم قسمتی از آنها را به عنوان داده ی train و ما بقی آن را به عنوان داده ی تست در نظر میگیریم و با پارامتر های مختلف بازی می کنیم تا ببینیم در چه شرایطی جواب ما کمترین خطا را دارد

بخش پنجم:

تابع calculate mistake را تغییر می دهیم زیرا عکس های ما تگ خورده اند و نیازی به محاسبه ی اختلاف نیست به عبارتی خروجی تابع ما یا درست است یا نه در نتیجه به این صورت محاسبه می کنیم

```
def colculate_mistakes(our_answer,correct_Answer):
    l = len(our_answer)
    sum = 0
    for i in range(0,l):
        if our_answer[i]!=correct_Answer[i]:
            sum = sum+ 1
    return sum
```

در این قسمت همه ی فایل های داخل فولدر Train را خوانده و هر تصویر را به ارایه 256 تایی تبدیل میکنیم و خروجی را هم بر اساس اسم فایل میتوانیم استخراج کنیم

```
train_x = []
train_y = []
for trainfile in train_files:
    y = int(trainfile.split('_')[0])
    img = cv2.imread(train_path+"\\"+str(trainfile),0)
    img_np = np.array(img)
    img_np = img_np.reshape(256)
    # img_np2 = np.reshape(img, [256, 1]) #256 D

    train_y.append(y)
    train_x.append(img_np)
    # print(img_np)

train_x = np.array(train_x)
train_y = np.array(train_y)
```

همین کار را برای فایل تست می کنیم

```
test_x = []
test_y = []
for testfile in test_files:
```

```
    y = int(testfile.split('_')[0])
    img = cv2.imread(testfile+"\\"+str(testfile),0)
    img_np = np.array(img)
    img_np = img_np.reshape(256)
    # img_np2 = np.reshape(img, [256, 1]) #256 D

    test_y.append(y)
    test_x.append(img_np)
    # print(img_np)

test_x = np.array(test_x)
test_y = np.array(test_y)
```

درنهایت شبکه ی عصبی خود را train می کنیم

```
train_network = MLPClassifier(hidden_layer_sizes=hidden_layer,
                              max_iter=num_of_iteration,
                              random_state=1,
                              ).fit(train_x,train_y)

tree_y = train_network.predict(test_x)

mistake = colculate_mistakes(test_y,tree_y)
print('colculated mistake: ' ,mistake/20)
```

سپس با انجام آزمایش های مختلف و تغییر دادن پارامتر ها سعی می کنیم به بهترین نتیجه برسیم

بخش ششم :

در اینجا ما در ابتدا عکس ها را نویز دار کرده و داخل فولدر noise_train می ریزیم.

```
train_path ="D:\\term6\AI\Project2\Part6\images\\train"
train_files = [f for f in listdir(train_path) if isfile(join(train_path, f))]

for trainfile in train_files:
    with Image(filename ="D:\\term6\AI\Project2\Part6\images\\train\\"
      +str(trainfile)) as img:
        img.noise("poisson", attenuate = 0.9)
        img.save(filename="D:\\term6\AI\Project2\Part6\images\\train_noise1\\"
      +str(trainfile))
```

سپس همین کار را برای داده های تست انجام می دهیم

```
test_path ="D:\\term6\AI\Project2\Part6\images\\test"
test_files = [f for f in listdir(test_path) if isfile(join(test_path, f))]

for testfile in test_files:
    with Image(filename ="D:\\term6\AI\Project2\Part6\images\\test\\"
      +str(testfile)) as img:
        img.noise("poisson", attenuate = 0.9)
        img.save(filename="D:\\term6\AI\Project2\Part6\images\\test_noise\\"
      +str(testfile))
```

سپس عکس های نویز دار را به عنوان ورودی و عکس های بدون نویز را به عنوان خروجی به شبکه ی عصبی خوب میدهیم . سپس آزمایش را انجام می دهیم

```
train_path ="D:\\term6\AI\Project2\Part6\images\\train"
train_noise_path = "D:\\term6\AI\Project2\Part6\images\\train_noise"
train_files = [f for f in listdir(train_path) if isfile(join(train_path, f))]
# print(train_files)
train_x = []
train_y = []
for trainfile in train_files:
    img = cv2.imread(train_path+"\\"+str(trainfile),0)
    img_np = np.array(img)
    img_np = img_np.reshape(256)
    # img_np2 = np.reshape(img, [256, 1]) #256 D
```

```
    noise_img = cv2.imread(train_noise_path+"\\"+str(trainfile),0)
    noise_img_np = np.array(img)
    noise_img_np = img_np.reshape(256)


    train_y.append(img_np)
    train_x.append(noise_img_np)
    # print(img_np)

train_x = np.array(train_x)
train_y = np.array(train_y)

test_path ="D:\\term6\AI\Project2\Part6\images\\test"
test_noise_path = "D:\\term6\AI\Project2\Part6\images\\test_noise"
test_files = [f for f in listdir(test_path) if isfile(join(test_path, f))]
# print(test_files)
test_x = []
test_y = []
for testfile in test_files:
    img = cv2.imread(test_path+"\\"+str(testfile),0)
    img_np = np.array(img)
    img_np = img_np.reshape(256)
    # img_np2 = np.reshape(img, [256, 1]) #256 D

    noise_img = cv2.imread(test_noise_path+"\\"+str(testfile),0)
    noise_img_np = np.array(img)
    noise_img_np = img_np.reshape(256)
    test_y.append(img_np)
    test_x.append(noise_img_np)
    # print(img_np)

test_x = np.array(test_x)
test_y = np.array(test_y)

hidden_layer = (100,100,100,20)
# hidden_layer = (100,1000,100)
num_of_iteration = 2000

train_netwoek = MLPRegressor( hidden_layer_sizes= hidden_layer,
                              max_iter=num_of_iteration,
                              random_state=1,
                              shuffle=True).fit(train_x, train_y)

tree_images = train_netwoek.predict(test_x)
```

و درنهایت چیزی که شبکه حدس میزند را در یک فایل دیگر به اسم network_predict می ریزیم

```
for i in range(len(tree_images)) :
    image = tree_images[i]
    my_img = np.reshape(image, [16, 16])
    cv2.imwrite("D:\\term6\AI\Project2\Part6\images\\network_predict"+ "\\"
+str(i)+'_predict.png', my_img)
```

برخی از نتایجی که از آزمایش های مختلف بدست آورده ام :

هرچه تعداد نقاط train را بیشتر کنیم دقت شبکه بیشتر می شود . برای تعداد نقاط تست هم همین طور می باشد منتها تاثیر نقاط train بیشتر است .

با افزایش دامنه ی train یادگیری بهتر رخ داد ولی افزایش دامنه ی تست باعث شد مقدار خطا بیشتر شود .

Number of iteration تا حدی که بالا می رود تاثیر روی بهبودی یاد گیری شبکه دارد اما از یه جایی به بعد هر چقدر آن را زیاد کنیم دیگر تاثیری ندارد

تعداد نورون ها در هر لایه به این صورت نیست که هر چقدر بیشتر باشد بهتر است . هر چه تعداد نورون ها بیشتر باشد پارامتر های بیشتری مورد بررسی قرار میگیرد که ممکن است آن پارامتر ها تاثیری در خروجی ما نداشته باشند پس افزایش تعداد نورون ها تا یک حدی تاثیر خوبی روی شبکه دارد .

برای بخش دوم که نویز اضافه می کنیم number of iteration را اگر بالا ببریم ممکن است پارامتر های نویز را در شبکه در نظر بگیرد

هرچه تعداد لایه ها بیشتر باشد تا یک حدی تاثیر خاصی دارد.

برخی دیگر از نتایج آزمایش ها را می توانید در عکس های بخش های قبلی ملاحظه کنید