پروژه ارقام دست نویس

(HANDWRITTEN DIGITS)

نام دانشجو:

هدیه مفتخری رستم خانی

شماره دانشجویی:

971113054

نام استاد راهنما :

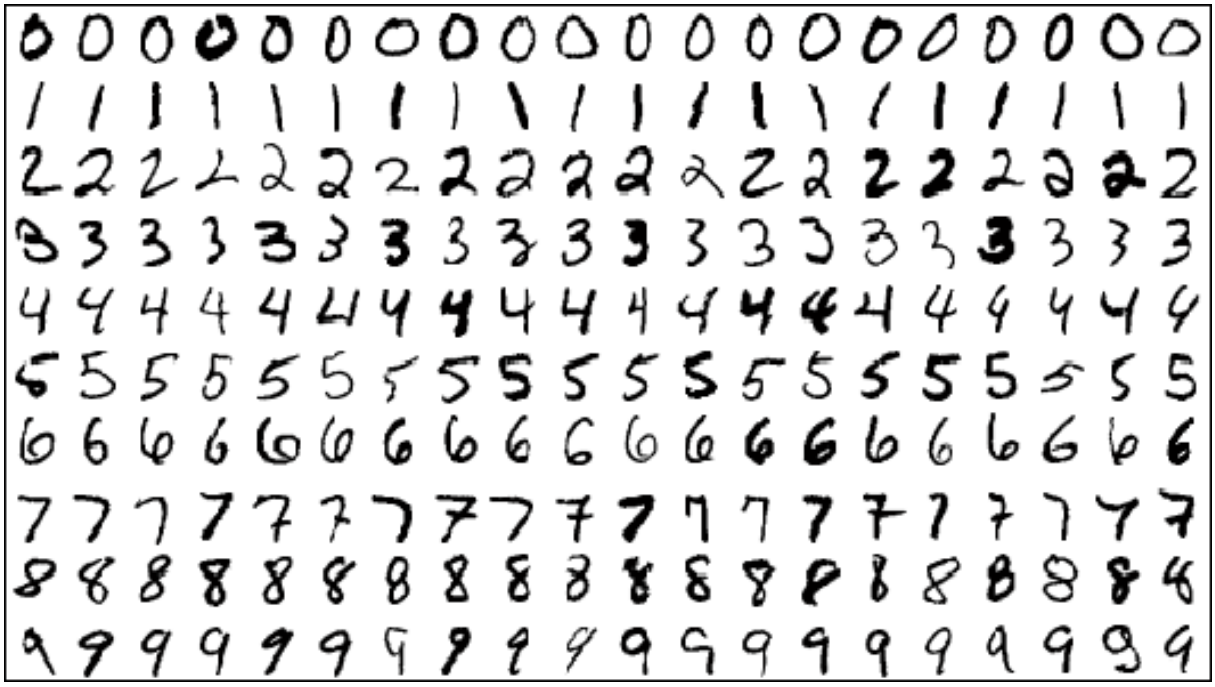
سرکارخانم آزاده طباطبایی

ترم پاییز 99

مقدمه :

در این پروژه قصد داریم تا ارقامی که به صورت دستنویس نوشته شده اند را به نحوی که توضیح داده خواهد شد در بیاوریم تا کامپیوتر متوجه شود که آن رقمی که ما نوشتیم در واقع چه عددیست.

به طور مثال شما روی یک کاغذ می نویسید 2 ما عکس عدد 2 را به کامپیوتر می دهیم و کامپیوتر آن عکس را به خانه های مربعی شکل به اسم بیت در می آورد مثلا عکس شما را به شکل یک جدول 28*28 بیت در می آورد که جا هایی که شما در آنجا نوشتید یا سیاهش کردید مقدار بیت آن خانه را 1 می دهد و جا هایی از عکس که سفید است و در آن ننوشتید در هر یک از خانه های جدول قرار بگیرد آن خانه 0 خواهد بود , حالا اگر فرد دیگری با دست خط دیگری بیاید و عکس عدد 2 ای که نوشته را به کامپیوتر بدهد با توجه به شبکه عصبی که ما طراحی کردیم کامپیوتر می فهمد که این عکس همان عدد 2 است. یعنی ازین به بعد هر عکسی از هر عدد 2 ای به شبکه ما بدهید شبکه ما خواهد فهمید که عدد 2 است.



شرح پروژه :

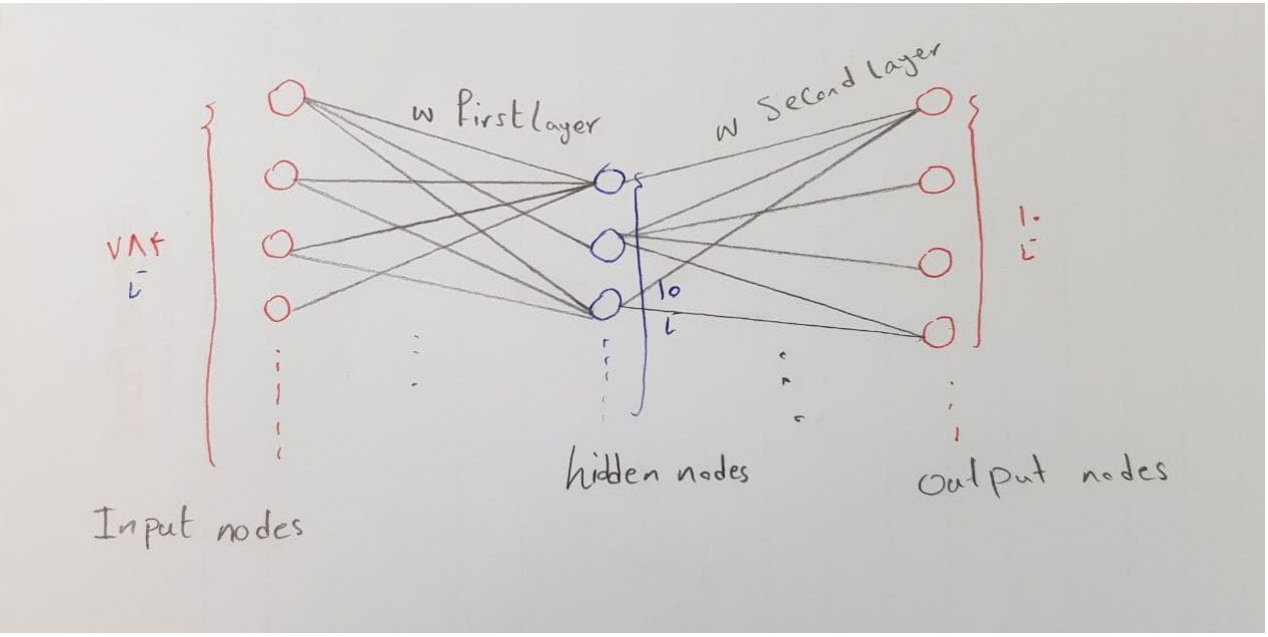
Mnist: یک dataset است که شامل 60000 داده آموزشیست یعنی 60000 عدد عکس در آن وجود دارد تا با استفاده از آن ها شبکه خود را آموزش دهیم و هر کدام ازین داده ها یا همان عکس ها یک label دارند که به ما می گوید این عکس چه عددیست همچنین در mnist یک مجموعه داده 10000 تایی از عکس هایی وجود دارد که برای تست کردن استفاده می شود که این ها هم label دارند.

ما در اینجا تنها از 1000 داده آموزشی برای آموزش شبکه خود استفاده می کنیم و از داد های تست فقط از 5 تای آن استفاده می کنیم.هر عکس در mnist به صورت یک آرایه 28*28 است یعنی هر داده یا همان عکس آن از 784 تا پیکسل یا بیت تشکیل شده.

چون ورودی شبکه عصبی که ما داریم به صورت یک آرایه تک بعدی از نود های ورودیست باید آرایه 28*28 ای که برای هر ورودیست را به یک آرایه 784*1 تبدیل کنیم یعنی پیکسل های عکس را به صورت یک آرایه 784*1 در بیاوریم و به شبکه به عنوان ورودی بدهیم.

خروجی شبکه هم به فرمت one-hot encoding است یعنی اگر مثلا عکس عدد 2 را به کامپیوتر بدهیم ابتدا می آید بیت های عکس آن را که 28*28 است به یک آرایه 784*1 تبدیل می کند هر درایه از این آرایه را به یک نود ورودی متناظر با آن در شبکه می دهد و پردازش هایش را که انجام داد خروجی شبکه را به شکل :

0010000000 که 10 بیت است می دهد .



توضیحات کد :

ابتدا کتاب خانه هایی که به آن نیاز داریم را اضافه می کنیم :

```
import keras
from keras.datasets import mnist
import numpy as np
```

داده ها را از mnist می گیریم و از داده های تمرین تنها 1000 تا و از داده های امتحان کردن 1000 تا را برداشته وبه ترتیب درون آرایه های X_train و X_test می ریزیم و خروجی که از X_train و X_test انتظار داریم را به ترتیب در y_train وy_test می ریزیم:

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()

X_train = X_train[:1000,:,:]
X_test = X_test[:1000,:,:]
y_train = y_train[:1000]
y_test  = y_test[:1000]
```

چون Y_Train و Y_test به صورت یک آرایه (عکس) 28*28 است و ما می خواهیم آن ها را با خروجی هایمان که به صورت one-hot encoding است مقایسه کنیم باید شکل Y_Train و Y_test را به صورت one-hot در بیاریم تا بیت به بیت آن را بتوانیم با بیت های خروجی که از شبکه خودمان می گیریم مقایسه کنیم :

```
# Convert to "one-
hot" vectors using the to_categorical function
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

عکس هایی که mnist به ما می دهد هر پیکسل آن می تواند دارای مقداری بین 0 تا 255 باشد اما ما می خواهیم هر بیت یا پیکسلی که در X_train یا X_test میریزیم فقط 0 یا 1 باشد پس ابتدا بر 255 آن را تقسیم می کنیم بعد عدد صحیح بدست آمده را در خانه های X_train و X_test می ریزیم :

```
#Each image has Intensity from 0 to 255
#so convert it to range 0 to 1
X_train = X_train/255
X_test = X_test/255
X_train = X_train.astype(int)
X_test= X_test.astype(int)
```

قبل تر گفتیم چون نود های ورودی مان باید 784 تا باشد آرایه 28*28 هر ورودی (عکس) را به صورت یک آرایه 1*784 تبدیل می کنیم به صورت زیر:

```
#Because of the matrix multipication we must change the shap
e of the array.
#(AxB * BxC = AxC) So we will shape the 28x28 array to 1x784
.
num_pixels = 784
X_train = X_train.reshape(X_train.shape[0],num_pixels)
X_test = X_test.reshape(X_test.shape[0],num_pixels)
```

مقادیرمان را به صورت زیر تعریف می کنیم دقت کنید W_firstlayer وزن هاییست که از نود های ورودی به نود های مخفی می رود و W_seclayyer آرایه ای از وزن هاییست که از نود های مخفی به نود خروجی می رود :

```
W_firtlayyer=np.random.random((784,10))
W_seclayyer=np.random.random((10,10))
net_hidden_nodes=np.zeros(10)
net_out_nodes=np.zeros(10)
Output_H=np.zeros(10)
Output_O=np.zeros(10)
sigma=0
sig_out=np.zeros(10)
sig_Hiden=np.zeros(10)
```

قسمت اصلی برنامه و پیاده سازی الگوریتم backpropagation در شبکه به شرح زیر انجام می شود:

قرار است به ازای هر کدام از 1000 داده ورودی , الگوریتمی را اجرا کنیم و این اجرای الگوریتم به ازای هر 1000 داده قرار است 2000 بار تکرار شود . یعنی حلقه تو در تو به شکل زیر :

```
for i1 in range(0,2000):# تعداد دفعات تکرار الگوریتم
    for num_input_data in range (0,1000):#تعداد دفعات تکرار به ازای داده ها
```

حال در حلقه درونی به تعداد داده های آموزشی (1000) به اجرای الگوریتم زیر می پردازیم :

ابتدا net را برای هر کدام از نود های مخفی و خروجی محاسبه می کنیم (اگر هر ورودی به یک نود در یال متناظر با آن ورودی ضرب شود و ما بیاییم این عمل را برای تمامی ورودی ها به یک نود انجام دهیم و در نهایت حاصل جمع تمامی این حاصل ضرب ها را حساب کنیم می شود net آن نود)

```
        net_hidden_nodes = np.matmul(X_train[num_input_data,:],W_firtlayyer)
        Output_H=1/(1+(np.exp(-net_hidden_nodes)))
        net_out_nodes = np.matmul(Output_H,W_seclayyer)#calculate output nodes' net
```

خروجی هر نود در واقع سیگوییدِ net آن نود است که در کد زیر برای نود خروجی حساب شده و چون برای بدست آوردن net نود های خروجی به خروجیِ نود های داخلی نیاز داشتیم در چند خط کد بالا آن را حساب کرده بودیم :

```
Output_O=1 / (1 + a)
```

برای بروز رسانی وزن ورودی به نود داخلی (W_firstlayer) و وزن نود خارجی (W_seclayer) فرمول های مخصوص به خودشان را داریم و در این فرمول ها SIGMOID را باید برای هر فرمول داشته باشیم تا درون آن قرار دهیم و مقدار محاسبه SIGMOID فرمولش در وزن های لایه اول و دوم متفاوت است که به ترتیب در چند خط کد زیر حساب شده :

```
        a=np.multiply(Output_O,(1-Output_O))
        b=np.subtract(y_train[num_input_data,:],Output_O)
        sig_out=np.multiply(a,b
        sigma += np.matmul(sig_out,W_seclayyer.T)
        a=np.multiply(Output_H,(1-Output_H))
        sig_Hiden=sigma*a
        sigma=0
```

حال که تمامی مقادیر مورد نیاز را برای بروز رسانی وزن ها داریم در چند خط کد زیر آن ها را قرار می دهیم و با توجه به فرمول ها به ترتیب وزن های لایه اول و دوم را حساب می کنیم:

```
        X_train_i=np.array(X_train[num_input_data,:])[np.newaxis]
        X_train_T=X_train_i.T
        b=np.array(sig_Hiden)[np.newaxis]
        W_firtlayyer += np.matmul(X_train_T,b)
        a=np.array(Output_H)[np.newaxis]
        W_seclayyer +=  np.matmul(a,sig_out)
```

حالا از داده های تست 5 تای اول آن ها را برمی داریم و به ورودی شبکه مان می دهیم تا ببینیم آیا خروجی شبکه (OUTPUT_O) آیا با (Y_test) همخوانی دارد یا نه در چند خط کد زیر کد تست کردن و خروجی گرفتن را می بینیم :

```
for cnt in range (0,5):
    net_hidden_nodes = np.matmul(X_test[cnt,:],W_firtlayyer)#calculate
 hidden
    Output_H=1/(1+(np.exp(-net_hidden_nodes)))
    net_out_nodes = np.matmul(Output_H,W_seclayyer)
    Output_O=1/(1+(np.exp(-net_out_nodes)))
    print (str(y_test[cnt]))
    print (str(Output_O))
```

و حالا خروجی هایمان را در زیر می بینیم که 80% آنها درست درآمده اند :

1.[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]

آرایه بالا اولین درایه از Y_test است و آرایه زیر خروجی شبکه عصبی ماست , اگر بزرگترین عدد آرایه زیر را 1 در نظر بگیریم و بقیه را 0 در نظر بگیریم می بینیم که خروجی شبکه ما با آرایه بالا می خواند

[0.07860919 0.11710958 0.10038208 0.1408893 0.05417305 0.117616

0.01818924 0.25673141 0.07532075 0.2396351]

به همین ترتیب برای 4 داده تست بعدی به صورت زیر میبینیم که :

2.[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]

[0.07012744 0.1236363 0.24953506 0.06514026 0.03986581 0.1162816

0.07271874 0.07785499 0.12765884 0.08413888]

3.[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]

[0.06445794 0.20975513 0.22182308 0.12680543 0.0793308 0.10218205

0.06223891 0.11837768 0.12977214 0.11892678]

خروجی داده سوم با خطای بسیار کمی درست است

4.[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

[0.31521772 0.05600026 0.1015411 0.08488709 0.05463423 0.10033474

0.06176037 0.04642115 0.17086489 0.17762898]

خروجی داده 5 ام اشتباه است , برای اینکه درصد اشتباهات از بین داده های تست پایین بیاید باید تعداد داده های آموزشی و تعداد دفعات تکرار الگوریتم را زیاد تر کنیم که در اینجا توان سیستم ما در همین اندازه است :

5.[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]

[0.02723111 0.07863484 0.09569381 0.05488465 0.03079344 0.052648

0.02105572 0.07962596 0.05484745 0.06579256]