گزارش

دراین قسمت به توضیح توابع مختلف نوشته شده در کد متلب می پردازیم. نکته ای که باید به آن توجه کرد که توضیح هر سمت از کد بلافاصله بعد از آن آورده شده است.

مسئله ۱: طبقه بندی ۲۶ حرف الفبا

تابع اصلی main.m

```
clc;
clear all;
close all;
```

پاک کردن صفحه ، پاک کردن متغیرهای قبلی و بستن پنجره های باز شده قبلی

```
load('data.mat');
load('data_normal.mat');
load('target.mat');
```

فرخوانی داده های مربوط به الفبای فارسی که شامل دیتای معمولی، دیتای نرمال شده و کلاس های مربوطه می باشد.

```
cr='ABCDEFGHIJKLMNOPQRSTUVWXYZ';
tar=zeros(length(target),1);
```

تشکیل بردار الفبا و تکشیل اولیه ماتریس target

```
for i=1:26
tar=tar+ i*(target==cr(i));
end
```

دراین قسمت برداری به نام tar تشکیل می شود که دراین بردار به جای هر حرف یک عدد قرار میگیرد. به عنوان مثال به جای A عدد۱ قرار میگرد. بنابراین اعداد موجود در این بردار از ۱ تا۲۶ می باشند.

```
Tarout=zeros(length(tar),5);
for i=1:length(tar)

        Tarout(i,:) = de2bi(tar(i),5,2,'left-msb')';

end
```

دراین قسمت ماتریس با نام Tarout تشکیل شده که به جای هر عدد در بردار tar ، معادل باینری آن گذاشته می شود.

```
trp=floor(length(data)*0.7);
```

trp ، تعداد داده های آموزش را تعیین می کند که ۷۰درصد تعداد کل داده هاست

```
training=data(1:trp,:);
testing=data(trp+1:end,:);
```

تشکیل ماتریس داده های آموزش و تست که به ترتیب ۷۰ و ۳۰ درصد داده های اصلی هستند.

```
training_class=Tarout(1:trp,:);
testing_class=Tarout(trp+1:end,:);
```

تشکیل ماتریس کلاس برای داده های آموزش و تست

```
hidden_layers = [35 12];
iterations = 500;
learning_rate = 0.01;
momentum = 0.05;
```

تعیین پارامترهای آموزش شبکه عصبی:

تعداد نرون های لایه ۱و۲ ، تعداد دفعات تکرار، نرخ آموزش و گشتاور مروبط به آپدیت

```
[model cc_train output_train] = train_mlp(training,
training_class, hidden_layers, iterations, learning_rate,
momentum);
```

این تابع برای آموزش شبکه عصبی به کار می رود. ورودی این تابع، داده ها ، کلاس ها و پارامترهای آموزش شبکه عصبی است و خروجی آن یک مدل برای شبکه عصبی و target های داده ها پس از آموزش است. منظور از مدل شبکه عصبی ، وزن های لایه ها و پارامترهای دیگر همچون mse و غیره می باشد .

```
[output_test cc_test] = test_mlp(model, testing,
testing_class);
```

این تابع برای تعیین target یا کلاس داده های تست به کار می رود.

```
g=output_test>=0.5;
```

دراین قسمت، مقادیر بیت های مربوط به خروجی شبکه عصبی round به ۰ یا ۱ می شود. نحوه تعیین کار بدین صورت است که اگر مقدار هر بیت از ۰/۵ باشد، بیت برابر با ۰ و اگر بیشتر باشد بیت برابر با ۱ می شود.

```
for i=1:length(g)
u(i)=bi2de(g(i,:),2,'left-msb')';
end
```

دراین قسمت معادل دسیمال عدد باینری مربوط به خروجی داده های تست ، محاسبه شده و کـلاس هـر داده تعیین می شود.

```
y=testing_class;
for i=1:length(g)
v(i)=bi2de(y(i,:),2,'left-msb')';
end
```

دراین قسمت نیز معادل دسیمال مربوط به عدد باینری کلاس واقعی هر داده محاسبه می شود.

```
corrected_precision= 100*sum(u==v)/6000
```

درنهایت نیز میزان درصد شناسایی درست بر اساس مقایسه کلاس واقعی و کلاس تخمـین شـذه شـده توسـط شبکه عصبی تعیین می شود.

تابع train_mlp:

```
function [model cc output] = train_mlp(input, target,
hidden, iterations, learning_rate, momentum)
```

ورودی این تابع ، داده های آموزش ، کلاس آنها و پارامترهای آموزش شبکه عصبی است.

```
model = [];
model.learning_rate = learning_rate;
model.momentum = momentum
```

دراین قسمت یک objet به نام model تشکیل می شود که شامل یکسری ویژگی است . دوتا از ویژگی های ذکر شده مربوط به نرخ یادگیری و گشتاور است .

```
[ntrain nInLayer] = size(input);
```

تعیین تعداد نرون های لایه ورودی و تعداد نمونه های آموزش بر اساس ماتریس داده ها

```
    [jnk nOutLayer] = size(target);
```

تعیین تعداد نرون های لایه خروجی بر اساس ماتریس کلاس ها یا targetها

```
    nNeurons = [nInLayer hidden nOutLayer];
```

برداری که تعداد نرون های همه لایه ها را مشخص می کند.

```
    nNeurons(nNeurons == 0) = [];
```

حذف نرون صفر

```
    nTransitions = length(nNeurons)-1;
```

تعیین تعداد مجموعه وزن های بین لایه ها . مثلا اگه سه لایه داشته باشم، این مقدار ۲ محاسبه می شود.

```
for i = 1:nTransitions
        model.weights{i} =
randn(nNeurons(i),nNeurons(i+1));
model.biases{i} = randn(1,nNeurons(i+1));
model.lastdelta{i} = 0;
    end
```

تعیین مقادیر اولیه ماتریس وزن و بایاس بین لایه ها به صورت تصادفی . دراین قسمت ماتریس وزن و بایاس به عنوان ویژگی درگیری از model اضافه می شوند.

```
for cntt = 1:iterations
```

حال وارد فاز اموزش می شویم. دراین قمست یک حلقه forنوشته می شود که به تعداد دفعات تکرار تعیین شده ،اموزش شبکه عصبی تکرار و وزن ها آپدیت می شود.

```
order = randperm(ntrain);
```

دراین قسمت داده های آموزش رو به هم ریخته و ترتیب انها را به هم میریزیم تا یک فاز تصادفی از داده های آموزش داشته باشیم.

```
for j = 1:ntrain
```

حال با نوشتن این حلقه for، برای هر داده آموزش ، وزن ها را آپدیت کرده و شبکه را آموزش می دهیم.

```
  inp1=input(order(j),:);
 targ1=target(order(j),:);
```

دراین قسمت بردار ورودی و target خروجی تعیین می شود.

```
activations = cell(length(model.weights)+1,1);
activations{1} = inp1;
```

تشکیل ماتریس activations: این ماتریس در حقیقت ماتریسی است که یک مرحله بعد از اعمال وزن ها در هر مرحله تشکیل می شود. به عبارتی خروجی تابع فعالساز می باشد. مقدار خروجی تابع فعالسر برای لایه ورودی را همان بردار ورودی در نظر می گیریم.

```
for i = 1:length(model.weights)
temp = activations{i} * model.weights{i} + model.biases{i};
activations{i+1} = 1./(1+exp(-(temp)));
    end
```

دراین قسمت وزن ها در هرلایه اعمال شده و سپس از تابع فعالساز عبور کرده و وارد لایه بعد می شود.

```
errors = cell(length(model.weights),1);
```

دراین قسمت یک cell تشکیل می شود که در هر سلول از این cell ماتریس errorبرای هر لایه وجود دارد.

```
run_error = (targ1 - activations{end});
```

میزان خطای شناسایی در لایه آخر برابر است با اختلاف targetاصلی و خروجی تابع فعاساز لایه آخر

```
for i = length(model.weights):-1:1
errors{i} = activations{i+1} .* (1-activations{i+1}) .*
(run_error);
 run_error = errors{i} * model.weights{i}';
    end
```

دراین قسمت خطا براساس روابط موجود در شبکه عصبی برای آپدیت وزنها در هر لایه محاسبه می شود.

```
for i = 1:length(model.weights)

model.weights{i} = model.weights{i} + model.learning_rate *
activations{i}' * errors{i};
model.biases{i} = model.biases{i} + model.learning_rate *
errors{i};

    end
```

دراین قسمت وزن های بین لایه ها و وزن های بایاس بر اساس میزان error محاسبه شـده و نـرخ یـادگیری و وزن قبلی آپدیت می شود.

```
 [output cc] = test_mlp(model, input, target);
```

در نهایت هم یکبار داده های ورودی به صورت یکجا وارد شبکه شده و خروجی TARGET آنها براساس آخرین وزن آپدیت شده تعیین می شود. به عبارتی داده ها ارزیابی یا VALIDATION می شوند.

تابع test_mlp.m

```
function [output cc] = test_mlp(model, input, target)
```

ورودی این تابع مدل تشکیل شده توسط مرحله آموزش، داده های تست و کلاس انها است و خروجی این تـابع کلاس تخمین زده شده توسط تابع تست و ضرایب کواریانس می باشد.

```
    [ntest nOutLayer] = size(target);
```

تعیین تعداد داده های تست و تعداد نرون های لایه خروجی

```
     output = zeros(ntest, nOutLayer);
```

تشکیل اولیه ماتریس خروجی

```
for i = 1:ntest
        temp = input(i,:);
```

از این قسمت به بعد قصد داریم که برای هر داده ورودی تست، وزن های بدست امده در مرحله آموزش را اعمال کرده و خروجی لایه آخر را بدست بیاوریم.

```
for j = 1:length(model.weights)
temp = temp * model.weights{j} + model.biases{j};
temp = 1./(1+exp(-temp));
end
output(i,:) = temp;
```

دراین قسمت وزن ها را در هر لایه اعمال کرده و پس از عبور از تابع فعالساز، خروجی در مرحله آخر بدست می آید.

```
warning('off', 'all')
```

علامت warning در command window ایجاد نمی شود

```
cc = corrcoef(target(:), output(:));
if(numel(cc)>1)
        cc = cc(2,1);
end
```

تشکیل ضرایب کوواریانس

مسئله دوم:

دراین مسئله قصد داریم که دو حرف الفبا را طبقه بندی کنیم. دراین مسئله توابـع train_mlp و test_mlp همان توابع در مسئله ۱ می باشند . بنابراین فقط به توضیح تابع main می پردازیم

تابع main.m

```
clc;
clear all;
close all;
```

پاک کردن صفحه ،پاک کردن متغیرهای قبلی، بستن پنجره های قبلی

```
data1=zeros(1016,100);
```

دراین قسمت یک ماتریس اولیه تشکیل می دهیم که تعداد سطرهای آن بـه تعـداد تصـاویر موجـود در فولـدر آموزش مربوط به حرف A و تعداد ستون های آن ۱۰۰ می باشد.

```
for i=1:712
```

دراین قسمت یه حلقه for نوشته می شود که ۷۱۲ بار (تعداد داده های آموزش) تکرار می شود . دراین حلقه قصد داریم که ماتریس هر تصویر را به صورت سطری تبدیل کنیم.

```
img=imread(['A_char_train\img (',num2str(i),').png']);
```

خواندن تصویر ورودی i ام و ذخیره در img

```
img=double(imresize(img,[10,10])>240);
```

درایم قسمت تصاویر را به سایز ۱۰ در ۱۰ تبدیل می کنیم.

```
    data1(i,:)=reshape(img,1,100);
```

دراین قسمت ماتریس ۱۰ در ۱۰ را به صورت سطری ۱ در ۱۰۰ تبدیل می کنیم.

```
for i=713:1016
    img=imread(['A_char_test\img (',num2str(i-
712),').png']);
    img=double(imresize(img,[10,10])>240);
    data1(i,:)=reshape(img,1,100);

end
```

دراین قسمت مشابه مرحله قبل ، یک ماتریس تشکیل می شود که تصاویر موجود در فولدر تست حرف A به صورت سطری ذخیره شده اند.

```
data2=zeros(1016,100);
for i=1:712

    img=imread(['S_char_train\img (',num2str(i),').png']);
    img=double(imresize(img,[10,10])>240);
    data2(i,:)=reshape(img,1,100);

end
```

دراین قسمت مشابه مرحله قبل ، یک ماتریس تشکیل می شود که تصاویر موجود در فولدر آموزش حرف S به صورت سطری ذخیره شده اند.

```
for i=713:1016
    img=imread(['S_char_test\img (',num2str(i-
712),').png']);
    img=double(imresize(img,[10,10])>240);
    data2(i,:)=reshape(img,1,100);

end
```

دراین قسمت مشابه مرحله قبل ، یک ماتریس تشکیل می شود که تصاویر موجود در فولدر آموزش حرف S بـه صورت سطری ذخیره شده اند.

```
trp=floor(length(data1)*0.7);
```

تعیین تعداد داده های آموزش

```
training=[data1(1:trp,:);data2(1:trp,:)];
testing=[data1(trp+1:end,:);data2(trp+1:end,:)];
training_class=[zeros(trp,1);ones(trp,1)];
testing_class=[zeros(length(testing)/2,1);ones(length(testi
ng)/2,1)];
```

تعیین داده های آموزش و تست و کلاس های آموزش و تست

```
rnp=randperm(length(training));
training=training(rnp,:);
training_class=training_class(rnp,:);
```

تصادفی کردن ترتیب داده های آموزش . دراین قسمت ترتیب داده های و کلای های آنها به صورت رندم جابـه جا می شوند.

```
hidden_layers = [30 10];
iterations = 500;
learning_rate = 0.01;
momentum = 0.05;
```

تعیین پارامترهای آموزش شبکه عصبی

```
[model cc_train output_train] = train_mlp(training,
training_class, hidden_layers, iterations, learning_rate,
momentum);
```

آموزش شبکه عصبی

```
[output_test cc_test] = test_mlp(model, testing,
testing_class);
```

تست کردن شبکه بر اساس داده های تست

```
[jnk est_class]
=min([output_test';0.5*ones(1,length(output_test))]);
```

تعیین کلاس داده های تست. چون خروجی شبکه عصبی برای این تمرین باید بین ۰ و ۱ باشد، اگر این خروجی از ۰/۵ کمتر ، کلاس خروجی صفر و اگر بیشتر از ۰/۵ باشد ، کلاس خروجی۱ می شود.

```
percent_correct =100* sum(est_class'==testing_class+1)/610
```

تعیین درصد صحت تشخیص داده های تست

```
num1=input(' enter pic_number from A_char_test folder...');
num2=input(' enter pic_number from S_char_test folder...');
```

در این قسمت از کاربر میخواهیم که یکی از تصاویر مروبط به هر یک از کاراکترها را انتخاب کند تا نتیجه تشخیص شبکه عصبی برای آنها مشخص شود.

```
p1=est_class(num1);
p2=est_class(num2+305);
```

دراین قسمت نتیجه تشخیص شبکه عصبی برای این دو تصویر تعیین می شود.

```
if(p1==1)
   sprintf(' pic number %d = A (corrected recognition)',
num1)
else
  sprintf(' pic number %d = S (wrong recognition) ', num1)

end
```

دراین قسمت نتیجه تشخیص شبکه عصبی برای تصویر اول در command window تعیین می شود.

```
if(p2==2)
   sprintf(' pic number %d = S (corrected recognition)',
num1)
else
  sprintf(' pic number %d = A (wrong recognition) ', num1)

end
```

دراین قسمت نتیجه تشخیص شبکه عصبی برای تصویر دوم در command window تعیین می شود.