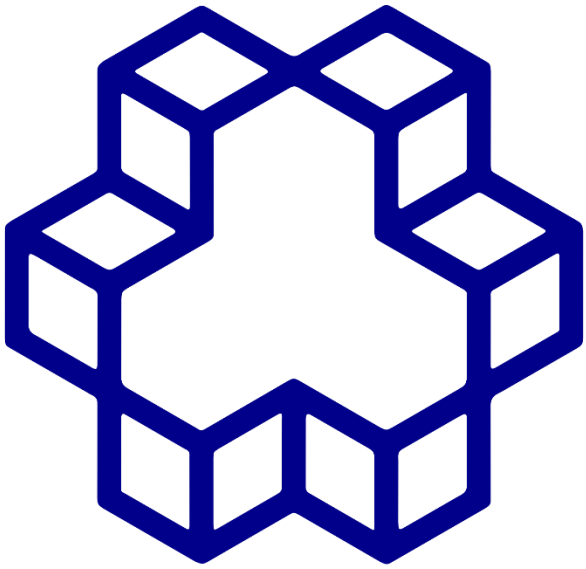


دانشگاه صنعتی خواجه نصیرالدین طوسی

سید عادل میرشرجی

مبانی هوش محاسباتی

پیاده سازی شبکه عصبی و آموزش به روش BGD

در این گزارش، نحوه پیاده سازی شبکه عصبی با یک لایه میانی و دو لایه میانی و آموزش آنها به روش BGD در Matlab توضیح داده شده و سپس نمودار خطای اعتبارسنجی و آموزش رسم شده و در آخر خطای نهایی هر یک از شبکه‌های عصبی بیان شده است و شبکه‌های عصبی یک و دو لایه میانی و روش BGD و SGD مقایسه شده اند.

هدف تخمین تابع f(x,y) است، در لایه میانی اول از تابع f (tansig) و در لایه میانی دوم از تابع g (logsig) استفاده میکنیم:

تابعی که می‌خواهیم تخمین بزنیم:

$$f(x,y) = (x^2 + y^2)humps(x)$$

توابع فعالساز لایه اول و دوم میانی به ترتیب از بالا به پایین:

$$f(x) = tansig(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

$$\frac{d(f(x))}{dx} = \frac{4e^{-2x}}{(1 + e^{-2x})^2}$$

$$g(x) = logsig(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{d(g(x))}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2}$$

در روش BGD در هر epoch برای تمام داده آموزشی feedforward انجام می‌شود و درهمین حین، هر بار مقدار خطای لایه آخر محاسبه و با حاصل جمع خطاهای iteration قبلی جمع می‌شود و در نهایت پس از آخرین iteration در feedforward، این حاصل جمع خطاهای لایه آخر

تقسیم بر تعداد داده‌های آموزشی می‌شود یا به عبارت دیگر میانگین می‌گیریم و سپس با این مقدار بدست آمده یکبار backpropagate می‌کنیم، بنابراین در هر epoch یکبار backpropagate می‌کنیم و وزنها و بایاسهای لایه‌ها بروزرسانی می‌شوند که این روش BGD است.

به طور مثال در این پیاده سازی تعداد ۷۰ داده آموزشی داریم، در هر epoch ۷۰ iteration در feedforward داریم، در هر iteration مقدار خطای لایه آخر محاسبه و با حاصل جمع خطاهای لایه آخر در iteration های قبلی جمع می‌شود. بعد از ۷۰امین iteration در feedforward میانگین این خطاها که آنها را باهم جمع کردیم گرفته میشود، یعنی حاصل جمع آنها تقسیم بر می‌شود۷۰ و سپس با توجه به خطای نهایی برای لایه آخر بدست آمده از این میانگین، یکبار backpropagate کرده و وزن‌ها و بایاس‌های تمام لایه ها را با روش دلتا بروزرسانی می‌کنیم.

الف)شبکه عصبی با یک لایه میانی:

پاک کردن حافظه و command window و بستن تمام plot ها:

```
clear;
clc;
close all;

```

حال داده‌ها را مقدار دهی می‌کنیم، به این صورت که از صفر تا یک با گام های ۰.۰۱ یک ماتریس ورودی می‌سازیم، یعنی کل داده‌های ورودی یک ماتریس ۲ در ۱۰۰ است که هر element این ماتریس به صورت رندوم مقداردهی شده است. و سپس ۷۰ داده اول برای آموزش،۲۰ داده بعدی برای اعتبارسنجی و ۱۰ داده آخر برای تست درنظر گرفته میشوند.

```
%-------------------------Initializing-----------------------------------------%
X=[0:0.01:1; 0:0.01:1];

Xin = X(:, 1:70);
D_Learn = (Xin(1,:).^ 2 + Xin(2,:) .^ 2) .* humps(Xin(1,:));
D_Learn = D_Learn / max(D_Learn(:));

X_valid = X(:, 71:90);
D_valid = (X_valid(1,:).^ 2 + X_valid(2,:) .^ 2) .* humps(X_valid(1,:));
D_valid = D_valid / max(D_valid(:));

X_test = X(:, 91:101);
D_test = (X_test(1,:).^ 2 + X_test(2,:) .^ 2) .* humps(X_test(1,:));
D_test = D_test/max(D_test(:));


```

حال به آموزش شبکه عصبی با روابط پیشرو(feedforward) و پس‌انتشارخطا بر اساس BGD می‌پردازیم. ابتدا مقادیر وزنها و بایاس ها در هر لایه را به صورت رندوم تعیین می کنیم و سپس شمارنده epoch و مقدار اولیه خطای اعتبارسنجی برای استفاده از while تعریف می کنیم.

```
%-------------------------Feedforward and Backpropagation----------------------------

eta=0.01;
epsilon=0.1;
%-------------------------16 neuron in 1st hidden layer
W1=rands(16,2);
Wb1=rands(16,1);
%-------------------------1 neuron in ouput layer
W2=rands(1,16);
Wb2=rands(1);
%-------------------------Kth epoch
k=0;
%-------------------------Validation error initializing
E_valid1=100;
```

حال روابط feedforward برای هر لایه به صورت زیر:

به این صورت که از لایه اول شروع کرده و net لایه را به صورت ماتریسی حساب می کنیم که برابر است با مجموع حاصل جمع ماتریس بایاس های این لایه با مجموع حاصل ضرب ماتریس وزنها در ورودی که در لایه اول ورودی داده های آموزشی هستند. سپس مقدار خروجی این لایه با محاسبه tansig، net لایه اول محاسبه می‌شود و در آخر نیز مقدار مشتق خروجی لایه اول برای استفاده در ادامه محاسبات نوشته شده است. سپس به سراغ لایه بعدی یعنی لایه خروجی رفته و net این لایه را بر

اساس خروجی لایه اول محاسبه کرده، خروجی لایه آخر به خاطر خطی بودن تابع فعالساز آن برابر خود net است و پرواضح است که مشتق آن برابر ۱ است و در آخر نیز مقدار خطای لایه آخر محاسبه می‌شود.

چون روش BGD است ابتدا برای تمام داده آموزشی (۷۰تا) feedforward را انجام می‌دهیم و در هر iteration مقدار خطای لایه آخر (e) با حاصل جمع خطاهای لایه آخر iteration های قبلی(ee) جمع میشود.

```
while(k < 100 && E_valid1 > epsilon)
    k = k+1;
    ee(k) = 0;
    for m = 1:70
        %-------------------------Feedforward
        %-------------------------1st layer
        net1 = W1 * Xin(:, m) + Wb1;
        O1 = tansig(net1);
        diff_O1 = 4 * exp(-2 .* net1) ./ (1 + exp(-2 .* net1)) .^2;
        %-------------------------output layer
        net2 = W2 * O1 + Wb2;
        O2 = net2;
        diff_O2 = 1;
        %-------------------------Calculating output layer error and
        %storing it
        e = D_Learn(:, m) - O2;
        ee(k) = ee(k) + e;
    end

```

حال روابط backpropagation برای هر لایه به صورت زیر:

چون از روش BGD استفاده میکنیم بعد از feedforward برای تمام داده‌های آموزشی با میانگین حاصل جمع خطاهای لایه آخر در هر iteration به backpropagate می‌پردازیم و مقادیر وزن‌ها و بایاس‌های تمام لایه ها را به روز رسانی می کنیم.

```
%-------------------------Backpropagation
%-------------------------output layer
W2 = W2 + eta * ee(k)/70 * O1';
Wb2 = Wb2 + eta * ee(k)/70;
%-------------------------1st layer
e1 = W2' * ee(k)/70;
delta1 = e1 .* diff_O1;
W1 = W1 + eta * delta1 * (Xin(:, m))';
Wb1 = Wb1 + eta * delta1;
```

قابل ذکر است که دو شرط توقف در نظر گرفته شده است: یکی اینک اگر به epoch ۱۰۰ برسیم یا اینک خطا اعتبار سنجی کمتر از epsilon که در ابتدا تعریف کردیم بشود که این دو شرط در while شروع فرآیند یادگیری قرار داده شده‌اند.

حال خطای اعتبار سنجی را بصورت زیر حساب می کنیم:

```
%-------------------------Validation error
WB1=ones(16,101);
for p=1:16
WB1(p,:)=Wb1(p,1)*ones(1,101);
end

net1_valid = W1 * X_valid + WB1(:,1:20);
O1_valid = tansig(net1_valid);

WB2 = Wb2 * ones(1,20);
net2_valid = W2 * O1_valid + WB2(:, 1:20);
O2_valid = net2_valid;

e_valid = D_valid - O2_valid;
E_valid = 0.5 * trace(e_valid * e_valid');
E_v1(k)= E_valid;
```

حال خطای آموزش را بصورت زیر حساب می کنیم:

```
    %-------------------------Learning error
    net1_Learn = W1 * Xin + WB1(:,1:70);
    O1_Learn = tansig(net1_Learn);

    WB2 = Wb2 * ones(1,70);
    net2_Learn = W2 * O1_Learn + WB2(:,1:70);
    O2_Learn = net2_Learn;

    e_Learn = D_Learn-O2_Learn;
    E_Learn = 0.5 * trace(e_Learn * e_Learn');
    E_L(k) = E_Learn;
end

```

در نهایت نمودارهای خطای اعتبارسنجی و آموزش را به صورت زیر در MATLAB رسم می کنیم:

```
%-------------------------Plots-----------------------------------------%
p = length(E_L);
m = 1:1:p;
figure;
plot(m, E_L,'g');
hold on;
plot(m, E_v1,'b');
title('Error of Learning (green) and Evaluation 1 (blue) Using Tansig');
xlabel('epoch')
ylabel('Learning and Evaluation 1 Error');

%-------------------------Test------------------------------------------%
WB1 = ones(16,101);
for p=1:16
```

و مرحله آخر که تست است:

```
%-------------------------Test------------------------------------------%
WB1 = ones(16,101);
for p=1:16
    WB1(p,:)=Wb1(p,1)*ones(1,101);
end

net1_test=W1*X_test+WB1(:,1:11);
O1_test=tansig(net1_test);

WB2=Wb2*ones(1,11);
net2_test=W2*O1_test+WB2(:,1:11);
O2_test=net2_test;

e_test = D_test-O2_test;
E_test = 0.5*trace(e_test * e_test');
```

ب)شبکه عصبی با دو لایه میانی:

تمام مراحل و عملیات مانند قسمت الف است با این تفاوت که یک لایه میانی جدید اضافه شده است که شامل ۱۶ نرون است و تابع فعالساز آن g (logsig) است.

مانند قسمت الف:

BGD_one_Layer.m | BGD_two_Layer.m

```
clear;
clc;
close all;

%-------------------------Initializing-----------------------------------------%
X=[0:0.01:1; 0:0.01:1];

Xin = X(:, 1:70);
D_Learn = (Xin(1,:).^ 2 + Xin(2,:) .^ 2) .* humps(Xin(1,:));
D_Learn = D_Learn / max(D_Learn(:));

X_valid = X(:, 71:90);
D_valid = (X_valid(1,:).^ 2 + X_valid(2,:) .^ 2) .* humps(X_valid(1,:));
D_valid = D_valid / max(D_valid(:));

X_test = X(:, 91:101);
D_test = (X_test(1,:).^ 2 + X_test(2,:) .^ 2) .* humps(X_test(1,:));
D_test = D_test/max(D_test(:));

```

حال آموزش با یک لایه اضافی نسبت به قسمت الف:

```
%-------------------------Feedforward and Backpropagation----------------------------

eta=0.01;
epsilon=0.1;
%-------------------------16 neuron in 1st hidden layer
W1=rands(16,2);
Wb1=rands(16,1);
%-------------------------16 neuron in 2nd hidden layer
W2=rands(16,16);
Wb2=rands(16,1);
%-------------------------1 neuron in ouput layer
W3=rands(1,16);
Wb3=rands(1);
%-------------------------Kth epoch
k=0;
%-------------------------Validation error initializing
E_valid1=100;
```

حال feedforward با یک لایه اضافی نسبت به قسمت الف:

لایه دوم اضافه شده است که تابع فعالساز آن logsig است و ورودی آن O1 و سایر موارد همانطور که توضیح داده شد هستند.

```
%-------------------------Starting Learning Process
while(k < 100 && E_valid1 > epsilon)
    k = k+1;
    ee(k) = 0;
    for m=1:70
        %-------------------------Feedforward
        %-------------------------1st layer
        net1 = W1 * Xin(:, m) + Wb1;
        O1 = tansig(net1);
        diff_O1 = 4 * exp(-2 .* net1) ./ (1 + exp(-2 .* net1)) .^2;
        %-------------------------2nd layer
        net2 = W2 * O1 + Wb2;
        O2 = logsig(net2);
        diff_O2 = exp(-net2) ./ ((1 + exp(-net2))).^2;
        %-------------------------output layer
        net3 = W3 * O2 + Wb3;
        O3 = net3;
        diff_O3 = 1;
        %-------------------------Calculating output layer error and
        %storing it
        e = D_Learn(:, m) - O3;
        ee(k) = ee(k) + e;
    end
```

حال backpropagation با یک لایه اضافی نسبت به قسمت الف:

همه چیز مانند قسمت الف یعنی یک لایه میانی است با این تفاوت که دلتای لایه اول از دلتای لایه دوم و دلتای لایه دو از دلتای لایه آخر محاسبه میشود.

```
    %-------------------------Backpropagation
    %-------------------------output layer
    W3 = W3 + eta * ee(k)/70 * O3';
    Wb3 = Wb3 + eta * ee(k)/70;
    %-------------------------2nd layer
    e2 = W3' * ee(k)/70;
    delta2 = e2 .* diff_O2;
    W2 = W2 + eta * delta2 * O1';
    Wb2 = Wb2 + eta * delta2;
    %-------------------------1st layer
    e1 = W2' * e2;
    delta1 = e1 .* diff_O1;
    W1 = W1 + eta * delta1 * (Xin(:, m))';
    Wb1 = Wb1 + eta *delta1;

```

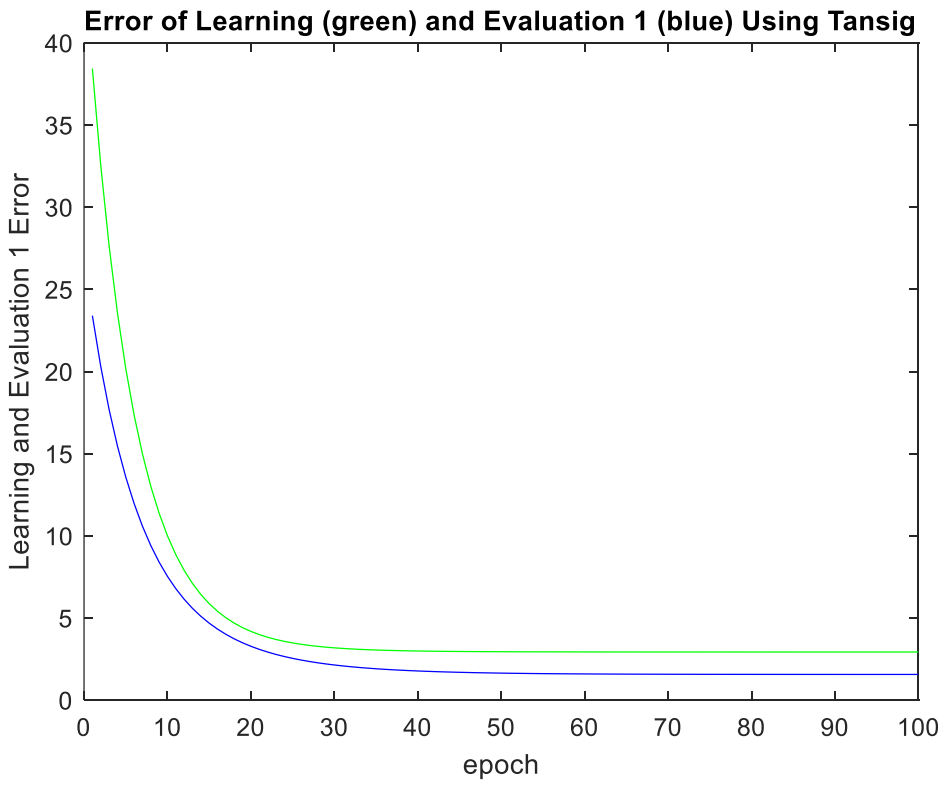
و در آخر هم محاسبه خطاهای اعتبارسنجی و یادگیری و رسم نمودار آنها و تست به صورت زیر:

```
    %-------------------------Validation error
    WB1=ones(16,101);
    WB2=ones(16,101);
    for p=1:16
    WB1(p,:)=Wb1(p,1)*ones(1,101);
    WB2(p,:)=Wb2(p,1)*ones(1,101);
    end

    net1_valid = W1 * X_valid + WB1(:,1:20);
    O1_valid = tansig(net1_valid);

    net2_valid = W2 * O1_valid + WB2(:,1:20);
    O2_valid = logsig(net2_valid);

    WB3 = Wb3 * ones(1,20);
    net3_valid = W3 * O2_valid + WB3(:,1:20);
    O3_valid = net3_valid;

    e_valid = D_valid - O3_valid;
    E_valid = 0.5 * trace(e_valid * e_valid');
    E_v1(k)= E_valid;

     %-------------------------Learning error
     net1_Learn = W1 * Xin + WB1(:,1:70);
     O1_Learn = tansig(net1_Learn);

     net2_Learn = W2 * O1_Learn + WB2(:,1:70);
     O2_Learn = logsig(net2_Learn);

     WB3 = Wb3 * ones(1,70);
     net3_Learn = W3 * O2_Learn + WB3(:,1:70);
     O3_Learn = net3_Learn;

     e_Learn = D_Learn-O3_Learn;
     E_Learn = 0.5 * trace(e_Learn * e_Learn');
     E_L(k) = E_Learn;
end
%-------------------------Plots-----------------------------------------%
p = length(E_L);
m = 1:1:p;
figure;
plot(m, E_L,'g');
hold on;
plot(m, E_v1,'b');
title('Error of Learning (green) and Evaluation 1 (blue) Using Tansig and Logsig');
xlabel('epoch')
ylabel('Learning and Evaluation 1 Error');
```

```
126  %-------------------------Test-----------------------------------------%
127  WB1 = ones(16,101);
128  WB2 = ones(16,101);
129  for p=1:16
130      WB1(p,:)=Wb1(p,1)*ones(1,101);
131      WB2(p,:)=Wb2(p,1)*ones(1,101);
132  end
133
134  net1_test=W1*X_test+WB1(:,1:11);
135  O1_test=tansig(net1_test);
136
137  net2_test = W2 * O1_test + WB2(:,1:11);
138  O2_test = logsig(net2_test);
139
140  WB3=Wb3*ones(1,11);
141  net3_test=W3*O2_test+WB3(:,1:11);
142  O3_test=net3_test;
143
144  e_test = D_test-O3_test;
145  E test = 0.5*trace(e test * e test');
```

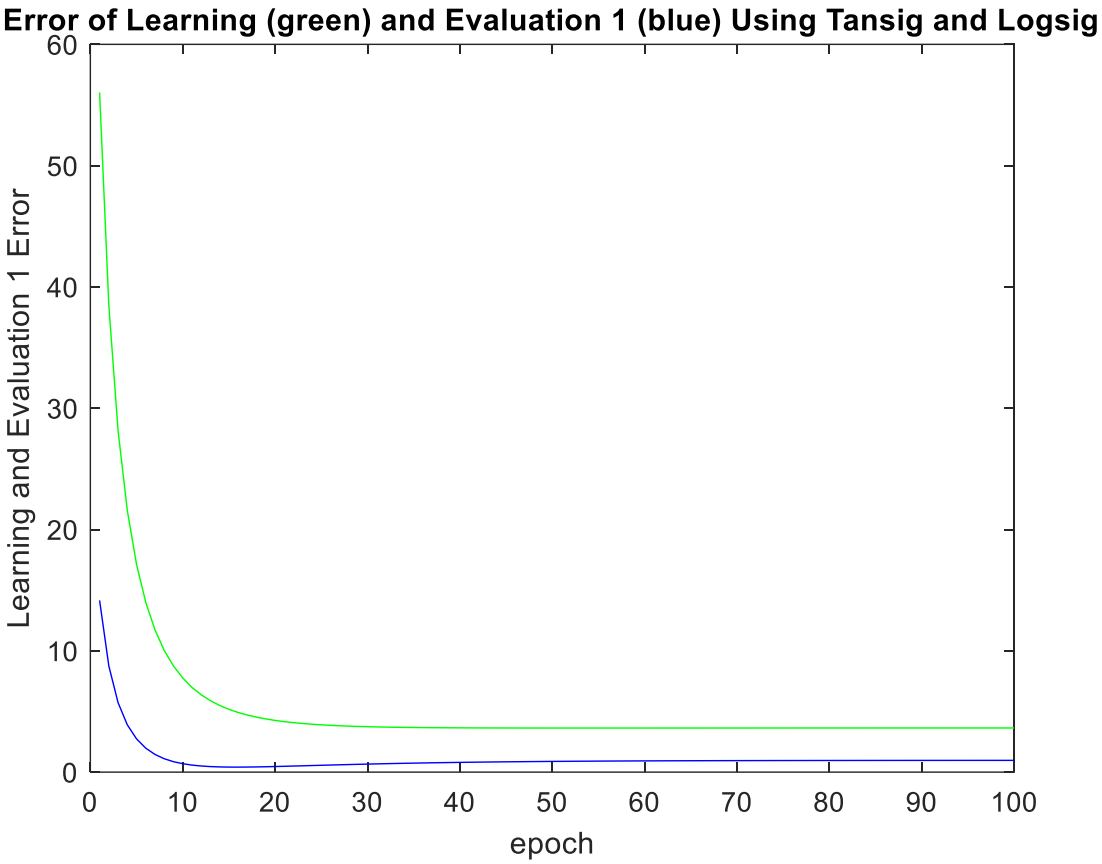
ج و د)نمایش نمودار و مقدار خطای نهایی قسمت الف و ب:

الف)



خطای نهایی = ۰.۹۶۲۷

ب)



خطای نهایی = ۰.۲۰۹۱

د) مقایسه و نتیجه‌گیری

با مقایسه شبه عصبی با یک لایه میانی و دو لایه میانی متوجه می‌شویم که شبکه عصبی با دولایه میانی پیچده‌تر است و آموزش آن هم از نظر پیچیدگی محاسبات (Computational Complexity) و هم از نظر پیچیدگی محاسباتی (Time Complexity) هزینه بیشتری دارد، اما دقت شبکه عصبی با دولایه میانی از شبکه عصبی با یک لایه میانی بیشتر است، اما خب این امر می‌تواند باعث overfitting در شبکه عصبی با لایه بیشتر نیز می‌تواند بشود. حال با مقایسه روش BGD و SGD در می‌یابیم که

سرعت روش BGD بیشتر از SGD است اما دقت کمتری در تخمین یک تابع یکسان با تعداد epoch یکسان دارد.