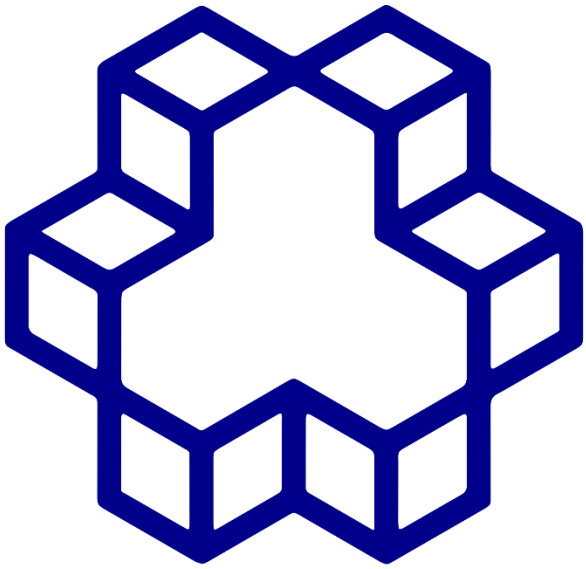


دانشگاه صنعتی خواجه نصیرالدین طوسی

سید عادل میرشرجی

مبانی هوش محاسباتی

پیاده سازی شبکه عصبی و آموزش به روش SGD

در این گزارش، نحوه پیاده سازی شبکه عصبی با یک لایه و دو لایه میانی و آموزش آنها به روش SGD 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.

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:

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

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

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

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

الف)شبکه عصبی با یک لایه میانی:

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

```
SGD_one_Layer.m  SGD_two_Layer.m
1  clear;
2  clc;
3  close all;
4
```

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 SGD می‌پردازیم. ابتدا مقادیر وزنها و بایاس ها در هر لایه را به صورت رندوم تعیین می‌کنیم و سپس شمارنده epoch و مقدار اولیه خطای اعتبارسنجی برای استفاده از while تعریف می‌کنیم.

```
%-------------------------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 iteration
k=0;
%-------------------------Validation error initializing
E_valid1=100;
%-------------------------Starting Learning Process
```

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.

```
%-------------------------Starting Learning Process
while(k < 100 && E_valid1 > epsilon)
    k=k+1;
    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
        e = D_Learn(:, m) - O2;
        ee(k,m) = e;
```

حال روابط backpropagation برای هر لایه به صورت زیر:

چون از روش SGD استفاده میکنیم در هر iteration با محاسبه دلتا وزنها و بایاس‌های هر لایه آپدیت می‌شوند. قابل ذکر است از لایه آخر شروع کرده و بر اساس خطای بدست آمده دلتای این لایه را حساب می‌کنیم و سپس وزنها و بایاس این لایه را آپدیت می‌کنیم و به سراغ لایه بعد رفته و مقدار خطای این لایه را با استفاده از دلتای لایه بعدی که از آن آمدیم حساب کرده و دلتای این لایه را بر اساس آن و مشتق تابع فعالساز این لایه حساب می‌کنیم و وزنها و بایاس‌های این لایه را نیز آپدیت می‌کنیم.

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

قابل ذکر است که دو شرط توقف در نظر گرفته شده است: یکی اینک اگر به ۱۰۰ epoch 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 می‌کنیم:

```
%-------------------------Validation error
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');
```

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:

```
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.

```
E_valid1=100;
%-------------------------Starting Learning Process
while(k < 100 && E_valid1 > epsilon)
    k=k+1;
    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
        e = D_Learn(:, m) - O3;
        ee(k,m)=e;
```

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.

```
58          %-------------------------Backpropagation
59          %-------------------------output layer
60          W3 = W3 + eta * e * O3';
61          Wb3 = Wb3 + eta * e;
62          %-------------------------2nd layer
63          e2 = W3' * e;
64          delta2 = e2 .* diff_O2;
65          W2 = W2 + eta * delta2 * O1';
66          Wb2 = Wb2 + eta * delta2;
67          %-------------------------1st layer
68          e1 = W2' * e2;
69          delta1 = e1 .* diff_O1;
70          W1 = W1 + eta * delta1 * (Xin(:, m))';
71          Wb1 = Wb1 + eta *delta1;
72      end
```

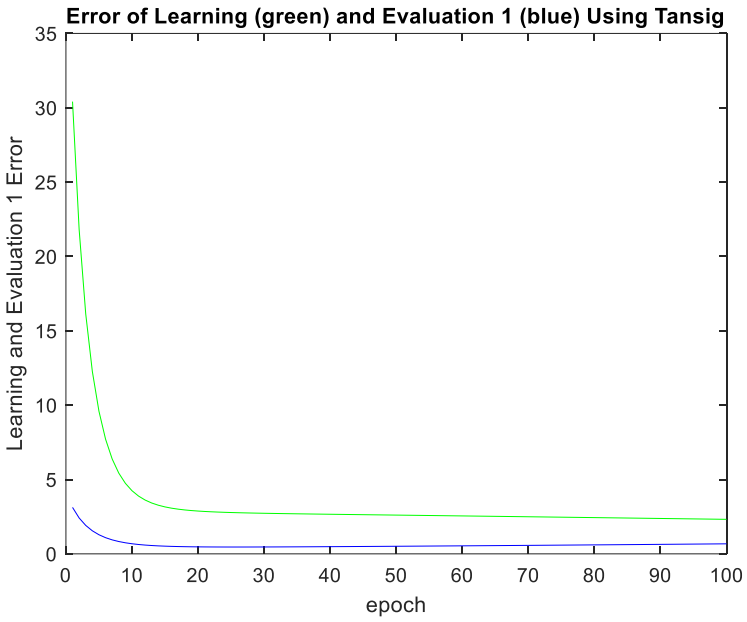
و در آخر هم محاسبه خطاهای اعتبارسنجی و یادگیری و رسم نمودار آنها و تست به صورت زیر:

```
80      %-------------------------Validation error
81      net1_valid = W1 * X_valid + WB1(:,1:20);
82      O1_valid = tansig(net1_valid);
83
84      net2_valid = W2 * O1_valid + WB2(:,1:20);
85      O2_valid = logsig(net2_valid);
86
87      WB3 = Wb3 * ones(1,20);
88      net3_valid = W3 * O2_valid + WB3(:,1:20);
89      O3_valid = net3_valid;
90
91      e_valid = D_valid - O3_valid;
92      E_valid = 0.5 * trace(e_valid * e_valid');
93      E_v1(k)= E_valid;
94
95      %-------------------------Learning error
96      net1_Learn = W1 * Xin + WB1(:,1:70);
97      O1_Learn = tansig(net1_Learn);
98
99      net2_Learn = W2 * O1_Learn + WB2(:,1:70);
100     O2_Learn = logsig(net2_Learn);
101
102     WB3 = Wb3 * ones(1,70);
103     net3_Learn = W3 * O2_Learn + WB3(:,1:70);
104     O3_Learn = net3_Learn;
105
106     e_Learn = D_Learn-O3_Learn;
107     E_Learn = 0.5 * trace(e_Learn * e_Learn');
108     E_L(k) = E_Learn;
109 end
110
```

```
%-------------------------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');

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

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

WB3=Wb3*ones(1,11);
net3_test=W3*O2_test+WB3(:,1:11);
O3_test=net3_test;

e_test = D_test-O3_test;
E_test = 0.5*trace(e_test * e_test');
```

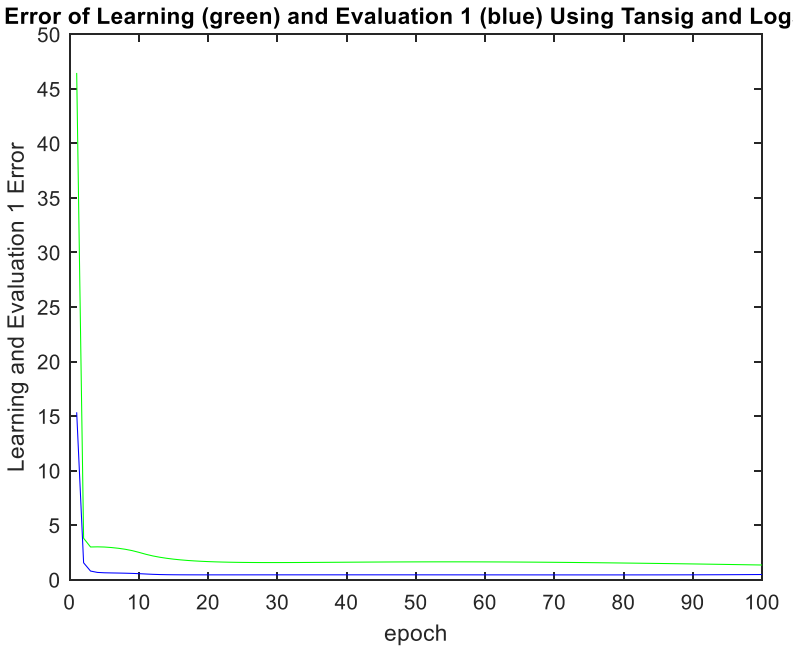
ج و د)نمایش نمودار و مقدار خطای نهایی قسمت الف و ب:

الف)



خطای نهایی = ۰.۰۸۴

ب)



خطای نهایی = ۰.۰۱۲۱