Xihao Zhu COMP502 HW4

P1.

Here is my variable test table

learn parameter	slope parameter	#of hidden layer	momentum term	epoch size	result	
0.05	1	10	0.8	200	0.04	No1 best
0.05	1	15	0.8	200	0.06	No2 best
0.05	1	5	0.8	200	0.08	not that good…
0.07	1	10	0.8	200	0.05(but fluctuates a lot)	No3 best
0.05	1	10	0.08	1000	0.17	bad.
0.05	1	10	0.08	800	0.12	bad

3 best parameter results No1, No2, No3:

No1:

(a)

For own parameters, I use 0.05 as learn parameter, 1 as slope parameter, 10 hidden layers, 0.8 as momentum term constant, 200 as epoch size. Stopping criteria is “stop when 50,000 learning steps are performed”.

Initial weights are:

```
w =

   -0.050000        0.070000      -0.040000        0.040000        0.040000      -0.050000
0.020000      0.010000    -0.020000
    0.080000        0.050000      -0.010000      -0.060000      -0.070000      -0.010000
-0.030000      0.040000    -0.040000

v =

   -0.020000
   -0.060000
    0.040000
    0.090000
    0.050000
   -0.050000
    0.090000
    0.050000
   -0.070000
```

-0.030000

(b)

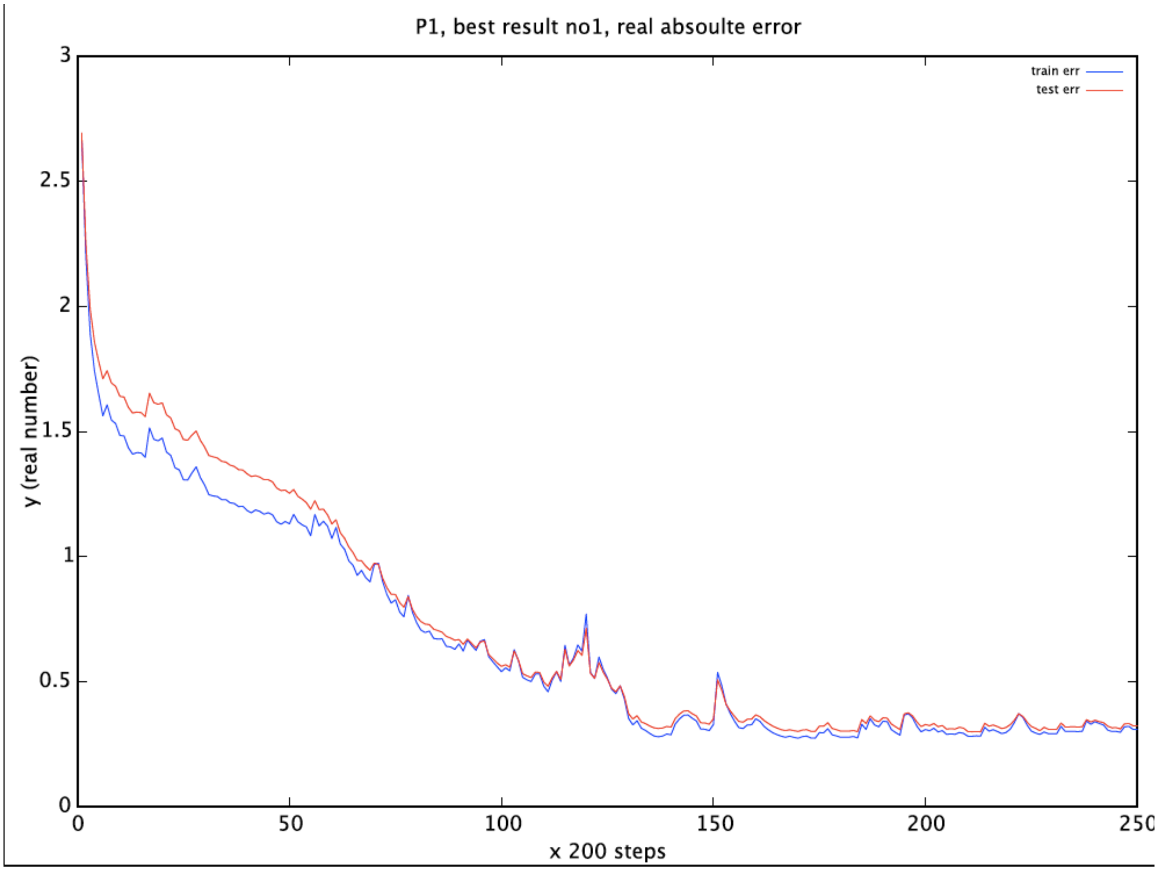
Learn step	Input vec	Desired	Actual	ek
5000	0.31598	-0.90834	-0.63253	-0.27581
10000	0.05245	-0.83657	-0.64877	-0.18781
15000	-0.3855	-0.60019	-0.46913	-0.13105
20000	0.26646	-0.89697	-0.82332	-0.073653
25000	-0.11237	-0.77221	-0.64679	-0.12543
30000	-0.29141	-0.67257	-0.58968	-0.082884
35000	-0.53822	-0.43109	-0.34116	-0.089926
40000	1 0.12608	-0.85973	-0.90121	0.04148
45000	1 -0.59081	-0.34938	-0.17634	-0.17304
50000	1 0.42721	-0.9312	-0.97158	0.040372

(c)

The error measurement is like this:

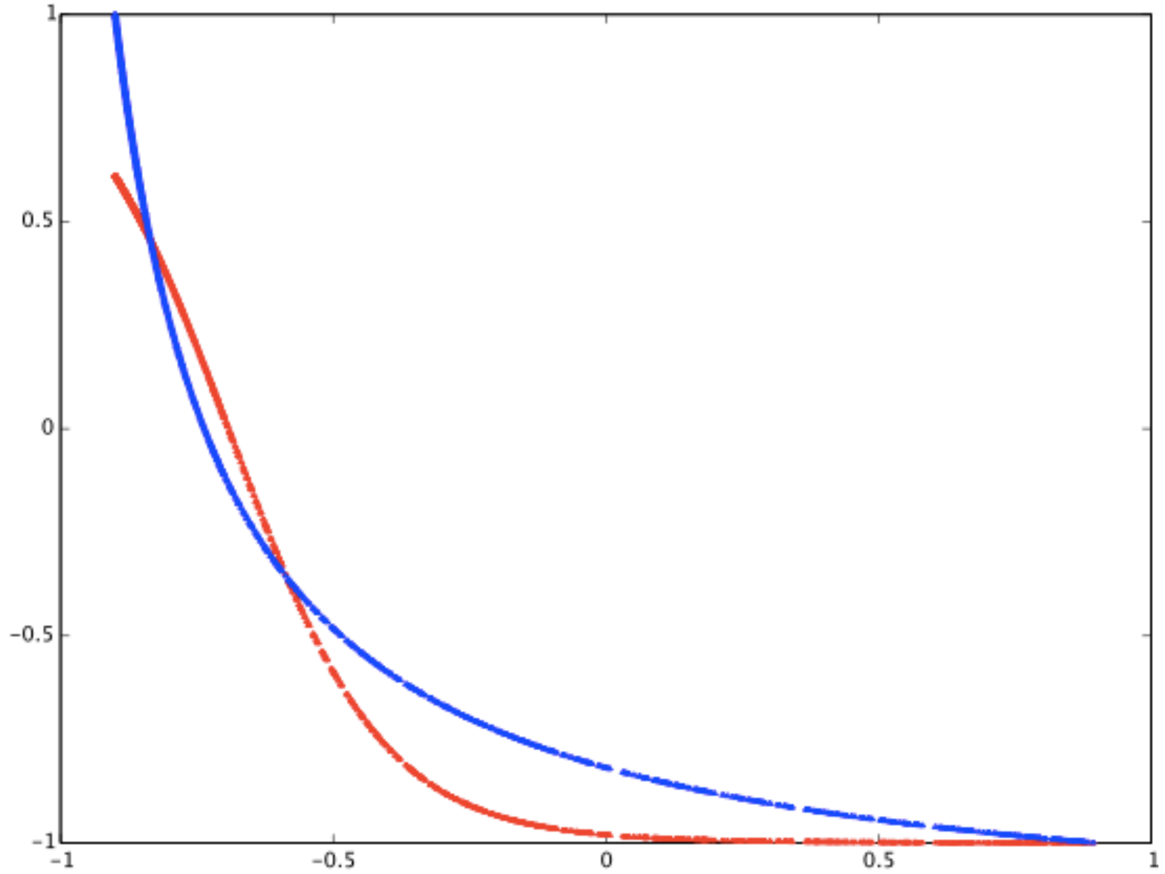
I use absolute error. The formula is to get sum of absolute value of every network's output(200 samples for train data and 100 for test data) minus desired output. Then scaling back the sum by multiplying by 4.5, so that the error is real error treating y as range of [1, 10].

Here is the plot



both the test error and train error are approaching 0.4. Therefore Error rate is about 0.4/(10-1)=4.44%

desired output VS calculated output is
results are not scaling back yet, but shapes are similar~
blue line is desired output, while red line is calculated output



No2.
(a)

For own parameters, I use 0.05 as learn parameter, 1 as slope parameter, 15 hidden layers, 0.8 as momentum term constant, 200 as epoch size. Stopping criteria is “stop when 50,000 learning steps are performed”.
Initial weights are:

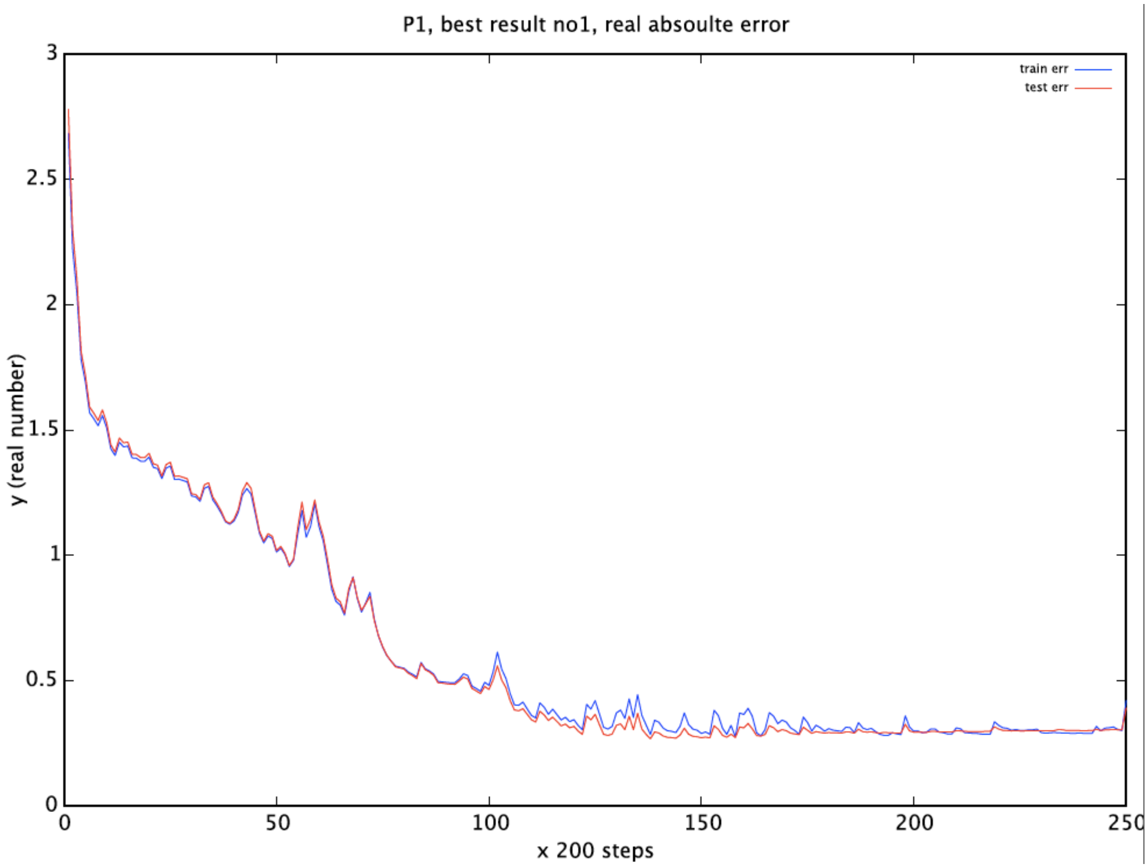
```
w =

  Columns 1 through 9:

   -0.0400000   -0.0900000    0.0100000   -0.0100000   -0.0200000   -0.0500000
-0.0200000    0.0500000    0.0400000
   -0.0800000          0.0400000      -0.0800000      -0.0200000          0.0300000
-0.0400000    0.0900000    0.0100000   -0.0050000

  Columns 10 through 14:

   -0.0500000    0.0300000    0.1000000    0.0900000    0.0900000
   -0.0300000    0.0100000   -0.0300000   -0.0300000   -0.0500000

v =

   -0.010000
    0.060000
    0.070000
    0.080000
   -0.080000
    0.040000
   -0.090000
    0.020000
   -0.090000
    0.090000
    0.030000
   -0.040000
    0.020000
    0.030000
   -0.030000
```

(b)

Learn step   Input vec   Desired   Actual   ek

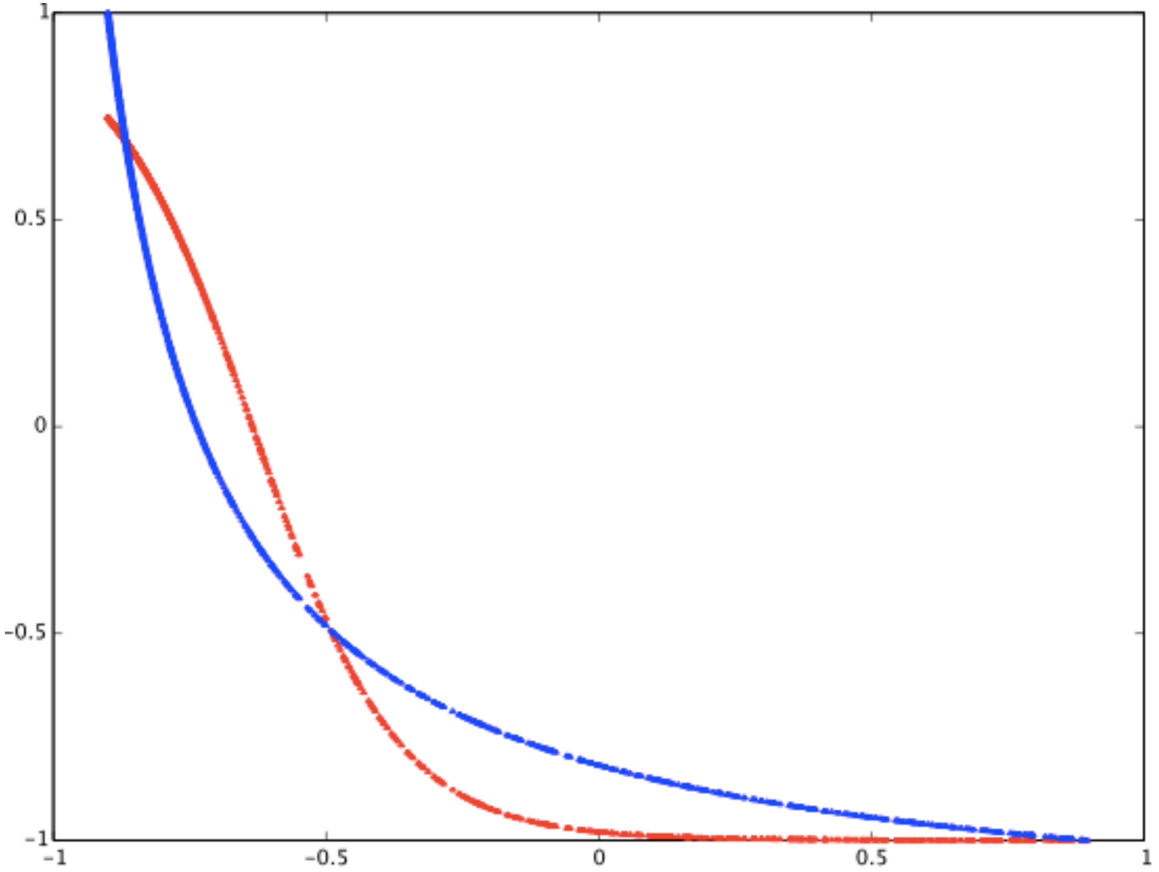
55000	-0.33059	-0.64458	-0.56816	-0.076426
10000	0.40975	-0.92784	-0.70333	-0.22451
15000	0.26529	-0.89669	-0.79353	-0.10316
20000	-0.44989	-0.53858	-0.38817	-0.15041
25000	-0.5593	-0.40024	-0.23664	-0.16359
30000	-0.3988	-0.58839	-0.48809	-0.10029
35000	-0.18729	-0.73527	-0.70888	-0.026388
40000	0.83638	-0.9927	-0.99027	-0.0024259
45000	-0.068562	-0.79132	-0.84028	0.048952
50000	0.78385	-0.9863	-0.99094	0.0046405

(c)
The error measurement is like this:
I use absolute error. The formula is to get sum of absolute value of every network's output(200 samples for train data and 100 for test data) minus desired output. Then scaling back the sum by multiplying by 4.5, so that the error is real error treating y as range of [1, 10].
(Title wrong, it should be best result No2)



both the test error and train error are approaching 0.4. Therefore Error rate is about 0.4/(10-1)=4.44%. The test error and train error are more similar than No1. This is interesing~

desired output VS calculated output is
results are not scaling back yet, but shapes are similar~
blue line is desired output, while red line is calculated output



No3.

(a)

For own parameters, I use 0.05 as learn parameter, 1 as slope parameter, 10 hidden layers, 0.8 as momentum term constant, 200 as epoch size. Stopping criteria is “stop when 50,000 learning steps are performed”.

Initial weights are:

```
w =

    0.0050000    -0.0100000     0.0100000     0.0500000    -0.0400000
0.0800000   -0.0100000    0.0800000    0.0050000
    0.1000000    -0.0900000     0.0100000     0.0300000     0.0100000
0.0800000   -0.0200000    0.0100000    0.0700000
```

```
v =

   -0.0100000
     0.1000000
   -0.0900000
     0.0100000
   -0.0400000
   -0.0900000
     0.0800000
     0.0500000
     0.0800000
   -0.0050000
```

(b)

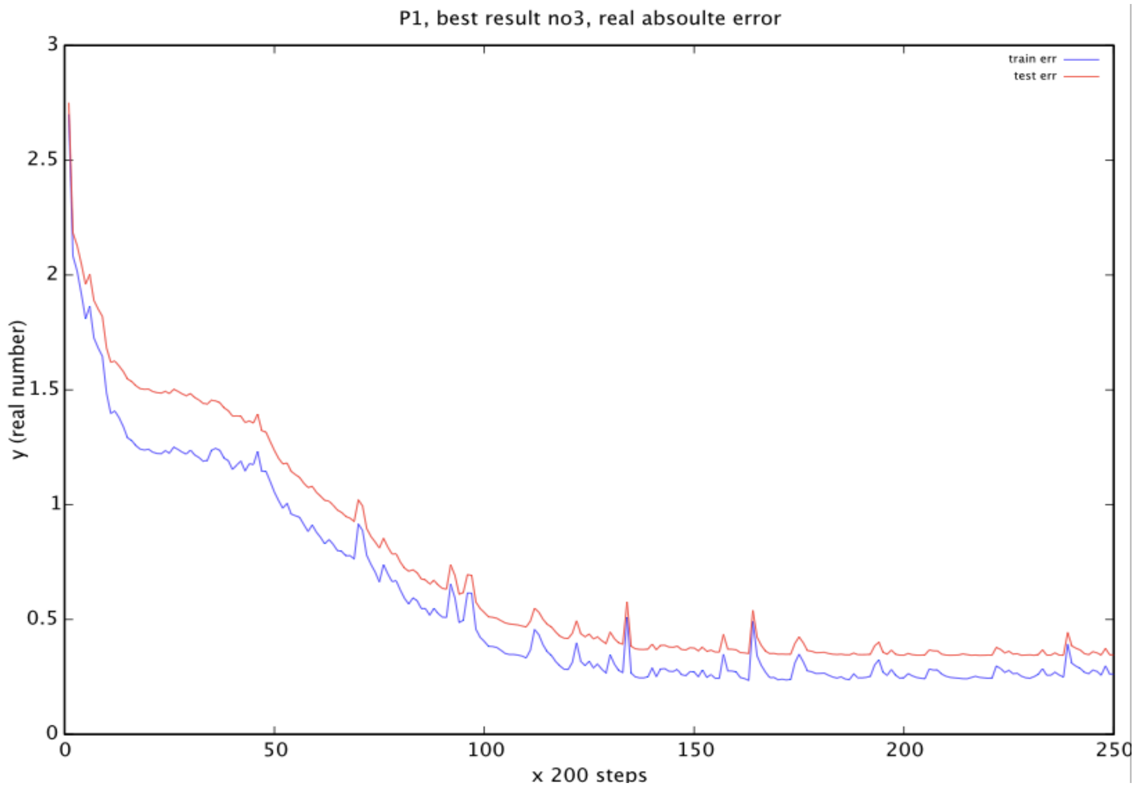
Learn step	Input vec		Desired	Actual	ek
5000	1	0.7139	-0.9772	-0.69972	-0.27748
10000	1	0.013675	-0.82314	-0.60667	-0.21647
15000	1	-0.0040455	-0.81669	-0.67072	-0.14597
20000	1	0.44367	-0.93431	-0.89957	-0.034738
25000	1	-0.76134	0.090151	-0.059618	0.14977
30000	1	0.11545	-0.85656	-0.86864	0.01208
35000	1	0.072002	-0.843	-0.86527	0.022265
40000	1	0.7565	-0.98282	-0.98465	0.0018287
45000	1	-0.09644	-0.77935	-0.8107	0.031346
50000	1	0.88091	-0.99786	-0.99135	-0.0065092
50000	1	0.42721	-0.9312	-0.97158	0.040372

(c)

The error measurement is like this:

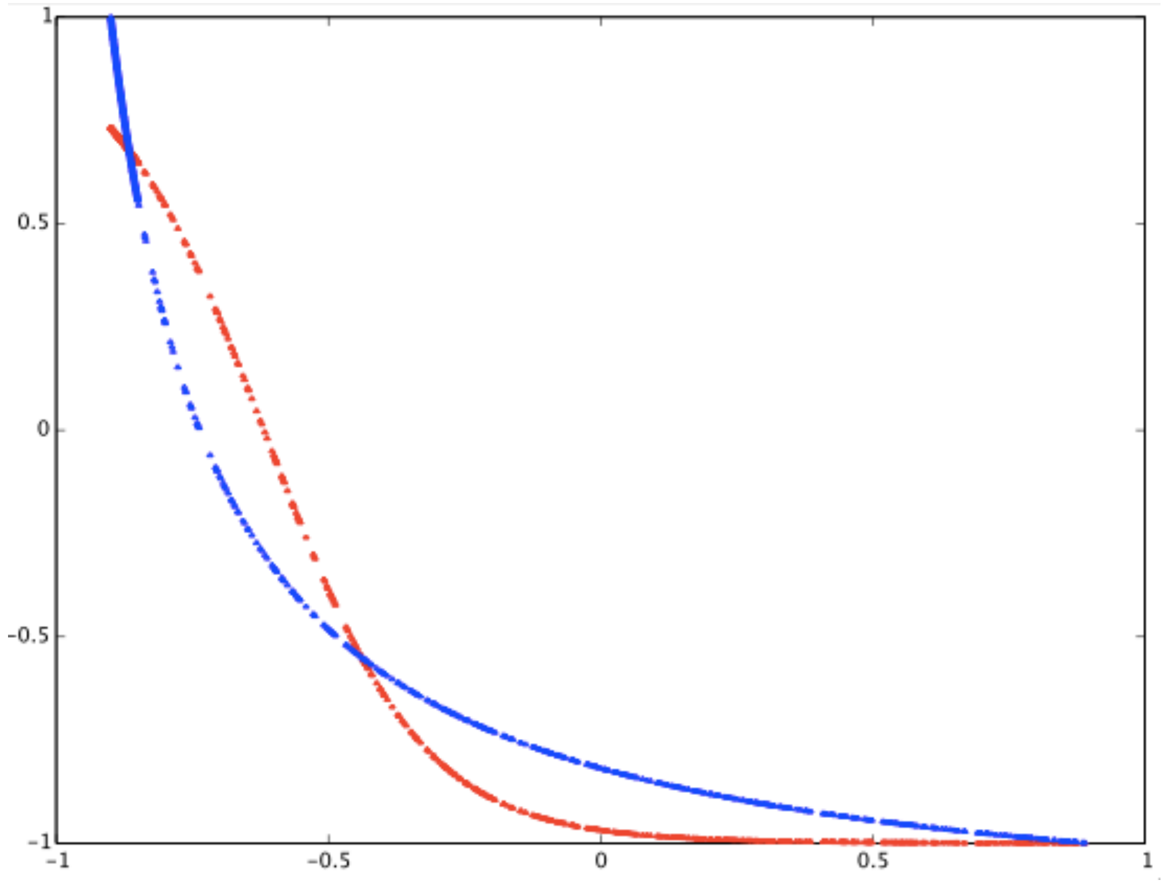
I use absolute error. The formula is to get sum of absolute value of every network's output(200 samples for train data and 100 for test data) minus desired output. Then scaling back the sum by multiplying by 4.5, so that the error is real error treating y as range of [1, 10].

Here is the plot



both the test error and train error are approaching 0.4. Therefore Error rate is about 0./(10-1)=5.56%

desired output VS calculated output is
results are not scaling back yet, but shapes are similar~
blue line is desired output, while red line is calculated output



P2.

(a)

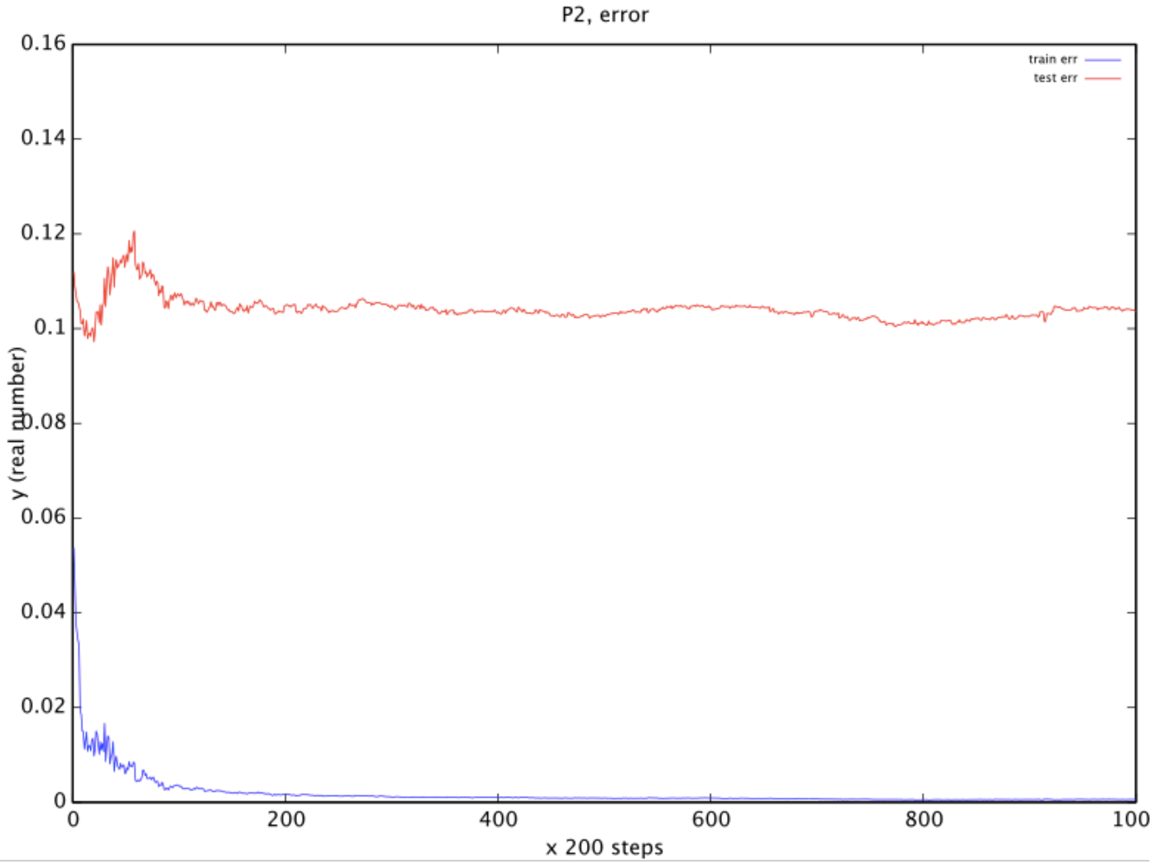
For own parameters, I use 0.05 as learn parameter, 1 as slope parameter, 10 hidden layers, 0.9 as momentum term constant, 200 as epoch size. Stopping criteria is “stop when 200,000 learning steps are performed”.

Initialized weights are:

```
w =

  Columns 1 through 6:

     0.0200000       -0.0300000          0.1000000       -0.0300000            0.0700000
-0.0700000
     0.0050000       -0.0600000          0.0300000          0.0700000       -0.0900000
-0.0900000
   -0.0900000       -0.0900000          0.0300000          0.0400000            0.0800000
-0.0400000
```

```
    0.0200000    -0.0700000      0.0700000    -0.0600000    -0.0800000
-0.0400000
   -0.0600000      0.0600000    -0.0400000    -0.0100000    -0.0700000
0.0050000

  Columns 7 through 9:

    0.0600000    0.0600000    0.1000000
    0.0500000   -0.0700000    0.0800000
   -0.0400000   -0.0400000   -0.0100000
    0.0200000    0.0100000    0.0900000
    0.0200000    0.0800000    0.0600000

v =

    0.090000    0.070000    0.080000
   -0.020000    0.100000    0.080000
    0.020000    0.080000   -0.040000
   -0.030000   -0.040000    0.020000
    0.060000    0.030000   -0.010000
   -0.070000    0.070000    0.100000
   -0.010000   -0.080000   -0.090000
   -0.050000   -0.050000   -0.090000
   -0.070000    0.060000   -0.070000
    0.020000   -0.050000    0.050000
```

(c)
The error measurement is like this:
I use absolute error. The formula is to get sum of absolute value of every network's output(200 samples for train data and 100 for test data) minus desired output.
Here is the plot(error without scaling back). Moreover, I changed output 0s all to -1.
So output [0 0 1] now is [-1 -1 1]



train data error is pretty small, nearly 0.001, but test data error is large, about 0.1, out of 2 total~which is about 5% error rate.

when I write my own program to use sign function(where f(x)=1 if x>0 and f(x)=-1if x<0) for network outputs, the result has no error with both test and train data. So the classification is correct after 200000 steps.