Special Topics: Artificial Neural Networks

Homework #1

Wesley Liao

2018-03-08

Contents

Test 1: Simple Single Neuron

Main file: multiinput_first.m

Data file: data_unscaled.csv

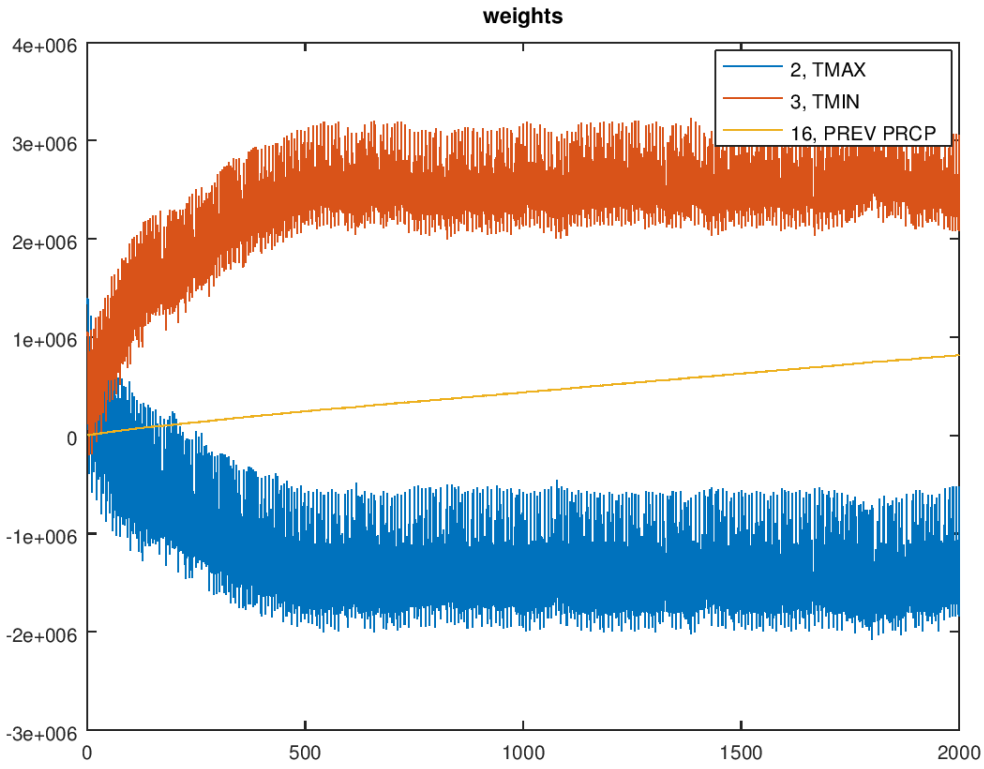
This first test uses a single neuron and 3 inputs. Only the original data is used. The network was set up to train using batch training, applying the weights and bias to the entire dataset, then summing the result to update the weights.

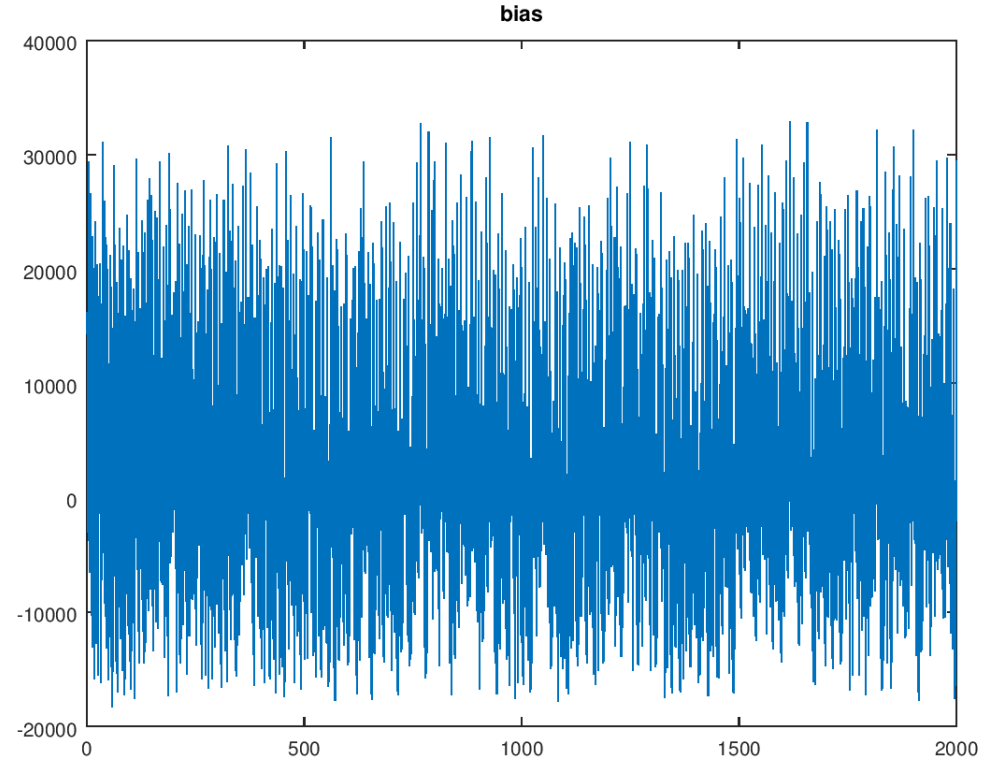
```
for i = 1:loops

  if mod(i, loops/100) == 0
    disp(floor(i/loops*100))
    fflush(stdout);
  endif

  a = hardlims(sum((repmat(w, rows(p), 1).*p)+b,2));
  w = w + lr_w.*sum((repmat(t, 1, columns(a)).-a).*p);
  b = b + lr_b.*sum((repmat(t, 1, columns(a)).-a));

  whist(i, :) = w;
  bhist(i, :) = b;

  accuracy = sum(or(and(a==1, t==1),and(a==-1, t==0)))/rows(t);
  sensitivity = sum(and(a==1, t==1))/sum(t==1);
  specificity = sum(and(a==-1, t==0))/sum(t==0);

  ahist(i, 1) = accuracy;
  ahist(i, 2) = sensitivity;
  ahist(i, 3) = specificity;
endfor
```

The network was first trained for 2000 iterations, resulting in an accuracy of 29.8%.

```
loops =  2000
features =

    2    3   16

training complete, weights:
w =

  -1.5392e+006   2.4336e+006   8.1626e+005

b = -1993.5
accuracy for all data:
accuracy =  0.29806
sensitivity =  0.99895
specificity =  0.0010030
```

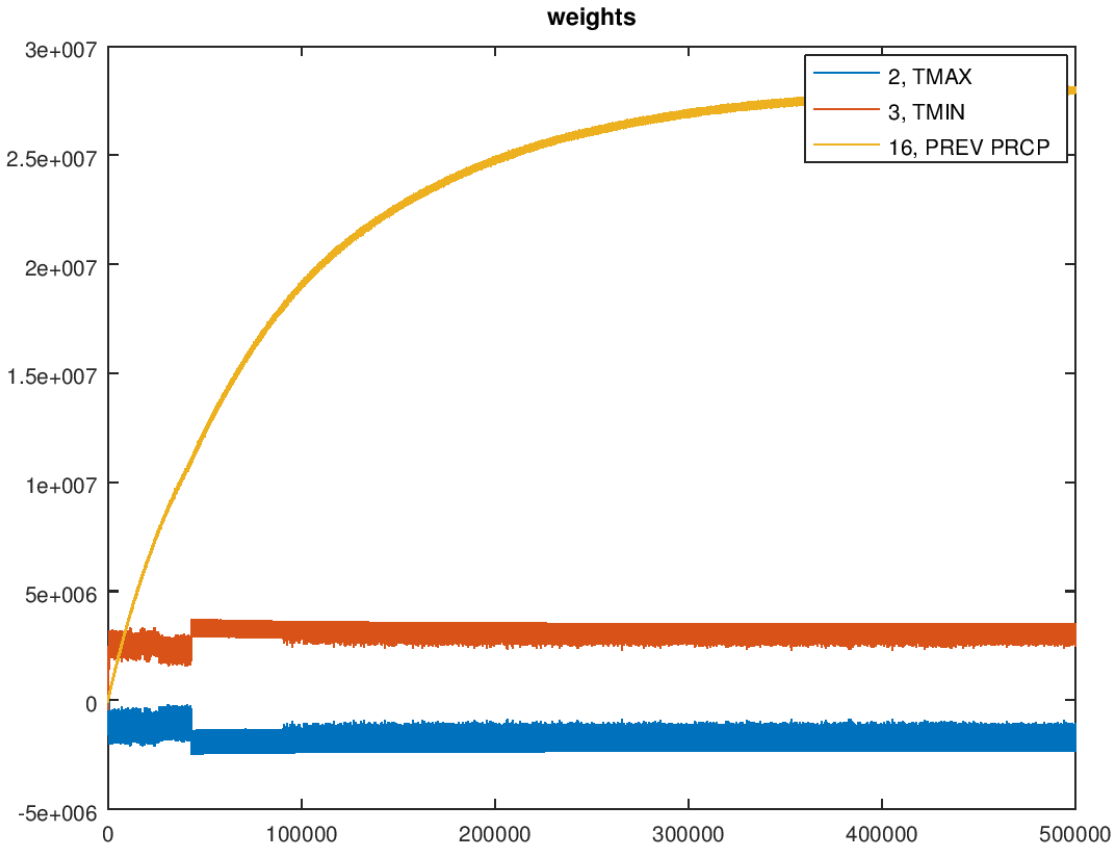
The weights for the temperature seem to center on a value but oscillate rapidly, while the weight for the previous day precipitation rose slowly throughout the training. The bias value oscillated wildly and did not converge.


weights
2, TMAX
3, TMIN
16, PREV PRCP
4e+006
3e+006
2e+006
1e+006
0
-1e+006
-2e+006
-3e+006
0
500
1000
1500
2000


bias
40000
30000
20000
10000
0
-10000
-20000
0
500
1000
1500
2000

Noticing that the weight for the previous precipitation had not settled, the training was ran for 500000 iterations, resulting in 78% accuracy:

```
loops =  500000

features =


    2    3   16

training complete, weights:
w =

  -1.2593e+006  3.5071e+006  2.7256e+007

b =   2.3271e+004

accuracy for all data:
accuracy =  0.78483
sensitivity =  0.33471
specificity =  0.97202
```



However, these results are suspicious. The network was retrained for another 500000 iterations, and resulted in only 29.6% accuracy, very close to the first run of only 2000.

```
loops =  500000
features =
    2    3   16

b = -9717.0
accuracy =  0.29642
sensitivity =  0.99841
```

This is because 29.7% of the days are raining in the dataset, and the data is not balanced. So the accuracy of even the 78% run is not really that good of a predictor. The network is just either predicting all rain or all dry.

Test 2: Simple Single Neuron with Scaled/Balanced Data and Feature Extraction

Feature Selection

Features extracted from the original data include:

- Mean temperature between the high and the low
- The past 10 days of mean temperatures
- The rolling average of up to the past 10 days of mean temperatures
- The change in temperature from the previous day
- The precipitation for the past 10 days
- The rolling average of up to the past 10 days of precipitation
- Cumulative and average rainfall for the current month and year, and previous month and year. (The cumulative totals and averages include only the days up to the previous day)
- The running streak of rainy and non-rainy days

After all the features were calculated, they were scaled from -1 to 1 using the (2*(value-min)/(max-min)-1 formula.

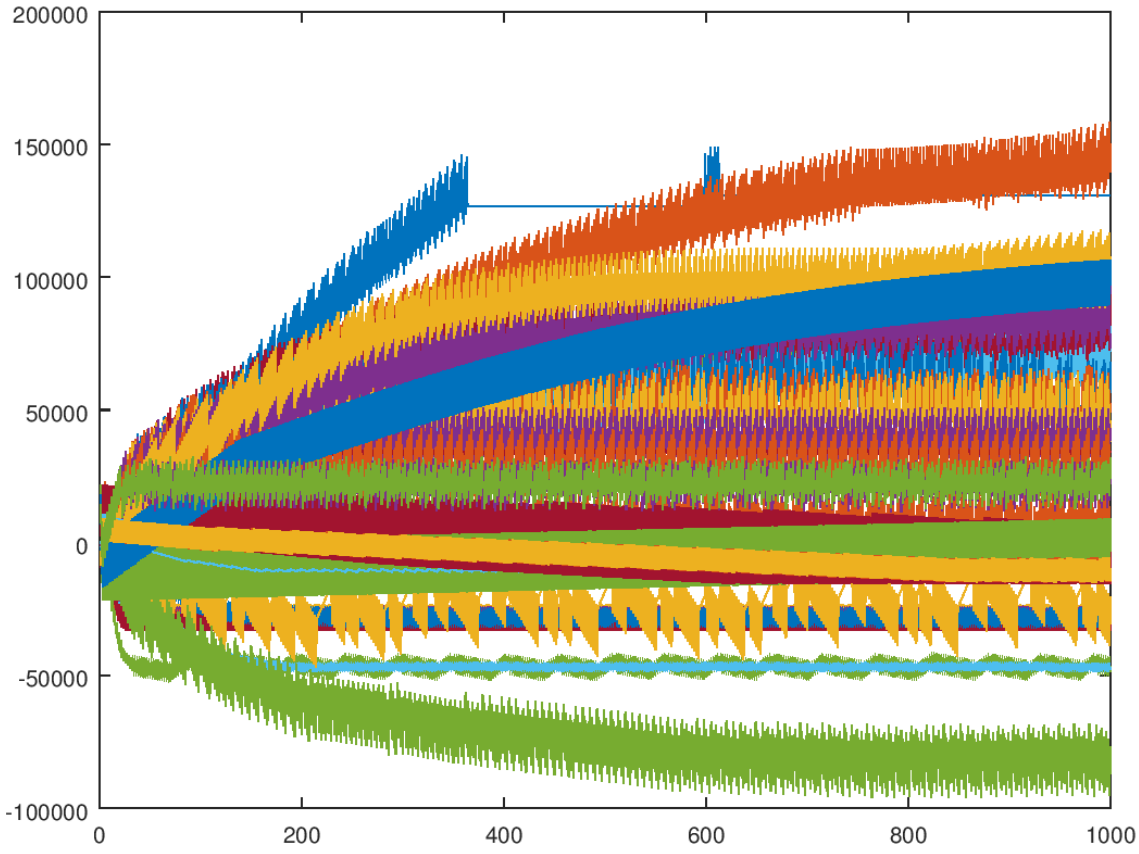
All features were independently trained on all the data using a single input single neuron with a weight and bias, and using the hardlim function to generate the output.

Main file: paramsel.m

Data file: data_scaled.csv

The results of the single feature accuracy test shown on the next page. Note: features marked with an * are intermediate data columns used to generate some of the other features and are not expected to be useful.

Parameter Name	#	Accuracy	Weight	Bias
PRCP	15	100%	130722.5975	130254
PREV_RAIN	30	71%	-7206	4900
TRANGE	26	67%	-48438.18203	6304
PREV_PRCP_AVG_4	23	65%	52299.09086	38210
RAINSTREAK	32	64%	73369.25	87828
TMAX	2	62%	-28625.63637	-4600
MONTH_CUM_AVG	38	61%	101927.9055	80418
PREV_PRCP_AVG_10	25	60%	43206.95831	21788
MONTH_CUM_RAIN	37	59%	30676.72729	18198
PREV_PRCP_AVG_5	24	58%	38308.42798	51000
TAVG	4	58%	-26404.9874	-6856
PREV_TAVG_AVG_10	14	56%	-32955.29705	1118
PREV_TAVG_1	5	56%	-30814.0921	-3982
PREV_TAVG_AVG_5	13	56%	-31967.2094	-1048
PREV_TAVG_AVG_4	12	56%	-31844.70719	-1700
PREV_TAVG_AVG_2	10	56%	-31249.76477	-3122
PREV_TAVG_AVG_3	11	56%	-31537.85555	-2312
PREV_TAVG_2	6	55%	-30829.77022	-3978
PREV_TAVG_3	7	55%	-30582.00016	-3866
PREV_TAVG_4	8	54%	-30584.43696	-3860
PREV_TAVG_5	9	54%	-30723.44839	-3882
DRYSTREAK	33	54%	-76947.40706	-72902
PREV_PRCP_1	16	53%	139433.6095	135526
TMIN	3	53%	-28404.67603	-4986
DAY_OF_YEAR*	39	52%	-10215.96167	9714
TMAX_DIFF	27	52%	-47680.96709	-7618
PREV_PRCP_2	17	51%	96708.3187	97748
TMIN_DIFF	28	51%	34512.9998	8206
STREAK	31	50%	-20282.44461	-19908
PREV_PRCP_4	19	50%	72909.49794	72466
YEAR*	41	50%	-10316.52174	10520
YEAR_CUM_AVG	43	50%	106539.2882	61042
PREV_PRCP_3	18	50%	99201.64938	58840
PREV_PRCP_5	20	50%	83629.82462	43974
PREV_PRCP_AVG_2	21	50%	93533.73661	57818
PREV_PRCP_AVG_3	22	50%	68024.66884	30298
PREV_YEAR_CUM_PRCP	44	50%	-14574.44952	-15110
PREV_YEAR_PRCP_AVG	45	50%	-14161.82331	-15110
PREV_YEAR_CUR_MONTH_CUM_PRCP	46	50%	30272.49533	-3972
PREV_YEAR_CUR_MONTH_PRCP_AVG	47	50%	32158.80421	-2818
DATE*	1	50%	-10888.17473	10888
DAY_OF_MONTH*	34	50%	-11050.39996	11164
MONTH*	36	50%	-10740.36404	11384
YEAR_CUM_RAIN	42	50%	-15501.22068	13016
LAST_DAY_OF_YEAR*	40	50%	9040	-8966
LAST_DAY_OF_MONTH*	35	50%	-14238	13762
TAVG_DIFF	29	50%	-29920.79066	-7048



Here are the weights of all the features being tuned simultaneously. The blue line at the top that converges is parameter 15, PRCP. This had an accuracy of 100%, which is expected because it is the actual precipitation of the current day. The rest of the parameters do not converge, although some oscillate less than others.

Data Balancing

The program datasplit_balanced_all.m takes the scaled data, balances it to contain an even number of rainy and dry days, and splits it into 3 groups: training, validation, and test. The data is balanced across all three of the groups, but each one contains a random set of days from the original data.

The program works by separating all the data into two sets, rainy days and dry days, then ordering the data in a random order. Data is taken in pairs from the rainy and dry sets, so that the datasets will be balanced. The two days have no correlation regarding the date they are taken from. The first third of the data is taken as the training data, the second as validation, and the remaining data is the test dataset.

Single Neuron Multi-input Network

With the data balanced and scaled, the performance can hopefully be improved.

Original Features

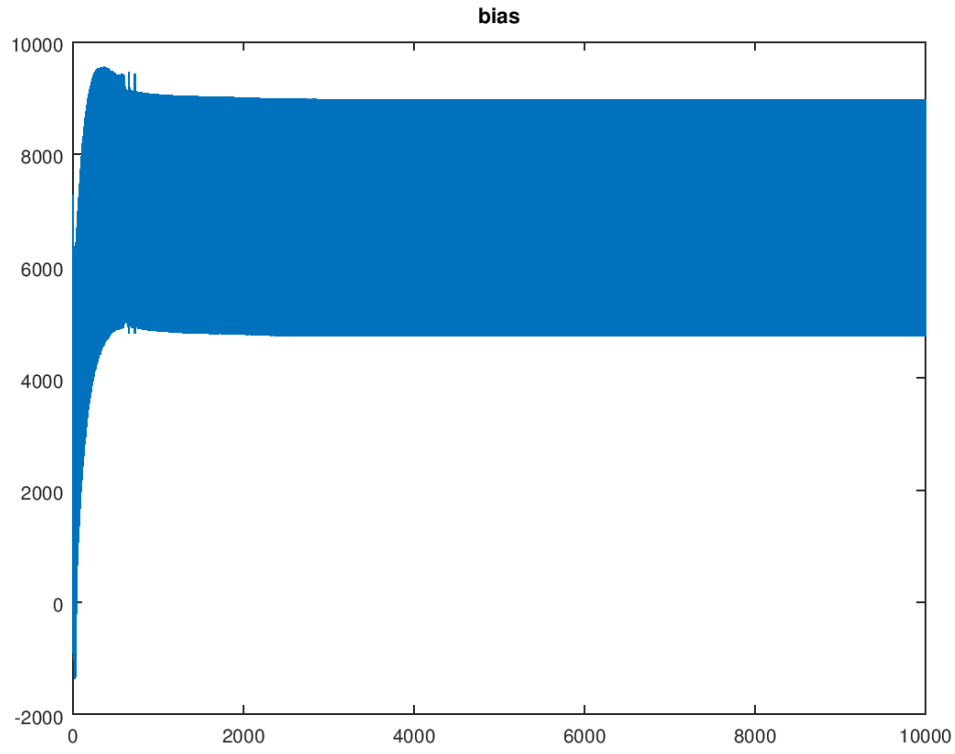
To demonstrate the effectiveness of scaling and balancing the data, first just the original features were used. Because the solution does not converge, and oscillates around a point, the training was set up to remember the best case values for the weights and bias, and save them for the final results. This resulted in an accuracy of about 65%.

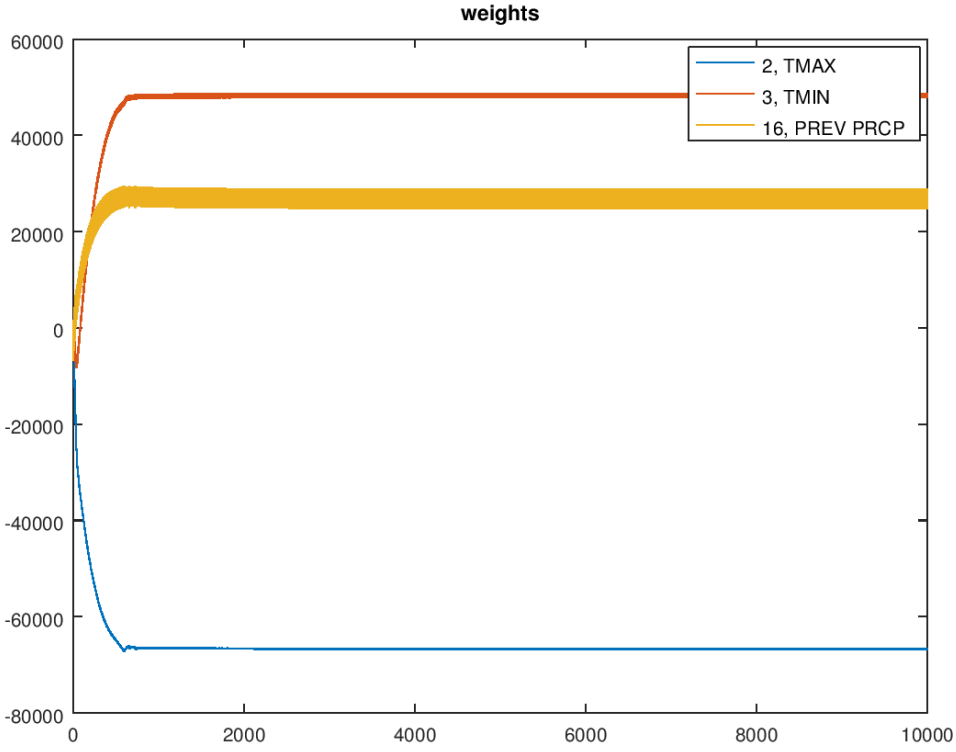
Main file: multiinput.m

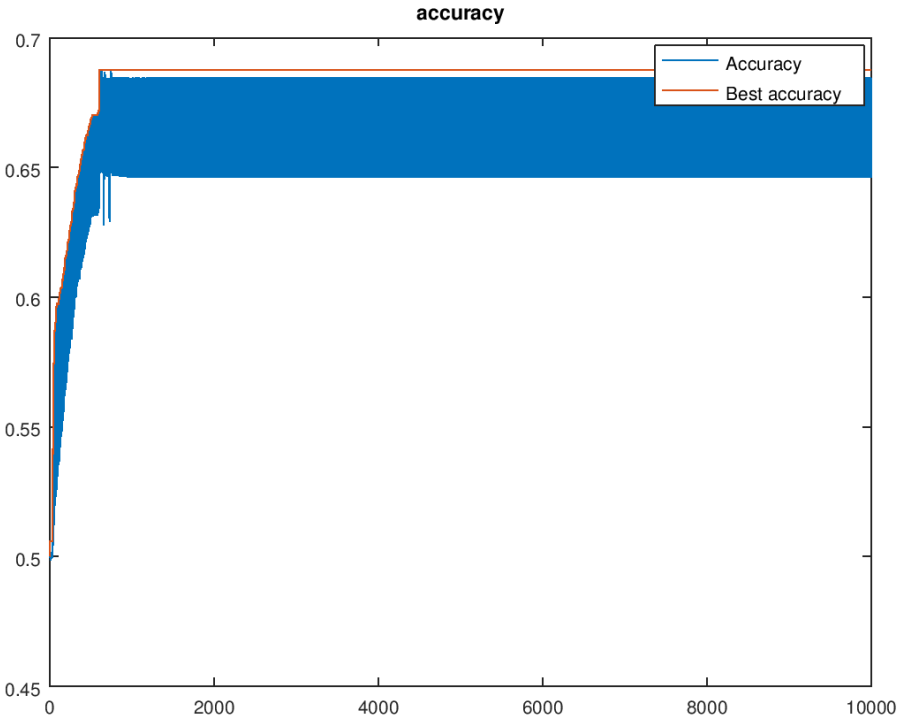
Data file: data_scaled.csv

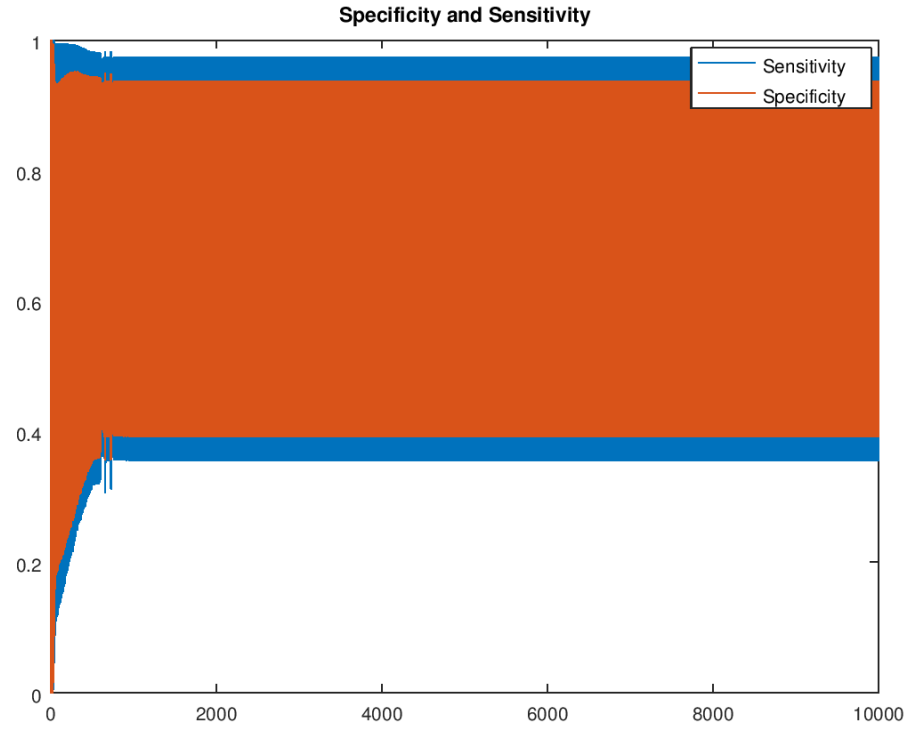
Output:

```
loops =  10000
features =

    2    3   16

training complete, weights:
w =

  -6.7150e+004  4.6560e+004  2.9358e+004

b =  5069.9

training only dataset:
accuracy =  0.68442
sensitivity =  0.97303
specificity =  0.39582

training and val datasets:
accuracy =  0.64988
sensitivity =  0.36127
specificity =  0.93848

training, val, and test datasets:
accuracy =  0.65156
sensitivity =  0.36817
specificity =  0.93504
```


bias
10000
8000
6000
4000
2000
0
-2000
0
2000
4000
6000
8000
10000


weights
60000
40000
20000
0
-20000
-40000
-60000
-80000
0
2000
4000
6000
8000
10000
2, TMAX
3, TMIN
16, PREV PRCP

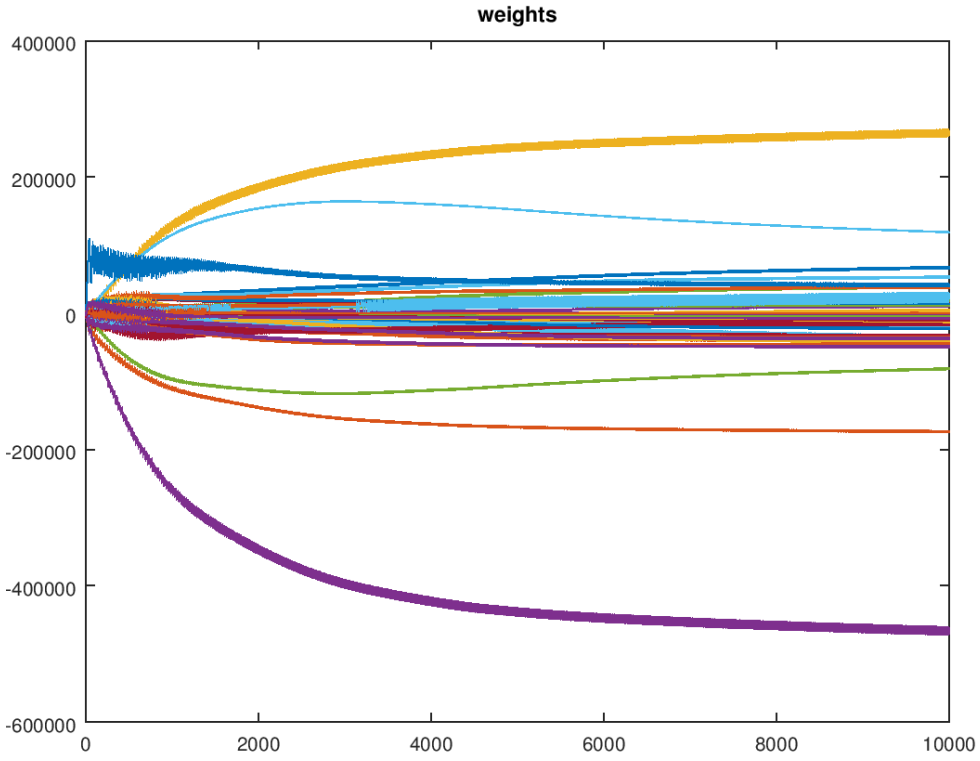

accuracy
Accuracy
Best accuracy
0.7
0.65
0.6
0.55
0.5
0.45
0
2000
4000
6000
8000
10000

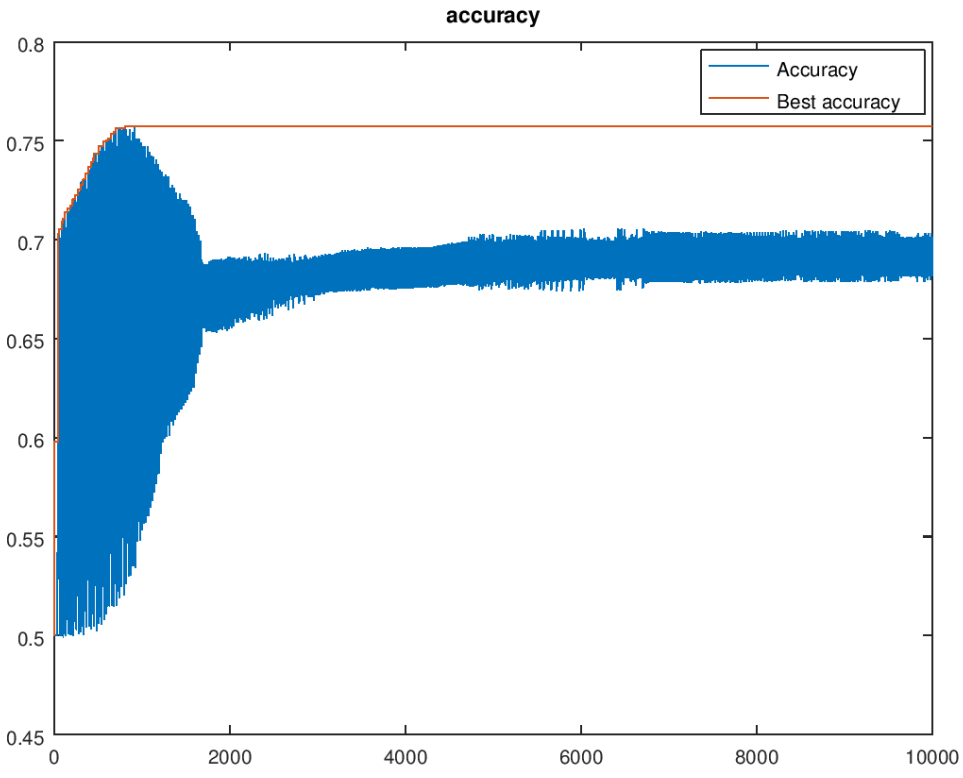

Specificity and Sensitivity
Sensitivity
Specificity
1
0.8
0.6
0.4
0.2
0
0
2000
4000
6000
8000
10000

Additional Features

The single neuron was run with all the available extracted features, and this increased accuracy to 73%.

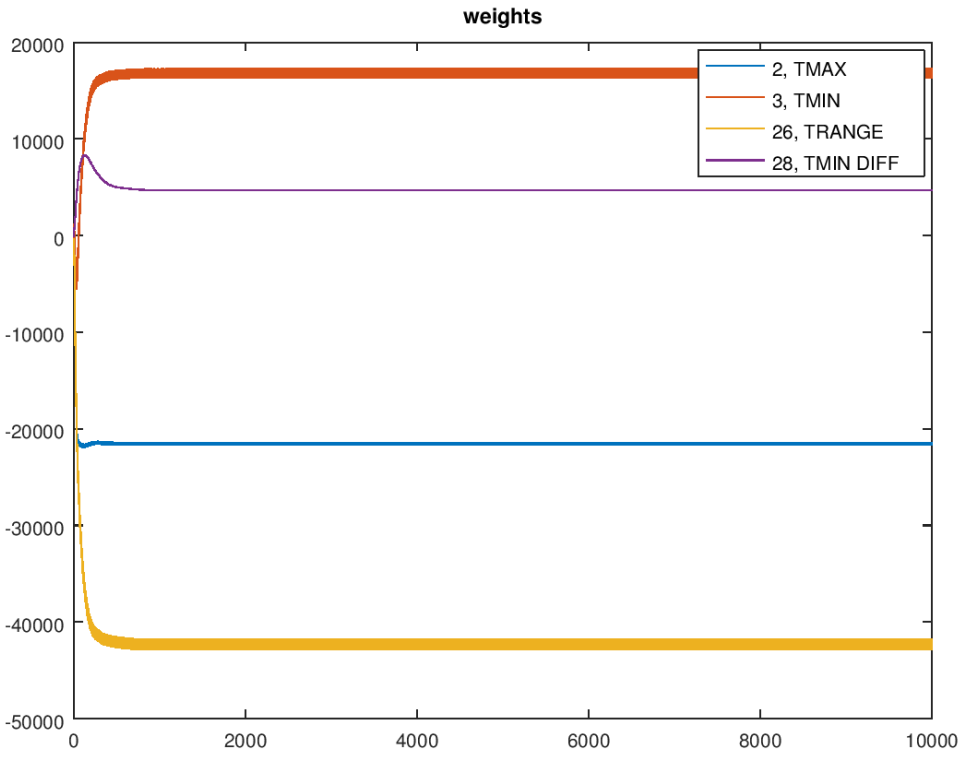
```
loops =  10000
features =

    1    2    3    4    5    6    7    8    9   10   11   12   13   14
16   17   18   19   20   21   22   23   24   25   26   27   28   29
30   31   32   33   34   35   36   37   38   39   40   41   42   43
44   45   46   47

training complete, weights:
w =

 Columns 1 through 18:

  -1.3422e+002  -1.0023e+005  1.1103e+005  -1.3511e+004  4.9107e+003
-7.0493e+003  -1.0215e+004  -1.5168e+004  -1.8707e+004  -1.0344e+003
-4.1900e+003  -7.0787e+003  -9.9189e+003  -1.9562e+004  2.4336e+004
1.1343e+003  -2.3082e+003  2
.1744e+003

 Columns 19 through 36:

  3.6563e+003  2.6277e+004  2.2899e+004  2.4327e+004  2.4988e+004
1.3906e+004  -2.3030e+005  -8.9055e+004  1.0080e+005  -3.3133e+004
6.4483e+004  -2.0975e+004  -2.6421e+001  -2.2458e+004  3.2738e+003  -
8.8120e+003  -1.9671e+003  9.5061
e+003

 Columns 37 through 46:

  1.1240e+004  -1.3795e+003  -1.0452e+004  -1.1661e+002  2.6783e+003
1.9970e+003  -3.1096e+003  -3.0100e+003  -1.2617e+003  -1.0774e+003

b =  2527.6
training only dataset:
accuracy =  0.68181
sensitivity =  0.41426
specificity =  0.94935
training and val datasets:
accuracy =  0.73018
sensitivity =  0.61107
specificity =  0.84930
training, val, and test datasets:
accuracy =  0.73041
sensitivity =  0.61339
specificity =  0.84751
```

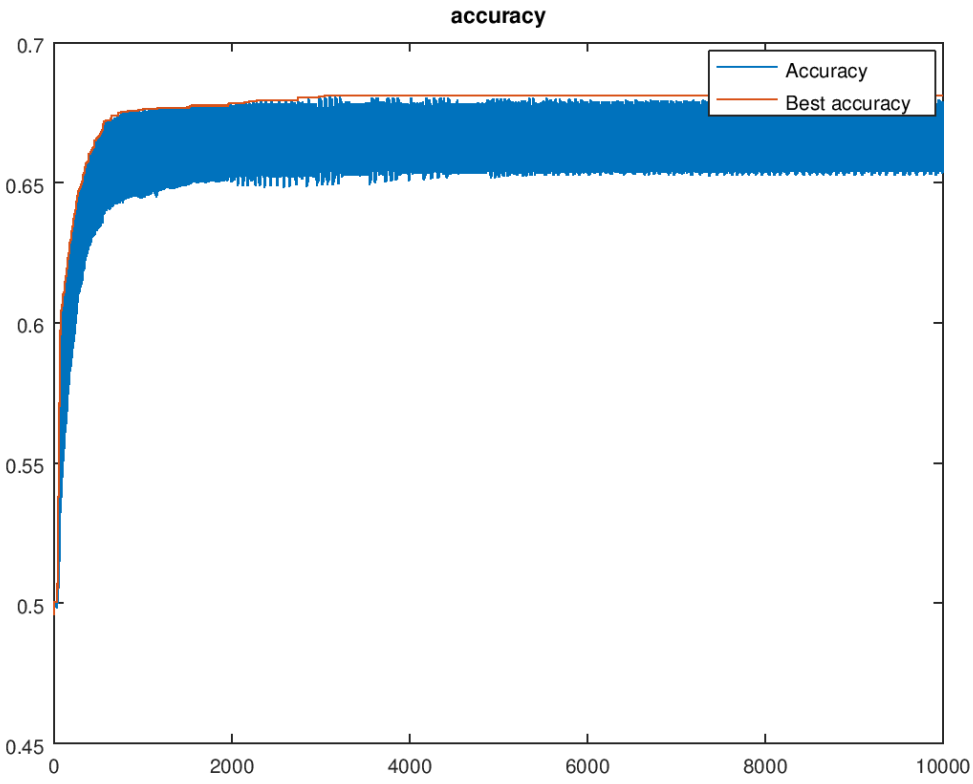

weights
400000
200000
0
-200000
-400000
-600000
0
2000
4000
6000
8000
10000


accuracy
Accuracy
Best accuracy
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0
2000
4000
6000
8000
10000

Using the values of the weights, the features were trimmed down to the top 4 largest weights, which reduced the accuracy to 63-65%.

```
loops =  10000
features =

    2    3   26   28

training complete, weights:
w =

  -2.1732e+004  1.6268e+004  -4.2845e+004  4.7545e+003

b =  256.75
training only dataset:
accuracy =  0.67231
sensitivity =  0.97633
specificity =  0.36829
training and val datasets:
accuracy =  0.63171
sensitivity =  0.32563
specificity =  0.93779
training, val, and test datasets:
accuracy =  0.63298
sensitivity =  0.33018
specificity =  0.93587
```




accuracy
Accuracy
Best accuracy
0.7
0.65
0.6
0.55
0.5
0.45
0
2000
4000
6000
8000
10000

Test 3: Widrow-Hoff Network

Implementation

The implementation is similar to the simple neuron network, with minor changes to the training methods and output function. This test also used batch training. When testing the network, the error would minimalize and then the weights would run away to infinity. The training was set up to stop as soon as the error gets worse than the previous iteration.

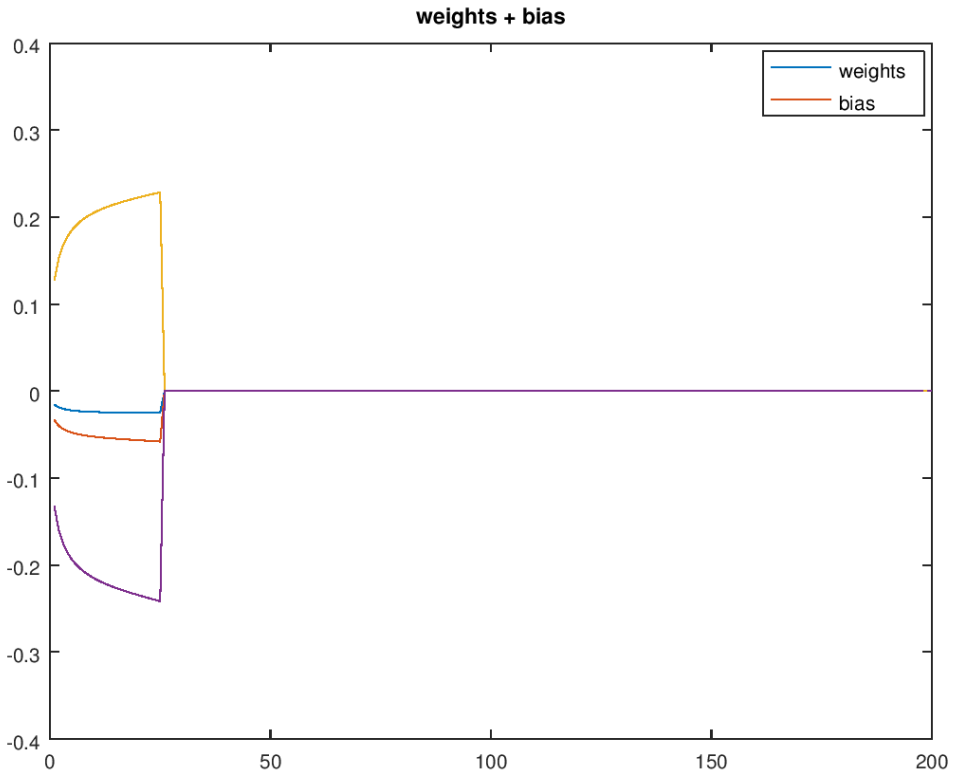
Because the output is a decimal value instead of a yes/no, I used the actual precipitation as the training data, and consider the output correct when the predicted precipitation is within 5% of the actual.

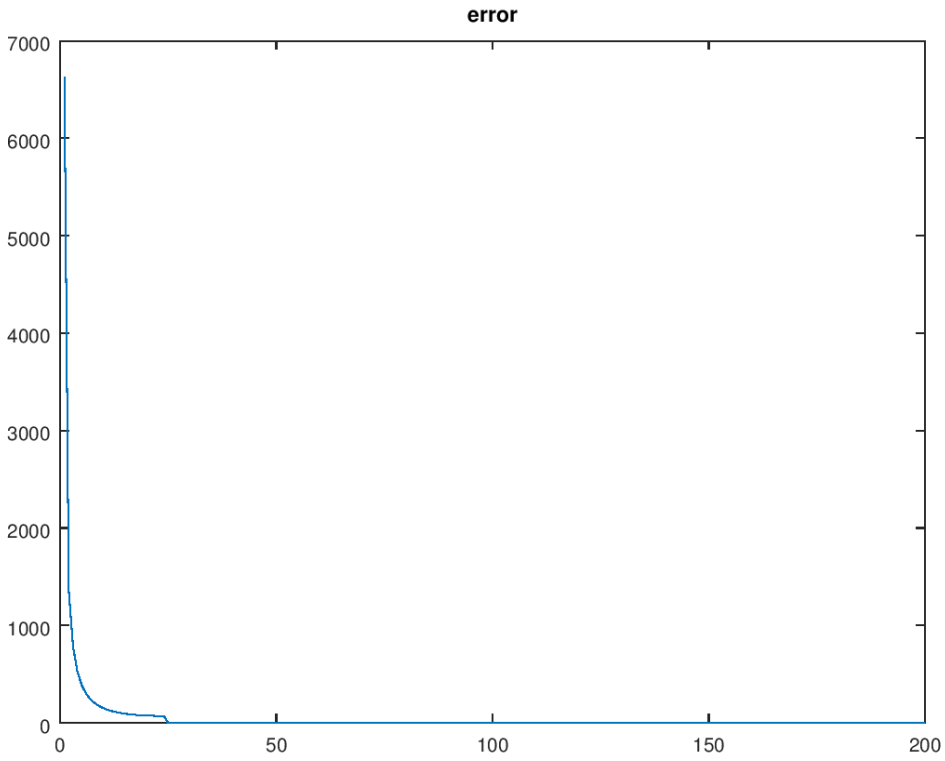
```
a = sum((repmat(w, rows(pl), 1).*pl)+b,2);
best_error = sum((t-a).^2);
for i = 1:loops

  if mod(i, loops/100) == 0
    disp(floor(i/loops*100))
    fflush(stdout);
  endif

  a = sum((repmat(w, rows(pl), 1).*pl)+b,2);
  w = w + (2.*lr_w.*sum(((repmat(t, 1, columns(a)).-a).^2).*pl));
  b = b + (2.*lr_b.*sum((repmat(t, 1, columns(a)).-a).^2));
  whist(i, :) = w;
  bhist(i, :) = b;
  accuracy = sum(abs(t-a)<.05)/rows(t);
%  accuracy = sum((a>-1)==(t>-1))/rows(t);
  error = sum((t-a).^2);
  if error <= best_error
    best_error = error;
  else
    break;
  endif

  ahist(i, 1) = error;
endfor
best_error
features
disp("training complete, weights:")
w
b
disp("accuracy for training only dataset: ")
accuracy
```

Original Features

The network was first tested on our original 3 features withan accuracy of 67%, very similar to the perceptron network.

```
best_error =  70.823
features =

    2    3   16

training complete, weights:
w =

  -0.024707  -0.058017   0.228591

b = -0.24165
accuracy for training only dataset:
accuracy =  0.64162
accuracy with training and val datasets:
accuracy =  0.67733
accuracy with training, val, and test datasets:
accuracy =  0.67642
```

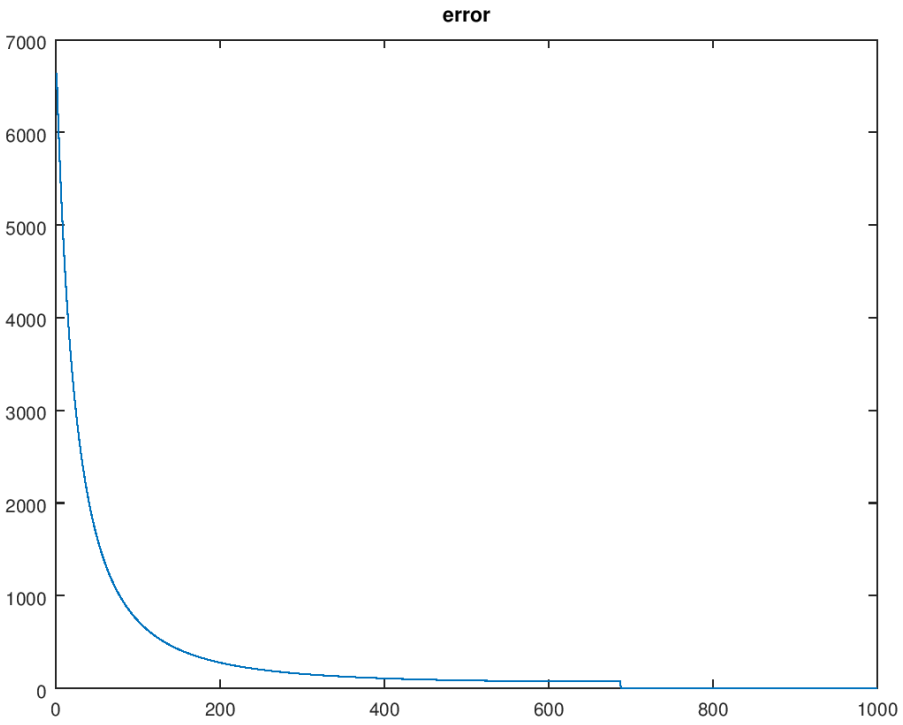


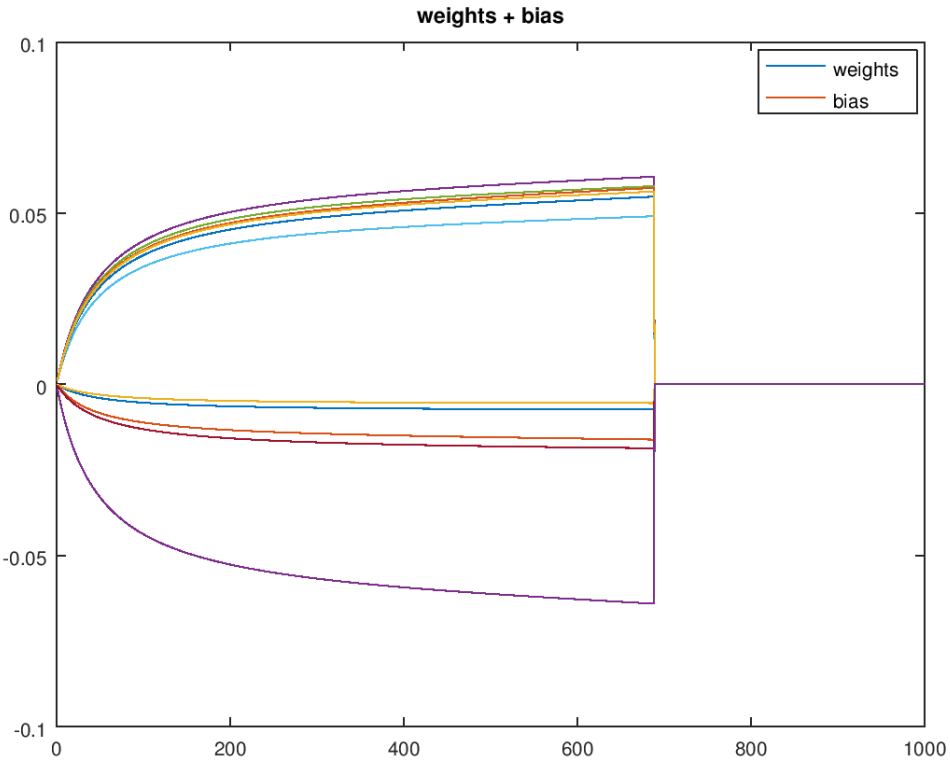

error
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200

Additional Features

The network was trained using features 2 3 4 16 21 25 26 31 33 38, resulting in an accuracy of 63%.

```
best_error =  72.515
features =

    2    3    4   16   21   25   26   31   33   38

training complete, weights:
w =

  -0.0073251  -0.0161523  -0.0054958   0.0606710   0.0578947
0.0491185  -0.0186668   0.0548398   0.0574050   0.0563210

b = -0.064103
accuracy for training only dataset:
accuracy =  0.63377
accuracy with training and val datasets:
accuracy =  0.63508
accuracy with training, val, and test datasets:
accuracy =  0.63615
```


error
7000
6000
5000
4000
3000
2000
1000
0
0
200
400
600
800
1000

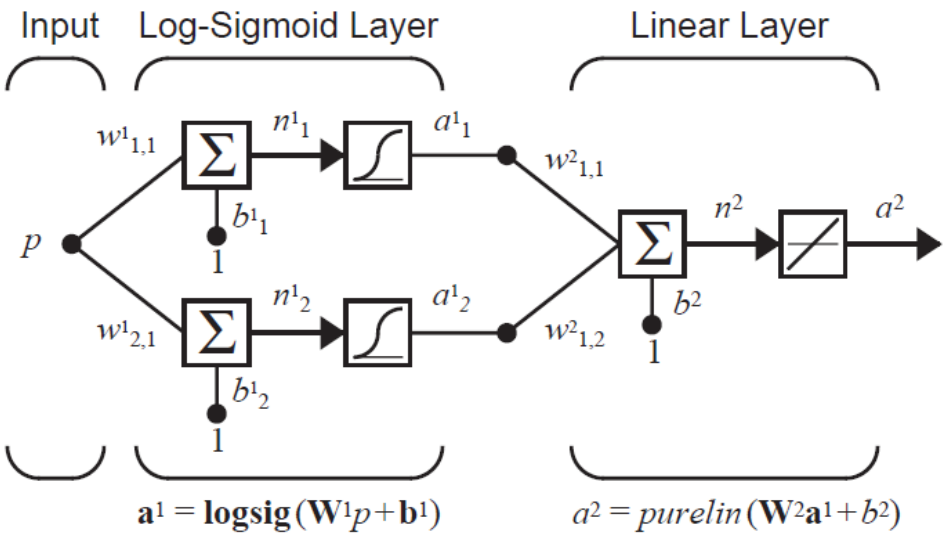

weights + bias
weights
bias
0.1
0.05
0
-0.05
-0.1
0
200
400
600
800
1000

Test 4: Multiple Layer Function Approximation

Implementation

Similarly to the Widrow-Hoff network, this also used the precipitation as the training data.

The network structure was based off of this network from the textbook, with a variable number of inputs and neurons in the log-sigmoid layer:



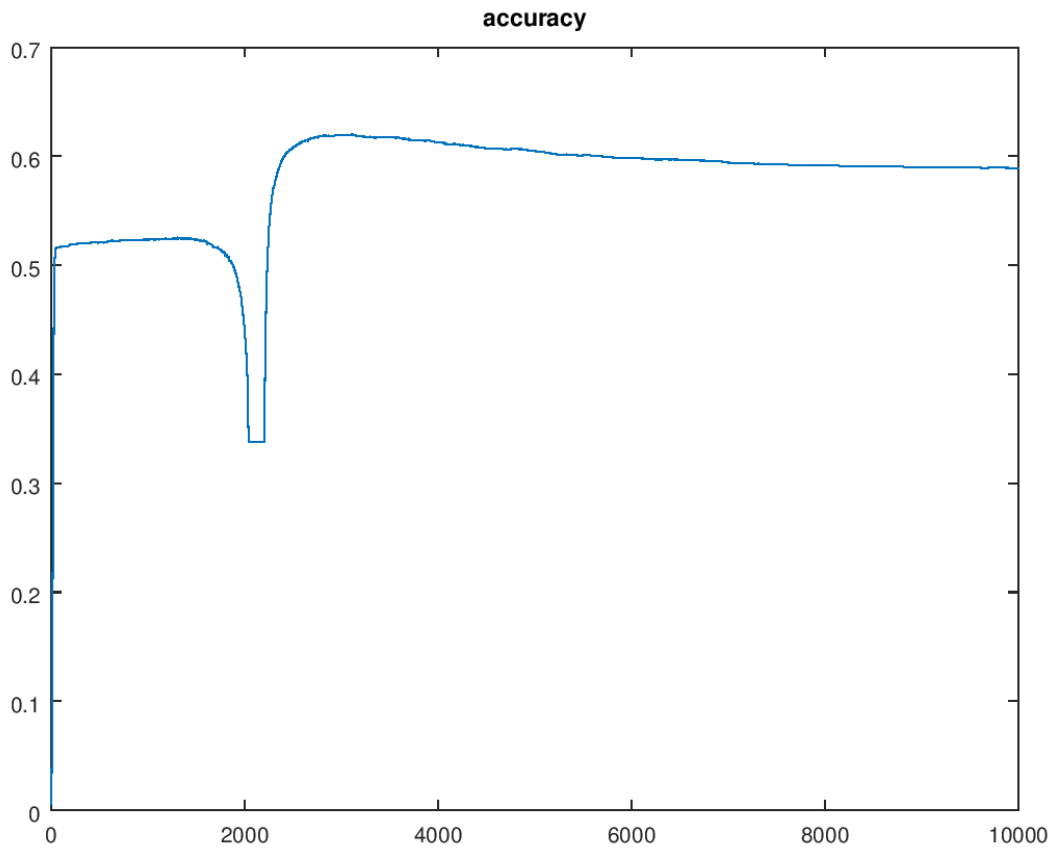
The network trains one layer of neurons at a time and trains in batches.

```
for i = 1:loops
  if mod(i, loops/100) == 0
    disp(floor(i/loops*100))
    fflush(stdout);
  endif

  for ii = 1:neurons
    a(ii,:, 1) = logsig(sum(w(ii, :, 1).*p+b(ii,:,1), 2));
    a(ii, :, 2) = w(ii,1,2).*a(ii,:, 1);
  endfor

  n2 = rot90( sum(a(:,:,2), 1)+b(1,1,1), -1);

  s2 = -2.*(t -n2);
  for ii = 1:neurons
    sl(:,ii) = rot90(((1 - a(ii,:,1)).*a(ii,:,1)),-1).*w(ii,1,2).* s2;

    w(ii,:, 2) = w(ii,:, 2) - sum((alpha.*s2).*rot90(a(ii,:,1),-1));
    w(ii,:, 1) = w(ii,:, 1) - sum((alpha.*sl(:,ii)).*rot90(a(ii,:,1),-1));
    b(ii, :, 1) =  b(ii, :, 1) - (alpha*sum(sl(:,ii)));
  endfor

  b(1, 1, 1) = b(1, 1, 1) - (alpha*sum(s2));

  error = sum((t - n2).^2)/rows(t);

%  accuracy = sum((n2>-1)==(t>-1))/rows(t);
  accuracy = sum(abs(t-n2)<.05)/rows(t);

  if accuracy > best_accuracy
    best_accuracy = accuracy;
    best_w = w;
    best_b = b;
  endif

  whist(i, :, 1) = w(1,:, 1);
  whist(i, :, 2) = w(2,:, 1);
  whist(i, :, 3) = w(1,:, 2);
  whist(i, :, 4) = w(2,:, 2);


  hist(i,1) = error;
  hist(i,2) = accuracy;

endfor
```

Original Features

The network was run with the original 3 features and only 2 neurons in the log sigmoid layer.

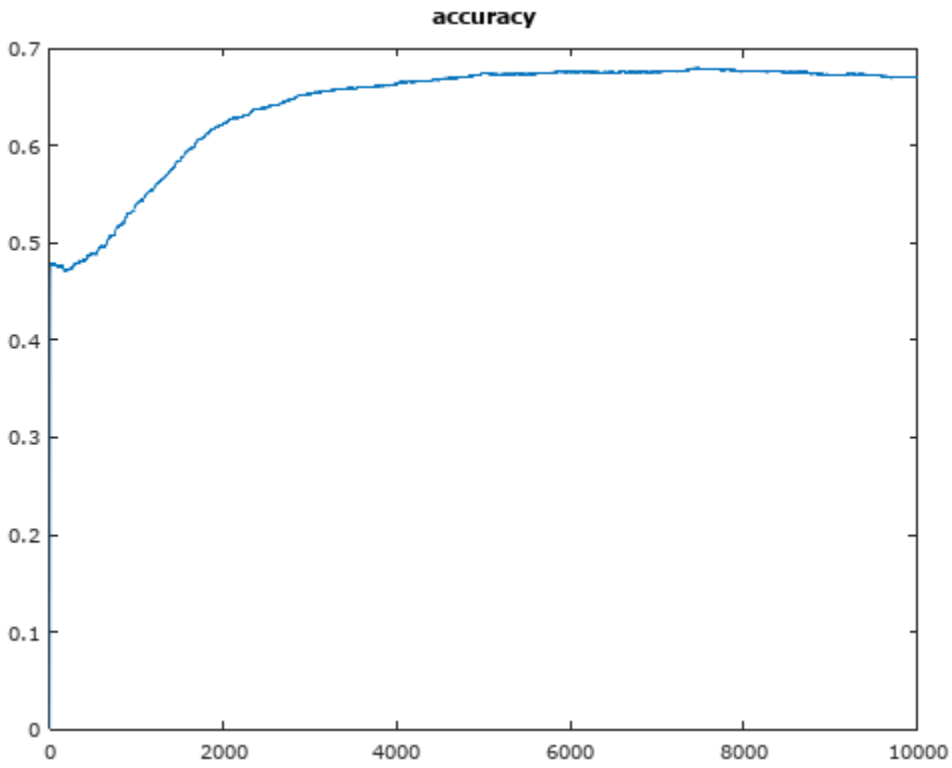
```
accuracy =  0.62029
features =

    2    3   16

training complete, weights:
accuracy for training only dataset:
accuracy =  0.62029
accuracy with training and val datasets:
accuracy =  0.61499
accuracy with training, val, and test datasets:
accuracy =  0.61326
```



The accuracy is not much better than the other networks tried so far.

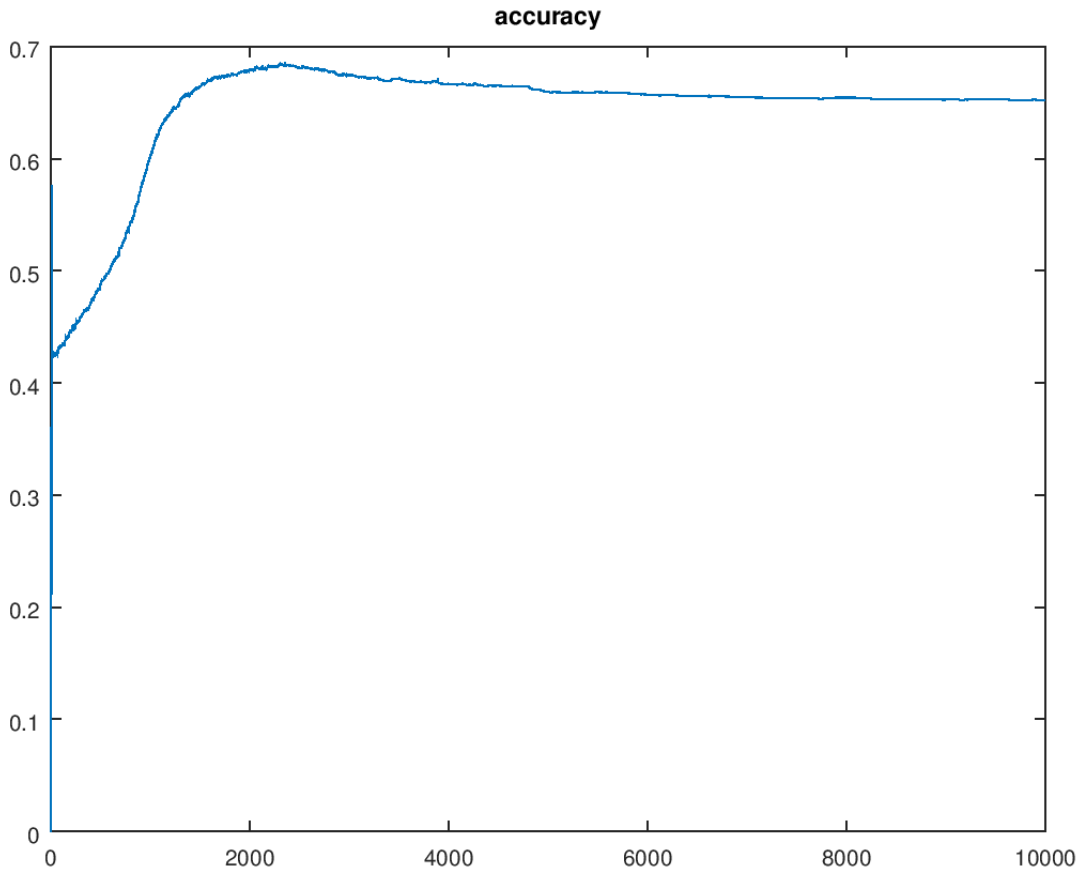
9 Neurons:

```
accuracy =  0.67960
features =

    2    3   16

training complete, weights:
accuracy for training only dataset:
accuracy =  0.67960
accuracy with training and val datasets:
accuracy =  0.68222
accuracy with training, val, and test datasets:
accuracy =  0.68193
```



25 Neurons:

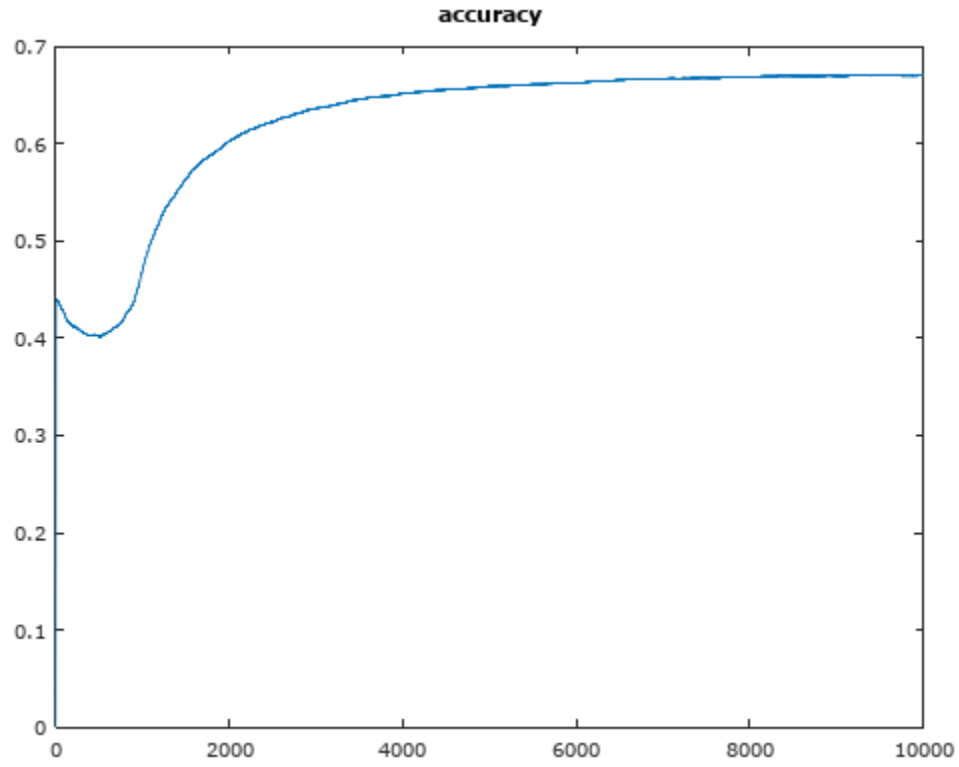
```
accuracy =  0.68428
features =

    2    3   16

training complete, weights:
accuracy for training only dataset:
accuracy =  0.68428
accuracy with training and val datasets:
accuracy =  0.67933
accuracy with training, val, and test datasets:
accuracy =  0.67693
```



Additional Features

More tests were completed with features added, using 9 neurons in the log sigmoid layer.

```
accuracy =  0.67080
features =

    2    3    4   16   21   25   26   31   33   38

training complete, weights:
accuracy for training only dataset:
accuracy =  0.67080
accuracy with training and val datasets:
accuracy =  0.67224
accuracy with training, val, and test datasets:
accuracy =  0.67096
```



Features [2 3 4 7 8 9 14 16 21 22 23 24 25 26 27 28 29 30 31 33 38 40], and 9 neurons:

```
accuracy =  0.63721
features =

    2    3    4    7    8    9   14   16   21   22   23   24   25   26
27   28   29   30   31   33   38   40

training complete, weights:
accuracy for training only dataset:
accuracy =  0.63721
accuracy with training and val datasets:
accuracy =  0.63832
accuracy with training, val, and test datasets:
accuracy =  0.63927
```

Increasing the number of features decreased the accuracy. The features [2 3 4 16 21 25 26 31 33 38] were retested with 25 instead of 9 neurons:

```
롨ʃaccuracy =  0.70823
features =

    2    3    4   16   21   25   26   31   33   38

training complete, weights:
accuracy for training only dataset:
accuracy =  0.70823
accuracy with training and val datasets:
accuracy =  0.70568
accuracy with training, val, and test datasets:
accuracy =  0.70633
```