**Создание сети Элмана для определения количества единиц в подаваемой на вход последовательности**

P – входная последовательность нулей и единиц

T – выходная последовательность для обучения сети

P= round ( rand (1,20)) *{Создаем последовательность из 20 нулей и единиц так, чтобы были*

*повторяющиеся единицы}*

1 1 0 1 0 1 0 1 1 0 1 1 0 0 0 0 1 1 0 0

T= [0 (P(1:end-1)+ P(2:end) ==2)] *{ Проверяет нет ли двух единиц подряд – если есть ставит*

*единицу}*

0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0

Psec = con2seq (P) {Преобразовываем из числовой последовательности в последовательность

векторов}

Tsec = con2seq (T)

net=newelm([0 1],[10 1], {'tansig','logsig'} ); *{Создаем сеть Элмана, с диапазоном входного сигнала 0 - 1, с 10 скрытыми и одним выходным нейронами, с функциями активации*

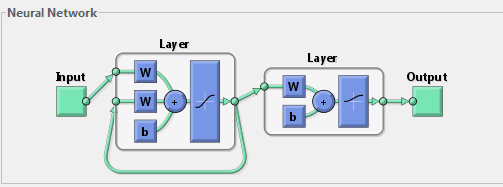
*гиперболический тангенс для скрытого слоя и и логистической для второго слоя }*

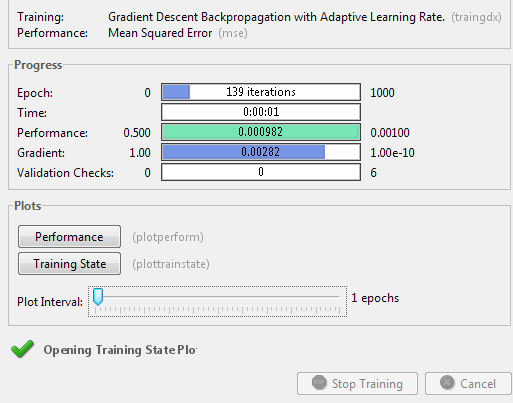
net.trainParam.goal=0.001; *{Задаем параметр для остановки обучения}*

net.trainParam.epochs=1000; {Залаем количество повторений}

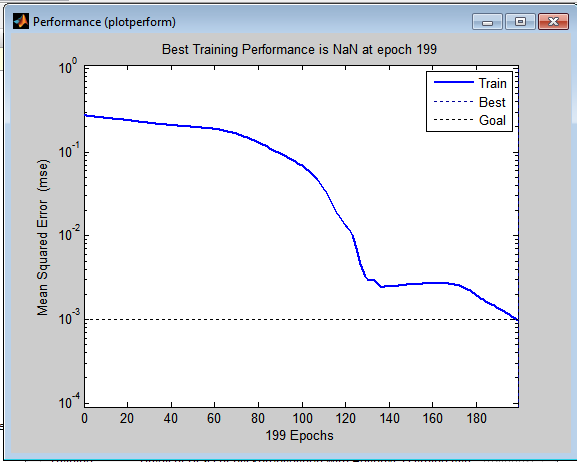
net = train (net, Psec, Tsec) *{Тренируем сеть, используя входную последовательность и*

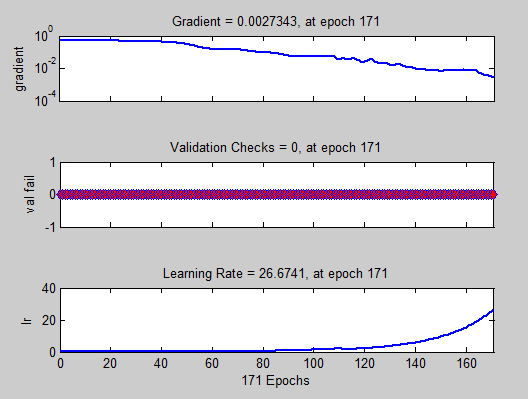
*выходную}*





Посмотрим на результаты обучения . Наилучшее приближение достигнуто на 199 шаге

Если нужный уровень не достигается, то нужно увеличить количество циклов обучения

Видно уменьшение градиента, проверка приближения и значение коэффициента обученности

Y = sim(net, Psec );  *{Проводим симуляцию работы сети на последовательности P}*

Y1 = seq2con(Y); *{Из векторного вида результат переводим в*

*последовательность}*

Y =

[0.0742] [0.9919] [0.0285] [0.0040] [0.0229] [5.6109e-004] [0.0116]

[4.7694e-004] [0.9765] [5.6207e-004] [0.0349] [0.9436] [0.0031]

[6.5438e-005] [0.0102] [2.2186e-004] [0.0377] [0.9352] [0.0320]

[8.5059e-005]

E = round ( T-Y1{1}) *{Находим разницу между эталонной последовательностью T*

E =

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

*и полученной Y1, если получаются все нули, то сеть*

*работает правильно}*

*Если не получается попробуйте поменять параметры создаваемой сети*

Попробуем подать другую последовательность на обученную сеть

P= round ( rand (1,20))

P = 1 1 0 1 1 0 0 0 1 1 0 0 1 0 0 0 1 1 0 1

>> Psec = con2seq (P)

Y = sim(net, Psec )

Y = [0.0342] [0.9925] [4.2521e-004] [0.0047] [0.9314] [0.0020] [0.0080] [4.2721e-004] [0.0011] [0.9855] [0.0137] [2.0794e-004] [0.0783] [0.0343] [8.6657e-004] [4.6659e-004] [0.0025] [0.9355] [0.0419] [0.0010]

Видно, что сеть опознала все 4 пары единиц. Если распознаются не все пары, то нужно поменять количество нейронов скрытого слоя или параметр точности обучения, в общем поэкспериментировать и найти такую конфигурацию, которая бы распознавала не только обучающую последовательность, но и любую другую.

Архитектура сети

net =

Neural Network object:

architecture:

numInputs: 1 количество входов

numLayers: 2 количество слоев

biasConnect: [1; 1] вектор смещения связи

inputConnect: [1; 0] вход связи

layerConnect: [1 0; 1 0] скрытый слой связи

outputConnect: [0 1] выход связи

numOutputs: 1 (read-only) задержка выхода

numInputDelays: 0 (read-only) задержка входа

numLayerDelays: 1 (read-only) задержка скрытого слоя

subobject structures: размерность матриц слоев, весов и смещений

inputs: {1x1 cell} of inputs

layers: {2x1 cell} of layers

outputs: {1x2 cell} containing 1 output

biases: {2x1 cell} containing 2 biases

inputWeights: {2x1 cell} containing 1 input weight

layerWeights: {2x2 cell} containing 2 layer weights

functions: используемые функции

adaptFcn: 'trains'

divideFcn: (none)

gradientFcn: 'calcgrad'

initFcn: 'initlay'

performFcn: 'mse'

plotFcns: {'plotperform','plottrainstate'}

trainFcn: 'traingdx'

**Сеть Элмана для детектирования амплитуды гармонического сигнала**

Создадим выборки из двух синусоид с амплитудами 1.0 и 2.0:

p1 = sin(1 : 20);

p2 = sin(1 : 20) \* 2;

t1 = ones(1, 20)

t1 =

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

t2 = ones(1, 20) \* 2

t2 =

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

Сформировать набор векторов входа и целей:

p = [p1 p2 p1 p2];

t = [t1 t2 t1 t2];

Сформировать обучающие последовательности из массивов ячеек:

Pseq = con2seq(p);

Tseq = con2seq(t);

Задать параметры сети:

R = 1;

S2 = 1;

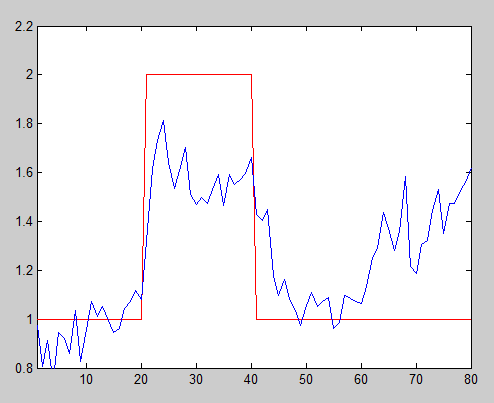
S1 = 10;

Создать сеть Элмана:

net = newelm([-2 2],[S1 S2],{'tansig','purelin'},'traingdx');

Задать параметры обучения:

net.trainParam.epochs = 1000;

>> net.trainParam.show = 25;

>> net.trainParam.goal = 0.01;

Провести обучение на входной и выходной последовательности векторов:

[net,tr] = train(net,Pseq,Tseq);

Проверить сеть на обучающей последовательности:

figure(2)

a = sim(net,Pseq);

time = 1:length(p);

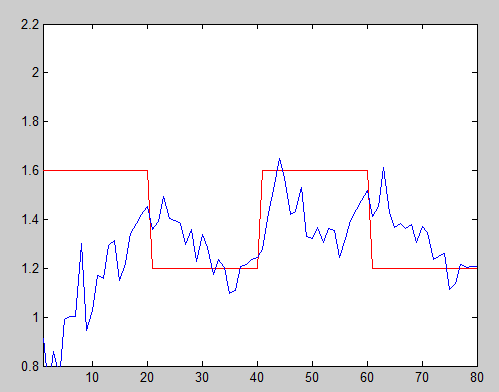
plot(time,t,'-r',time,cat(2, a {:} ),'-b')

axis([1 80 0.8 2.2]) % Поскольку наш синус выводит 80 значений, то 1 80 . Для удобства выведения на график амплитуд 1 и 2 одновременно размах функции - 0.8 2.2

Как следует из анализа рисунка, сеть справляется с решением задачи детектирования амплитуды на наборах обучающего множества, хотя и находит 0.9 и 1.8. Однако неясно, как она будет вести себя на других наборах входа. Обладает ли построенная сеть Элмана свойством обобщения? Попробуем проверить это, выполнив следующие исследования. Подадим на сеть набор сигналов, составленный из двух синусоид с амплитудами 1.6 и 1.2 соответственно:

p3 = sin(1:20)\*1.6;

t3 = ones(1,20)\*1.6;

p4 = sin(1:20)\*1.2;

t4 = ones(1,20)\*1.2;

pg = [p3 p4 p3 p4];

tg = [t3 t4 t3 t4];

pgseq = con2seq(pg);

figure(3)

a = sim(net,pgseq);

ime = 1:length(pg);

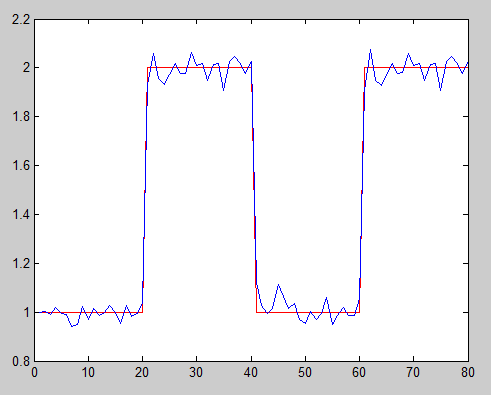
plot(time,tg,'-r',time,cat(2, a {:} ),'-b')

axis([1 80 0.8 2.2])

На этот раз сеть хуже справляется с задачей. Сеть стремится детектировать значение амплитуды, но делает это не очень точно. Улучшенное обобщение могло быть получено, обучая сеть на большее количество амплитуд, чем только на значения 1.0 и 2.0. Использование трех или четырех гармонических сигналов с различными амплитудами может привести к намного лучшему датчику амплитуд, но здесь возникают проблемы для визуализации результатов. Поэтому попробуем просто изменить параметры сети.

Изменим архитектуру сети Элмана, например, так

net = newelm([-2 2],[20 20 1],{'tansig', 'tansig','purelin'},'traingdx');

И изменим параметры обучения

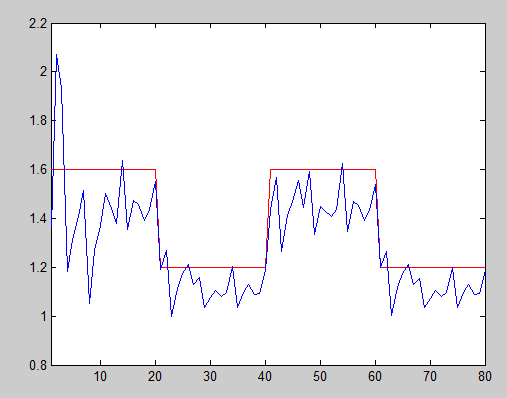
net.trainParam.epochs = 10000;

>> net.trainParam.show = 50;

>> net.trainParam.goal = 0.001;

Проведем обучение и увидим насколько улучшились результаты

Теперь подадим нашу проверочную последовательность

Видим, что и в этом случае результаты гораздо лучше

Таким образом, можно подобрать параметры для довольно точного определения амплитуды.