Основные команды среды MATLAB для работы с нейронными сетями.
NEURAL NETWORK TOOLBOX

NEWFF - Реализует однонаправленную сеть, обучаемую с применением алгоритма обратного распространения

Синтаксис:

net = newff
net = newff(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF)

Описание:

net = newff создает новую сеть с использованием диалогового окна

NEWFF(PR, [S1 S2...SNl], {TF1 TF2...TFNl}, BTF, LF, PF) в качестве входных параметров использует:
PR - Rx2 матрица минимальных и максимальных значений строк входной матрицы с размерностью RxQ. Для получения матрицы PR можно использовать функцию minmax;
Si – количество нейронов в i – ом слое, N1 – количество слоев;
TFi - функция активации i - го слоя, по умолчанию = 'tansig';
BTF – обучающая функция обратного распространения, по умолчанию='trainlm';
BLF – алгоритм подстройки весов и смещений (обучающий алгоритм), по умолчанию = 'learngdm';
PF - функция оценки функционирования сети, по умолчанию = 'mse';

и возвращает однонаправленную сеть, состоящую из N слоев.

В качестве функций активации TFi можно использовать любые дифференцируемые функции активации, например TANSIG, LOGSIG, или PURELIN.

В качестве обучающей функции BTF можно использовать любые функции на основе алгоритма обратного распространения, например TRAINLM, TRAINBFG, TRAINRP, TRAINGD, и т.д

Следует обратить внимание, что функция TRAINLM используется по умолчанию, поскольку она обеспечивает наиболее быстрое обучение. Но она требует много памяти. Поэтому, если Вы получаете сообщение "out-of-memory error", необходимо попытаться выполнить одно из следующих действий:

(1) замедлить процедуру обучения TRAINLM, снизив требования к объему оперативной памяти путем установки параметра NET.trainParam.mem_reduc равным 2 или более (см. HELP TRAINLM)

(2) использовать функцию обучения TRAINBFG, которая работает медленнее, но требует меньше памяти, чем TRAINML.

(3) использовать функцию обучения TRAINRP, которая работает медленнее, но требует меньше памяти, чем TRAINBFG.

В качестве обучающей функции BLF можно использовать любые из алгоритмов обратного распространения. Например LEARNGD или LEARNGDM.

В качестве функции оценки функционирования сети могут быть использованы любые дифференцируемые функции, например MSE или MSEREG.

Примеры:

Имеем набор входных значений P и соответствующих им выходных эталонных значений T. Задача - создать сеть для нахождения выходных значений по входным.

```
P = [0 1 2 3 4 5 6 7 8 9 10];

T = [0 1 2 3 4 3 2 1 2 3 4];
```

Создаем двухслойную однонаправленную сеть. Диапазон входных значений от 0 до 10, первый слой состоит из 5 нейронов с функциями активации TANSIG, второй слой содержит один нейрон с функцией активации PURELIN. Для обучения используем функцию TRAINLM:

```
net = newff([0 10],[5 1],{'tansig' 'purelin'});
```

Моделируем сеть. Подаем на вход сети входные значения P. Получаем выходные значения Y:

```
Y = sim(net,P);
```

Строим график, демонстрирующий отклонения выходных значений Y от целевых значений T для необученной сети:

```
plot(P,T,P,Y,'o')
```

Обучаем (тренируем) сеть. Количество эпох тренировки – 50:

```
net.trainParam.epochs = 50;

net = train(net,P,T);
```

Снова моделируем (теперь уже обученную сеть):

```
Y = sim(net,P);
```

Строим график для обученной сети:

```
plot(P,T,P,Y,'o')
```

Алгоритм:

Однонаправленные нейронные сети состоят из Nl слоев, использующих весовую функцию DOTPROD , входную функцию сети NETSUM и выбранную функцию активации. Первый слой получает веса, приходящие с входа. Каждый последующий слой получает веса от

предыдущего слоя. Все слои обладают смещениями. Последний слой является выходом сети. Веса и смещения каждого слоя инициализируются функцией INITNW. Адаптация выполняется с помощью функции TRAINS, которая подстраивает веса в соответствии с выбранной обучающей функцией. Тренировка сети выполнена с помощью выбранной функции. Оценка функционирования сети выполняется с помощью выбранной функции оценки.

SIM - Моделирует нейронную сеть

Синтаксис:

[Y,Pf,Af,E,perf] = sim(net,P,Pi,Ai,T)

[Y,Pf,Af,E,perf] = sim(net,{Q TS},Pi,Ai,T)

[Y,Pf,Af,E,perf] = sim(net,Q,Pi,Ai,T)

Описание:

SIM моделирует нейронную сеть

[Y,Pf,Af,E,perf] = SIM(net,P,Pi,Ai,T) использует:
net – сеть;
P – входы сети;
Pi – начальные задержки входов, по умолчанию – нули;
Ai – начальные задержки слоев, по умолчанию – нули;
T – целевые значения, по умолчанию – нули;
Y – выходы сети;
Pf – конечные входные задержки;
Af – конечные задержки слоев;
E – ошибки сети;
perf – параметр функционирование сети.

Аргументы Pi, Ai, Pf, и Af являются необязательными и их необходимо использовать лишь в тех случаях, когда сети содержат входы и слои с задержками.

Аргументы SIM могут быть двух форматов: массив ячеек или матрица. Формат в виде массива ячеек является наиболее легким для описания. Он наиболее удобен для описания сетей с многими входами и выходами и позволяет представлять последовательность входов:
P - NixTS массив ячеек, каждая ячейка P{i,ts} – матрица размером RixQ;
T - NtxTS массив ячеек, каждая ячейка T{i,ts} – матрица размером VixQ;
Pi - NixID массив ячеек, каждая ячейка Pi{i,k} – матрица размером RixQ;
Ai - NlxLD массив ячеек, каждая ячейка Ai{i,k} – матрица размером SixQ;
Y - NOxTS массив ячеек, каждая ячейка Y{i,ts} – матрица размером VixQ;
E - NtxTS массив ячеек, каждая ячейка E{i,ts} – матрица размером VixQ;
Pf - NixID массив ячеек, каждая ячейка Pf{i,k} – матрица размером RixQ;
Af - NlxLD массив ячеек, каждая ячейка Af{i,k} – матрица размером SixQ,

где:
Ni = net.numInputs

Nl = net.numLayers,
No = net.numOutputs
ID = net.numInputDelays
LD = net.numLayerDelays
TS = number of time steps
Q = batch size
Ri = net.inputs{i}.size
Si = net.layers{i}.size
Ui = net.outputs{i}.size

Столбцы Pi, Pf, Ai, и Af упорядочены от самых больших задержек к текущим:
Pi{i,k} = i – й вход в момент времени ts=k-ID.
Pf{i,k} = i – й вход в момент времени ts=TS+k-ID.
Ai{i,k} = i – й выход слоя в момент времени ts=k-LD.
Af{i,k} = i – й выход слоя в момент времени ts=TS+k-LD.

Матричный формат может быть использован тогда, когда должен быть смоделирован только один шаг по времени
(TS = 1). Это удобно только для сетей с одним входом и с одним выходом. Однако, он может быть использован для сетей с большим количеством входов и выходов. Каждый аргумент матрицы находится путем накопления элементов соответствующих аргументам массивов элементов в одну матрицу:
P – матрица (сумма Ri)xQ;
T – матрица (сумма Vi)xQ;
Pi – матрица (сумма Ri)x(ID*Q);
Ai – матрица (сумма Si)x(LD*Q);
Y – матрица (сумма Ui)xQ;
E - матрица (сумма Vi)xQ;
Pf – матрица (сумма Ri)x(ID*Q);
Af – матрица (сумма Si)x(LD*Q).

[Y,Pf,Af] = SIM(net,{Q TS},Pi,Ai) используется для сетей, которые не имеют входа, таких как сети Хопфилда.

Примеры:

Используем NEWP для того, чтобы создать перцептронный слой с 2-х элементным входом (диапазон входных значений [0,1;1]), и с единственным нейроном:

```
net = newp([0 1;0 1],1);
```

Перцептрон моделируется для индивидуального вектора, для пакета из трех векторов и для последовательности трех векторов:

```
p1 = [.2; .9]; a1 = sim(net,p1)

p2 = [.2 .5 .1; .9 .3 .7]; a2 = sim(net,p2)

p3 = {[.2; .9] [.5; .3] [.1; .7]}; a3 = sim(net,p3)
```

NEWLIND используется для того, чтобы создать линейный слой с 3 элементным входом и с двумя нейронами:

net = newlin([0 2;0 2;0 2],2,[0 1]);

Линейный слой моделируется для последовательности из двух векторов и начальной временной задержки, задаваемой по умолчанию (все нули):

p1 = {[2; 0.5; 1] [1; 1.2; 0.1]};

[y1,pf] = sim(net,p1)

Слой моделируется для еще трех векторов с использованием предыдущих задержек в качестве новых начальных:

p2 = {[0.5; 0.6; 1.8] [1.3; 1.6; 1.1] [0.2; 0.1; 0]};

[y2,pf] = sim(net,p2,pf)

Функция NEWELM использована для того, чтобы создать сеть Элмана (Elman) с 1 элементным входом, за которым следует первый слой из 3 TANSIG нейронов и второй слой из двух PURELIN нейронов. Так как сеть является сетью Элмана, она содержит линию задержки с задержкой 1 от слоя 1 к слою2:

net = newelm([0 1],[3 2],{'tansig','purelin'});

Сеть Элмана моделируется для последовательности из 3 величин с использованием начальных задержек, установленных по умолчанию:

p1 = {0.2 0.7 0.1};

[y1,pf,af] = sim(net,p1)

Сеть моделируется еще для 4 значений, использующих конечные задержки предыдущего моделирования в качестве новых начальных задержек:

p2 = {0.1 0.9 0.8 0.4};

[y2,pf,af] = sim(net,p2,pf,af)

Алгоритм:

SIM использует следующие свойства для моделирования сети NET:

NET.numInputs, NET.numLayers

NET.outputConnect, NET.biasConnect

NET.inputConnect, NET.layerConnect

Эти свойства определяют значения весов и смещений сети, а также количество задержек, связанных с каждым весом:

NET.inputWeights{i,j}.value

NET.layerWeights{i,j}.value

NET.layers{i}.value

NET.inputWeights{i,j}.delays

NET.layerWeights{i,j}.delays

Следующие свойства функции указывают, каким образом SIM применяет значения весов и смещений к входам, чтобы получить выход каждого слоя:

NET.inputWeights{i,j}.weightFcn

NET.layerWeights{i,j}.weightFcn

NET.layers{i}.netInputFcn

NET.layers{i}.transferFcn

TRAIN - Тренировка нейронной сети

Синтаксис:

[net,tr,Y,E,Pf,Af] = train(NET,P,T,Pi,Ai,VV,TV)

Описание:

Тренирует сеть NET в соответствии с NET.trainFcn и NET.trainParam.

TRAIN(NET,P,T,Pi,Ai) в качестве входных параметров использует:
NET – сеть;
P – входы сети;
T – целевые значения, по умолчанию – нули;
Pi – начальные входные задержки, по умолчанию – нули;
Ai – начальные задержки слоев, по умолчанию – нули;
VV – структура векторов верификации, по умолчанию = [];
TV – структура тестовых векторов, по умолчанию = [];
и возвращает:
NET – новая сеть (тренированная);
TR – результат тренировки (количество эпох и функция выполнения);
Y – выходы сети;
E – ошибки сети;
Pf – окончательные входные задержки;
Af – окончательные задержки слоев.

Следует отметить, что T – необязательный параметр и используется только при обучении сетей, для которых необходимы эталонные значения. Pi и Pf также необязательные параметры и используются только для сетей с задержками входов и слоев. Необязательные аргументы VV и TV описаны ниже.

Аргументы TRAIN могут быть двух форматов: массив ячеек или матрица. Формат в виде массива ячеек является наиболее удобным для описания. Он наиболее удобен для описания сетей с многими входами и выходами и позволяет представлять последовательность входов:
P - NixTS массив ячеек, каждая ячейка P{i,ts} – матрица размером RixQ;
T - NtxTS массив ячеек, каждая ячейка T{i,ts} – матрица размером VixQ;
Pi - NixID массив ячеек, каждая ячейка Pi{i,k} – матрица размером RixQ;
Ai - NlxLD массив ячеек, каждая ячейка Ai{i,k} – матрица размером SixQ;
Y - NOxTS массив ячеек, каждая ячейка Y{i,ts} – матрица размером VixQ;
E - NtxTS массив ячеек, каждая ячейка E{i,ts} – матрица размером VixQ;
Pf - NixID массив ячеек, каждая ячейка Pf{i,k} – матрица размером RixQ;
Af - NlxLD массив ячеек, каждая ячейка Af{i,k} – матрица размером SixQ,

где
Ni = net.numInputs
Nl = net.numLayers
Nt = net.numTargets
ID = net.numInputDelays
LD = net.numLayerDelays
TS = number of time steps
Q = batch size
Ri = net.inputs{i}.size
Si = net.layers{i}.size
Vi = net.targets{i}.size

Столбцы Pi, Pf, Ai, and Af упорядочены от самых больших задержек к текущим:
Pi{i,k} = i – й вход в момент времени ts=k-ID.
Pf{i,k} = i – й вход в момент времени ts=TS+k-ID.
Ai{i,k} = i – й выход слоя в момент времени ts=k-LD.
Af{i,k} = i – й выход слоя в момент времени ts=TS+k-LD.

Матричный формат может быть использован тогда, когда должен быть смоделирован только один шаг по времени (TS = 1). Это удобно только для сетей с одним входом и с одним выходом. Однако, он может быть использован для сетей с большим количеством входов и выходов. Каждый аргумент матрицы находится путем накопления элементов соответствующих аргументам массивов элементов в одну матрицу:
P – матрица (сумма Ri)xQ;
T – матрица (сумма Vi)xQ;
Pi – матрица (сумма Ri)x(ID*Q);
Ai – матрица (сумма Si)x(LD*Q);
Y – матрица (сумма Ui)xQ;
E - матрица (сумма Vi)xQ;
Pf – матрица (сумма Ri)x(ID*Q);
Af – матрица (сумма Si)x(LD*Q).

Если VV and TV заданы, они должны быть пустыми матрицами или должны иметь структуру со следующими полями:
VV.P, TV.P – входы верификация/тест;
VV.T, TV.T - эталонные значения верификация/тест (по умолчанию – нули);
VV.Pi, TV.Pi – начальные входные задержки верификация/тест (по умолчанию – нули);
VV.Ai, TV.Ai – задержки слоев верификация/тест (по умолчанию – нули).

Векторы верификации используются для того, чтобы остановить тренировку раньше, если дальнейшая тренировка на исходных векторах будет ухудшать приближение к векторам верификации. Тестовый вектор функционирования может быть использован для измерения того, насколько хорошо сеть обеспечивает согласование между исходными векторами и векторами верификации. Если в качестве VV.T, VV.Pi, or VV.Ai выбраны пустые матрицы или массивы элементов, то будут использованы значения по умолчанию. Это правило справедливо также и для TV.T, TV.Pi, TV.Ai.

Не все функции тренировки поддерживают векторы верификации и тестовые векторы. Только те, которые не игнорируют аргументы VV и TV arguments.

Примеры:

Вход P и эталонные значения T определяют простую функцию, которую мы можем нарисовать:

```
p = [0 1 2 3 4 5 6 7 8];

t = [0 0.84 0.91 0.14 -0.77 -0.96 -0.28 0.66 0.99];

plot(p,t,'o')
```

Используем NEWFF , чтобы создать двухслойную однонаправленную сеть. Сеть имеет вход (диапазон входных значений от 0 до 8), слой из 10 нейронов с функцией активации TANSIG и слой из 1 нейрона с функцией активации PURELIN. Для обучения используется алгоритм обратного распространения и функция тренировки TRAINLM:

```
net = newff([0 8],[10 1],{'tansig' 'purelin'},'trainlm');
```

Моделируем сеть и сравниваем результат моделирования с эталонными значениями:

```
y1 = sim(net,p)

plot(p,t,'o',p,y1,'x')
```

Тренируем сеть 50 эпох до достижения ошибки 0.01 и моделируем снова:

```
net.trainParam.epochs = 50;

net.trainParam.goal = 0.01;

net = train(net,p,t);

y2 = sim(net,p)

plot(p,t,'o',p,y1,'x',p,y2,'*')
```

Алгоритм:

TRAIN вызывает функцию, которая задается NET.trainFcn, использующую параметры тренировки, задаваемые NET.trainParam.

Обычно одна эпоха тренировки определяется как однократное предъявление всех входных векторов. Затем сеть модифицируется в соответствии с результатами всех таких предъявлений.

Тренировка продолжается до тех пор, пока не будет достигнуто заданное значение ошибки функционирования или пока не будет достигнуто одно из условий остановки из заданных в соответствии с видом выбранной функции NET.trainFcn.

Некоторые функции отличаются от этого стандарта тем, что предъявляется только один вектор (или последовательность) за каждую эпоху. Входной вектор (или последовательность) выбирается случайным образом для каждой эпохи из конкурирующих входных векторов (или последовательностей). NEWC и NEWSOM возвращают сети, которые используют TRAINR.

COMPET - Конкурентная функция активации

Синтаксис:

A = compet(N)
info = compet(code)

Описание:

COMPET – функция активации. Функция активации вычисляет выход слоя по его входу.

COMPET(N) имеет один входной аргумент – N - SxQ матрицу входных векторов (столбцов) и возвращает выходные векторы с единицей в позиции, где входной вектор имеет максимальное значение и нулями в остальных позициях.

COMPET(code) – возвращает информацию об этой функции.

Значения "code":
'deriv' – вид производной от функции активации;
'name' - полное название;
'output' – диапазон выходных значений;
'active' – диапазон входных значений.

COMPET не имеет производной.

Примеры.

Задаем входной вектор N, вычисляем выход и представляем в графическом виде вход и выход.

```
n = [0; 1; -0.5; 0.5];
a = compet(n);
subplot(2,1,1), bar(n), ylabel('n')
subplot(2,1,2), bar(a), ylabel('a')
```

Использование в сети.

Для использования функции необходимо вызвать NEWC или NEWPNN.

Для того, чтобы использовать в сети функцию COMPET, необходимо выполнить следующие установки:

NET.layers{i,j}.transferFcn установить как 'compet'.

MINMAX - Вычисляет минимальные и максимальные значения строк матрицы

Синтаксис:

pr = minmax(p)

Описание:

Для матрицы P с размерностью RxQ функция PR=MINMAX(P) возвращает Rx2 матрицу PR минимальных и максимальных значений для каждой строки матрицы P.

Пример:

p = [0 1 2; -1 -2 -0.5]

pr = minmax(p)

Список функций Neural Network Toolbox

Функции сгруппированы по назначению.

Функции для анализа
rrsurf – поверхность ошибки нейрона с единственным входом
maxlinlr – максимальная скорость обучения для линейного нейрона

Функции отклонения
boxdist – расстояние между двумя векторами
dist – евклидова весовая функция отклонения
linkdist – связанная фунция отклонения
mandist – весовая функция отклонения Манхеттена

Функции графического интерфейса
nntool – вызов графического интерфейса пользователя

Функции инициализации слоя
initnw – функция инициализации Нгуен-Видроу (Nguyen-Widrow)
initwb – функция инициализации по весам и смещениям

Функции обучения

learncon – обучающая функция смещений
learngd – обучающая функция градиентного спуска
learngdm – обучающая функция градиентного спуска с учетом моментов
learnh – обучащая функция Хэбба
learnhd – обучающая функция Хэбба с учетом затухания
learnis – обучающая функция instar
learnk – обучающая функция Кохонена
learnlv1 – LVQ1 обучающая функция
learnlv2 – LVQ2 обучающая функция
learnos – outstar обучающая функция
learnp – обучающая функция смещений и весов перцептрона
learnpn - обучающая функция нормализованных смещений и весов перцептрона
learnsom – обучающая функция самоорганизующейся карты весов
learnwh – правило обучения Уидроу-Хоффа (Widrow-Hoff)

Линейные функции поиска
srchbac – одномерная минимизация с использованием поиска с возвратом
srchbre – одномерная локализация интервала с использованием метода Брента (Brent)
srchcha - одномерная минимизация с использованием метода Караламбуса (Charalambous)
srchgol - одномерная минимизация с использованием золотого сечения
srchhyb - одномерная минимизация с использованием гибридного бисекционного поиска

Функции вычисления производных от входов сети
dnetprod – вычисление производной от входов сети с перемножением входов
dnetsum – вычисление производной от входов сети с суммированием входов

Входные функции сети
netprod – функция произведения входов
netsum – функция суммирования входов

Функции инициализации сети
initlay – функция послойной инициализации сети

Функции использования сети
adapt – разрешает адаптацию сети
disp – отображает свойства нейронной сети
display – отображает имена переменных и свойства сети
init – инициализация нейронной сети
sim – моделирование нейронной сети
train – тренировка нейронной сети

Функции создания новой сети
network – создание нейронной сети пользователя
newc – создание конкурентного слоя
newcf – создание каскадной направленной сети
newelm – создание сети обратного распространения Элмана (Elman)
newff – создание однонаправленной сети
newfftd – создание однонаправленной сети с входными задержками
newgrnn – создание обобщенной регрессонной нейронной сети
newhop – создание рекуррентной сети Хопфилда
newlin – создание линейного слоя
newlind – конструирование линейного слоя

newlvq – создание квантованной сети
newp – создание перцептрона
newpnn – конструирование вероятностной нейронной сети
newrb – конструирование сети с радиальным базисом
newrbe – конструирование точной сети с радиальными базисными функциями
newsom – создание самоорганизующейся карты

Функции производных функционирования
dmae – средняя абсолютная ошибка вычисления производной
dmse – средне-квадратичная ошибка производной
dmsereg - средне-квадратичная ошибка производной w/reg
dsse – суммарная квадратичная ошибка производной

Функции выполнения
mae - средняя абсолютная ошибка
mse – средне-квадратичная ошибка
msereg - средне-квадратичная ошибка w/reg
sse - суммарная квадратичная ошибка

Функции графики
hintonw – график Хинтона для матрицы весов
hintonwb – график Хинтона для матрицы весов и векторов смещений
plotbr – график функционирования сети при регулярной тренировке (Bayesian)
plotep – изображение положений весов и смещений на поверхности ошибки
plotes – изображение поверхности ошибок единичного входного нейрона
plotpc – изображение линии классификации в векторном пространстве перцептрона
plotperf – графическое представление функционирования сети
plotpv - графическое представление входных целевых векторов
plotsom – графическое представление самоорганизующейся карты
plotv – графическое представление векторов в виде линий, выходящих из начала координат
plotvec - графическое представление векторов различными цветами

Функции предварительной и пост обработки
postmnmx – ненормализованные данные, которые были нормализованы посредством prenmmx
postreg – линейный регрессионный анализ выходов сети по отношению к целевым значениям обучающего массива
poststd – ненормированные данные, которые были нормированы с помощью функции prestd
premnmx – нормирование данных в диапазоне от –1 до +1
prepca – анализ главных компонент для входных данных
prestd – нормирование данных к единичному стандартному отклонению и нулевому среднему
tramnmx – преобразование данных с предварительно вычисленными минимумом и максимумом
trapca – преобразование данных с использованием PCA матрицы, вычисленной с помощью функции prepca
trastd – преобразование данных с использованием предварительно вычисленных значений стандартного отклонения и среднего

Функции поддержки Simulink

gensim – генерация блока Simulink для моделирования нейронной сети

Топологические функции

gridtop – топологическая функция в виде сеточного слоя
hextop – топологическая функция в виде гексагонального слоя
randtop – топологическая функция в виде случайного слоя

Функции тренировки

trainb – пакетная тренировка с использованием правил обучения для весов и смещений
trainbfg – тренировка сети с использованием квази–Ньютоновского метода BFGS
trainbr – регуляризация Bayesian
trainc – использование приращений циклического порядка
traincgb – метод связанных градиентов Пауэлла-Била (Powell-Beale)
traincgf - метод связанных градиентов Флетчера-Пауэлла (Fletcher-Powell)
traincgp - метод связанных градиентов Полака-Рибира (Polak-Ribiere)
traingd – метод градиентного спуска
traingda – метод градиентного спуска с адаптивным обучением
traingdm - метод градиентного спуска с учетом моментов
traingdx - метод градиентного спуска с учетом моментов и с адаптивным обучением
trainlm – метод Левенберга-Маркара (Levenberg-Marquardt)
trainoss – одноступенчатый метод секущих
trainr – метод случайных приращений
trainrp – алгоритм упругого обратного распространения
trains – метод последовательных приращений
trainscg - метод шкалированных связанных градиентов

Производные функций активации

dhardlim – производная ступенчатой функции активации
dhardlms – производная симметричной ступенчатой функции активации
dlogsig – производная сигмоидной (логистической) функции активации
dposlin – производная положительной линейной функции активации
dpurelin – приозводная линейной функции активации
dradbas – производная радиальной базисной функции активации
dsatlin – производная насыщающейся линейной функции активации
dsatlins – производная симметричной насыщающейся функции активации
dtansig – производная функции активации гиперболический тангенс
dtribas – производная треугольной функции активации

Функции активации

compet – конкурирующая функция активации
hardlim – ступенчатая функция активации
hardlims – ступенчатая симметричная функция активации
logsig – сигмоидная (логистическая) функция активации
poslin – положительная линейная функция активации
purelin – линейная функция активации
radbas – радиальная базисная функция активации
satlin – насыщающаяся линейная функция активации
satlins – симметричная насыщающаяся линейная функция активации
softmax – функция активации, уменьшающая диапазон входных значений
tansig – функция активации гиперболический тангенс
tribas – треугольная функция активации

Полезные функции

calca – вычисляет выходы сети и другие сигналы
calca1 – вычисляет сигналы сети для одного шага по времени
calce – вычисляет ошибки слоев
calce1 - вычисляет ошибки слоев для одного шага по времени
calcgx – вычисляет градиент весов и смещений как единственный вектор
calcjejj – вычисляет Якобиан
calcjx – вычисляет Якобиан весов и смещений как одну матрицу
calcpd – вычисляет задержанные входы сети
calcperf – вычисление выходов сети, сигналов и функционирования
formx – формирует один вектор из весов и смещений
getx – возвращает все веса и смещения сети как один вектор
setx – устанавливает все веса и смещения сети в виде одного вектора

Векторные функции

cell2mat – объединяет массив элементов матриц в одну матрицу
combvec – создает все комбинации векторов
con2seq – преобразует сходящиеся векторы в последовательные векторы
concur – создает сходящиеся векторы смещений
ind2vec – преобразование индексов в векторы
mat2cell – разбиение матрицы на массив элементов матриц
minmax – вычисляет минимальные и максимальные значения строк матрицы
normc – нормирует столбцы матрицы
normr – нормирует строки матрицы
pnormc – псевдо-нормировка столбцов матрицы
quant – дискретизация величины
seq2con – преобразование последовательных векторов в сходящиеся векторы
sumsqr – сумма квадратов элементов матрицы
vec2ind – преобразование векторов в индексы

Функции инициализации весов и смещений

initcon – "сознательная" функция инициализации
initzero – инициализация с установкой нулевых значений весов и смещений
midpoint – инициализация с установкой средних значений весов
randnc – инициализация с установкой нормализованных значений столбцов весовых матриц
randnr - инициализация с установкой нормализованных значений строк весовых функций
rands – инициализация с установкой симметричных случайных значений весов и смещений
revert – возвращение весам и смещениям значений, соответствующих предыдущей инициализации

Функции весовых производных

ddotprod – производная скалярного произведения

Весовые функции

dist – Евклидово расстояние
dotprod – весовая функция в виде скалярного произведения
mandist – весовая функция – расстояние Манхеттена
negdist – весовая функция – отрицательное расстояние
normprod – нормированное скалярное произведение