НЕЙРОННЫЕ СЕТИ

ПЕРСЕПТРОН
И
СЛОЙ
КОХОНЕНА

Задание

1.Из данных варианта создать две выборки: обучающую и тестирующую так, чтобы в них было примерно поровну экземпляров каждого класса

2.По обучающей выборке (по 2 каким-либо классам) создать персептрон и обучить его

3.Протестировать персептрон по обучающей и тестирующей выборкам тех же двух классов, определив процент верно распознанных объектов (по каждому классу)

4.Построить графики разделяющей поверхности и визуальное представление результатов тестирования

5.По обучающей выборке (по всем классам) построить конкурирующую сеть (сеть Кохонена) и обучить ее

6.Протестировать конкурирующую сеть по обучающей и тестирующей выборкам, определив процент верно распознанных объектов (по каждому классу)

7.Построить графики разделяющей поверхности и визуальное представление результатов тестирования

8.Оформить результаты в виде таблицы

Процент верно классифицированных объектов обучающей выборки с помощью персептрона	Процент верно классифицированных объектов тестирующей выборки с помощью персептрона	Процент верно классифицированных объектов обучающей выборки с помощью сети Кохонена	Процент верно классифицированных объектов тестирующей выборки с помощью сети Кохонена

Краткое описание Neural Network Toolbox

Neural Network Toolbox - набор инструментов для работы с ИНС (искусственные нейронные сети) предоставляет функции и приложения для моделирования сложных нелинейных систем.

Neural Network Toolbox поддерживает обучение с учителем для сетей прямого распространения, сетей на основе радиальных базисных функций и динамических сетей. Он также поддерживает обучение без учителя для карт самоорганизации и конкурирующих слоев. Благодаря этому набору инструментов, становится возможным проектирование, обучение, визуализация и моделирование нейронных сетей. Для повышения скорости обучения и обращения к большим массивам данных, пользователь может распределять вычисления и данных между многоядерными процессорами, графическими процессорами и компьютерными кластерами, используя пакет для параллельных вычислений.

Краткое описание Neural Network Toolbox

Ключевые понятия:

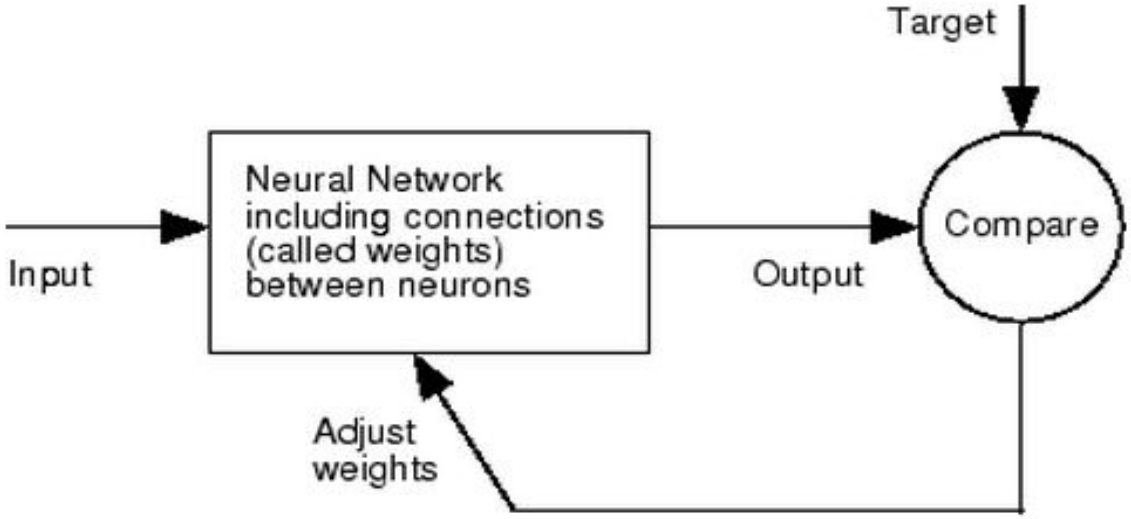
- Сети обучающиеся с учителем: многослойные, радиальные базисные, LVQ, нелинейной авторегрессии и рекуррентные сети.
- Сети обучающиеся без учителя: карты самоорганизации, конкурирующие слои.
- Приложения для аппроксимации и фильтрации сигналов, распознавания образов и кластеризации.
- Модульное представление сети для управления и визуализации сетей различных размеров

Введение в нейронные сети

Нейронные сети состоят из простых элементов функционирующих параллельно. Название этих элементов пришло из биологических нервных систем. Как и в природе, связи между элементами четко определены функцией сети. Пользователь способен обучать нейронную сеть выполнению определенной функции путем настройки значений связи между этими элементами (веса синапсов).

Обычно нейронные сети настроены или обучены так, чтобы конкретный входной вектор приводил к конкретному выходному сигналу.

Введение в нейронные сети



На рисунке настройка сети проходит по принципу сравнения выходного сигнала и целевого вектора (Target). Это происходит до тех пор, пока выход сети не соответствует целевому вектору. Как правило, для обучения сети требуется достаточно большое количество таких пар (входной вектор/целевой вектор).

Нейронные сети могут быть обучены для выполнения сложных функций в различных областях деятельности, включая распознавание образов, синтез речи, систем управления и многих других.

Как использовать NNT?

Существует 4 способа использовать набор инструментов для работы с ИНС.

- Первый способ – через графический интерфейс пользователя(GUIs). Интерфейс обеспечивает простой и удобный способ доступа к возможностям набора инструментов.
- Второй способ – использование инструментов посредством стандартной командной строки в системе MATLAB. Операции командной строки обеспечивают большую гибкость, чем GUIs, но несколько сложнее в использовании. При знакомстве с набором инструментов графический интерфейс будет более информативен.
- Третий способ – использование скриптов и функций. Это расширенные возможности для создания своих собственных нейронных сетей. Есть возможность создавать сети с произвольными связями и обучать их, используя существующие функции из инструментария.
- Четвертый способ – использование набора инструментов, изменяя любую стандартную функцию из этого набора.

Описанные уровни использования подходят и для новичка, и для эксперта в области ИНС.

Где применяются нейронные сети?

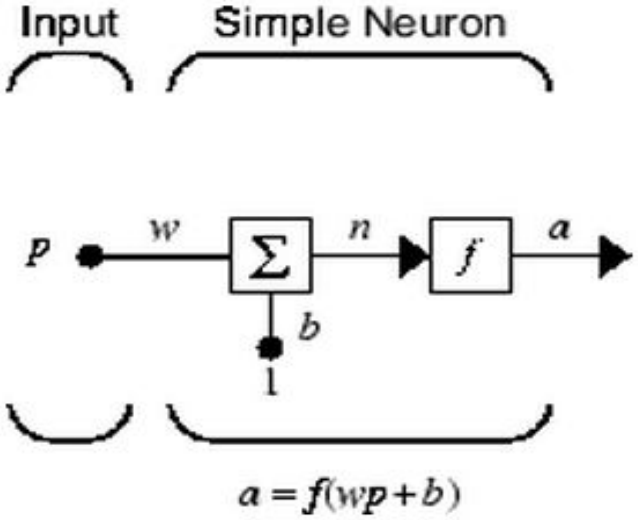
Отрасль	Практическое применение
Аэрокосмическая	Автопилотирование, моделирование траектории полета и т.д.
Автомобильная	Автомобильная система автоматического управления
Банковское дело	Оценка заявки на кредит
Оборона	Наведение оружия, распознавание лиц, новые виды сенсоров, радаров
Электроника	Управление технологическими процессами, анализ отказа чипа, синтез голоса
Развлечения	Анимация, спец. эффекты
Финансы	Реальная оценка недвижимости, рейтинг корпоративных облигации
Медицина	Анализ раковых клеток, проектирование протезов
Нефтегазовая	Разведка месторождений
Телекоммуникации	Сжатие данных, перевод речи в режиме реального времени
Транспортная	Проектирование маршрутов

Основные этапы проектирования ИНС

Приведены стандартные этапы проектирования нейронной сети. Всего их можно поделить на 7 этапов:

1. Сбор данных
2. Инициализация сети
3. Конфигурирование сети
4. Инициализация весов и смещений
5. Обучение сети
6. Проверка сети
7. Использование конечным пользователем

Модель нейрона
Простой нейрон



Элементарной ячейкой нейронной сети является нейрон.

Три отдельных функциональных операций:

- Скалярный вход сигнал p умножается на весовой коэффициент w. В результате получаем скалярную величину wp.
- Далее взвешенный вход wp суммируется со смещением b. В результате имеем величину n.
- Величина n – аргумент функции активации f, выход функции активации - скалярная величина a.

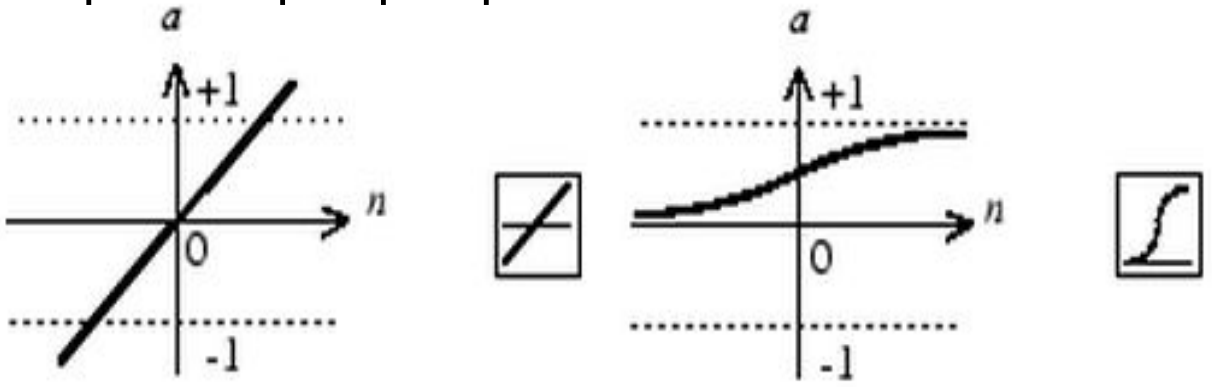
В некоторых случаях смещение b не используется.

Функции активации (передаточные функции)

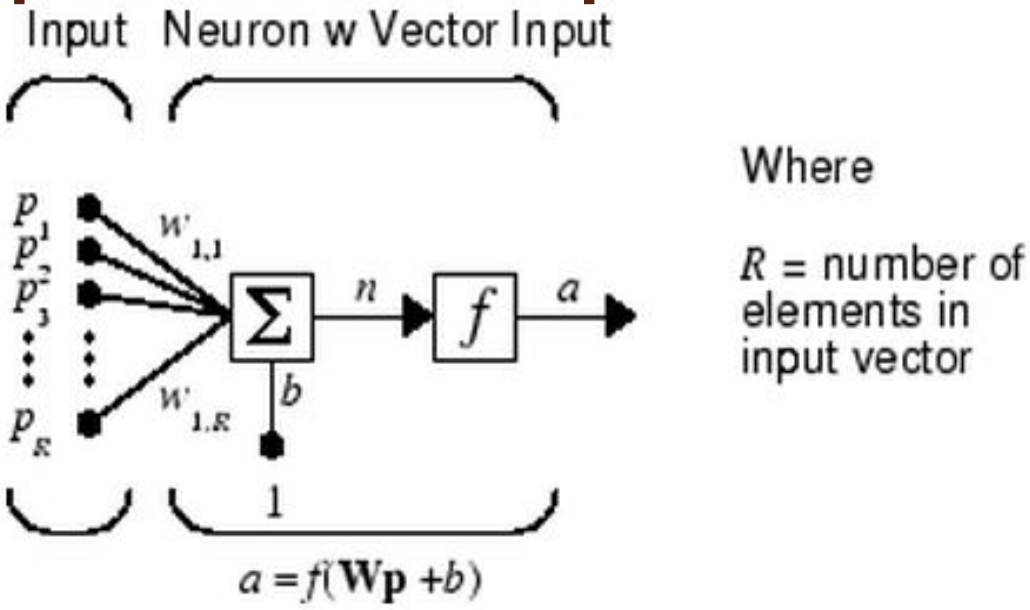
Набор инструментов содержит большой набор передаточных функций.

Две наиболее распространенные функции:

- Нейронные с линейной функцией активации используется в выходном слое.
- Сигмоидальная функция активации, аргумент функции может принимать значения от -∞ до +∞, а выход изменяется в диапазоне от 0 до 1. Благодаря свойству дифференцируемости эта функция часто используется в сетях с обучением на основе метода обратного распространения ошибки.



Нейрон с векторным входом

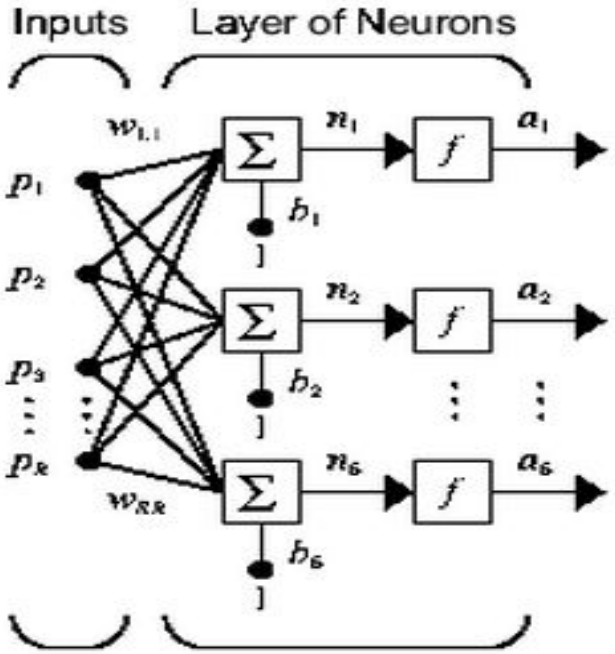


Простой нейрон может обрабатывать входные сигналы, представленные в виде вектора.

Нейрон с одним вектором входа p с R элементами: $p_1, p_2, \ldots, p_R$

Каждый элемент входа умножается на веса $w_{11}, w_{12}, \ldots, w_{1R}$ соответственно и взвешенные значения передаются на сумматор. Нейрон имеет смещение b, которое суммируется со взвешенной суммой входов. В результате получен аргумент n функции активации: $n = w_{11}p_1 + w_{12}p_2 +, \ldots, +w_{1R}p_R + b$.

Архитектура нейронных сетей



Where

R = number of elements in input vector

S = number of neurons in layer

Однослойная сеть с R входными элементами и S нейронами.

Каждый элемент вектора входа соединен со всеми входами нейрона и это соединение задается матрицей весов W.

Каждый i-ый нейрон включает суммирующий элемент, который формирует скалярный выход n(i). Совокупность скалярных функций n(i) объединяется в S-элементный вектор входа n функции активации слоя.

Выходы слоя нейронов формируют вектор-столбец a. $a = f(W * p + b)$

Количество входов R в слое может не совпадать с количеством нейронов S.

Архитектура нейронных сетей

Элементы вектора входа передаются через матрицу весов W.

Индексы строк матрицы W указывают адресатов весов нейронов, а индексы столбцов – какой источник является входом для этого веса.

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ & & & \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

Персептрон

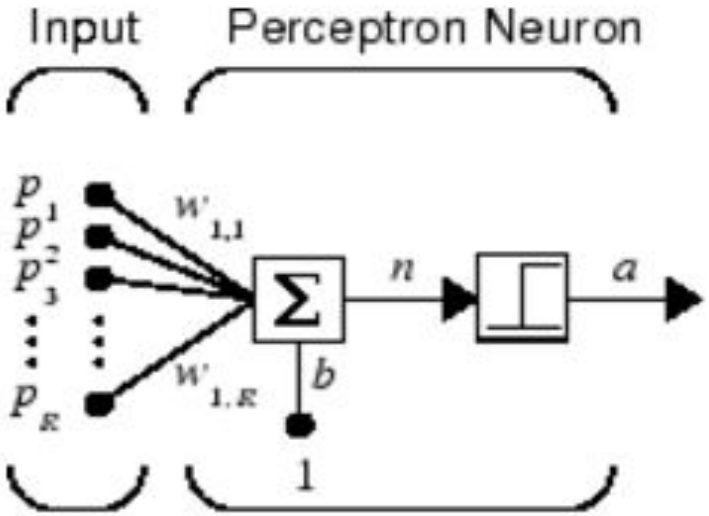
Большое количество моделей персептрона представлено в основополагающей работе Розенблатта.

Простейшая из таких моделей – однослойный персептрон, веса и смещение которого могут быть настроены таким образом, чтобы решить задачу классификации входных векторов.

Методика обучения называется правило обучения персептрона.

Персептроны лучше всего подходят для простых задач классификации образов. Это очень быстрый и надежный тип сетей для решения подобных задач. Кроме того, понимание принципа работы персептрона обеспечивает хорошую основу для понимания более сложных сетей.

Персептрон
Модель нейрона

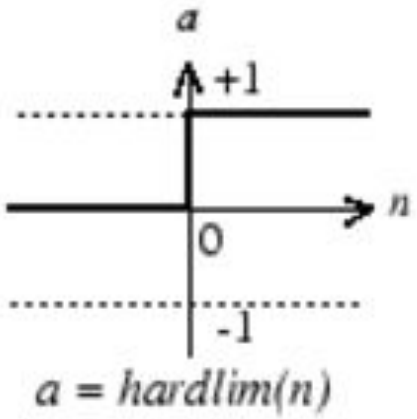


Where

R = number of elements in input vector

Нейрон, используемый в модели персептрона, имеет ступенчатую функцию активации hardlim с жесткими ограничениями.

Каждый элемент вектора входа персептрона взвешен с соответствующим весом w_{1j}, и их сумма является входом функции активации. Функция активации с жесткими ограничениями возвращает два значения: 0 и 1.

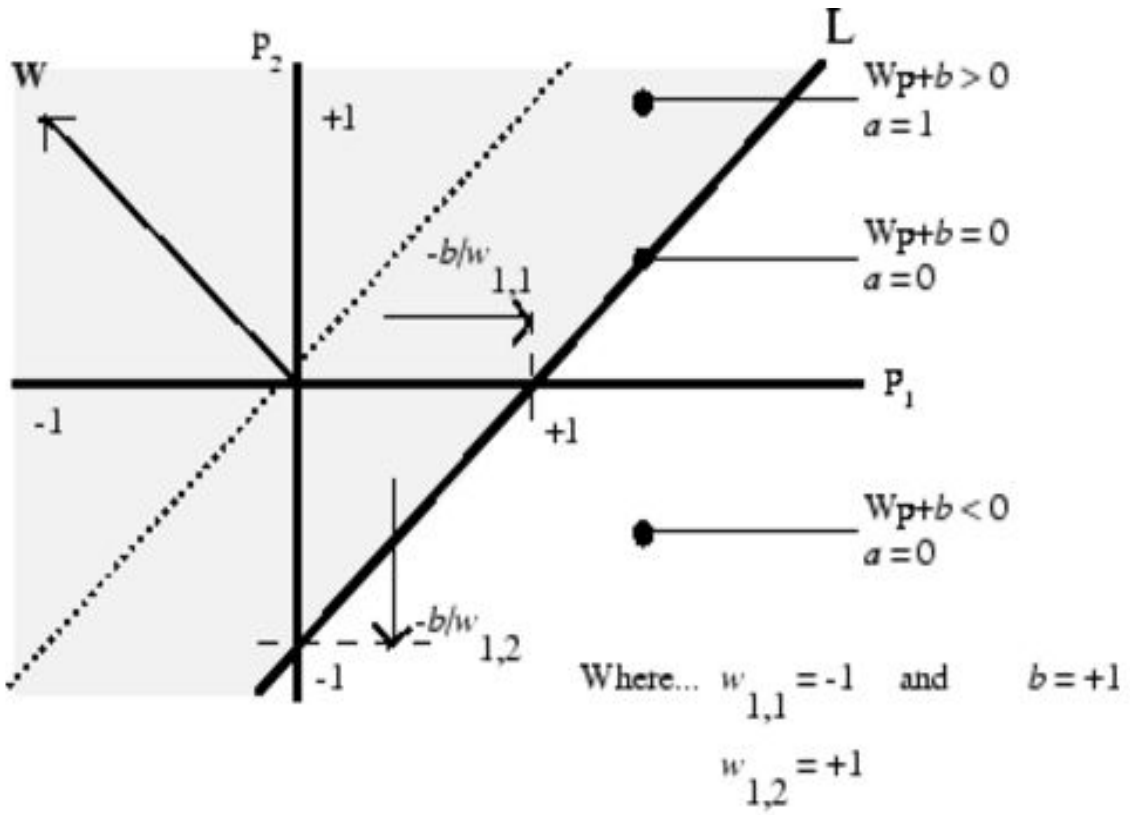


Hard-Limit Transfer Function

Персептрон
Модель нейрона

Нейрон персептрона возвращает 1, если вход функции активации $n \geq 0$; в противном случае 0.

Функция активации с жесткими ограничениями придает персептрону способность классифицировать векторы входа, разделяя пространство входов на 2 области.



Персептрон
Модель нейрона

Пространство входов делится на 2 области разделяющей линией L, которая для мерного случая задается уравнением:
.
$$Wp + b = 0$$

Эта линия перпендикулярна к вектору весов W и смещена на величину b.

Векторы входа выше линии L соответствуют положительному потенциалу нейрона, следовательно выход персептрона для этих векторов будет равен 1; векторы входа ниже линии L соответствуют выходу персептрона, равному 0.

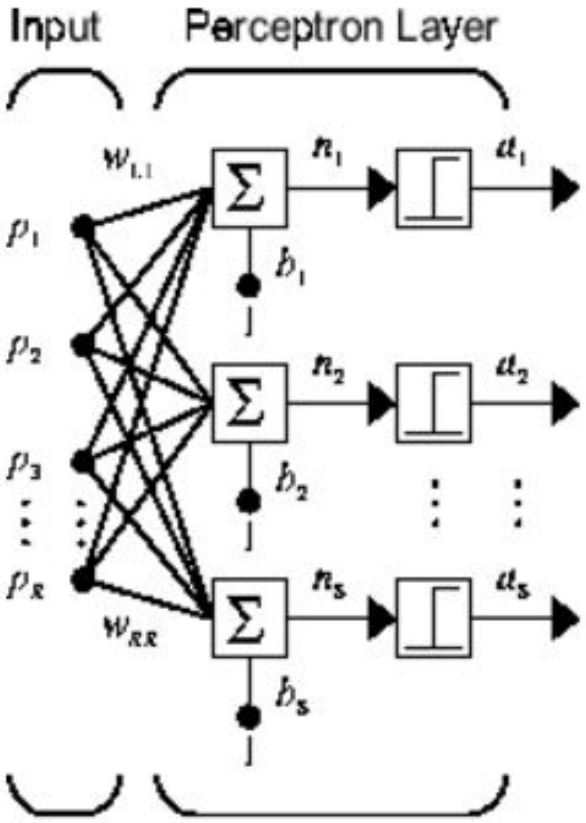
При изменении значений смещения и весов граница линии L изменяет свое положение. Персептрон без смещения всегда формирует распределяющую линию, проходящую через начало координат.

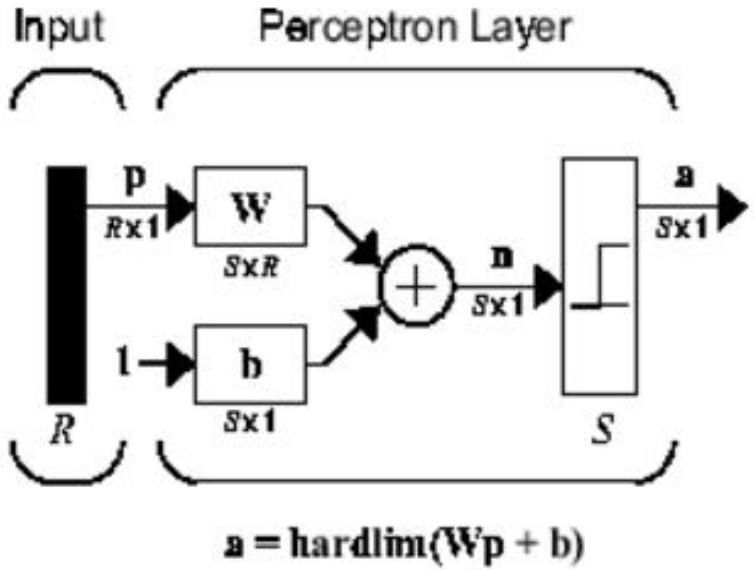
В случае, когда размерность вектора входа превышает 2, разделяющей границей будет служить гиперплоскость.

Архитектура персептрона

Персептрон состоит из единственного слоя, включающего S нейронов связанных с R входами.

W_{ij} -коэффициенты передачи от j-го входа к i-му нейрону.





Where

R = number of elements in input

S = number of neurons in layer

Обучение персептрона

Персептроны обучаются с учителем. При обучении с учителем задается множество примеров требуемого поведения сети, которое называется обучающим множеством

$$\mathbf{p}_1\mathbf{t}_1, \mathbf{p}_2\mathbf{t}_1, \ldots, \mathbf{p}_Q\mathbf{t}_Q$$

$p_1, p_2, \ldots, p_Q$ - входы персептрона,

$t_1, t_2, \ldots, t_Q$ - требуемые (целевые) выходы.

Цель обучения – уменьшить погрешность $e = t - a$, которая равна разности между реакцией нейрона a и вектором цели t.

Правило обучения персептрона learnp рассчитывает необходимые изменения весовых коэффициентов и смещений с учетом входного вектора p и погрешности e. Целевой вектор t может включать только значения 0 и 1, т.к. используется функция активации hardlim. Каждый раз при выполнении функции learnp будет происходить перенастройка параметров персептрона. Доказано, что если решение существует, то процесс обучения персептрона сходится за конечное число итераций.

Обучение персептрона

Если смещение не используется, функция learnp ищет решение, изменяя только вектор весов W Это приводит к нахождению разделяющей линии, перпендикулярной вектору W и которая должным образом разделяет векторы входа.

Правило настройки (обучения) персептрона можно записать, связав изменение вектора веса ΔW с погрешностью $e = t - a$:

$$\Delta W = \begin{cases} 0, \text{если } e = 0; \\ p, \text{если } e = 1; \\ -p, \text{если } e = -1. \end{cases}$$

Все 3 случая можно записать иначе: $\Delta W = (t - a)p = ep$

Аналогичное выражение для изменения смещения (смещение рассматривается как вес единичного входа): $\Delta b = (t - a)1 = e$

В случае нескольких нейронов эти соотношения выглядят:

$$\begin{cases} \Delta W = (t - a)p^T = ep^T; \\ \Delta b = (t - a) = e. \end{cases}$$

Конечный вид правила обучения персептрона:

$$\begin{cases} W^{new} = W^{old} + ep^T; \\ b^{new} = b^{old} + e. \end{cases}$$

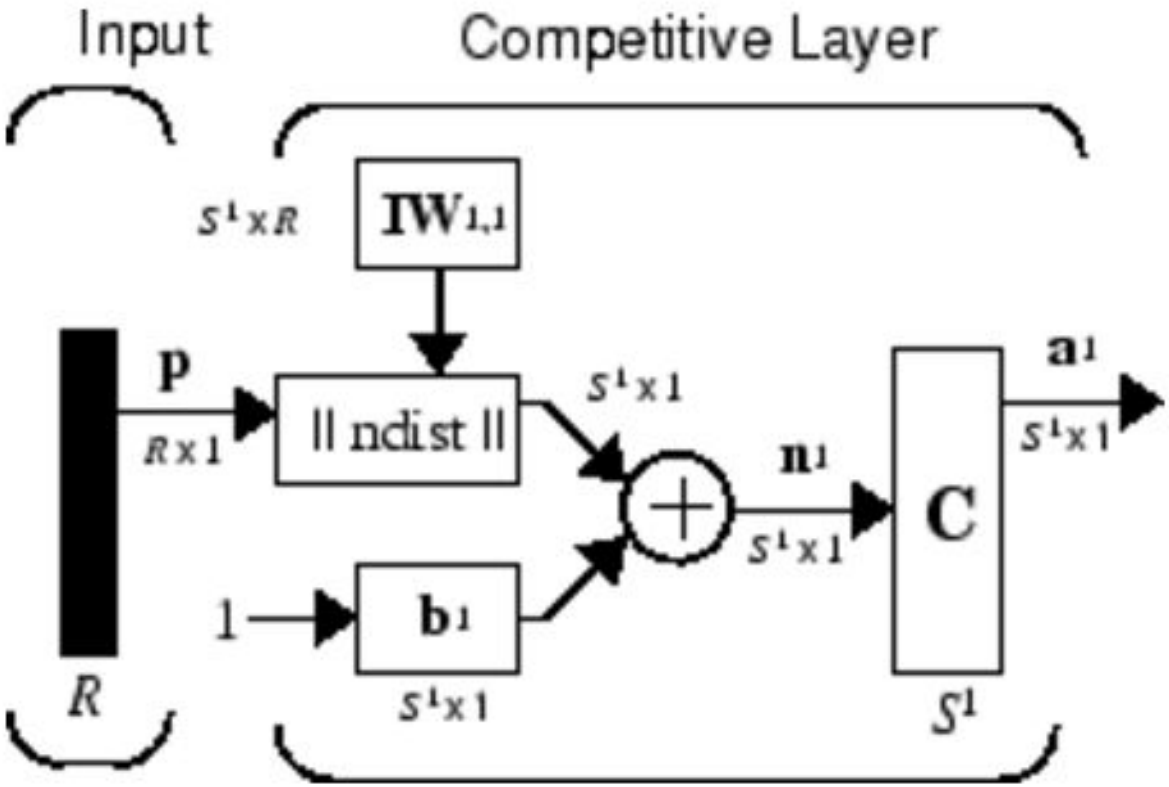
Слой Кохонена (конкурирующий слой)

В процессе анализа больших информационных массивов данных неизменно возникают задачи, связанные с исследованием топологической структуры данных, их объединением в группы (кластеры). Такие задачи могут быть успешно решены с применением специального класса самоорганизующихся нейронных сетей.

Свойство самоорганизации является одним из наиболее привлекательных свойств нейронных сетей. Таких свойством обладают самоорганизующиеся сети, описанные финским ученым Т. Кохоненом.

Нейроны самоорганизующейся сети могут быть обучены выявлению групп (кластеров) векторов входа, обладающих некоторыми общими свойствами.

Архитектура сети Кохонена



Блок ndist вычисляет отрицательные евклидовые расстояния между вектором входа p и строками матрицы весов IW^{11}.

Вход функции активации n^1 - это результат суммирования вычисленного расстояния с вектором смещения b.

Архитектура сети Кохонена

Если все смещения нулевые, максимальное значение n^1 не может превышать 0. Нулевое значение возможно только тогда, когда вектор входа p оказывается равным вектору веса одного из нейронов. Если смещения отличны от 0, то возможны и положительные значения для элементов вектора n^1.

Конкурирующая функция активации анализирует значения элементов вектора n^1 и формирует выходы нейронов, равные 0 для всех нейронов, кроме одного нейрона-победителя, имеющего на входе максимальное значение. Таким образом, вектор выхода слоя a^1 имеет единственный элемент, равный 1, который соответствует нейрону-победителю, а остальные равны 0.

Обучение сети Кохонена (learnk)

Отличительной чертой процесса обучения данной сети является то, что необходимо настроить веса синапсов нейронов, а не минимизировать ошибку обучения.

Обучение происходит по правилу Кохонена:

$$_iIW_{11}(q) = {}_iIW_{11}(q-1) + \alpha(p(q) - {}_iIW_{11}(q-1))$$

Правило Кохонена представляет собой рекуррентное соотношение, которое обеспечивает коррекцию строки i матрицы весов добавлением взвешенной разности вектора входа и значения строки на предыдущем шаге. Таким образом, вектор веса, наиболее близкий к вектору входа, модифицируется так, чтобы расстояние между ними стало еще меньше. Результат такого обучения будет заключаться в том, что победивший нейрон, вероятно, выиграет конкуренцию и в том случае, когда будет представлен новый входной вектор, близкий к предыдущему, и его победа менее вероятна, когда будет представлен вектор, существенно отличающийся от предыдущего. Когда на вход сети поступает все большее и большее число векторов, нейрон, являющийся ближайшим, снова корректирует свой весовой вектор. В конечном счете, если в слое имеется достаточное количество нейронов, то каждая группа близких векторов окажется связанной с одним из нейронов слоем.

Правило настройки смещений (learncon)

Одно из ограничений любого конкурирующего слоя состоит в том, что некоторые нейроны оказываются незадействованными. Это проявляется в том, что нейроны, имеющие начальные весовые векторы, значительно удаленные от векторов входа, никогда не выигрывают конкуренции, независимо от того как долго продолжается обучение.

В результате оказывается, что такие векторы не используются при обучении и соответствующие нейроны никогда не оказываются победителями. Их называют «мертвыми» нейронами.

Чтобы исключить такую ситуацию и сделать нейроны чувствительными к поступающим на вход векторам, используются смещения, которые позволяют нейрону стать конкурентным с нейронами-победителями. Этому способствует положительное смещение, которое добавляется к отрицательному расстоянию удаленного нейрона.

Правило настройки смещений (learncon)

В начале процедуры настройки всем нейронам конкурирующего слоя присваивается одинаковый параметр активности:

$$c_0 = \frac{1}{N}$$

N – количество нейронов конкурирующего слоя, равное числу кластеров. В процессе настройки learncon корректирует этот параметр таким образом, чтобы его значения для активных нейронов становились больше, а для неактивных меньше.

Формула для вектора приращений параметров активности:

$$\Delta c = lr * (a_i^1 - c)$$

lr - параметр скорости настройки, a_i^1 - вектор, элемент которого равен 1 (все остальные равны 0).

Правило настройки смещений (learncon)

Для всех нейронов, кроме нейрона-победителя, приращения отрицательны.

Параметр активности связан со смещением соотношением:

$$b = \exp(1)./c$$

Смещение для нейрона-победителя уменьшится, а смещение для остальных нейронов немного увеличится.

Формула для расчета приращений вектора смещений:

$$\Delta b = \exp(1 - \log(c)) - b$$

Увеличение смещений для неактивных нейронов позволяет расширить диапазон покрытия входных значений, и неактивный нейрон начинает формировать кластер.

Правило настройки смещений (learncon)

Два преимущества функции learncon:

- Если нейрон не выигрывает конкуренции, потому что его вектор весов существенно отличается от векторов, поступающих на вход сети, то его смещение по мере обучения становится достаточно большим и он становится конкурентоспособным. Как только нейрон начинает побеждать, его смещение начинает уменьшаться.

- Настройка смещений позволяет выравнивать значения параметра активности и обеспечивает притяжение приблизительно одинакового количества векторов входа. Если один из кластеров притягивает большее число векторов входа, чем другой, то более заполненная область притягивает дополнительное количество нейронов и будет поделена на меньшие по размерам кластеры.

Команды Matlab для работы с нейронными сетями (работа с персептроном)

Net=newp(minmax,p)

Создание персептрона

Minmax – матрица минимальных и максимальных значений признаков

P – число нейронов

Обучение персептрона выполняется с помощью функции **train**, которая корректирует веса и смещения в соответствии с обучающим правилом.

net = train(net,P,T)

Вход:

net - сеть

P - входы - матрица RxQ из Q входных векторов – столбцов размерности f R каждый.

T - целевые значения - матрица SxQ matrix из Q целевых

Команды Matlab для работы с нейронными сетями (работа с персептроном)

Распознавание по нейронной сети

Y = **sim**(net,P)

Входные данные:

net - сеть

P - вектора - столбцы с исходными данными

Выходные данные:

Y - вектора – столбцы с результатами

Команды Matlab для работы с нейронными сетями (работа с персептроном)

С помощью функции **plotpv** можно изображать пары вход-выход (для двух или трех признаков).

plotpv(P,T) выводит вектора- столбцы P символами в соответствии с T.

С помощью функции **plotpc** изображается разделяющая поверхность координатной плоскости на основе весов и смещения сети при числе признаков 2 или 3.

plotpc(W,B), где:

W - S x R матрица весов

B - S x 1 вектор смещений

Команды Matlab для работы с нейронными сетями (работа с персептроном)

Функция **minmax** определяет диапазоны каждого признака матрицы

pr = minmax(P)

Входной параметр

P - матрица R x Q

Выходной параметр

pr - матрица R x 2 минимальных и максимальных значений для каждой строки P.

Команды Matlab для работы с нейронными сетями (работа с персептроном)

Пример с распознаванием ирисов Фишера:

```
load fisheriris
tr1=meas(1:25,:);
tr2=meas(51:75,:);
trr=[tr1;tr2];
tr=trr';
out(1:25)=0;
out(26:50)=1;
cr1=meas(26:50,:);
cr2=meas(76:100,:);
crr=[cr1;cr2];
cr=crr';
P=tr(1:3,:);
P1=cr(1:3,:)
mmx=minmax(P);
```

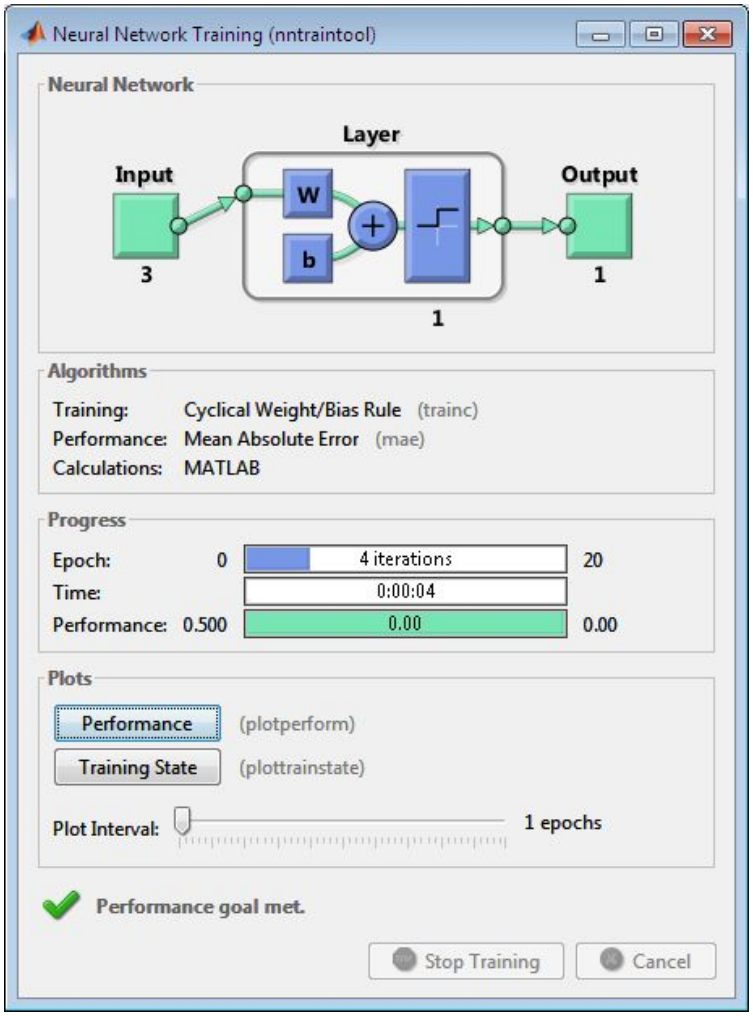
Команды Matlab для работы с нейронными сетями (работа с персептроном)

```
net=newp(mmx,1);
subplot(2,2,1);
plotpv(P,out);
hold on
plotpc(net.IW{1},net.b{1});%Разделяющая поверхность  по весам и смещениям
grid on
net.trainParam.epochs = 20;%Число итераций
[net,rez] = train(net,P,out);
rez
subplot(2,2,2);
hold off
plotpv(P,out);
```

Команды Matlab для работы с нейронными сетями (работа с персептроном)

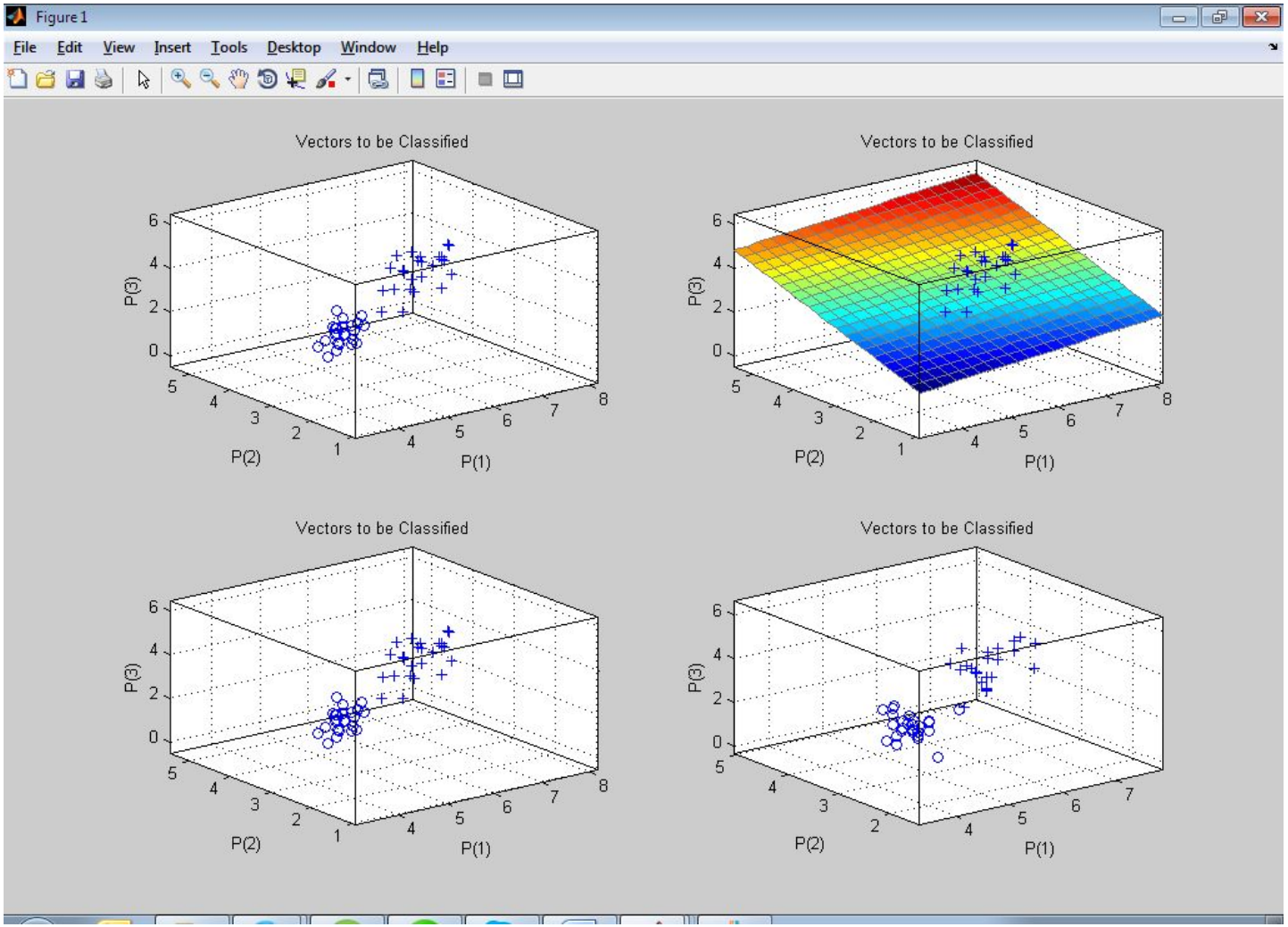
```
hold on;
plotpc(net.IW{1},net.b{1});%Разделяющая поверхность  по весам и смещениям
grid on
hold off;
a1=sim(net,P);%Распознавание объектов обучающей выборки
subplot(2,2,3);
plotpv(P,a1);
grid on
a2=sim(net,P1);%Распознавание объектов контролирующей выборки
subplot(2,2,4);
plotpv(P1,a2);
grid on
```

Архитектура сети



Команды Matlab для работы с нейронными сетями (работа с персептроном)

Результат работы скрипта



Команды Matlab для работы с нейронными сетями (работа с сетью Кохонена)

Для формирования слоя Кохонена предназначена функция **competlayer**(число классов). Она возвращает слой конкурирующей сети.

Функция **configure**(сеть, матрица наблюдений) осуществляет первоначальную настройку весов. Она возвращает настроенную сеть.

Функция **train**(сеть,обучающая выборка) обучает сеть на обучающей выборке. Она возвращает обученную сеть.

Функция **net**(выборка) выполняет вычисление по сети и возвращает матрицу, число строк в которой равно числу нейронов, а число столбцов числу объектов. Все элементы каждого столбца, кроме одного, равны 0. Единичный столбец определяет номер победившего нейрона.

Команды Matlab для работы с нейронными сетями (работа с сетью Кохонена)

Пример с ирисами Фишера, в котором используются все 3 класса и 3 признака для визуализации.

```
load fisheriris
tr1=meas(1:25,1:3);
tr2=meas(51:75,1:3);
tr3=meas(101:125,1:3);
trr=[tr1;tr2;tr3];
tr=trr';
cr1=meas(26:50,1:3);
cr2=meas(76:100,1:3);
cr3=meas(126:150,1:3);
crr=[cr1;cr2;cr3];
cr=crr';
```

Команды Matlab для работы с нейронными сетями (работа с сетью Кохонена)

```
subplot(2,2,1)
plot3(tr(1,:),tr(2,:),tr(3,:),'.b');
grid on
title('Обучающая выборка');
xlabel('1 признак');
ylabel('2 признак');
zlabel('3 признак');
net = competlayer(3,.1);
net = configure(net,tr);
w = net.IW{1};
subplot(2,2,2)
plot3(tr(1,:),tr(2,:),tr(3,:),'.b');
grid on
hold on;
```

Команды Matlab для работы с нейронными сетями (работа с сетью Кохонена)

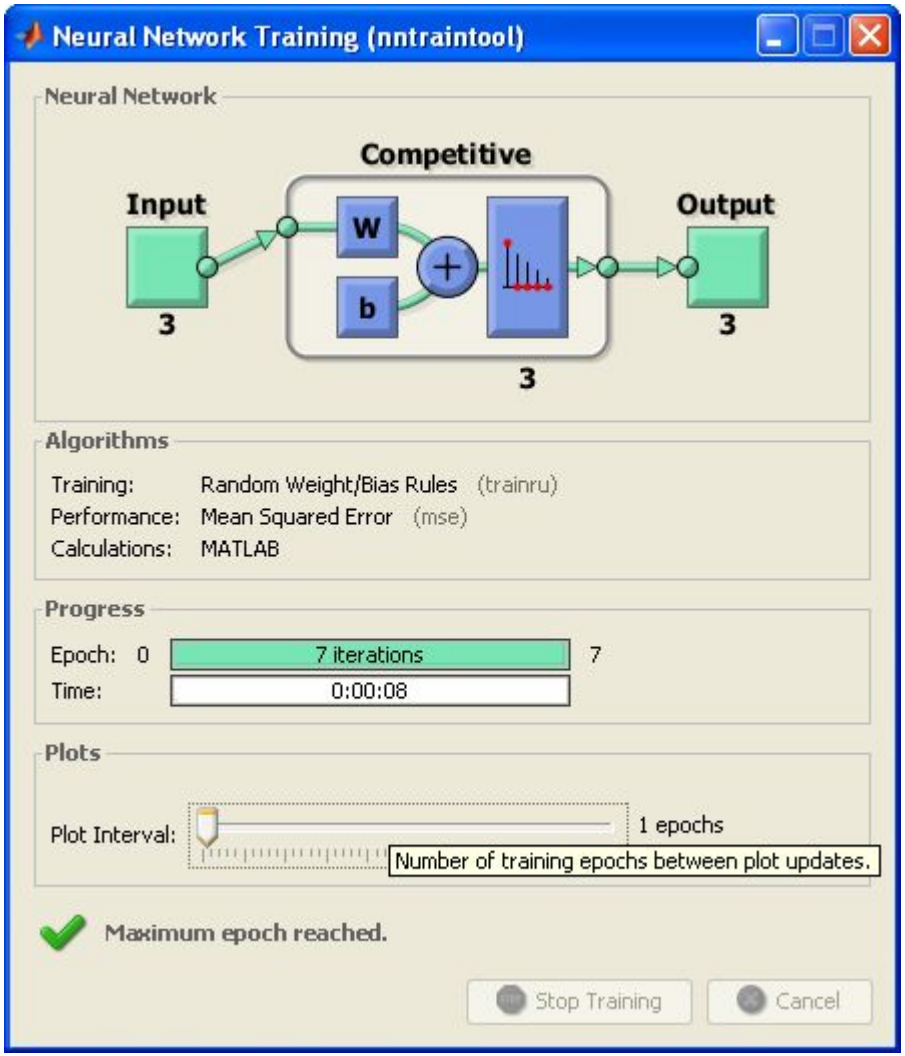
```
circles = plot3(w(:,1),w(:,2),w(:,3),'om');
title('Обучающая выборка и веса');
net.trainParam.epochs = 7;
net = train(net,tr);
w = net.IW{1};
delete(circles);
plot3(w(:,1),w(:,2),w(:,3),'om');
y = net(tr);
for i=1:75
    kl_tr(i)=find(y(:,i)==1);
end
```

Команды Matlab для работы с нейронными сетями (работа с сетью Кохонена)

```
psymb={'+r' '*c' 'xk'};
subplot(2,2,3)
plot3(tr(1,:),tr(2,:),tr(3,:),'.b');
hold on
for i=1:3
    cl=find(kl_tr==i);
    scatter3(tr(1,cl),tr(2,cl),tr(3,cl),psymb{i})
end
grid on
title('Обучающая выборка и результаты распознавания');
```

Команды Matlab для работы с нейронными сетями (работа с сетью Кохонена)

Архитектура построенной сети и ход обучения:



Команды Matlab для работы с нейронными сетями (работа с сетью Кохонена)

Выводимые графики: