Лабораторная работа №3

Тема: Классификация с помощью слоя Кохонена

Цель работы: изучение модели слоя Кохонена и алгоритма обучения без учителя; создание и исследование модели слоя Кохонена в системе MATLAB.

Основные положения

В естественной биологической системе трудно себе представить наличие учителя и сам процесс обучения. Тем не менее, человек (и не только) способен выполнять классификацию без учителя, что дает основания полагать, что объективно существуют алгоритмы обучения без учителя. То есть в обучающей выборке для образов неизвестны правильные (желаемые) выходные реакции. Главная черта, делающая обучение без учителя привлекательным – это его самостоятельность. Процесс обучения, как и в случае обучения с учителем заключается в подстройке синаптических коэффициентов сети.

Для реализации этого подхода необходимо решить две основные проблемы: 1) разработать методы разбиения образов на классы без учителя – этап обучения; 2) выработать правила отнесения текущего входного образа к некоторому классу – этап распознавания. Очевидно, разбиение на классы должно быть основано на использовании достаточно общих свойств классифицируемых объектов. В один класс должны попасть в некотором смысле похожие образы. При этом, как правило, используют компактность образов. В этом случае каждому классу в пространстве признаков соответствует обособленная группа точек. Тогда задача классификации сводится к разделению в многомерном пространстве на части множества точек. Процесс разбиения множества образов на классы называют кластеризацией. Визуально в двумерном (или трехмерном) пространстве

разделение множества точек часто не представляет особых проблем. Необходимо формализовать способ проведения границы между классами.

Существует тип нейронных сетей, который основан на конкуррентном обучении (competitive learning), где нейроны выходного слоя соревнуются за право активации. При этом активным становится обычно один нейрон - победитель в сети нейронов. Таким образом, здесь реализуется принцип “победитель получает все”.

Обучение по Хеббу

Очевидно, что подстройка синапсов может проводиться только на основании информации доступной в нейроне, т.е. его состояние Y_j и уже имеющихся коэффициентов W. Исходя из этих соображений и по аналогии с известными принципами самоорганизации нервных клеток построен алгоритм обучения Хебба, который основан на следующем правиле коррекции весов:

$$W_{ij}(t) = W_{ij}(t-1) + \alpha y_i^{(n-1)} y_j^{(n)} \qquad (3.1)$$

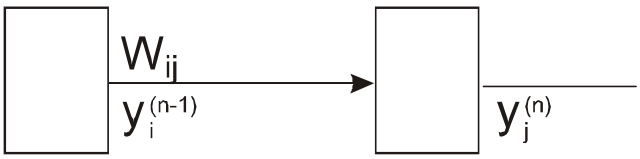

Рис. 3.1. Нейронная сеть.

$y_j^{(n-1)}$ – выходное значение i-го нейрона (n-1)-го слоя

$y_j^{(n)}$ – выходное значение j-го нейрона n-го слоя

$W_{ij}(t), W_{ij}(t-1)$ - весовые коэффициенты синапса, соединяющего i-й и j-й нейроны на итерациях t и t-1 соответственно.

α – коэффициент обучения.

Очевидно, что при обучении данным методом усиливаются связи между соседними нейронами.

Существует также дифференциальный метод обучения Хебба, в котором коррекция весов производится по формуле:

$$W_{ij}(t) = W_{ij}(t-1) + \alpha[y_i^{(n-1)}(t) - y_i^{(n-1)}(t-1)][y_j^{(n)}(t) - y_j^{(n)}(t-1)] \quad (3.2)$$

Из формулы видно, что сильнее всего обучаются синапсы соединяющие те нейроны выходы которых наиболее динамично изменяются.

Полный алгоритм обучения выглядит следующим образом:

1. На стадии инициализации всем весовым коэффициентам присваивается небольшое случайное значение.
2. На входы сети подается входной образ и сигнал возбуждения распространяется по всем слоям сети.
3. На основании поученных выходных значений Y_i, Y_j по формулам (3.1) и (3.2) производится изменение весовых коэффициентов.
4. Переход на шаг 2, пока выходы сети не стабилизируются с заданной точностью. При этом на втором шаге цикла попеременно предъявляются все образы обучающего множества.

Вид отклика на каждый класс входных образов неизвестен заранее и представляет собой произвольное сочетание состояний нейронов обусловленное случайным распределением весов на стадии инициализации. Несмотря на это сеть обобщает похожие образы, относя их к одному классу.

Обучение по Кохонену

Этот алгоритм предусматривает подстройку синаптических коэффициентов на основании их значений на предыдущей итерации

$$W_{ij}(t) = W_{ij}(t-1) + \alpha[y_j^{(n-1)} - W_{ij}(t-1)] \quad (3.3)$$

Видно, что здесь обучение сводится к минимизации разности между входными синапсами нейронов поступающих с выходов нейронов предыдущего слоя и весовыми коэффициентами его синапсов.

Полный алгоритм обучения имеет приблизительно такую же структуру, как и алгоритм Хебба, но на шаге 3 из всего слоя выбирается 1 нейрон, значение синапсов которого максимально подходят на входной образ и подстройка весов этого нейрона выполняется по формуле 3. Таким образом,

здесь реализуется принцип «победитель получает все». Часто такой подход называют конкурирующим обучением.

Приведенная процедура разбивает множество входных образов на кластеры присущие входным данным.

В правильно обученной сети для текущего входного образа *X* активизируется только один выходной нейрон-победитель, соответствующий кластеру, который попадает в текущий входной образ *X*.

Методы определения нейрона-победителя

Существует два метода определения нейрона победителя.

Первый основан на скалярном произведении $\overline{X} * \overline{W}$, т.е. максимальное значение дает выигравший нейрон. При этом обычно вектора $\overline{X}$ и $\overline{W}$ нормализуются.

Для каждого выходного нейрона вычисляется скалярное произведение

$$y_0 = \sum_i W_{i0} * x_i = \overline{W}_0^t * \overline{x} \qquad \forall o \neq k, y_o \leq y_k \tag{3.4}$$

и выбирается нейрон победитель.

Далее устанавливаются значения: для каждого нейрона победителя: $y_k = 1$, для остальных: $y_{o \neq k} = 1$.

Как только определяется победитель, его веса корректируются следующим образом:

$$\overline{W}_k(t+1) = \frac{\overline{W}_k(t) + \alpha\left[\overline{x}(t) - \overline{W}_k(t)\right]}{\left\|\overline{W}_k + \alpha\left[x(t) - W_k(t)\right]\right\|} \tag{3.5}$$

где деление сохраняет вектора $\overline{W}$ нормализованными.

Согласно этому выражению вектор весов вращается («подкручивается») в сторону вектора *X*.

В процессе обучения, каждый раз, когда предъявляется текущий вектор $\overline{x}$, определяется ближайший к нему весовой вектор нейрона победителя, который «подкручивается» по направлению к $\overline{x}$. В результате вектора весов

нейронов вращаются по направлению к тем областям, где находится много входных образов (т.е. кластеров), что показано на рисунке 3.3.

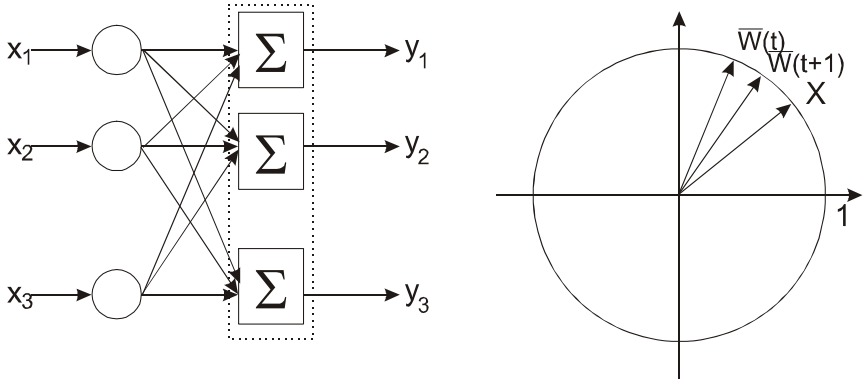

Рис. 3.2.Вращение весового вектора в процессе обучения.

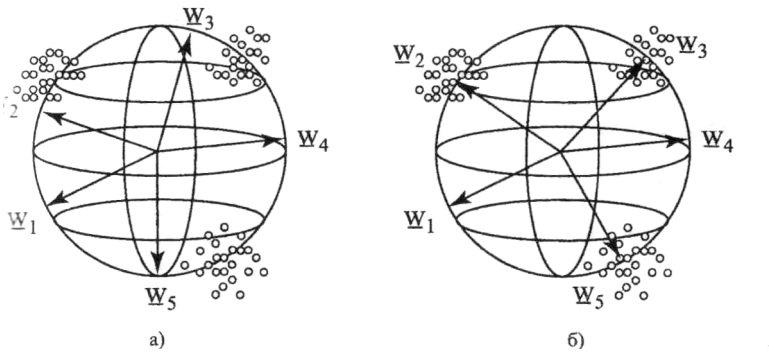

Рис. 3.3 Установление весовых векторов нейронов в центры кластеров.

Второй метод основан на вычислении евклидова расстояния. Т.е. выбирается *k*-й нейрон для которого евклидово расстояние удовлетворяет условию:

$$K : \left\| \overline{W}_k - \bar{x} \right\| \le \left\| \overline{W}_o - \bar{x} \right\| \forall o \qquad (3.6)$$

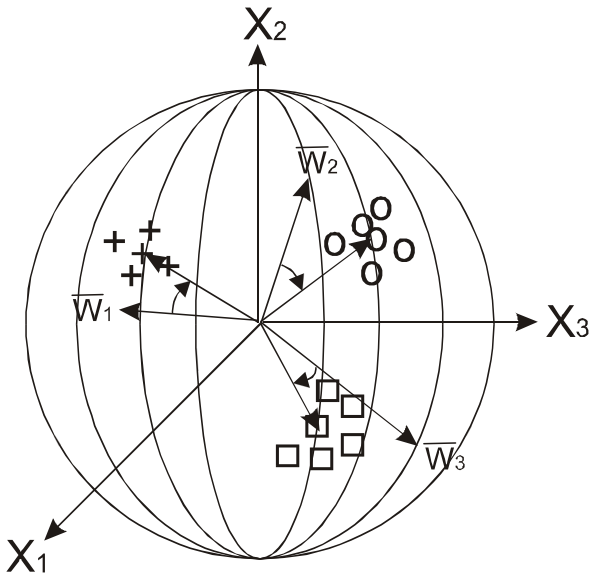

Рис. 3.4. Установление весовых векторов нейронов в центры кластеров.

Очевидно, что если вектора $\overline{X}$ и $\overline{W}$ нормализованы, то этот метод дает те же результаты, что и первый.

После определения победителя его веса корректируются по той же формуле

$$\overline{W_k}(t+1) = \overline{W_k}(t) + \alpha\left(\overline{x}(t) - \overline{W_k}(t)\right) \qquad (3.7)$$

Пример. Фрагменты программы создания и обучения сети:

```
P = [.1 .8   .1 .9; .2 .9 .1 .8]; %массив обучающих точек
…
net = newc([0 1; 0 1],2);          % создание сети
…
net = train(net,P);                % обучение сети
…
A= [0 1 -1; 0 1 -1];                 % массив проверочных точек
a=sim(net,A);                       % моделирование работы НС
…
```

Порядок выполнения работы

1. Построить нейронную сеть, которая производит классификацию на основе слоя Кохонена с помощью функции newc. Произвести начальную инициализацию весов нейронной сети с помощью функции init.

2. Построить обучающую выборку, которая позволяет правильно классифицировать заданную проверочную выборку. Координаты точек обучающей выборки должны быть подобраны геометрически. Для этого первоначально нанести точки проверочной выборки на график, ориентировочно построить вокруг каждой из этих точек несколько точек обучающей выборки, затем записать координаты этих точек в массив обучающей выборки. На каждую из точек проверочной выборки должно приходиться не менее 3 точек обучающей выборки.

3. Показать обучающую выборку на графике с помощью функции plot. На графике должны быть координатные оси, подписи по осям и название самого графика.

4. Произвести обучение нейронной сети на составленной обучающей выборке с использованием функции train.

5. Используя обученную нейронную сеть, произвести классификацию проверочного множества, с помощью функции sim. Показать результат классификации в командном окне matlab и на графике.

Таблица 3.1.

Индивидуальные задания

№ варианта	Кол-во классов для классификации	Координаты точек проверочного множества, номер класса, к которому принадлежит данная точка
1	3	[0;0]-1; [1;1]-2; [-1;-1]-3;
2	4	[2;1]-1; [1;0]-2; [0;-1]-3;
3	5	[0;0]-1; [1;1]-1; [-1;-1]-3;
4	6	[1;0]-1; [-1;1]-2; [-1;-1]-2;
5	7	[0;0]-1; [1;1]-2; [-1;-1]-3;
6	8	[0;0]-1; [1;1]-1; [-1;-1]-1;
7	3	[-1;0]-1; [-1;1]-2; [-1;-1]-3;
8	4	[0;1]-1; [1;-1]-2; [-1;-1]-3;
9	5	[2;0]-1; [1;1]-2; [-1;1]-2;
10	6	[0;0]-1; [1;1]-2; [-1;-1]-1;
11	7	[0;0]-1; [1;1]-1; [-1;-1]-2;
12	8	[0;0]-1; [1;1]-2; [-1;-1]-3;
13	3	[0;0]-1; [1;1]-1; [-1;-1]-1;
14	4	[0;1]-1; [1;-1]-2; [-1;-1]-3;

Содержание отчета

1. Титульный лист.
2. Задание, учитывая свой вариант.
3. Теоретические сведения.
4. Представить графическое и табличное представление обучающего множества.
5. Представить графическое и табличное представление проверочных точек.
6. Представить программу, написанную в среде MATLAB, и результаты классификации.
7. Выводы.

Контрольные вопросы

1. Архитектура нейронной сети.
2. Правило нахождения количества нейронов в сети для распознавания заданного числа классов.
3. Алгоритм обучения.
4. Анализ информации, выдаваемой функцией display.
5. Результат, который возвращает функция sim.
6. Параметры функций newc.

Лабораторная работа №4

Тема: Прогнозирование временных рядов с использованием нейронных сетей

Цель работы: изучение способа прогнозирования временных ряда с помощью НС в системе MATLAB.

Понятие прогнозирования временных рядов

Ряд наблюдений $x(t_1)$, $x(t_2)$,…, $x(t_N)$ анализируемой величины x(t), произведенных в последовательные моменты времени t_1, t_2,..., t_N, называется временным рядом (ВР). В общем виде моменты наблюдений могут быть произвольными, но классически считается, что во временных рядах равноотстоящие моменты наблюдений.

Задача одношагового прогнозирования может быть сформулирована так: имеется временной ряд $\bar{Y}$, представленный своими значениями, лагами (ЛАГ – от англ. lag — запаздывание, временной лаг – показатель, отражающий отставание или опережение во времени одного явления по сравнению с другими) в предшествующие моменты времени Y_{t-k}, Y_{t-k+1}, Y_t, где t-текущий момент времени, k-глубина исторической выборки. Необходимо найти величину изменения ее значения, которое произойдет в следующий момент времени, на основе анализа прошлых значений, а также истории изменения других факторов, влияющих на динамику прогнозируемой величины. При этом важным может являться как максимально низкая величина ошибки прогнозирования, так и правильное определение характера дальнейшего изменения значения прогнозируемой величины (увеличение или уменьшение).

При решении задачи прогнозирования требуется найти значение Y (или несколько величин, вектор) в будущий момент времени на основе известных предыдущих значений $\bar{Y}, \bar{V}, \bar{C}$: $Y_{t+1} = g(\bar{Y}, \bar{V}, \bar{C})$