Лабораторная работа №6

Исследование состязательных сетей и сетей векторного квантования

1 Цель работы

Углубление теоретических знаний в области обучения нейросетей без учителя, исследование свойств алгоритмов обучения состязательных сетей на основе правил Кохонена, приобретение практических навыков обучения самоорганизующихся карт Кохонена и сетей векторного квантования при решении задач классификации.

2 Основные теоретические положения

2.1 Обучение без учителя и самоорганизующиеся ИНС

При обучении без учителя **отсутствует информация о правильности реакции** ИНС на тот или иной входной вектор. Нейронная сеть самостоятельно обнаруживает взаимосвязь входных векторов и преобразует их в ассоциированную с ними выходную реакцию, т.е. в ИНС происходят процессы самоорганизации.

Самоорганизующиеся ИНС способны открывать отношения подобия во входных данных, обеспечивая тем самым **выделение во входном пространстве классов.**

Существует два основных типа самоорганизующихся ИНС:

1) ИНС, основанные на правиле обучения Хебба;

2) состязательные ИНС.

2.2 Правило обучения Хебба

Правило обучения Хебба заимствовано из биологических нейронных сетей. В соответствии с этим правилом **веса синаптических связей изменяются пропорционально активности** нейронов. Правило обучения Хебба можно записать в виде

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q)\, p_j(q), \tag{7.1}$$

где $a_i(q)$ – выход i- го нейрона; $p_j(q)$ – входное воздействие на j-ом входе нейрона. Изменение веса связи в соответствии с правилом не требует привлечения сведений о желаемой реакции нейрона.

Правило Хебба может быть записано в векторной форме

$$\mathbf{W}(q) = \mathbf{W}(q-1) + \alpha \mathbf{a}(q)\mathbf{p}^{\mathrm{T}}(q). \tag{7.2}$$

Как в любом правиле обучения без учителя, обучение выполняется по реакциям на последовательность предъявляемых входных образов **p**(1)**,** **p**(2),…,**p**(Q).

Приведенная форма правила Хебба **не ограничивает рост весов** связей по мере обучения. Поэтому в правило Хебба вводят **затухание**

$$\mathbf{W}(q)=\mathbf{W}(q-1)+\alpha\mathbf{a}(q)\mathbf{p}^{\mathrm{T}}(q)-\gamma\mathbf{W}(q-1)=(1-\gamma)\mathbf{W}(q-1)+\alpha\mathbf{a}(q)\mathbf{p}^{\mathrm{T}}(q), \quad (7.3)$$

где γ – скорость затухания. В этом случае максимальный вес связи ограничивается значением w_{ij} =α/ γ (получается, если все **a** и **p** равны 1).

2.3 Простая распознающая сеть Instar

Рассмотрим простую сеть из одного нейрона (рисунок 7.1).

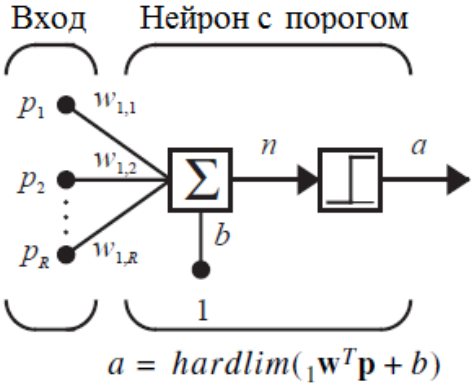

Рисунок 7.1 – Сеть Instar

Нейрон будет активен, когда $_1\mathbf{w}^T\mathbf{p} \geq -b$. Скалярное произведение будет максимальным при $\mathbf{p} = {}_1\mathbf{w}$. Т.е. сеть активна, когда **p** близко к $_1$**w.** Подбирая *b*, мы можем регулировать степень схожести **p** и $_1$**w.**

Если установить $b = -\|{}_1\mathbf{w}\|\|\mathbf{p}\|$, то нейрон будет активен, когда **p** точно соответствует направлению $_1$**w.** В этом случае нейрон будет распознавать только образ, соответсвующий $_1$**w.**

Если мы хотим, чтобы сеть распознавала образы, близкие к $_1$**w,** надо уменьшить *b*. Чем меньше *b*, тем большое число входных образов будут активировать сеть.

2.4 Состязательные сети

Рассмотрим слой нейронов, в котором на выходе нейрона с наибольшим значением сетевой функции устанавливается единичная активность. Такая сеть называется *состязательной* или *соревновательной*. Нейрон, имеющий наибольшее значение сетевой функции, называют нейроном-победителем, выигравшим “состязание”.

Состязательную функцию преобразования обозначают как **a= compet(n)** (рисунок 7.2). Она обеспечивает нахождение индекса *i** нейрона-победителя и устанавливает на его выходе 1:

$$a_i = \begin{cases} 1, i = i^* \\ 0, i \neq i^* \end{cases}, \text{ где } n_{i^*} \geq n_i, \forall i, \quad \text{и} \quad i^* \leq i, \forall n_i = n_{i^*}$$

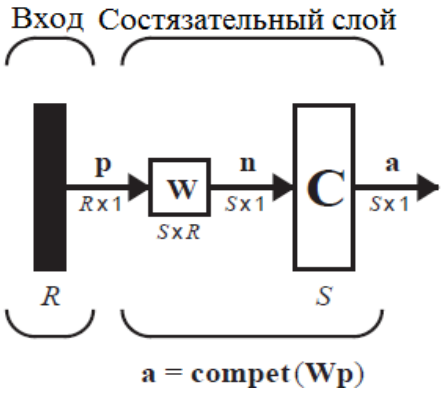

Рисунок 7.1 – Состязательная сеть

В состязательной сети устанавливается 1 на выходе того, нейрона, чей вектор весов **w** *по направлению "ближе" всего к входному вектору* **p.**

2.5 Правило обучения Кохонена

Отдельные нейроны состязательной сети на уровне сетевой функции соответствуют Instar-нейронам. Для решения задачи классификации с помощью состязательной сети можно строкам матрицы весов связей **W** присвоить значения желаемых векторов-прототипов классов. Т.к. векторы-прототипы нам заранее неизвестны, то для обучения сети можно использовать ассоциирующее Instar правило (7.3).

Поскольку для состязательной сети $a_i(q) \neq 0$ только для нейрона-победителя i^*, то

$$_i\mathbf{w}(q) = _i\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - _i\mathbf{w}(q-1)) = (1-\alpha)_i\mathbf{w}(q-1) + \alpha\mathbf{p}(q), \quad i = i^*. \tag{7.4}$$

$$_i\mathbf{w}(q) = _i\mathbf{w}(q-1), \quad i \neq i^*. \tag{7.5}$$

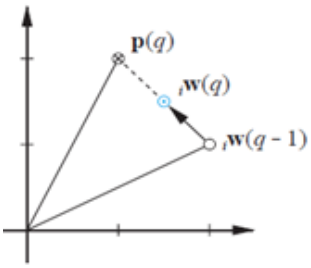

Рисунок 7.3

Правило (7.4-7.5) называют **правилом Кохонена**. В соответствии с этим правилом на каждой итерации вектор-строка $_i\mathbf{w}(q\text{-}1)$ матрицы весов, ближайшая к входному вектору $\mathbf{p}(q)$ (или имеющая наибольшее значение скалярного произведения с $\mathbf{p}(q)$), смещается по направлению к входному вектору (рисунок 7.2), формируя новое значение вектора весов $_i\mathbf{w}(q)$.

2.6 Самоорганизующиеся карты признаков (SOM)

SOM (self-organizing feature maps), предложенные Кохоненым, сначала определяют нейрон-победитель i^* (как в состязательной сети), а затем веса всех нейронов **в окрестности нейрона-победителя** N_{i^*} обновляются на основе правила Кохонена

$$_i\mathbf{w}(q) =_i \mathbf{w}(q-1)+\alpha(\mathbf{p}(q)-_i \mathbf{w}(q-1)) = (1-\alpha)_i\mathbf{w}(q-1)+\alpha\mathbf{p}(q), \quad i \in N_{i^*}. \tag{7.6}$$

Окрестность N_{i^*} содержит индексы всех нейронов, которые находятся на расстоянии радиуса d от нейрона–победителя i^*: $N_i(d)=\{j, d_{ij} \leq d\}$. Когда предъявляется входной вектор $\mathbf{p}(q)$ веса нейрона-победителя и нейронов в его окрестности смещаются в сторону вектора $\mathbf{p}(q)$. После многократных предъявлений соседние нейроны будут обучены распознаванию похожих входных векторов.

Карта признаков отображает образы из входного n-мерного пространства в пространство меньшей размерности. Кроме сокращения размерности пространства, карта признаков обеспечивает сохранение отношений топологического соседства входных образов. При этом геометрическая близость выходных классов на карте признаков означает близость соответствующих образов во входном пространстве признаков.

Примеры круговых окрестностей с радиусами d=1 и d=2 для двумерной карты изображены на рисунке 7.4.

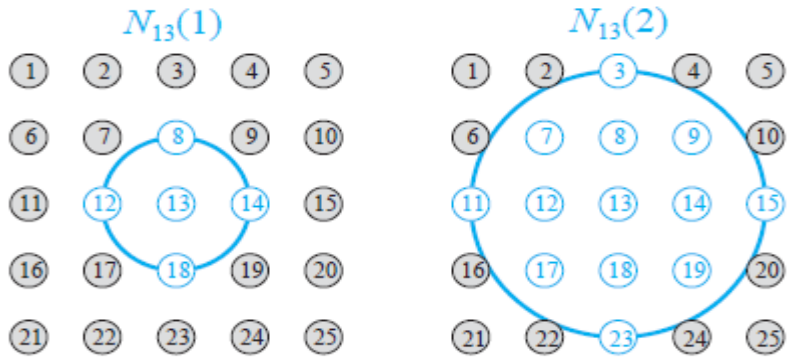

Рисунок 7.4 – Примеры круговых окрестностей

Нейроны могут быть также организованы в 1-мерные и 3-х мерные карты. Кохоненым были предложены **прямоугольные и гексагональные окрестности** с целью повышения эффективности реализации.

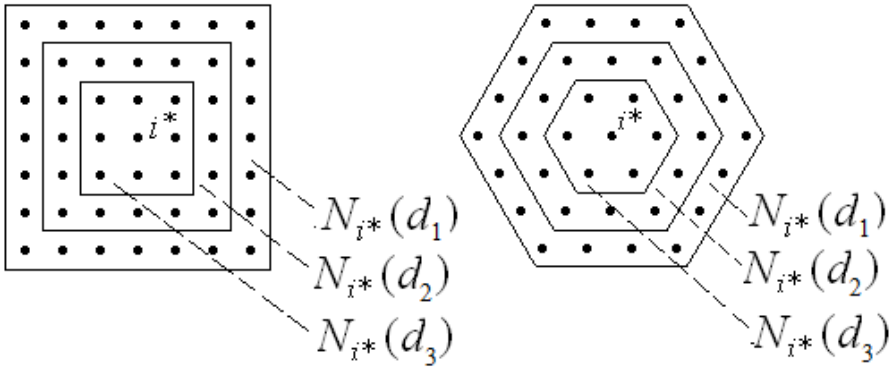

Рисунок 7.5 – Прямоугольная и гексогональные окрестности

На практике множество $N_{i*}(d)$ и коэффициент обучения меняют свои значения динамически в процессе обучения. Процесс обучения начинается при расширенном множестве $N_{i*}(d)$ и больших значениях α . По мере обучения количество НЭ, включаемых в $N_{i*}(d)$ уменьшается, одновременно уменьшается и значение α. При обучении учёт нейронов, образующих окрестность $N_{i*}(d)$, осуществляется с помощью функции соседства $\Lambda(i,i^*)$, которая равна 1 при $i=i^*$ и уменьшается по мере увеличения расстояния от нейрона i^* до нейрона i в выходном топологическом пространстве. В этом случае правило обучения записывается в виде:

$$\Delta_i \mathbf{w}(q) = \alpha(q) \cdot \Lambda(i,i^*)(\mathbf{p}(q) -_i \mathbf{w}(q-1)). \quad \textbf{(7.7)}$$

Например, функция $\Lambda(i,i^*)$ может быть задана выражением

$$\Lambda(i,i^*) = \exp(-|r_{ii^*}|^2 / 2d^2), \quad (7.8)$$

где r_{ii^*} – расстояние между нейронами i и i^* в выходном двумерном пространстве; d – параметр, задающий “диаметр” функции соседства. Диаметр d и скорость обучения α уменьшаются в зависимости от q , например по экспоненте.

2.7 Обучение на основе векторного квантования (LVQ)

LVQ (learning vector quantization) сети являются гибридными (рисунок 7.6).

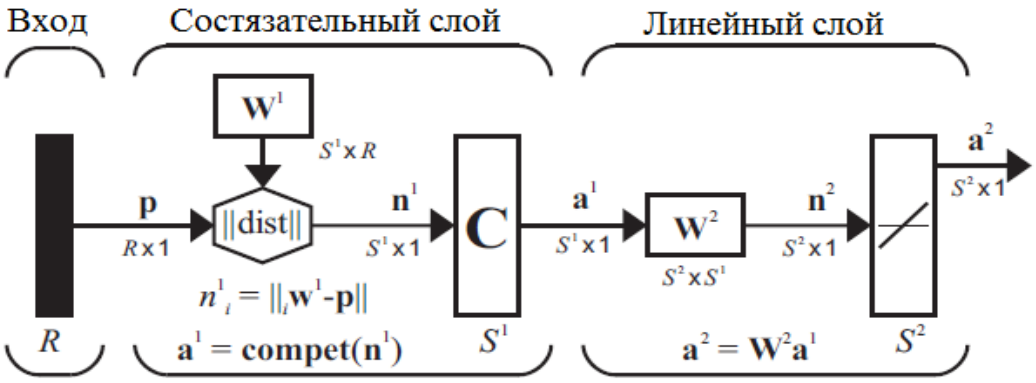

Рисунок 7.6 – Сеть векторного квантования

Как и в состязательной сети, 1-ый слой определяет подклассы входного пространства. При этом вместо скалярного произведения вычисляется прямое расстояние между векторами. Это позволяет не выполнять нормализацию входных векторов.

Второй слой комбинирует подклассы в один класс. Для второго слоя столбцы матрицы $\mathbf{W}^2$ представляют подклассы, а строки – классы. $\mathbf{W}^2$ содержит в каждом столбце по одной 1. Единицы в строках отображают класс подкласса:

$$(w_{ki}^2 = 1) \Rightarrow \text{подкласс } i \text{ принадлежит классу } k \quad . \qquad (7.9)$$

Процесс комбинации подклассов в классы позволяет сетям LVQ создавать сложные границы между классами.

2.8 Правило обучения сетей векторного квантования

Правило комбинирует обучение без учителя и с учителем. Оно использует множество обучающих примеров $\{\mathbf{p}_1,\mathbf{t}_1\}$, $\{\mathbf{p}_2,\mathbf{t}_2\},\ldots,\{\mathbf{p}_Q,\mathbf{t}_Q\}$. Каждый целевой вектор содержит только одну единицу, которая обозначает класс принадлежности входного вектора. Перед началом обучения формируется матрица $\mathbf{W}^2$ и нейронам 1-го слоя назначается класс принадлежности (выходной нейрон). Обычно равное количество нейронов 1-го слоя связывается с каждым выходным нейроном.

Затем веса 1-го слоя обучаются **на основе модифицированного правила Кохонена**. Если **p** классифицируется корректно ($i^* \in$ классу k), то

$${}_{i^*}\mathbf{w}^1(q) = {}_{i^*}\mathbf{w}^1(q-1) + \alpha(\mathbf{p}(q) - {}_{i^*}\mathbf{w}^1(q-1)), \text{ если } a_{k^*}^2 = t_{k^*} = 1 \quad . (7.10)$$

Если **p** классифицируется не корректно ($i^* \notin$ классу k), то

$${}_{i^*}\mathbf{w}^1(q) = {}_{i^*}\mathbf{w}^1(q-1) - \alpha(\mathbf{p}(q) - {}_{i^*}\mathbf{w}^1(q-1)), \text{ если } a_{k^*}^2 = 1 \neq t_{k^*} = 0 \quad . \quad (7.11)$$

Таким образом, нейроны скрытого слоя смещаются в строну векторов, которые попадают в класс, для которого они формируют подкласс, и удаляются от векторов, которые относятся к другому классу.

Правило LVQ чувствительно к инициализации весов 1-го первого слоя. Из-за этого некоторые нейроны не могут «попасть» в свои подклассы. Имеется **усовершенствованное правило LVQ2.** Если возникает ситуация неверной классификации, то дополнительно подстраиваются со знаком «+» веса ближайшего нейрона к входному вектору, который даёт верную классификацию.

2.7 Функции Scilab Neural Network 2.0 для обучения состязательных сетей и сетей векторного квантования

В модуле Scilab Neural Network 2.0 реализованы следующие функции для обучения без учителя:

- ann_COMPET — функция обучения состязательной сети;
- ann_COMPET_run — функция моделирования состязательной сети;
- ann_COMPET_visualize2d — состязательная сеть с 2D анимацией;
- ann_COMPET_visualize3d — состязательная сеть с 3D анимацией;
- ann_SOM — ИНС в виде самоорганизующейся карты (блочное обучение);

- ann_SOM_online — ИНС в виде самоорганизующейся карты (последовательное обучение;
- ann_SOM_run — функция моделирования ИНС в виде самоорганизующейся карты;
- ann_SOM_visualize2d — ИНС в виде самоорганизующейся карты с 2D визуализацией;
- ann_SOM_visualize3d — ИНС в виде самоорганизующейся карты с 3D визуализацией.

Для обучения и моделирования сетей векторного квантования в модуле Scilab Neural Network 2.0 имеются следующие функции:

- ann_LVQ1 — ИНС векторного квантования ;
- ann_LVQ_run — функция моделирования ИНС векторного квантования.

Рассмотрим функцию для обучения состязательной сети ann_COMPET (код функции с комментариями приведен в приложении А). Синтаксис вызова функции:

```
[W,b] =  ann_COMPET(P,N)
[W,b] =  ann_ COMPET(P,N,lr,lr_c,itermax),
```

где P – матрица обучающих примеров; N – число нейронов в состязательном слое (число классов); lr – скорость обучения (по умолчанию 0.1); lr_c – скорость обучения для смещения (по умолчанию 0.1); itermax – максимальное число эпох обучения (по умолчанию 100); W – выходная матрица весов связей; b – вектор смещений.

Ниже приведен пример применения функции:

```
x = rand(2,10);
x(:,1:5) = x(:,1:5) + 1;
plot(x(1,:),x(2,:),'.');
[W,b] =  ann_COMPET(x,4);close;
plot(W(:,1), W(:,2), 'or');
```

В примере формируется двумерный случайный массив чисел x в диапазоне от 0 до 1 с равномерным распределением. Затем первая часть массива со столбцами с 1 по 5 увеличивается на 1. Массив x представляет собой матрицу обучающих примеров, которые на рисунке 7.7 обозначены голубыми точками. Вызов ann_COMPET(x,4) обеспечивает обучение состязательной сети для разделения обучающих примеров на 4 класса. Результаты обучения (значения весов нейронов) представлены красными кружочками на рисунке 7.7.

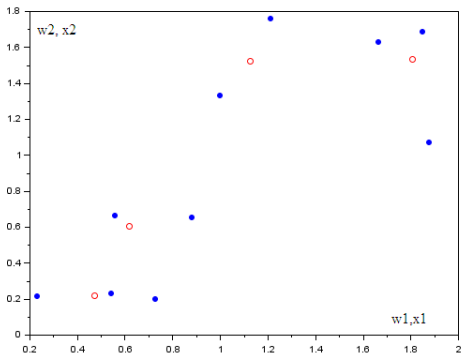

Рисунок 7.7 – Пример обучения состязательной сети

Функции ann_COMPET_visualize2d и ann_COMPET_visualize3d вызываются аналогично. Эти функции позволяют визуализировать несложные примеры обучения либо на плоскости, либо в 3-мерном пространстве.

Функция ann_SOM предназначена для блочного обучения состязательной сети, реализуемой в виде самоорганизующейся карты. Синтаксис вызова функции:

```
W = ann_SOM(P)
W = ann_SOM(P,N,itermax,steps,NS,topfcn,distfcn),
```

где P – матрица обучающих примеров; N – структура карты признаков (по умолчанию - 8x8); itermax – максимальное число эпох обучения (по умолчанию - 200); steps – число шагов сокращения NS (по умолчанию - 100); NS – начальный размер окрестности (по умолчанию - 3); topfcn – функция, определяющая топологию сети (по умолчанию ann_som_gridtop – прямоугольная сетка); distfcn – функция, определяющая способ вычисления расстояния соседства (по умолчанию ann_som_linkdist).

Ниже приведен пример применения функции ann_SOM:

```
x = rand(2,10);
x(:,1:5) = x(:,1:5) + 1;
W =  ann_SOM(x,[2 2]);
[y,classes] = ann_SOM_run(W,x)
```

В примере формируется массив обучающих данных и создается сеть в видесамоорганизующейся карты признаков Кохонена размером 2x2 с прямоугольной топологией. В результате обучения функция ann_SOM вернет матрицу весов W. Вызов функции моделирования ann_SOM_run обеспечивает проверку функционирования обученной сети. Т.к. сеть содержит 4 нейрона, то обучающие примеры будут отнесены к одному из 4-х классов, распознаваемых сетью. При этом значением переменной y для каждого примера из x являются бинарные векторы с 1 в позиции нейрона-победителя, а значением переменной classes – номер класса каждого входного примера.

Функция ann_SOM_online имеет сходный набор и вызывается аналогично ann_SOM. Отличие заключается в том, что функция осуществляет обучение в последовательном режиме, т.е. матрица весов корректируется после

предъявления каждого обучающего примера. Функции ann_SOM_visualize2d и ann_SOM_visualize3d осуществляют визуализацию процесса обучения в блочном режиме (вызываются с тем же набором параметров, что и ann_SOM).

В состав модуля Scilab Neural Network 2.0 для работы с самоорганизующимися картами входят следующие функции, определяющие топологию карт и способ вычисления расстояний соседства:

- ann_som_boxdist — функция блочного расстояния;
- ann_som_eudist — функция евклидова расстояния;
- ann_som_gridtop — функция прямоугольной топологии;
- ann_som_hextop — функция гексагональной топологии;
- ann_som_linkdist — функция расстояния связи;
- ann_som_mandist — функция манхэттенского расстояния;
- ann_som_randtop — функция случайной топологии.

Для визулизации топологии самоорганизующей сети можно использовать функции ann_som_plot2d и ann_som_plot3d. Примеры использования этих функций для визулизации топологии сетей в 2- и 3-х мерном пространствах:

```
N = [5,5];
y = ann_som_hextop(N);
ann_som_plot2d(y);

N = [5,5,5];
y = ann_som_hextop(N);
ann_som_plot3d(y);
```

Результаты визуализации изображены на рисунке 7.8.

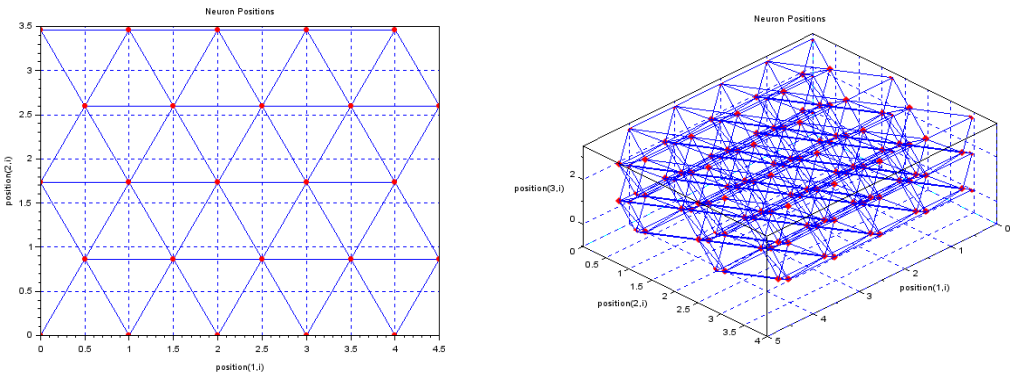

Рисунок 7.8 – 2D и 3D гексагональные топологии

Для обучения и моделирования сетей векторного квантования применяются соответственно функции ann_LVQ1 и ann_LVQ_run. Форматы вызова функций:

```
[W,b] =  ann_LVQ1(P,T,N2);
[W,b] =  ann_LVQ1(P,T,N2,lr,itermax);
[y,classes] = ann_LVQ_run(W,P),
```

где T – целевые классы, задаваемые в виде бинарных векторов-состояний выходов сети; N2 – количество нейронов состязательного слоя (количество подклассов). Ниже приведен пример применения этих функций для разделения множества входных данных на 4 класса:

```
x = rand(2,10);
x(:,1:5) = x(:,1:5) + 1;
T = [1 1 1 1 1 0 0 0 0 0; 0 0 0 0 0 1 1 1 1 1]
[W,b] =  ann_LVQ1(x,T,4);
[y,classes] = ann_LVQ_run(W,x).
```

3. Варианты заданий и программа работы

3.1 Повторить теоретический материал, относящийся к алгоритмам обучения состязательных нейронных сетей и сетей векторного квантования [1, 5].

3.2 Сформировать в соответствии с таблицей 7.1 множество входных данных для обучения SOM. Для генерации данных использовать генератор случайных чисел с равномерным распределением в диапазоне, указанном в таблице 7.1.

Таблица 7.1 – Варианты заданий

Вариант	Пространство	Функции топологии и расстояния	Кол-во классов	Диапазон значений вх.данных
1	2D	gridtop, eudist	16	0.0-4.0
2	2D	hextop, linkdist	16	0.0-4.0
3	2D	randtop, mandist	16	0.0-4.0
4	2D	gridtop, linkdist	36	0.0-6.0
5	2D	hextop, mandist	8	0.0-8.0
6	2D	randtop, eudist	8	0.0-8.0
7	2D	gridtop, mandist	25	0.0-5.0
8	3D	hextop, mandist	27	0.0-3.0
9	3D	randtop, linkdist	27	0.0-3.0
10	3D	gridtop, eudist	27	0.0-3.0
11	3D	hextop, eudist	8	0.0-2.0
12	3D	randtop, linkdist	8	0.0-2.0
13	3D	gridtop, mandist	8	0.0-2.0
14	3D	hextop, linkdist	64	0.0-4.0
15	3D	randtop, linkdist	64	0.0-4.0

3.3 Написать программу, создающую и обучающую SOM, заданной топологии в соответствии с таблицей 7.1. на сгенерированных данных.

3.4 Визуализировать топологию SOM на начальном этапе и после обучения.

3.5 Сгенерировать множество векторов, представляющих центры классов, равномерно распределив их в заданных диапазонах входного пространства. Подать на вход сети эти векторы и определить номера нейронов-победителей, распознающих эти векторы.

3.6 Используя данные, сгенерированные в п. 3.2. и п.3.5 создать и обучить LVQ сеть, которая:

- содержит в скрытом слое такое же число нейронов как SOM, заданная по варианту в соответствии с таблицей 7.1;
- группирует каждые 2 (при четном числе подклассов) или 3 (при нечетном числе подклассов) подкласса, формируемые скрытым слоем, в один класс.

3.7 Оценить корректность классификации векторов, сгенерированных в п.3.5

3.8. Подготовить и защитить отчет.

4. Методические рекомендации по выполнению работы

4.1 При выполнении работы используйте примеры, приведенные в подпункте 2.8.

5. Содержание отчета

5.1 Цель работы.

5.2 Вариант задания.

5.3 Схемы SOM и LVQ сетей заданной архитектуры, топология SOM до обучения и после обучения, листинги программ с комментариями, таблица с векторами-прототипами и номерами нейронов-победителей для SOM, таблица с векторами- прототипами и номерами классов для LVQ сети.

5.4 Выводы

6. Контрольные вопросы

6.1 Что понимается под термином «обучение без учителя»?

6.2 Назовите основные типы самоорганизующихся ИНС.

6.3 Сформулируйте и запишите правило обучения Хебба в его исходном виде.

6.4 Запишите правило обучения Хебба с затуханием.

6.5 Нарисуйте схему сети Instar и запишите правило её обучения.

6.6 Нарисуте схему состязательной сети и определите функцию её преобразования.

6.7 Запишите правило обучения Кохонена для состязательной сети. Как графически интепретируется правило?

6.8 Запишите правило обучения Кохонена для самоорганизующейся карты.

6.9 Объясните понятие окрестности для нейрона-победителя. Приведите примеры разных видов окрестностей.

6.10 Сформулируйте правило обучения SOM с использованием функции соседства. Определите функцию соседства.

6.11 Нарисуйте схему сети векторного квантования. Объясните процесс вычислений в этой схеме.

6.12 Как назначаются веса второго слоя сети векторного квантования?

6.13 Сформулируйте модифицированное правило Кохонена для сети векторного квантования.

6.14 Какие проблемы возникают при обучении состязательных сетей и как их решают на практике?

6.15 В чем особенность усовершенстовованного правила обучения LVQ2?

6.16 Объясните и приведите пример вызова функции ann_COMPET.

6.17 Объясните и приведите пример вызова функции ann_SOM.

6.18 Объясните и приведите пример вызова функции ann_LVQ1.

Приложение А. Функция обучения состязательной сети

```
function [W, b]=ann_COMPET(P, N, lr, lr_c, itermax)
// Функция обучения состязательной сети.
//
// Обработка входных аргументов функции
rhs=argn(2);

if rhs < 2; error("Expect at least 2 arguments, P and N");end
if rhs < 3; lr = 0.1; end  // Скорость обучения по умолчанию 0.1
if rhs < 4; lr_c = 0.1; end // Скорость обучения смещения по умолчанию-  0.1
if rhs < 5; itermax = 100; end // Число эпох обучения по умолчанию - 100

if lr == []; lr = 0.1; end
if lr_c == []; lr_c = 0.1; end
if itermax == []; itermax = 100; end

[W, b] = ann_compet_init(P, N); // инициализация сети
iter_span = itermax;

// Инициализация GUI прогресса обучения
handles = ann_training_process();
handles.itermax.string = string(itermax);
handles.msecurrent.visible = 'off';
handles.msemin.visible = 'off';
handles.msemax.visible = 'off';
handles.mse.visible = 'off';
handles.msetitle.visible = 'off';
handles.gdcurrent.visible = 'off';
handles.gdmin.visible = 'off';
handles.gdmax.visible = 'off';
handles.gd.visible = 'off';
handles.gdtitle.visible = 'off';
 Q = size(P,2); // Определение размера обучающего множества примеров

for itercnt = 1:itermax // цикл эпох обучения
   for Pcnt = 1:size(P,2) // Цикл по всем  обучающим примерам из P
      //n = W*P(:,Pcnt);
      q = fix(rand(1,'uniform')*Q)+1; // q – формируем случайный номер примера
      p = P(:,q);       // Выбираем случайный пример из P
      n = ann_negdist(W,p);  // вычисление отрицательного расстояния между примером и весами нейронов сети
      a = ann_compet_activ(n + b); // вычисление состязательной функции
```

```
    W(find(a),:) = W(find(a),:) +lr*(p'-W(find(a),:)); // Правило обучения Кохонена (формула 7.4)
    b(find(a)) = b(find(a)) - 0.2; // Обновление смещений активного нейрона
    b(find(~a)) = lr_c.*b(find(~a));  // Обновление смещений проигравших нейронов
  end

  handles.iter.value = round((itercnt/iter_span)*100);
  handles.itercurrent.string = string(itercnt);

end

endfunction
```

Список рекомендованной литературы

1. Бондарев В.Н. Искусственный интеллект: Учеб. пособие для студентов вузов / В. Н. Бондарев, Ф. Г. Аде. — Севастополь: Изд-во СевНТУ, 2002. — 613 с.
2. Ерин С.В. Scilab – примеры и задачи: практическое пособие / С.В. Ерин – М.: Лаборатория «Знания будущего», 2017. – 154 с.
3. Медведев, В.С. Нейронные сети. MATLAB 6 / В.С. Медведев, В.Г. Потемкин; под общ. ред. В.Г. Потемкина. — М.: ДИАЛОГ-МИФИ, 2002. — 496 с.
4. Хайкин С. Нейронные сети: Полный курс. Пер. С англ. / С. Хайкин. — М.: Изд. «Вильямс», 2006. — 1104 с.
5. Hagan M.T. Neural Network Design. The 2nd edition [Электронный ресурс] /M.T.Hagan, H.B.Demuth, M.H.Beale, O.D. Jesus. . — Frisco, Texas, 2014 . — 1012 p. Режим доступа: https://www.hagan.okstate.edu/NNDesign.pdf. —Последний доступ: 14.01.2019. —Название с экрана.