РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
А В Т О М Е Т Р И Я
№ 2 1997

УДК 519.68

С. Г. Пудов
(Новосибирск)

ОБУЧЕНИЕ КЛЕТОЧНО-НЕЙРОННОЙ АССОЦИАТИВНОЙ ПАМЯТИ

Исследуется клеточно-нейронная ассоциативная память (КНАП). Предлагаются два алгоритма обучения, основанные на методе обучения персептрона, улучшающие способности КНАП по исправлению искажений хранимых образов. Для КНАП без автосвязи получено выражение, позволяющее ввести автосвязь после завершения процесса обучения. Проведены эксперименты по хранению и распознаванию образов, представляющих собой буквы, цифры и т. д. В результате моделирования получен следующий результат: введение автосвязи в КНАП, обученную по второму алгоритму, улучшает возможности сети по исправлению 1-искажений в 7—10 раз.

Введение. Клеточно-нейронная ассоциативная память (КНАП) возникла как синтез искусственных нейронных сетей (ИНС) [1] и алгоритма параллельных подстановок (АПП) [2] и формально может быть интерпретирована как ИНС с локальными связями. Вследствие локальности связей КНАП гораздо легче реализовать в «железе», чем полносвязные ИНС, в которых большое число связей между нейронами является главным препятствием для их производства. Ограничение числа связей каждого нейрона упрощает реализацию нейронных сетей, но при этом уменьшает их возможности. Поэтому исследованию свойств сетей с ограниченным числом связей нейронов посвящено несколько работ [3—5], в которых делаются попытки оценить их способности к хранению образов и исправлению искажений.

Кроме архитектуры связей, большую роль играют алгоритмы обучения, поскольку в остальном именно они определяют все свойства обученной нейронной сети. Для полносвязных ИНС существуют методы обучения, с помощью которых достигаются лучшие динамические свойства сети [6, 7], но эти методы в большинстве своем используют преобразования матриц размерностью $N \times N$, где N — число нейронов в сети, что связано с большой сложностью вычислений и не позволяет проводить процесс обучения в той же структуре, что и ИНС. Поэтому в качестве отправной точки для исследования был выбран алгоритм обучения персептрона [8, 9], который в случае полносвязных сетей обладает хорошими характеристиками по хранению информации.

Статья организована следующим образом: в разд. 1 даются основные определения КНАП и исследуются ограничения, накладываемые локальностью связей; в разд. 2 приводятся два алгоритма обучения; в разд. 3 исследуется влияние автосвязи на исправление искажений и приводится выражение для веса автосвязи; в разд. 4 описывается система моделирования, в которой проводились все эксперименты, и результаты проведенных экспериментов.

1. **Основные понятия и динамические свойства.** 1.1. *Формальное представление КНАП.* Определим клеточно-нейронную ассоциативную память [10] тройкой: $\mathbf{N} = \langle \mathbf{C}, \mathbf{W}, \Phi \rangle$, где $\mathbf{C}$ — прямоугольный массив размерностью $m \times n$, состоящий из *клеток* (или *нейронов*) с состояниями $c_{ij} \in \{-1, 1\}$; $\mathbf{W} = \{W_{ij}\}$ является множеством *весовых векторов* вида $W_{ij} = (w_1, \ldots, w_q)$, причем w_k обозначает действительное число, характеризующее связь между нейроном

с координатами (i, j) и его k-м соседом; **Ф** — правило функционирования КНАП.

Множество пар координат формирует дискретное пространство $M = \{(i, j): i = 0, ..., m - 1;\ j = 0, ..., n - 1\}$, называемое *пространством имен*. Множество всех клеточных массивов с одним и тем же пространством имен M будем обозначать $\mathbf{C}(M)$. Иногда бывает удобным явно выделить в клетке ее *имя* (i, j) и *состояние* c_{ij}, т. е. записывать клетку в виде $(c_{ij}, (i, j))$. В пространстве имен M зададим *именующие функции* $\varphi(i, j)$: $M \Rightarrow M$, определяющие существование межклеточных связей. Для каждой клетки множество других клеток, с которыми она связана, формирует ее *соседство* и задается *шаблоном связей*:

$$T(i, j) = \{\varphi_1(i, j), ..., \varphi_q(i, j)\},$$

где $\varphi_k(i, j) \neq \varphi_l(i, j)$ для любых $(i, j) \in M$ и любых $k, l = 1, ..., q, k \neq l$.

Отображение $S: M \Rightarrow Q(T)$, где $Q(T)$ — множество подмножеств $C \in \mathbf{C}(M)$, порожденное шаблоном $T(i, j)$ для всех $(i, j) \in M$, заданное в виде

$$S(i, j) = \{(x_1, \varphi_1(i, j)), ..., (x_q, \varphi_q(i, j))\}, \tag{1}$$

называется *функцией соседства*. Из нее можно извлечь набор переменных: $X(i, j) = (x_1, ..., x_q)$, называемый *вектором переменных состояний*. В нашем случае именующие функции представляют собой простейшие функции сдвига, например, $\varphi_1(i, j) = (i + 1, j)$ дает имя соседа «справа» от клетки (i, j), $\varphi_2(i, j) = (i, j - 1)$ дает имя соседа «сверху» и т. д. Однако на граничных клетках некоторые полученные имена соседей могут выходить за пределы множества имен M. В этих случаях во избежание проблем, связанных с краевыми эффектами, будем полагать состояния таких «виртуальных» соседей равными нулю. На рис. 1 показаны шаблоны связей и функция соседства. Значение функции (1) в определенной клетке массива $C \in \mathbf{C}(M)$ называется *клеточным соседством* и получается подстановкой $c_{\varphi_k(i,j)}$ на x_k в формуле (1). В дальнейшем значение $X(i, j)$ на некоторой клетке (i, j) массива $C \in \mathbf{C}(M)$ будем обозначать $C_{ij} = (c_1, ..., c_q)$, где c_l — состояние соседа клетки (i, j) с номером l. Кроме того, нам понадобится обозначение

$$D_{ij} = c_{ij}C_{ij} = \{d_1, ..., d_q\},$$

означающее *приведенное соседство* клетки (i, j). Таким образом, C_{ij} и W_{ij} — это векторы, состоящие из q элементов, причем их нумерация согласована, т. е. l-й элемент представляет собой состояние l-го соседа и вес связи с l-м соседом соответственно. Поэтому эти векторы можно рассматривать как элементы q-мерного векторного пространства, в котором определено стандартное скалярное произведение: $\langle C_{ij}, W_{ij}\rangle = \sum_l c_l w_l$, а длины векторов вычисляются следующим образом: $\| W_{ij}\| = (\langle W_{ij}, W_{ij}\rangle)^{1/2}$.

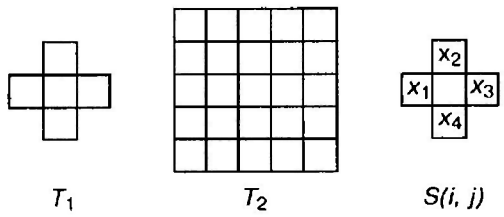


Рис. 1. Пространственное представление шаблонов связей T_1 и T_2 и функции соседства, порожденной T_1

Правило **Φ** функционирования КНАП описывается следующей итерационной процедурой.

Процедура. *Пусть* $C(t)$ *обозначает массив* **C** *после итерации* t. *Тогда:*

1) *все нейроны в* $C(t)$ *синхронно вычисляют следующую функцию:*

$$f(C_{ij}, W_{ij}) = \begin{cases} 1, & \text{если } \langle C_{ij}, W_{ij} \rangle > 0, \\ -1 & \text{иначе,} \end{cases} \tag{2}$$

и ее результат становится состоянием нейрона c_{ij} *на итерации* $(t + 1)$;

2) *если* $C(t + 1) = C(t)$, *тогда* $C(t) = \Phi(C(0))$ *есть результат вычислений, который соответствует устойчивому состоянию КНАП.*

Действуя таким путем, КНАП производит хранение и распознавание множества образов (являющихся устойчивыми состояниями) $\{P^0, \ldots, P^{L-1}\}$, $P^K \in \mathbf{C}(M)$, которые называются *прототипами* и заданы как клеточные массивы. Распознаваемые образы подаются на вход путем установки КНАП в соответствующее начальное состояние $C(0)$. Применение вышеописанной итерационной процедуры к любому $C(0)$ либо приводит к устойчивому состоянию, которое наиболее «похоже» на $C(0)$, либо возникают колебания, т. е. последовательность состояний сети описывается следующим образом: $C(t) = C^1$, $C(t + 1) = C^2$, $C(t + 2) = C^1, \ldots$. На рис. 2 показан процесс распознавания образа P' для КНАП, где хранятся P^1 и P^2.

Соответствие каждого прототипа устойчивому состоянию КНАП должно обеспечиваться соответствующим определением значений весовых векторов, которое происходит в *процессе обучения* КНАП. Процедура обучения должна обеспечивать следующие динамические свойства обученной КНАП:

i) *индивидуальная устойчивость*, т. е. каждый прототип должен быть устойчивым состоянием КНАП;

ii) *k-аттрактивность* устойчивых состояний, т. е. задано число k такое, что каждое входное состояние, отличающееся от прототипа P^K в l клетках (l-искажение), $l \leq k$, должно быть распознано как P^K;

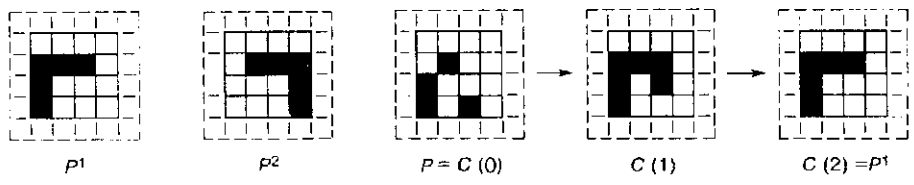


Рис. 2. Процесс распознавания образа P для КНАП:

P^1, P^2 — хранимые прототипы; $C(0), C(1), C(2)$ — процесс восстановления P^1 при начальном состоянии $C(0)$; W — множество весовых векторов (табл. 1) с функцией соседства $S(i, j)$ из рис. 1. Черный цвет обозначает состояние 1, белый — −1, пунктиром показаны «виртуальные» соседи, состояние которых равно 0

Таблица 1

j, i	0	1	2	3
0	(0, 2, 0, 0)	(0, 2, −2, 0)	(0, 2, −2, 2)	(0, 0, 0, 2)
1	(0, 0, 2, 0)	(−2, 2, −2, 0)	(−2, 0, −2, 0)	(2, 0, 2, 0)
2	(2, 0, 2, 0)	(−2, 2, 2, 0)	(−2, 0, 2, 2)	(2, 0, 2, 0)
3	(2, 0, 0, 0)	(2, 2, 0, 0)	(1, 2, 0, 2)	(2, 0, 0, 0)

iii) *минимум ложных образов*, т. е. должно быть как можно меньше устойчивых состояний, отличающихся от прототипов.

1.2. *Индивидуальная устойчивость.* Свойство индивидуальной устойчивости (i) множества прототипов $\{P^0, ..., P^{L-1}\}$ может быть записано в следующем виде:

$$\Phi(P^K) = P^K \text{ для всех } K = 0, ..., L-1.$$

Поскольку в (2) использовался знак «>», то это условие соответствует *строгой устойчивости* (strong stable) [9] каждого прототипа. Теперь введем понятие *локальной различимости* прототипов аналогично понятию *линейной различимости* [8] для полносвязных сетей.

Определение 1. *Прототипы $P^0, ..., P^{L-1}$ локально различимы в КНАП, если существует такой набор весовых векторов $\mathbf{W} = \{W_{ij}\}$, что условие*

$$\langle D_{ij}^K, W_{ij} \rangle > 0 \tag{3}$$

выполняется для всех $K = 0, ..., L-1$ и всех $(i, j) \in M$.

Очевидно, что в КНАП с весовыми векторами, которые удовлетворяют (3) для каждого $(i, j) \in M$, прототипы $P^0, ..., P^{L-1}$ будут индивидуально устойчивы. И наоборот, если в некоторой КНАП прототипы являются индивидуально устойчивыми, то они, очевидно, будут локально различимыми.

Теперь перейдем к описанию адаптированного алгоритма обучения персептрона:

Алгоритм 1. *Пусть даны прототипы $P^0, ..., P^{L-1}$, которые должны храниться в КНАП.*

Шаг 1. *Из данных прототипов организуем бесконечную последовательность $P^0, ..., P^{L-1}, P^0, ..., P^{L-1}, P^0, ...$, полученную из исходного множества прототипов его циклическим повторением, и введем сквозную нумерацию всех членов данной последовательности, т. е. сформируем цепочку образов $\{P^{(t)} \mid P^{(t)} = P^S$ при $t \equiv S \pmod{L}\}$. Период обучения, в течение которого на вход КНАП были поданы прототипы $P^0, ..., P^{L-1}$, будем называть* **макроитерацией** *обучения.*

Шаг 2. *Начальные значения компонент векторов W_{ij}^0 выбираем произвольно для всех $(i, j) \in M$.*

Шаг 3. *Весовые векторы изменяются согласно следующей итеративной процедуре для каждого $(i, j) \in M$:*

$$W_{ij}^{t+1} = \begin{cases} W_{ij}^t & \text{при } \langle D_{ij}^{(t)}, W_{ij}^t \rangle > 0, \\ W_{ij}^t + D_{ij}^{(t)} & \text{иначе.} \end{cases} \tag{4}$$

Вычисления прекращаются, если в течение одной макроитерации весовые векторы не менялись.

Алгоритм 1 сходится за конечное число итераций, если исходные прототипы были локально различимы. В таком случае в обученной КНАП все прототипы будут индивидуально устойчивы. Если же процесс обучения не завершается, то это означает, что исходные прототипы не являются локально различимыми.

В некоторых случаях уже заранее можно сказать, что прототипы не могут храниться одновременно в некоторой КНАП. Эту и некоторую другую информацию о прототипах можно узнать, опираясь на приведенные ниже теоремы, которые будут даны без доказательств. Для формулировки некоторых теорем нам понадобится

Определение 2. *Пусть P^G и P^H — два прототипа, тогда множество номеров соседей*

$$Id_{ij} = \{k: p_k^G p_k^H = p_{ij}^G p_{ij}^H\}$$

будем называть **идентифицирующим множеством,** *а те соседние клетки, чьи номера вошли в это множество, назовем* **идентифицирующими клетками.**

Другими словами, если состояния клеток p_{ij}^G и p_{ij}^H совпадают, то совпадают состояния идентифицирующих клеток и наоборот.

Теорема 1 [11]. *Если два прототипа P^G и P^H из $\mathbf{C}(M)$ локально различимы, то на каждом нейроне $(i, j) \in M$ мощность идентифицирующего множества не меньше единицы, т. е.*

$$|Id_{ij}| \geq 1 \; \forall \; (i, j) \in M.$$

Данная теорема дает необходимое условие локальной различимости, поэтому если в исходном множестве прототипов хоть одна пара не имеет идентифицирующих клеток на некотором нейроне, то процесс обучения не сойдется за конечное число итераций. То, что попарная локальная различимость не влечет за собой выполнимости (3) для всех $P^0, \ldots, P^{L-1}$, $L > 2$, вытекает из следующей теоремы.

Теорема 2 [11]. *Для любого прототипа P^0 существуют три образа (P^1, P^2, P^3) таких, что все они попарно локально различимы, но в совокупности для них условие (3) невыполнимо.*

1.3. *k-аттрактивность.* Рассмотрим свойство (ii) — k-аттрактивность устойчивых состояний. Для полносвязных сетей был предложен алгоритм, обеспечивающий 1-аттрактивность прототипов [9]. Его идея заключается в следующем: на вход нейронной сети для обучения подаются не только сами прототипы, но и все их 1-искажения, причем на последних сеть заставляют распознавать исходный прототип. Говоря более формально, этот алгоритм описывается следующим образом:

Алгоритм 2. *Пусть даны прототипы $P^0, \ldots, P^{L-1}$, которые должны храниться в КНАП.*

Шаг 1. *Из данных прототипов организуем бесконечную последовательность $P^0, \ldots, P^{L-1}, P^0, \ldots, P^{L-1}, P^0, \ldots$, полученную из исходного множества прототипов его циклическим повторением, и введем сквозную нумерацию всех членов данной последовательности, т. е. сформируем цепочку образов $\{P^{(t)} \mid P^{(t)} = P^S$ при $t \equiv S \pmod{L}\}$.*

Шаг 2. *Начальные значения компонент векторов W_{ij}^0 выбираем произвольно для всех $(i, j) \in M$.*

Шаг 3. *Для каждого прототипа $P^{(t)}$ строятся все его 1-искажения $\hat{P}^{(t)}$ (их $m \times n$), которые подаются на вход КНАП для обучения, а весовые векторы изменяются согласно следующей итеративной процедуре для каждого $(i, j) \in M$:*

$$W_{ij}^{t+1} = \begin{cases} W_{ij}^t & \text{при } \langle \hat{D}_{ij}^{(t)}, W_{ij}^t \rangle > 0, \\ W_{ij}^t + D_{ij}^{(t)} & \text{иначе,} \end{cases} \tag{5}$$

где $\hat{D}_{ij}^K$ — приведенное соседство нейрона (i, j) в некотором 1-искажении $\hat{P}^{(t)}$.

Основное достоинство данного алгоритма — гарантия 1-аттрактивности прототипов в случае успешной сходимости процесса обучения. К недостаткам алгоритма 2 можно отнести большее время обучения по сравнению с алгоритмом 1, поскольку на вход теперь подается Lmn образов (в лучшем случае Lq, q вследствие локальности связей нейронов [11]). Кроме того, данный алгоритм хотя и легко обобщается на случай k-аттрактивности, но реализовать его становится трудно уже при $k = 2$.

Аналогично предыдущим теоремам можно привести необходимое условие сходимости алгоритма 2 (обобщенного на случай k-аттрактивности).

Теорема 3 [11]. *Для сходимости алгоритма 2 (обобщенного на случай k-аттрактивности) необходимо, чтобы для любых двух прототипов P^G и*

P^{II} на каждом нейроне $(i, j) \in M$ мощность идентифицирующего множества была не меньше $2k + 1$, т. е.

$$|Id_{ij}| \geq 2k + 1 \; \forall \; (i, j) \in M.$$

Приведенные теоремы помогают при имитационном моделировании КНАП на компьютерах. Это выполняется следующим образом. В процессе обучения отмечаются на прототипах те нейроны, которые изменили весовые векторы. Если обучение по алгоритму 1 на каком-то нейроне не завершается, то проверяется условие (3) для тех прототипов, на которых этот нейрон не может обучиться. Непосредственная проверка выделенных прототипов в 60—80 % случаев показала, что условия теоремы 1 не выполнялись (т. с. прототипы не были попарно локально различимыми). В таких случаях небольшая коррекция одного или нескольких прототипов может привести к завершению обучения на этом нейроне, и если таких нейронов немного, то путем некоторого изменения исходных прототипов можно добиться сходимости процесса обучения для всей КНАП. Как показали эксперименты, такое обучение с некоторой корректировкой прототипов позволяет хранить в КНАП порядка $(1{,}5 \div 2{,}5)q$ образов для $q \sim 15 \div 25$, причем все они индивидуально устойчивы.

2. **Метод обучения.** 2.1. *Обоснование метода.* Остановимся теперь на проблеме достижения k-аттрактивности прототипов. Для этого проанализируем как алгоритм 1, так и алгоритм 2. Как известно, чем больше в обученной КНАП значения

$$m_{ij}^K = \langle D_{ij}^K, W_{ij} \rangle,$$

тем лучше сеть распознает искажения прототипов [12]. Из описания алгоритма 1 видно, что он гарантирует в случае сходимости процесса обучения $m_{ij}^K > 0 \; \forall \; (i, j) \in M, \; \forall \; K = 0, \ldots, L - 1$, т. е. индивидуальную устойчивость прототипов. Теперь подробно остановимся на рассмотрении второго алгоритма обучения. В алгоритме 2 изменение весовых векторов производится в соответствии со знаком следующей величины:

$$h_{ij}^K = \langle \hat{D}_{ij}^K, W_{ij} \rangle, \tag{6}$$

где $\hat{D}_{ij}^K$ — 1-искажение приведенного соседства D_{ij}^K. Допустим, что искажен сосед с номером s, тогда, следуя [12], перепишем (6) в виде

$$h_{ij}^K = \langle D_{ij}^K, W_{ij} \rangle - 2d_s^K w_s, \tag{7}$$

где $d_s^K w_s$ имеет смысл влияния s-го соседа на величину h_{ij}^K. Выберем максимальное по всем соседям значение $d_s^K w_s$ (пусть это будет r-й сосед) и будем обозначать его

$$\text{maxim}_1^K = d_r^K w_r = \max_s \{d_s^K w_s\}. \tag{8}$$

Если теперь для этого номера r величина (7) больше нуля, то величина (6) будет больше нуля для любого 1-искажения соседства D_{ij}^K. Таким образом, вместо того, чтобы на вход КНАП подавать все 1-искажения прототипа и на каждом из них вычислять (5), можно поступить следующим образом.

1. На вход КНАП подать для обучения прототип P^K.

2. Вычислить максимальное влияние 1-искажений по (8) и соответствующий номер соседа r.

3. Вычислить величину h_{ij}^K по формуле (7) для полученного на предыдущем шаге номера r и изменить весовой вектор аналогично (4), только вместо $\langle D_{ij}^K, W_{ij} \rangle$ использовать h_{ij}^K.

Очевидно, что проведенные преобразования, не нарушив свойств алгоритма 2, значительно упростили его с реализационной точки зрения. И кроме того, теперь преобразованный алгоритм можно не только обобщить на случай k-аттрактивности, но и легко реализовать это обобщение. Для этого необходимо вычислить максимальное влияние k-искажений по формуле

$$\text{maxim}_k^K = \max(\underbrace{d_{s_1}^K w_{s_1} + \ldots + d_{s_k}^K w_{s_k}}_{k \text{ слагаемых}}), \tag{9}$$

а вместо (7) с нулем сравнивать величину

$$\langle D_{ij}^K, W_{ij} \rangle - 2\text{maxim}_k^K. \tag{10}$$

З а м е ч а н и е. В формуле (9) максимум сумм из k слагаемых означает не что иное, как сумму k первых элементов множества $\{d_s^K w_s : s = 1, \ldots, q\}$, упорядоченного по убыванию значений его элементов. Следовательно, сложность вычисления максимального влияния k-искажений растет линейно с увеличением k.

Обобщая все вышеизложенное, заметим, что если мы хотим получить k-аттрактивность прототипов в обученной КНАП, то для этого можно воспользоваться алгоритмом 1, в формуле (4) которого необходимо $\langle D_{ij}^K, W_{ij} \rangle$ заменить на (10). Такую модификацию алгоритма 1 будем обозначать как *алгоритм* 1^k.

Однако теперь встает самый главный вопрос: как выбрать максимально возможное k? Одно из решений — делать это в процессе обучения, т. е. в случае, если на нейроне обучение сошлось для некоторого k, попытаться продолжить процесс обучения, но уже для $k + 1$. К недостаткам такого подхода можно отнести его сложность, поскольку необходимо уметь определять несходимость процесса обучения для выбранного k, а затем возвращаться к предыдущему значению $k - 1$, и, кроме того, большой разрыв между вычисляемыми параметрами обучения: имеется в виду большая разница между maxim_k и maxim_{k+1}. Поэтому был разработан алгоритм, в котором учитывались эти недостатки.

2.2. *Алгоритм 3.* Идея нового алгоритма состоит в следующем. Из (10) и (9) видно, что в случае успешного завершения обучения по алгоритму 1^k гарантируется $m_{ij}^K > \text{maxim}_k^K$, причем чем больше maxim_k^K, тем лучше. Если maxim_k^K в (10) заменить некоторым параметром обучения α_{ij}, то в результате получим $m_{ij}^K > \alpha_{ij}$. Основная проблема — выбрать такой алгоритм изменения параметров обучения α_{ij}, чтобы, во-первых, получить максимально возможную аттрактивность прототипов и, во-вторых, избежать возвратов к предыдущим значениям параметра. Предлагается следующий вариант: в течение макроитерации обучения вычислять значение $\mu_{ij} = \min_K m_{ij}^K$, и если оно больше значения α_{ij} на этой же макроитерации, то новое значение параметра обучения полагать $\alpha_{ij} = \mu_{ij}$. Этим обеспечивается, во-первых, максимально возможное значение m_{ij}^K. Во-вторых, вычисление параметра обучения гораздо проще, чем в (9). И в-третьих, разница между параметрами обучения динамически изменяется в процессе обучения. Теперь перейдем к описанию самого алгоритма.

Алгоритм 3. *Пусть даны прототипы* $P^0, \ldots, P^{L-1}$, *которые должны храниться в КНАП.*

Шаг 1. *Из данных прототипов организуем бесконечную последовательность* $P^0, \ldots, P^{L-1}, P^0, \ldots, P^{L-1}, P^0, \ldots$, *полученную из исходного множества прототипов его циклическим повторением, и введем сквозную нумерацию всех членов данной последовательности, т. е. сформируем цепочку образов* $\{P^{(t)} \mid P^{(t)} = P^S \text{ при } t \equiv S(\text{mod}L)\}$.

Шаг 2. *Начальные значения компонент векторов* W_{ij}^0 *выбираем произвольно для всех* $(i, j) \in M$, *параметр обучения* $\alpha_{ij}^0 = 0$, $r = 0$ — *счетчик макроитераций.*

Шаг 3. *Начало макроитерации;* $\mu_{ij}^t = \infty$; *выполнение шага 4 для всего исходного множества прототипов.*

Шаг 4. *Весовые векторы изменяются согласно следующей итеративной процедуре для каждого* $(i, j) \in M$:

$$W_{ij}^{t+1} = \begin{cases} W_{ij}^t & \text{при } \langle D_{ij}^{(t)}, W_{ij}^t \rangle > \alpha_{ij}^t; \\ W_{ij}^t + D_{ij}^{(t)} & \text{иначе;} \end{cases}$$

$$\mu_{ij}^{t+1} = \min(\mu_{ij}^t, \langle D_{ij}^{(t)}, W_{ij}^t \rangle).$$

Шаг 5. *Если выполнено условие остановки и весовые векторы в течение макроитерации не менялись, то вычисления прекращаются; иначе если весовые векторы менялись, то на шаг 3. Если условие остановки не выполнено, то новый параметр обучения вычисляется по следующей формуле:*

$$\alpha_{ij}^{r+1} = \max(\alpha_{ij}^r, \mu_{ij}^t),$$

и переход на шаг 3.

Отметим некоторые свойства алгоритма 3:

— если за время макроитерации обучения r весовой вектор изменялся, то параметр обучения не меняется, т. е. $\alpha^{r+1} = \alpha^r$, а значит, на начальном этапе данный алгоритм полностью совпадает с алгоритмом 1 ($\alpha_{ij}^0 = 0$), что обеспечивает индивидуальную устойчивость прототипов;

— если весовой вектор не менялся, то параметр обучения увеличивается и становится равным минимальному скалярному произведению;

— после обучения имеем $m_{ij}^K > \alpha_{ij}$, где $\alpha_{ij} > 0$ — последний параметр обучения перед остановкой алгоритма. Если же хоть один из $\alpha_{ij} = 0$, то в этом случае нельзя гарантировать даже индивидуальную устойчивость прототипов.

Теперь рассмотрим условия остановки алгоритма 3. Чем больше число проведенных макроитераций, тем больше значения m_{ij}^K. Однако величины m_{ij}^K зависят также от длин весовых векторов (из свойств скалярного произведения), а поскольку во время обучения длины весовых векторов увеличиваются, соответственно увеличиваются и m_{ij}^K, хотя относительные величины $m_{ij}^K / \| W_{ij} \|$ могут изменяться мало. Исходя из этого, приведем несколько из возможных условий остановки: 1) ограничение определенным заранее заданным числом макроитераций; 2) слежение за длиной весового вектора $\| W_{ij} \|$ и прекращение изменения параметров после достижения ею некоторого значения; 3) слежение за ростом относительных величин $m_{ij}^K / \| W_{ij} \|$ и прекращение изменения параметров обучения либо по достижении ими некоторых заранее заданных значений, либо при замедлении темпов их роста. Заметим, что выбор ограничений во всех вариантах полностью зависит от исходной задачи: размерности связей, числа прототипов и в немалой степени от самих прототипов.

Тесты некоторых моделей показали, что алгоритм 3 обеспечивает хороший уровень оптимизации m_{ij}^K уже тогда, когда число макроитераций всего на 20—30 больше, чем нужно для сходимости алгоритма 1.

2.3. *Ускорение алгоритмов обучения.* В дальнейших рассуждениях как весовые векторы, так и векторы состояний соседей будут рассматриваться как элементы векторного пространства R^q. Из описания предыдущих алгоритмов видно, что изменение весового вектора происходит следующим образом:

$$W_{ij}^{t+1} = W_{ij}^t + D_{ij}^{(t)},$$

где длина вектора $D_{ij}^{(t)}$ зависит от числа связей нейрона. В случае когда длина весового вектора еще относительно невелика, как, например, в начале обучения, такая добавка сильно изменяет направление вектора W_{ij}^{t+1}. Когда же процесс обучения близится к концу, то длина весового вектора становится достаточно большой по сравнению с длиной вектора $D_{ij}^{(t)}$, и поэтому такие добавки имеют гораздо меньшее влияние, что приводит к более точной корректировке положения весового вектора W_{ij}^{t+1}. Кроме того, в конце процесса обучения много итераций, существенно не меняя положения весового вектора, увеличивают только его длину. Эти размышления наводят на мысль о том, что можно уменьшить время сходимости алгоритма, принудительно увеличивая длину весового вектора в процессе обучения сети. Это можно делать следующим образом: через три—четыре макроитерации обучения умножать весовые векторы на 2. Как показало моделирование, такая процедура ускоряет сходимость алгоритма 1 примерно в 2 раза.

3. **Введение автосвязи в КНАП.** 3.1. *Исправление 1-искажений.* Во всех вышеописанных методах обучения предполагается, что в КНАП нет автосвязи, т. е. вес связи равен нулю, хотя известно, что положительная автосвязь уменьшает число циклических состояний, поэтому введение автосвязи во многом улучшило бы свойства КНАП. Однако ввести ее на этапе обучения оказалось невозможным, поскольку исследуемые алгоритмы итерационные, а прибавление единицы к весу автосвязи на каждой итерации приводит к неконтролируемому росту значения этого веса. Поэтому было принято решение ввести автосвязь после успешного завершения обучения, причем для определения веса связи (который обозначим w_0) использовались следующие соображения.

1. В КНАП без автосвязи состояние любого нейрона на следующей итерации распознавания полностью определяется состоянием его соседства, т. е. не будет зависеть от предыдущего его состояния. После введения автосвязи новое состояние нейрона зависит от предыдущего, поэтому необходимо из условия минимума ложных хранимых образов (iii), чтобы влияние соседства на состояние нейрона было больше, чем влияние автосвязи, т. е. после введения автосвязи мы хотим иметь (вес w_0 относится к нейрону (i, j))

$$\langle W_{ij}, D_{ij}^K \rangle \pm w_0 > 0 \; \forall \; K = 0, \ldots, L-1.$$

Таким образом получено ограничение сверху:

$$|w_0| < \min_K \langle W_{ij}, D_{ij}^K \rangle. \tag{11}$$

2. Введение автосвязи должно максимально улучшать способности сети по исправлению k-искажений. Для $k = 1$ это означает, что если состояние нейрона (i, j) в 1-искажении $\hat{P}^K$ равно p_{ij}^K, то оно останется таким же и после проведения одной итерации распознавания при

$$\langle W_{ij}, \hat{D}_{ij}^K \rangle + w_0 > 0, \tag{12}$$

где $\hat{D}_{ij}^K$ — 1-искажение приведенного соседства D_{ij}^K. Таким образом, положительная автосвязь уменьшает влияние 1-искажений на состояние нейрона, причем чем больше ее вес, тем устойчивее нейрон к 1-искажениям в своей окрестности. Учитывая все вышеизложенное, из (11) и (12) получаем результат:

$$w_0 = \min_K \langle W_{ij}, D_{ij}^K \rangle - \delta,$$

где $\delta > 0$ — некоторое маленькое число. Для моделей с целочисленным типом данных полагаем δ равным минимальному положительному числу — единице, т. е.

$$w_0 = \min_K \langle W_{ij}, D_{ij}^K \rangle - 1. \tag{13}$$

3.2. *Радиус устойчивости нейрона.* Теперь выясним, как автосвязь, вычисленная по (13), влияет на исправление k-искажений для любых $k \geq 1$. Для этого нам потребуются некоторые понятия. Положим

$$h_{ij}^K = \begin{cases} \langle W_{ij}, D_{ij}^K \rangle & \text{без автосвязи,} \\ \langle W_{ij}, D_{ij}^K \rangle + w_0 & \text{с автосвязью,} \end{cases} \tag{14}$$

величина h_{ij}^K имеет смысл модуля потенциала нейрона (i, j) на прототипе P^K. Из (10) имеем, что нейрон (i, j) будет правильно распознавать свое состояние при любых k-искажениях его соседства D_{ij}^K для тех k, которые удовлетворяют условию

$$h_{ij}^K - 2\operatorname{maxim}_k^K > 0,$$

где maxim_k^K определяется по формуле (9). Поскольку величину maxim_k^K можно ограничить сверху значением $k\max_l |w_l|$, то из

$$h_{ij}^K - 2k\max_l |w_l| > 0$$

получаем следующую оценку для k ($\max_l |w_l|$ в случае отсутствия автосвязи обозначает максимум компонент весового вектора W_{ij}, а в случае с автосвязью — максимум из всех весов связей, включая и вес автосвязи):

$$k < \frac{h_{ij}^K}{2\max_l |w_l|}.$$

Отсюда следует, что на состояние нейрона (i, j) не будут влиять никакие k-искажения любого прототипа как минимум для тех k, которые будут меньше

$$k_{ij} = \min_K \frac{h_{ij}^K}{2\max_l |w_l|}. \tag{15}$$

Число k_{ij} будем называть *радиусом устойчивости* нейрона (i, j) (нейрон устойчив к искажениям, если их число не превосходит радиуса устойчивости).

3.3. *Влияние автосвязи на радиус устойчивости нейрона.* Задача состоит в том, чтобы, во-первых, выяснить влияние веса автосвязи (13) на радиус устойчивости нейрона, а во-вторых, получить выражение для веса автосвязи, максимально увеличивающего радиус устойчивости. Для этого получим и сравним между собой выражения для радиуса устойчивости k_{ij} без автосвязи и k'_{ij} с автосвязью. Если соединить формулы (14) и (15), то получим выражения для радиуса устойчивости нейрона с автосвязью

$$k'_{ij} = \inf_K \frac{\langle W_{ij}, D_{ij}^K \rangle + w_0}{2\max_l |w_l|} \tag{16}$$

и без автосвязи

$$k_{ij} = \inf_{K} \frac{\langle W_{ij}, D_{ij}^{K} \rangle}{2\max\limits_{i>0} |w_i|}. \tag{17}$$

Из этих формул еще не видно, как влияет введение автосвязи на изменение радиуса устойчивости нейрона, поэтому соединим их с формулой (13) и получим соотношения

$$k'_{ij} = \frac{2w_0 + 1}{2\max\limits_{i>0} |w_i|}, \tag{18}$$

$$k_{ij} = \frac{w_0 + 1}{2\max\limits_{i>0} |w_i|}, \tag{19}$$

из которых можно вывести нужную нам зависимость. Для этого рассмотрим два случая: 1) $w_0 < \max\limits_{i>0} |w_i|$, поскольку знаменатели в (18) и (19) одинаковы, то радиус устойчивости k'_{ij} будет больше k_{ij}; 2) $w_0 > \max\limits_{i>0} |w_i|$, следовательно, поскольку $\min\limits_{i} |w_i| = w_0$, то $k'_{ij} = (2w_0 + 1)/(2w_0) < 2$, т. е. использование формул (13) в некоторых случаях уменьшает радиус устойчивости ($k'_{ij} < k_{ij}$), так как, например, если до введения автосвязи радиус устойчивости был больше двух, то после введения автосвязи он стал меньше двух. Однако заметим, что при этом $k'_{ij} > 1$, а это дает гарантию распознавания 1-искажений.

Итак, мы получили, что слишком большой вес автосвязи плохо сказывается на исправлении k-искажений для $k \geq 2$. Как ограничить вес автосвязи? Для ответа на этот вопрос введем параметр a и выразим значение веса автосвязи через максимум модулей компонент весового вектора, т. е. положим $w_0 = a\max\limits_{i>0} |w_i|$, а для определения a рассмотрим два случая: 1) пусть $a \leq 1$, тогда из (17) и (16) получим $k'_{ij} = k_{ij} + a/2$, откуда следует, что максимальное увеличение радиуса устойчивости будет при $a = 1$; 2) пусть $a \geq 1$, тогда из (17) и (16) $k'_{ij} = k_{ij}/a + 1/2$, следовательно, в обоих случаях максимальное увеличение радиуса устойчивости нейрона после введения автосвязи произошло при $a = 1$, т. е. наилучшим значением веса автосвязи будет $w_0 = \max\limits_{i>0} |w_i|$.

Таким образом, мы получили окончательный ответ для веса автосвязи, максимально увеличивающего радиус устойчивости нейрона: если $w_0 = \min\limits_{K} \langle W_{ij}, D_{ij}^{K} \rangle - 1 < \max\limits_{i>0} |w_i|$, то радиус устойчивости k'_{ij} становится больше, чем k_{ij}, а если $\min\limits_{K} \langle W_{ij}, D_{ij}^{K} \rangle - 1 > \max\limits_{i>0} |w_i|$, то, полагая $w_0 = \max\limits_{i>0} |w_i|$, получаем максимальное увеличение радиуса устойчивости нейрона на $1/2$ по сравнению со случаем без автосвязи. Если записать все вышеизложенное одним выражением, то получим следующее окончательное значение для веса автосвязи:

$$w_0 = \min(\max_{i>0} |w_i|, \min_{K} \langle W_{ij}, D_{ij}^{K} \rangle - 1), \tag{20}$$

который можно считать оптимальным с точки зрения увеличения радиусов устойчивости.

4. **Результаты моделирования.** 4.1. *Описание системы моделирования ALT.* Моделирование проводилось в системе ALT (Animating Language Tools), которая комбинирует текстовые и графические методы представления параллельных клеточных вычислений [2]. Она включает в себя транслятор со

специального высокоуровневого языка описания во внутреннее представление. ALT характеризуется следующими свойствами: языком высокого уровня для описания понятий и конструкций АПП, графической формой представления клеточных массивов и соседств, визуализацией вычислительного процесса, возможностью наблюдать за любой клеткой массива во время вычислительного процесса.

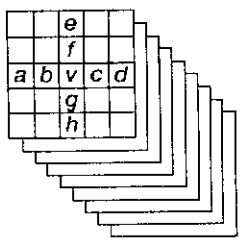
КНАП в системе ALT представляются в виде многослойного массива. Каждый массив имеет уникальное имя, которое участвует при описании алгоритмов в ALT-системе. Значения, хранимые в массиве, кодируются цветом, и затем содержимое массива отображается на экране компьютера. Например, клеточно-нейронная сеть, не имеющая автосвязи, представляется в виде многослойного массива из $q + 1$ слоев, в котором на нулевом, лицевом слое находятся клетки, содержащие состояния нейронов сети, а на k-м слое расположены веса связи w_k, причем если нейрон имеет координаты (i, j) на лицевом слое, то компоненты его весового вектора располагаются на соответствующих слоях в клетках с теми же координатами. Функция соседства также представляется трехмерным массивом, на первом слое которого именами переменных обозначено пространственное расположение соседей относительно центрального нейрона, а под центральной клеткой в глубь массива переменными обозначается относительное расположение компонент весового вектора (рис. 3, *a*). В дальнейшем эти переменные используются при написании функций; пример функции, вычисляющей новое значение компонент весового вектора, показан на рис. 3, *b*.

4.2. *Результаты экспериментов.* Проведены следующие эксперименты.

Эксперимент 1. Обучение, обеспечивающее индивидуальную устойчивость (алгоритм 1), проводилось на 50 прототипах с использованием ускоряющего метода. Процесс обучения завершился за 12 макроитераций (вместо 25 без ускорения). При этом сделано восемь корректировок прототипов, из них в шести случаях использовалось условие теоремы 1 для обеспечения локальной различимости прототипов; в двух остальных случаях конфликтовали четверки прототипов (теорема 2). После проведенной коррекции все прототипы были индивидуально устойчивы. Таким образом, получено соотношение $L/q \geq 2$. Проведена проверка способности обученной КНАП исправлять 1-искажения. Для этого выбрано шесть прототипов, и на вход сети подано по 400 1-искажений каждого из них. В табл. 2 для каждого прототипа из этой шестерки приведено число 1-искажений, не исправленных обученной КНАП. Из таблицы видно, что в среднем 30—35 % 1-искажений не исправилось.

Эксперимент 2. На тех же 50 прототипах КНАП обучалась по алгоритму обучения с параметром (алгоритм 3). Число макроитераций обучения было выбрано равным 60, причем на первых 25 макроитерациях обучение проводилось с ускорением. Поскольку алгоритм 3 на начальном этапе совпадает с алгоритмом 1, то в обученной КНАП все прототипы были индивидуально устойчивыми. Проверка способностей обученной КНАП исправлять 1-иска-

a



b

```
learn()
{
    w1 = w1 + va,
    w2 = w2 + vb,
     ⋮    ⋮    ⋮
    wq = wq + vh;
}
```

Рис. 3. Представление в системе ALT:
a — клеточно-нейронного массива; *b* — процедуры вычисления весов связей

Таблица 2

Прототип	Алгоритм 1	Алгоритм 3
A	145 (141)	140 (20)
B	148 (96)	97 (13)
C	125 (111)	90 (2)
D	116 (93)	82 (10)
F	145 (108)	91 (10)
I	128 (105)	119 (8)

Таблица 3

Прототип	26 прототипов	10 прототипов
A	49 (6)	2 (0)
B	32 (5)	0 (0)
C	30 (0)	6 (0)
D	45 (5)	4 (2)
F	47 (2)	6 (2)
I	57 (4)	5 (0)

жения проводилась так же, как и в эксперименте 1, и на тех же прототипах. Результаты эксперимента приведены в той же табл. 2. Эта КНАП не исправляла в среднем 20—30 % 1-искажений, т. е. улучшение по сравнению с экспериментом 1 на 5—10 %.

Эксперимент 3. Исследовалось влияние автосвязи на исправление 1-искажений. В модели, обученные в ходе экспериментов 1 и 2, была введена автосвязь, вес связи рассчитывался по формуле (20). Аналогично предыдущим экспериментам проверялась способность КНАП с автосвязью исправлять искажения. Результаты экспериментов приведены в табл. 2 в скобках. Из таблицы видно, что введение автосвязи улучшило исправление 1-искажений, причем в модели, обученной по алгоритму 3, улучшение было в 7—10 раз.

Эксперимент 4. Поскольку при соотношении $L/q \geq 2$ трудно надеяться на хорошее исправление 1-искажений, то было проведено два эксперимента для $L/q \simeq 1$ и $L/q \simeq 1/2$. Обучение проводилось по алгоритму 3 с 60 макроитерациями обучения. Первое множество прототипов состояло из 26 образов, обозначающих буквы латинского алфавита, второе — из 10 первых букв от *A* до *J*. Результат исправления 1-искажений обученными моделями приведен в табл. 3, откуда видно, что при соотношении $L/q \simeq 1/2$ КНАП с автосвязью исправила практически все 1-искажения. Те нейроны, искаженное состояние которых не исправилось, имели всего по 1-й идентифицирующей клетке.

Эксперимент 5. Цель этого эксперимента — определить примерную зависимость между радиусами устойчивостей нейронов в обученной КНАП и числом макроитераций обучения с параметром. Обучалась сеть на 26 прототипах (буквы латинского алфавита) по алгоритму 3 при разном числе макроитераций, и подсчитывалось число нейронов, у которых радиус устойчивости был меньше 1. Число таких нейронов в зависимости от числа макроитераций обучения приведено в табл. 4.

Примечание. Обучение по ускоренному алгоритму 1 завершилось за восемь макроитераций. Из таблицы видно, что после 30 итераций обучения число клеток с $k_{ij} < 1$ практически не меняется и составляет 7 % от общего числа нейронов.

Все описанные эксперименты проводились на одном и том же клеточно-нейронном массиве размером 20 × 20 нейронов с шаблоном связей T_2, показанным на рис 1, т. е. каждый нейрон имел 24 соседа в моделях КНАП без автосвязи. В моделировании использовались прототипы, представляющие собой символы латинского алфавита, цифры и различные символы типа скобок, стрелок и т. д., нарисованные «цветом» +1 толщиной в одну клетку (рис. 4). Все исходное множество прототипов насчитывало 50 символов, причем в некоторых случаях использовалась только часть этого множества.

Таблица 4

Число макроитераций	Число нейронов с $k_{ij} < 1$
15	34
20	28
30	27
40	27
60	27
100	25

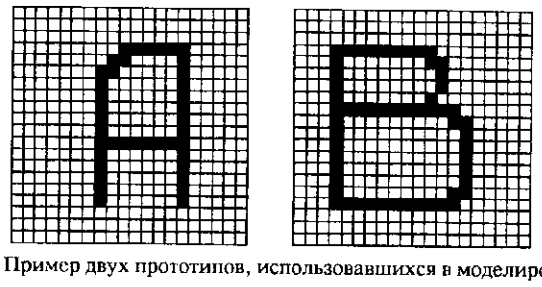


Рис. 4. Пример двух прототипов, использовавшихся в моделировании

Заключение. Исследованы методы обучения, на основании которых предложено два алгоритма обучения КНАП. Первый алгоритм (алгоритм 1^k) позволяет получить максимальную k-аттрактивность прототипов, его сложность растет менее чем линейно с увеличением k, но этот алгоритм сложен в реализации. Второй алгоритм обучения (алгоритм 3) имеет тот же порядок сложности, что и алгоритм обучения персептрона по Розенблатту, его реализация значительно проще, чем первого алгоритма, но он не всегда обеспечивает максимально возможную аттрактивность прототипов. Также приведено выражение для веса автосвязи, позволяющее ввести автосвязь в обученную КНАП, при этом максимально увеличиваются способности сети по исправлению искажений с точки зрения увеличения радиусов устойчивостей нейронов. Проведенные эксперименты показали значительное улучшение (в 7—10 раз) возможностей КНАП, обученной по алгоритму 3, исправлять 1-искажения прототипов.

СПИСОК ЛИТЕРАТУРЫ

1. **Hopfield J. J., Tank D. W.** Computing with neural circuits: a model // Science. 1986. **233**. P. 625.
2. **Achasova S., Bandman O., Markova V., Piskunov S.** Parallel substitution algorithm // Theory and Application. Singapore: World Scientific, 1995.
3. **Zhang J., Zhang L., Yan D. et al.** Local interconnection neural network and its optical implementation // Opt. Commun. 1993. **102**. P. 13.
4. **Arenzon J. J., Lemke N.** Simulating highly diluted neural network // Journ. Phys. A. Math. Gen. 1994. **27**. P. 5161.
5. **Liu D., Michel A.** Sparsely interconnected artificial neuron networks for associative memories // Lecture Notes in Comp. Sci. 1993. **606**. P. 155.
6. **Perzonas L., Guyon I., Dreyfus G.** Collective computational properties of neural networks: New learning mechanism // Phys. Rev. A. 1986. **34**. P. 4217.
7. **Michel A. M., Farell J. A., Sun H.** Analysis and synthesis techniques for hopfield type synchronous discrete time neural networks with application to associative memory // IEEE Trans. 1990. **CS-37**, N 11. P. 1356.
8. **Rosenblatt F.** Principles of Neurodynamics. Washington: Spartan, 1959.
9. **Zhuang X., Huang Y., Yu F. A.** Design of hopfield content-addressable memories // IEEE Trans. Signal Process. 1994. **42**, N 2. P. 492.
10. **Bandman O. L.** Cellular-neural computations, formal model and possible applications // Lecture Notes in Computer Science. 1995. **964**. P. 21.
11. **Bandman O. L., Pudov S. G.** Stability of stored patterns in cellular-neural associative memory // Bull. of the Novosibirsk Computer Center. Ser. Computer Science, 1996, N 4.
12. **Cottrell M.** Stability and attractability in associative memory networks // Biological Cybernetics. 1988. **58**. P. 129.

Поступила в редакцию 15 июля 1996 г.