МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ВОЛЖСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО
УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

В. И. Капля, Е.В. Капля

МОДЕЛИ И МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Электронное учебное пособие

Волжский
2019

УДК 004.8(07)
ББК 32.813я73
К 203

Р е ц е н з е н т ы:
канд. тех. наук, доцент, зав. кафедрой “Автоматизация технологических процессов и производств”, филиала ФГБОУ ВО «Национальный исследовательский университет «МЭИ» в г. Волжском
Болдырев И. А.,
инженер-программист второй категории ООО «Волгопромавтоматика»
Поспеев Ю.М.

Печатается по решению редакционно-издательского совета
Волгоградского государственного технического университета

Капля, В. И.
Модели и методы искусственного интеллекта [Электронный ресурс] : учебное пособие / В. И. Капля, Е.В. Капля ; ВПИ (филиал) ВолгГТУ, – Электрон. текстовые дан. (1 файл: 1,08 МБ). – Волжский, 2019. – Режим доступа: http://lib.volpi.ru. – Загл. с титул. экрана.
ISBN 978-5-9948-3466-4

Содержит сведения о моделях и методах интеллектуальных систем, которые могут использоваться в автоматических системах управления. Изложение теоретического материала сопровождается демонстрационными примерами.

Рекомендуется для использования в учебном процессе по техническим специальностям, в том числе по направлению бакалавриата 09.03.04 «Программная инженерия», при изучении дисциплин «Системы искусственного интеллекта» и «Ведение в разработку интеллектуальных систем», а также направления 15.04.04 «Автоматизация технологических процессов и производств» (уровень магистратуры), при изучении дисциплины «Интеллектуальные системы» для студентов дневной, вечерней и заочной форм обучения.

Ил. 16, библиограф.: 19 назв.

ISBN 978-5-9948-3466-4

Содержание

Введение

Внедрение интеллектуальных элементов и систем в автоматизированное производство и другие сферы человеческой деятельности базируется на росте быстродействия и адресного пространства цифровых процессоров, что позволяет реализовать сложные и объёмные алгоритмы интеллектуальной обработки информации. Моделям и методам построения интеллектуальных систем посвящено большое число научных книг, статей и программных продуктов, так как для описания и исследования интеллектуальных систем ввиду их разнообразия необходимо использовать разнообразные аналитические соотношения и проверять их истинность методами имитационного моделирования.

Интеллектуальные системы предназначены для повышения уровня автоматизации информационных и технических систем, повышения их производительности и качества работы. Интеллектуальные системы освобождают от рутинных действий человека, формируют информационное и операционное поле для продуктивной, творческой и комфортной работы.

Пути реализации интеллектуальных систем базируются на ряде принципиально отличающихся моделей:

- алгоритмические модели,
- экспертные модели,
- предикатные модели,
- нейросетевые модели,
- динамические модели,
- рекуррентные модели,
- самоорганизующиеся карты Кохонена,

- стохастические модели.

Модели интеллектуальных систем являются средством описания структуры решаемых интеллектуальных задач: решения логических уравнений, обучения, кластеризации. Перечень методов, используемых в работе интеллектуальных систем, включает: методы минимизация целевых функций, методы стохастической аппроксимации, методы оптимального управления.

В данном пособии рассматриваются модели и методы, используемые для разработки интеллектуальных систем. Выбраны базовые модели и методы, которые позволяют сформировать обобщающее представление о возможностях интеллектуальных систем и которые доступны для исследования в процессе лабораторных или практических занятий. Основное внимание уделяется обучаемым и самообучаемым нейронным сетям, кроме того, приводятся сведения об алгоритмах кластеризации данных и о динамических моделях нейронов. Изложение теоретического материала сопровождается демонстрационными примерами решения задач обучения и кластеризации.

1. Алгоритмы обучения нейронных сетей

Модели нейронных сетей позволяют создавать интеллектуальные модули систем управления различными объектами.

Решение задач обучения нейронных сетей реализуется путём использоваться следующих итерационных алгоритмов:

1. Алгоритмы локальной оптимизации с вычислением частных производных первого порядка:

- градиентный алгоритм (метод наискорейшего спуска),
- методы с одномерной и двумерной оптимизацией целевой функции в направлении антиградиента,
- метод сопряженных градиентов,
- методы, учитывающие направление антиградиента на нескольких шагах алгоритма.

2. Алгоритмы локальной оптимизации с вычислением частных производных первого и второго порядка:

- метод Ньютона,
- методы оптимизации с разреженными матрицами Гессе,
- квазиньютоновские методы,
- метод Гаусса-Ньютона,
- метод Левенберга-Марквардта.

3. Стохастические алгоритмы оптимизации:

- поиск в случайном направлении,
- имитация отжига,
- метод Монте-Карло (численный метод статистических испытаний).

4. Алгоритмы глобальной оптимизации (задачи глобальной оптимизации решаются с помощью перебора значений переменных, от которых зависит целевая функция).

1.1. Структура нейронных сетей

Нейронные сети состоят из множества связанных нейронов. Структура и свойства межнейронных связей может быть достаточно разнообразной, что даёт возможность решать с помощью нейронных сетей широкий круг задач интеллектуальной обработки информации.

Первичная модель нейрона была впервые предложена американскими учёными МакКаллок (McCalloch) и Питц (Pitt) в 1943. Модель нейрона (рис.1.1) представляет собой элементарный скалярный вычислитель с нелинейным преобразователем результата скалярного умножения вектора входного сигнала *X* и вектора весовых коэффициентов *W* нейрона.

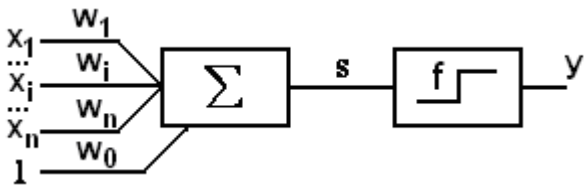

Рис.1.1. Первичная модель нейрона

Вектор входного сигнала $X = (x_1, x_2, ..., x_n)^T$ скалярно умножается на вектор весовых (синаптических) коэффициентов $(w_1, w_2, ..., w_n)^T$ и к результату прибавляется смещение w_0. Сигнал синаптической суммы s поступает на вход блока активационной функции *f(s)*. Активационная функция исполняет роль ограничителя выходного сигнала, что присуще биологическим системам. Она определят вид нелинейности характеристик нейрона, расширяет его функциональные возможности. Математические зависимости модели нейрона имеют следующий вид:

$$s = \sum_{i=1}^{n} w_i \cdot x_i + w_0, \qquad y = f(s).$$

В качестве активационной функции используются либо ступенчатые функции, либо непрерывные функции. Первый тип функций проще

реализуется, а второй необходим для аналитических исследований свойств нейронов. Смещение w_0 предназначено для устранения тривиального нулевого выходного сигнала при нулевом входном векторе.

Базовой структурой нейронных сетей считается многослойная структура, которая подразумевает разбиение нейронов на группы, образующие слои, то есть у нейронов слоя существует общий вектор входных сигналов, а выходы нейронов образуют вектор выходных сигналов нейронного слоя.

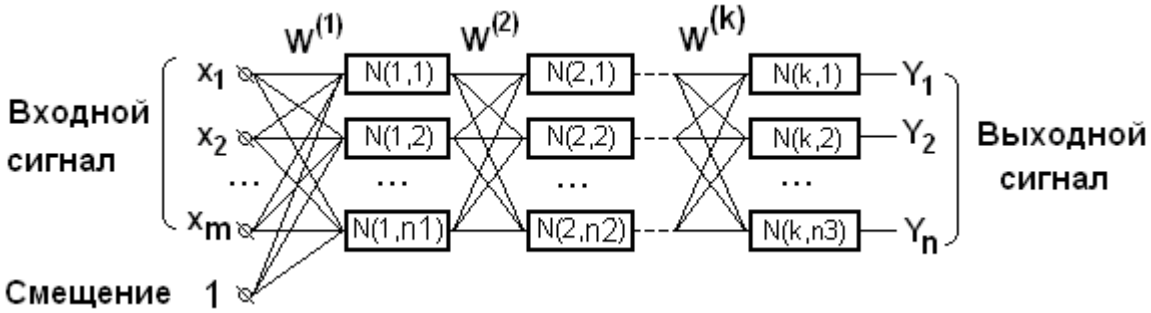

Рис.1.2. Многослойная полносвязанная нейронная сеть

Математическая модель НС, приведенной на рис.1.2, основана на матрицах весовых коэффициентов и имеет следующий вид:

$$Y = \tilde{f}(W^{(k)} \cdot \tilde{f}(\ldots \cdot \tilde{f}(W^{(2)} \cdot \tilde{f}(W^{(1)} \cdot X))\ldots)),$$

где $\tilde{f}(v)$- оператор поэлементного вычисления активационной функции $f(v_i \cdot)$ для элементов векторного аргумента v. Матрицы весовых коэффициентов объединяют нейроны слоя: каждой строке матрицы соответствует набор весовых коэффициентов соответствующего нейрона. Если нейронная сеть является полносвязанной, то перестановка строк в матрицах внутренних слоев не изменит выходной вектор.

Нейронные сети являются универсальными аппроксиматорами [1,2], так как на них распространяется теорема Колмогорова-Арнольда: непрерывная функция нескольких переменных может быть представлена в виде суперпозиций и операций сложения функций одной переменной.

1.2. Генетический алгоритм обучения нейронных сетей

Генетический алгоритм – это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов естественного отбора. Алгоритм является разновидностью эволюционных вычислений, с помощью которых решаются оптимизационные задачи с использованием методов эволюции, таких как наследование, мутации, отбор и кроссинговер. Метод требует использования обучающей выборки.

Генетический алгоритм обучения состоит в совмещении процедур случайного и направленного поиска оптимального решения задачи обучения. Рассматривается сразу несколько вариантов решения (*популяция*), которые случайным образом заполняют область допустимых решений. Из текущей популяции строится новое поколение путем применения специальных процедур.

Обучение нейронной сети с помощью генетического алгоритма осуществляется в процессе следующих этапов:

1. Инициализация весовых коэффициентов случайными значениями, которые берутся из заданного диапазона значений.
2. Вычисление параметра качества на всей обучающей выборке для каждого члена популяции. Обычно, параметр качества равен квадрату модуля вектора рассогласования выходного сигнала и соответствующего вектора обучающей выборки.
3. Селекция – выбор векторов весовых коэффициентов, которые будут участвовать в формировании новой популяции. Выбор осуществляется путём ранжирования по параметру качества.
4. Скрещивание (кроссинговер) – процедура случайного выбора двух разных членов популяции и последующее формирование нового члена

популяции путём смешивания по заданному правилу параметров выбранных членов популяции.

5. Мутация – изменение случайно выбранных в популяции векторов весовых коэффициентов по алгоритму, изменяющему компоненты вектора по заданному правилу с использованием случайных чисел.
6. Формирование новой популяции – объединение результатов применения в заданной пропорции процедур селекции, скрещивания и мутации в общую популяцию с неизменной численностью.

Достоинством генетического алгоритма является то, что он может применяться для решения сложных формализованных задач. В отличие от алгоритма обратного распространения ошибки, генетический алгоритм не требует вычисления дифференциальных поправок для компонент векторов весовых коэффициентов.

Применение генетического алгоритма к обучению нейронных сетей сводится к преобразованию матриц весовых коэффициентов по заданным и, в принципе, неоднозначно определяемым правилам. Хорошие результаты обучения дают следующие правила скрещивания и мутации матриц весовых коэффициентов:

$$W_{CROSS(A,B)}^{(p+1)} = W_A^{(p)} \cdot (1 - k_{CROSS}) + W_B^{(p)} \cdot k_{CROSS}$$

$$W_{MUT(A)}^{(p+1)} = -W_A^{(p)} + k_{MUT} \cdot (RND(1) - 0.5)$$

где p - номер популяции, $W_A^{(p)}, W_A^{(p)}$ - скрещиваемые матрицы (предки), k_{CROSS} - коэффициент скрещивания, $W_{CROSS(A,B)}^{(p)}$ - результирующая матрица (потомок), $W_{MUT(A)}^{(p+1)}$ - результат мутации, k_{MUT} - коэффициент мутации. Функция $RND(1)$ является генератором случайных чисел, подчиненных равномерному закону распределения и принадлежащих интервалу $[0,1]$. Коэффициент скрещивания k_{CROSS} рекомендуется назначать из интервала $[0,1]$. Коэффициент k_{MUT} является

масштабирующим, и если весовые коэффициенты генерируются в интервале $[-1,+1]$, то его можно приравнять 1.

Пример применения генетического алгоритма для обучения трехслойной НС распознаванию изображений приведен на рис.1.3. Об успешности процесса обучения целесообразно судить по величине невязки выходного сигнала.

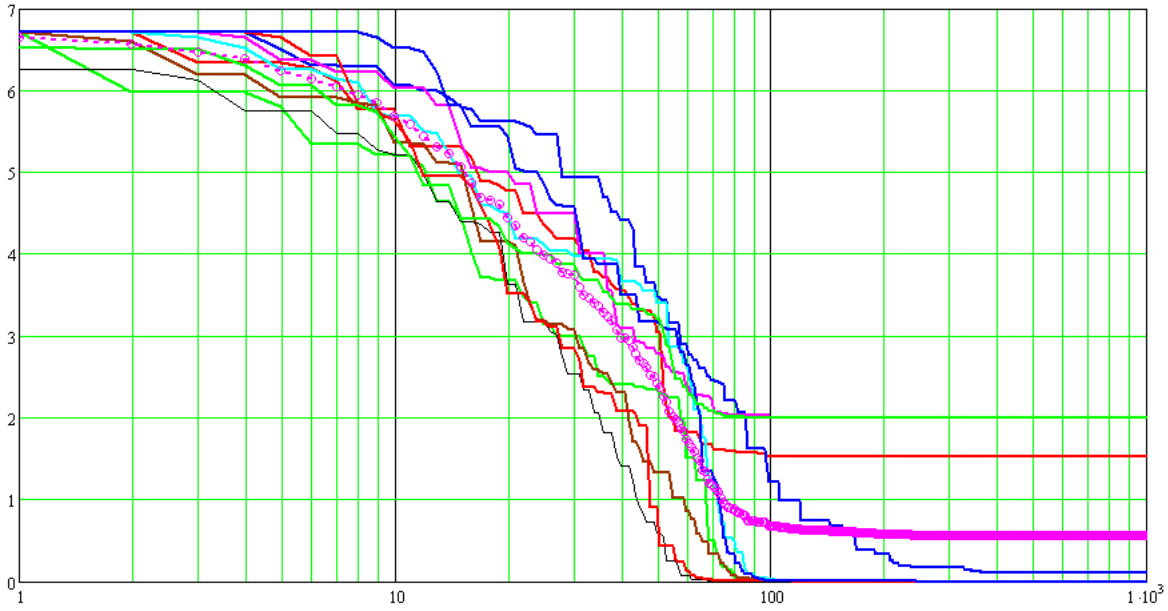

Рис.1.3. График изменения невязки в процессе обучения

На рисунке приведены зависимости невязки от номера шага обучения для 10-ти особей (наборов весовых матриц). Графики демонстрируют две особенности алгоритма:

- сочетание плавного и скачкообразного изменения невязки,
- некоторые особи прекращают обучаться.

Плавное изменение невязки соответствует процедуре скрещивания, а скачи – появлению удачных особей-мутантов.

Недостаток генетического алгоритма обучения: реализация алгоритма требует больших вычислительных ресурсов, по сравнению с методом обратного распространения ошибки.

Основное преимущество генетического алгоритма обучения состоит в том, что он не зависит от внутренней структуры обучаемой системы.

Алгоритмы МОРО и Хебба применимы только для обучения слоистых нейронных сетей без обратных связей, генетический алгоритм может быть применен к любому виду нейронных сетей или других видов систем трансформации информации.

1.3. Метод обратного распространения ошибки

Наиболее просто метод обратного распространения ошибки представляется в матричной форме. Прямое распространение сигнала по нейронной сети, состоящей из N слоев, описывается следующей системой уравнений:

$$s^{(1)} = W^{(1)} \cdot \begin{bmatrix} X \\ 1 \end{bmatrix}, \qquad Y^{(1)} = \overrightarrow{f(s^{(1)})}, \ldots,$$

$$s^{(i)} = W^{(i)} \cdot Y^{(i-1)}, \qquad Y^{(i)} = f(s^{(i)}), \ldots,$$

$$s^{(N)} = W^{(N)} \cdot Y^{(N-1)}, \quad Y^{(N)} = f(s^{(N)}).$$

Вектор входного сигнала наращивается единичной компонентой, так как необходимо учитывать величину смещения в каждом слое нейронов. Обратное распространение сигнала ошибки по нейронной сети позволяет вычислить корректирующие коэффициенты с помощью следующих уравнений:

$$\delta^{(N)} = (Y^{(N)} - U)\frac{\partial f(s^{(N)})}{\partial s^{(N)}}, \ldots,$$

$$\delta^{(i)} = \left(W^{(i+1)}\right)^T \cdot \delta^{(i+1)} \cdot \frac{\partial f(s^{(i)})}{\partial s^{(i)}}, \ldots,$$

$$\delta^{(1)} = \left(W^{(2)}\right)^T \cdot \delta^{(2)} \cdot \frac{\partial f(s^{(1)})}{\partial s^{(1)}}.$$

Коррекция весовых коэффициентов нейронной сети проводится с помощью следующих уравнений:

$$\Delta W^{(1)} = -\eta \cdot \delta^{(1)} \cdot [\, X^T, 1\,], \ldots, \qquad W^{(1)} = W^{(1)} + \Delta W^{(1)},$$

$$\Delta W^{(i)} = -\eta \cdot \delta^{(i)} \cdot Y^{(i)T}, \ldots, \qquad W^{(i)} = W^{(i)} + \Delta W^{(i)},$$

$$\Delta W^{(N)} = -\eta \cdot \delta^{(N)} \cdot Y^{(N)T}, \qquad W^{(N)} = W^{(N)} + \Delta W^{(N)}.$$

где η – коэффициент скорости обучения.

Приведенные выше уравнения являются компактной формой описания метода обратного распространения ошибки, используемого для обучения нейронных сетей. Применение данных уравнений позволяет строить модели нейронной сети с произвольным числом слоев.

1.4. Алгоритм роя частиц

Алгоритм Particle Swarm Optimization (PSO) был впервые предложен в работе [3,4]. Позднее вошел в употребление термин «swarm intelligence» [5], поскольку разработанная вычислительная технология оказалась пригодной для решения широкого круга сложных задач, относящихся к искусственному интеллекту. В отечественной технической литературе эти английские термины переводятся обычно как «стайный интеллект» или «ройный интеллект», а также «стайная оптимизация» или «оптимизация роем частиц».

Пусть, например, стая птиц занята поиском корма. Каждая птица отыскивает его самостоятельно и наблюдает за другими птицами. Как только одна из птиц заметит признаки еды, она подает сигналы остальным, которые летают неподалеку. Уровень сигнала зависит от количества пищи. Получив сигналы от соседей, все птицы корректируют свои траектории. Так формируется локальное взаимодействие, распространяющееся на всю стаю. В итоге стая собирается в месте с максимальным количеством корма, т. е. находит глобальный экстремум целевой функции.

Искусственные частицы, перемещающиеся в *N*-мерном поисковом пространстве, могут вести себя подобно стае птиц (или рою насекомых). Аналогом корма здесь является заданная целевая функция, минимум которой требуется найти. По аналогии с эволюционными вычислениями можно утверждать, что рой подобен популяции, а частицы (индивидуалы) соответствуют элементам популяции. Частицы летают в поисковом пространстве, изменяя направление в соответствии с собственным опытом и опытом соседей.

В модели PSO каждая частица обладает вектором скорости и вектором позиции. При оптимизации *N*-мерной функции такие векторы имеют размерность *N*. Как и в биологической жизни, предполагается, что каждая частица регулирует свои векторы позиции и скорости в соответствии с собственным опытом (когнитивная составляющая), а также по информации, полученной от других членов социума (социальный опыт). Когнитивный опыт частицы понимается как ее знание о лучшей позиции, в которой она сама находилась, а социальное знание частицы – как знание о лучшей позиции, через которую прошла одна из частиц в одной группе с ней. В задаче оптимизации лучшая позиция частицы понимается как позиция, в которой минимизируется значение функции.

Состояние каждой частицы роя характеризуется:

1). Тремя *N*-мерными векторами:

- X – текущая позиция частицы в поисковом пространстве;
- *G* – лучшая позиция, найденная всем роем;
- *v* – скорость движения частицы.

2). Двумя скалярными значениями, описывающими качество решения задачи:

- *P* – фитнес-частицы (значение целевой функции в текущей позиции);
- *G* – фитнес-группы, в которую входит частица (значение целевой функции в лучшей позиции).

Изменение положения частицы задается простым добавлением *v*-вектора к *X*-вектору:

$$X_i = X_i + v_i. \qquad (1.1)$$

Изначально значение вектора скорости генерируется случайным образом в пределах $[-v_{\max}, v_{\max}]$, где $v_{\max}$ – максимально допустимое значение.

Скорость частицы модифицируется по формуле

$$v_i = v_i + c_1 r_1 (P_i - X_i) + c_2 r_2 (G - X_i), \qquad (1.2)$$

где i – номер частицы; v_i – вектор скорости; X_i – вектор позиции частицы; P_i – вектор лучшей позиции, которой достигала частица; G – вектор лучшей позиции, которой достигала группа из I частиц; c_1 , c_2 – константы скорости обучения, управляющие соответственно социальной и познавательной компонентой; r_1 , r_2 – случайные числа в диапазоне [5], которые служат для поддержания разных траекторий частиц при поиске.

Константы скорости обучения контролируют степень важности индивидуального познания и социального знания. Частица одновременно обновляет свою позицию по формуле (1.1) относительно лучшей позиции группы и собственной лучшей позиции, которая была в прошлом.

Если c_2 имеет преобладающее значение, то частица будет обследовать узкую область поискового пространства, а если преобладает c_1, то происходит поиск в большом пространстве.

В формулу (1.2) может вводиться фактор инерции *W*:

$$v_i = W \cdot v_i + c_1 r_1 (P_i - X_i) + c_2 r_2 (G - X_i).$$

Вначале фактор инерции $W = 1$, а потом он постепенно уменьшается во времени по формуле

$$W = W_{max} - (W_{max} - W_{mix}) \cdot n/N,$$

где n – номер текущей итерации; N – заданное число итераций.

При инициализации частицы распределяются случайным образом в поисковом пространстве, их скорость получает случайные значения в допустимом диапазоне. Таким образом, работу алгоритма оптимизации PSO можно описать на основе метода наименьших квадратов. Если ввести обозначения:

$$K = c_1 r_1 (P_i - X_i),$$

$$S = c_2 r_2 (G - X_i),$$

то можно назвать величину K когнитивной компонентой скорости, или ностальгией частицы, поскольку она характеризует стремление частицы вернуться в то место, где ей было хорошо в прошлом. Величина S описывает социальные нормы, которым должен удовлетворять индивид, т. е. движение по направлению к позиции лучшей частицы группы.

Решение считается полученным, если оно удовлетворяет поставленным критериям или истекло время, отведенное на поиск решения.

Большое значение для работы алгоритма имеет выбор топологии группы, в которой участвует частица. Возможны две разные топологии: кольцевая и звездообразная. Соответственно различают два варианта: global best (gbest) и local best (lbest).

В варианте gbest используется звездообразная топология, и частица получает информацию от всех частиц группы. В варианте lbest применяется кольцевая топология, и частица получает информацию только от ближайших соседей. Однако области соседства частиц перекрывают друг друга. Таким образом, gbest можно рассматривать как специальный случай lbest, в котором область соседства расширена на весь рой.

На рис. 1.4 приведен пример движения частицы в двумерном пространстве поиска. С течением времени лучшие позиции частицы и группы становятся одинаковыми.

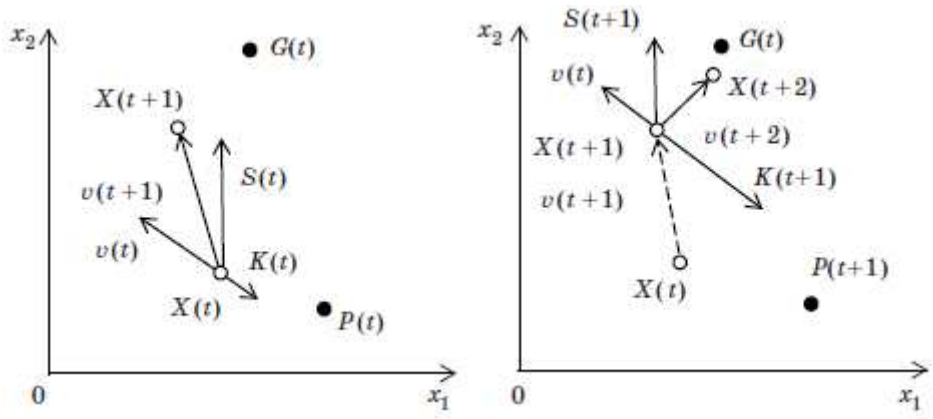

Рис. 1.4. Движение частицы в двумерном пространстве

Алгоритм PSO имеет очевидное сходство с генетическим алгоритмом:

• оба алгоритма являются стохастическими;

• решение ищется на базе популяции индивидуумов;

• начальная популяция генерируется случайным образом;

• для работы алгоритма требуется расчет фитнеса каждого индивидуума;

• области возможного использования алгоритма совпадают.

Но существуют и отличия:

• число настраиваемых параметров алгоритма PSO меньше;

• эволюционные операторы отсутствуют, частицы не «умирают».

Алгоритм PSO, описываемый формулами (1.1) и (1.2), обеспечивает более быструю сходимость, чем генетический алгоритм [6], однако при его использовании может возникать ряд особых проблем, к которым относятся:

• преждевременная сходимость, связанная с уменьшением скорости движения частиц;

• проблемная зависимость алгоритма, которая приводит к сильному влиянию значения констант скорости обучения на решение в конкретной задаче.

Для устранения этих проблем в классический алгоритм PSO вносятся модификации [7,8], в частности, для управления процессом поиска в [8]

предложено использовать нечеткий логический регулятор, работа которого основана на экспертных знаниях в виде лингвистических правил.

1.5. Метод расширяющегося нейронного газа

Расширяющийся нейронный газ (GNG) – это алгоритм, позволяющий осуществлять адаптивную кластеризацию входных данных, то есть не только разделить пространство на кластеры, но и определить необходимое их количество, исходя из особенностей самих данных. Это новый класс вычислительных механизмов. Количество и расположение искусственных нейронов в пространстве признаков не задается заранее, а вычисляется в процессе обучения моделей в соответствии с особенностями входных данных, самостоятельно подстраиваясь под них.

«Расширяющийся нейронный газ – это алгоритм, позволяющий осуществлять адаптивную кластеризацию входных данных, то есть не только разделить пространство на кластеры, но и определить необходимое их количество исходя из особенностей самих данных. Расширяющийся нейронный газ не требует априорной информации о данных, таких как оценка количества кластеров или форма кластеров» [3]. В данной модели не фиксировано соседство узлов, а динамически меняется по мере улучшения кластеризации. Переменными являются не только отношения соседства, но и число нейронов-кластеров.

Начиная всего с двух нейронов, алгоритм последовательно изменяет (по большей части увеличивает) их число, одновременно создавая набор связей между нейронами, наилучшим образом отвечающий распределению входных векторов. У каждого нейрона имеется внутренняя переменная, в которой накапливается «локальная ошибка». Соединения между узлами характеризуются переменной, называемой «возраст» [4].

- В начале работы алгоритма создаются два узла W_s и W_t (здесь и далее, узел – это нейрон) с векторами весов, разрешенными распределением

входных векторов, и нулевыми значениями локальных ошибок. Пространственная интерпретация весов W_s и W_t – это координаты.

- Узлы соединяются связью, которой можно установить возраст. На начальном этапе возраст равен 0.
- Затем на вход нейронной сети подаётся вектор X – координата кластеризуемого множества.
- На следующем этапе находятся два нейрона S и T, ближайших к X (допустим S ближе, чем T), то есть узлы с векторами весов W_s и W_t, такими, что $\|W_s - X\|$ - минимальное, а $\|W_t - X\|$ – второе минимальное значение расстояния среди всех узлов.
- Обновляется локальная ошибка наиболее близкого нейрона – победителя S, к ней добавляется квадрат расстояния между векторами W_s и X.

$$E_s \Rightarrow E_s + \|W_s - X\|^2.$$

Эта процедура приводит к тому, что наиболее часто выигрывающие узлы, то есть те, в окрестности которых попадает наибольшее количество входных сигналов, имеют наибольшее значение ошибки, а значит, именно эти области становятся главными кандидатами на «уплотнение» путём добавления новых узлов.

- Нейрон-победитель S и все его топологические соседи (то есть все нейроны N, имеющие соединение с победителем) смещаются в сторону входного вектора на расстояния, равные долям ε_w и ε_n от полного расстояния.

$$W_s \Rightarrow W_s + \varepsilon_w (W_s - X)$$

$$W_n \Rightarrow W_n + \varepsilon_n (W_n - X)$$

Смещение узлов в сторону входного вектора на данном шаге означает, что победитель стремится «усреднить» своё положение среди входных

сигналов, расположенных в его окрестностях. При этом лучший нейрон немного «подтягивает» в сторону сигнала и своих соседей.

- Увеличить на 1 возраст всех соединений, исходящих от победителя S.
- Удалите соединения, длина которых больше, чем a_{max}. Если это приводит к изолированным нейронам (без исходящих соединений), то удалите такие нейроны.
- Если два лучших нейрона S и T соединены, обнулить возраст их связи. В противном случае создать связь между ними.
- Удалить все соединения, возраст которых превышает максимальный возраст. Если после этого имеются нейроны, не имеющие связей с другими узлами, удалить эти нейроны.
- Если номер текущей итерации кратен λ, и предельный размер сети не достигнут, то создать новый нейрон R по правилам. Со временем после нескольких циклов смещений накапливается информация, на основании которой принимается решение о месте, в котором должен быть добавлен новый нейрон. Этот процесс представляет собой коррекцию переменных ошибок всех нейронов слоя. Это необходимо для того, чтобы сеть «забывала» старые входные векторы и лучше реагировала на новые сигналы. Таким образом, достигается возможность использовать ***расширяющийся нейронный газ*** для адаптации нейросети под нестационарные, а именно, медленно дрейфующие распределения входных сигналов.
- Найти нейрон U с наибольшей локальной ошибкой.
- Среди соседей U найти нейрон V с максимальной ошибкой.
- Создать узел R «посередине» между U и V:

$$W_r = \frac{W_u + W_v}{2}.$$

- Заменить связь между U и V на связи между U и R, R и V.

- Уменьшить ошибки нейронов U и V , установить значение ошибки нейрона R.

$$E_u \Rightarrow E_u * \alpha, \qquad E_v \Rightarrow E_v * \alpha, \qquad E_r \Rightarrow E_u .$$

- Большое значение этой ошибки служит указанием на то, что соответствующий нейрон лежит в области небольшого числа нейронов.
- Каждый раз, когда для случайно выбранного X определяется ближайший к нему нейрон W, локальная ошибка для последнего E_j получает приращение $\|W_j - X\|^2$.
- Если критерий остановки не достигнут, то переходите к шагу 3. В качестве критерия остановки можно использовать количество вставленных нейронов.

Когда деформации в топологии объектов малы и постепенны между последовательными кадрами в последовательности изображений, мы можем использовать информацию предыдущих карт для размещения нейронов без сброса процесса обучения. Используя эту особенность GNG, мы достигаем высокого ускорения для процесса представления.

Например, на рис.1.5 представлены некоторые объекты с цветом закраски в качестве общего признака распознаваемых объектов. В качестве переднего плана объектов использовано белое поле слева и закрашенное поле в качестве фона справа.

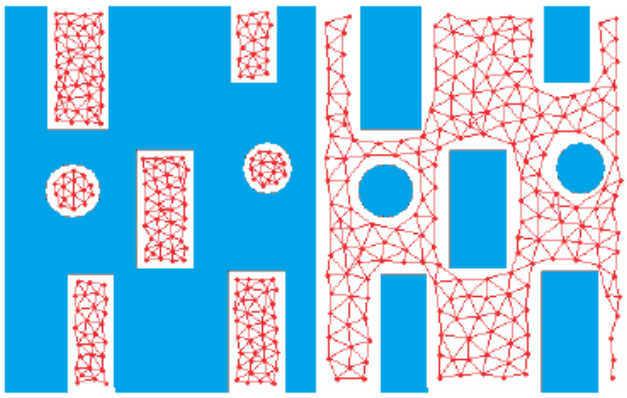

Рис. 1.5. Представление объектов со схожими визуальными свойствами, как переднего плана, так и заднего плана

Чтобы проиллюстрировать возможности GNG для представления топологических деформаций в объектах, мы адаптировали карты к форме объекта, которая меняет свою топологию с компактного квадрата на четыре маленьких квадрата в четыре этапа (кадры), получая графики, которые представляют топологию формы объекта вдоль последовательности изображений, но без сброса процесса обучения для любого изображения.

На рис.1.6 показана исходная последовательность изображений, используемых в качестве входного пространства для самоорганизующейся карты, где из однородного квадрата на первом изображении (слева) создаются четыре маленьких квадрата на последнем изображении (справа). Результаты работы алгоритма, решающего задачи адаптации GNG, устанавливающей визуальные свойства изображаемых объектов.

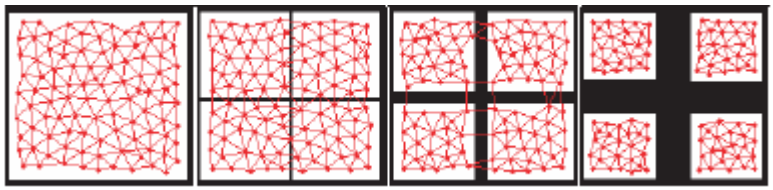

Рис.1.6. Результаты адаптации GNG к изменениям входного пространства

Из первой карты (слева), новые карты получаются на основе предыдущей, без перезагрузки процесса обучения. Эта особенность GNG позволяет ускорить представление последовательности изображений.

Как видно из последовательности изображений, карта способна разделить нейроны на четыре группы, представляющие различные квадраты в исходных изображениях, когда расстояние между ними больше, чем средняя длина ребер, соединяющих нейроны.

Рис.1.7 представляет последовательность деформаций от малого круга до эллипса и, наконец, до квадрата, используемого в качестве входного пространства для GNG. Показаны результаты адаптации карты без сброса алгоритма обучения между кадрами.

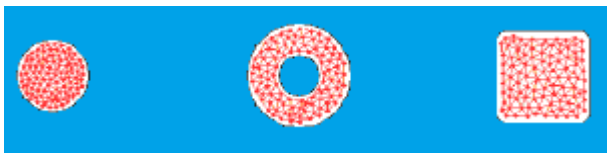

Рис.1.7. Деформация объекта с адаптацией GNG

Параметры, используемые для моделирования: N=100, λ = 1000 для первой карты и 10000-20000 для последующих карт, E_w = 0,1, E_n = 0,001, α=0,5, β=0,95, a_{max} = 250.

Вычислительные затраты на представление последовательности деформаций очень низки, по сравнению с методами, основанными на адаптации новой карты для любого кадра последовательности, так как наш метод не сбрасывает алгоритм для новых кадров. Эта функция предоставляет методу возможности в режиме реального времени.

В этой статье мы продемонстрировали способность GNG к представлению двумерных объектов. Устанавливая подходящую функцию преобразования, модель способна адаптировать свою топологию к форме объекта. Затем получается простое, но очень богатое представление объектов.

Модель, благодаря своему собственному процессу адаптации, способна разделить себя таким образом, что она может характеризовать различные фрагменты из объекта или различные объекты в одном и том же изображении. Кроме того, GNG может представлять деформации в объектах топологии, представляя их вдоль последовательности изображений без сброса процесса обучения. Эта функция ускоряет процесс представления и отслеживания объектов.

1.6. EM-алгоритм для разделения смеси случайных величин

EM-алгоритм (англ. expectation-maximization = ожидание-максимизация) – алгоритм, используемый в математической статистике для нахождения оценок максимального правдоподобия параметров вероятностных моделей, в случае, когда модель зависит от некоторых скрытых переменных.

EM-алгоритм может использоваться для разделения смеси случайных величин, которые естественно интерпретировать как элементы разных множеств, то есть решить задачу вероятностного распознавания элементов эти множества.

Каждая итерация алгоритма состоит из двух шагов.

- На E-шаге (expectation) вычисляется ожидаемое значение функции правдоподобия, при этом скрытые переменные рассматриваются как наблюдаемые.
- На M-шаге (maximization) вычисляется оценка максимального правдоподобия, таким образом увеличивается ожидаемое правдоподобие, вычисляемое на E-шаге.
- Затем это значение используется для E-шага на следующей итерации. Алгоритм выполняется до сходимости.

Как правило, EM-алгоритм применяется для решения задач двух типов.

- К первому типу можно отнести задачи, связанные с анализом действительно неполных данных, когда некоторые статистические данные отсутствуют в силу каких-либо причин.
- Ко второму типу задач можно отнести те задачи, в которых функция правдоподобия имеет вид, не допускающий удобных аналитических методов исследования, но допускающий серьезные упрощения, если в задачу ввести дополнительные «ненаблюдаемые» (скрытые, латентные)

переменные. Примерами прикладных задач второго типа являются задачи распознавания образов, реконструкции изображений. Математическую суть данных задач составляют задачи кластерного анализа, классификации и разделения смесей вероятностных распределений.

Пусть плотность распределения на множестве X имеет вид смеси k распределений:

$$p(x) = \sum_{j=1}^{k} w_j p_j(x), \qquad \sum_{j=1}^{k} w_j = 1, \quad w_j \geq 0,$$

где $p_j(x)$ – функция правдоподобия j-й компоненты смеси, w_j – ее априорная вероятность.

Пусть функции правдоподобия принадлежат параметрическому семейству распределений $\varphi(x,\theta)$ и отличаются только значениями параметра:

$$p_j(x) = \varphi(x,\theta_j).$$

Задача разделения смеси заключается в том, чтобы, имея выборку X^m случайных и независимых наблюдений из смеси $p(x)$, зная число k и функцию φ, оценить вектор параметров:

$$\Theta = (w_1,...,w_k,\theta_1,...,\theta_k).$$

Идея алгоритма заключается в следующем. Искусственно вводится *вспомогательный вектор скрытых переменных* G, обладающий двумя важными свойствами.

- С одной стороны, он может быть вычислен, если известны значения вектора параметров Θ.
- С другой стороны, поиск максимума правдоподобия сильно упрощается, если известны значения скрытых переменных.

EM-алгоритм состоит из итерационного повторения двух шагов.

- На **E-шаге** вычисляется ожидаемое значение (expectation) вектора скрытых переменных G по текущему приближению вектора параметров Θ.

  Обозначим через $p(x,\theta_j)$ плотность вероятности того, что объект x получен из j-й компоненты смеси. По формуле условной вероятности:

$$p(x,\theta_j) = p(x)P(\theta_j \mid x) = w_j p_j(x).$$

  Введём обозначение:

$$g_{ij} \equiv P(\theta_j \mid x_i).$$

  Это неизвестная апостериорная вероятность того, что обучающий объект x_i получен из j-й компоненты смеси. Возьмём эти величины в качестве скрытых переменных.

$$\sum_{j=1}^{k} g_{ij} = 1,$$

  для любого $i=1,\ldots,m$, так как имеет смысл полной вероятности принадлежать объекту x_i одной из k компонент смеси. Из формулы Байеса для всех i,j следует соотношение:

$$g_{ij} = \frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)}.$$

- На **M-шаге** решается задача максимизации правдоподобия (maximization) и находится следующее приближение вектора Θ по текущим значениям векторов G и Θ.

  Будем максимизировать логарифм полного правдоподобия:

$$Q(\Theta) = \ln \prod_{i=1}^{m} p(x_i) = \sum_{i=1}^{m} \ln \sum_{j=1}^{k} w_j p_j(x_i) \to \max_{\Theta}.$$

Решая оптимизационную задачу Лагранжа с ограничением на сумму коэффициентов w_j, для $j=1,\ldots,k$ находим:

$$w_j = \frac{1}{m}\sum_{i=1}^{m} g_{ij} ,$$

$$\theta_j = \arg\max \sum_{i=1}^{m} g_{ij} \ln \varphi(x_i, \theta) .$$

Таким образом, М-шаг сводится к вычислению весов компонент w_j, как средних арифметических и оцениванию параметров компонент θ_j путём решения k независимых оптимизационных задач.

Отметим, что разделение переменных оказалось возможным благодаря удачному введению скрытых переменных.

Смесь нормальных распределений

Далее часто будет встречаться смесь нормальных распределений, поэтому выпишем для нее результаты Е- и М-шага алгоритма: $\theta = (w_1,...,w_k;\mu_1,...,\mu_k;\sigma_1,...,\sigma_k)$ – вектор параметров,

$p_j(x) = N(x;\mu_j,\sigma_j) = \frac{1}{\sqrt{2\pi}\sigma_j}\exp\left(-\frac{(x-\mu_j)^2}{2\sigma_j^2}\right)$ – плотность распределения.

- Е-шаг

$$g_{ij} = \frac{w_j N(x_i;\mu_j,\sigma_j)}{\sum_{s=1}^{k} w_s N(x_i;\mu_j,\sigma_j)} .$$

- М-шаг. Для $j=1,...,k$ вычисляем:

$$w_j = \frac{1}{m}\sum_{i=1}^{m} g_{ij} ,$$

$$\mu_j = \frac{1}{m w_j}\sum_{i=1}^{m} g_{ij} x_i ,$$

$$\sigma_j^2 = \frac{1}{m w_j}\sum_{i=1}^{m} g_{ij} (x_i - \mu_j)^2 .$$

Недостатки основного алгоритма:

1. Основной алгоритм *неустойчив по начальным данным* (то есть тем, которые инициализируют вектор параметров θ на первой итерации), так как он находит локальный экстремум, значение которого может оказаться гораздо ниже, чем глобальный максимум. В зависимости от выбора начального приближения алгоритм может сходиться к разным точкам. Также может сильно варьироваться скорость сходимости.
2. Основной алгоритм *не позволяет определять количество k компонент смеси*. Эта величина является структурным параметром алгоритма.

В связи с этим разработаны модификации основного алгоритма, направленные на устранение данных недостатков.

Медианные модификации EM-алгоритма

Для противодействия неустойчивости алгоритма по начальным данным можно использовать медианные модификации. Их смысл заключается в том, что наиболее «неустойчивые» этапы выполнения EM-алгоритма заменяются более устойчивыми. В частности, на М-шаге моментные оценки максимального правдоподобия заменяются более устойчивыми (робастными) оценками медианного типа.

Далее будут приведены конечные результаты для смеси нормальных распределений. Математическое обоснование данного метода можно посмотреть в [2].

Первая медианная модификация

Введем для $i=1,\ldots,m$, $j=1,\ldots,k$ следующие величины:

$$\pi_{ij} = \frac{g_{ij}}{\sum_{i=1}^{m} g_{ij}},$$

имеющие смысл некой вероятности. Введем также «фиктивные» случайные величины ξ_j, принимающие значения x_i с вероятностями η_{ij}. Переупорядочим значения $x_1,\ldots,x_m$ случайной величины ξ_j по неубыванию,

одновременно переставляя соответствующие данным значениям вероятности. Пусть $\eta_{(ij)}$ – вероятность, соответствующая значению $x_{(i)}$.

Положим:

$$I_j = \min\left(i : \pi_{(1j)} + \pi_{(2j)} + ... + \pi_{(ij)} \geq \frac{1}{2} \right).$$

Тогда $\mu_j = med\xi_j = x_{I,j}$.

Пусть $m_j = med\left|\xi_j - \mu_j\right| = x_{I,j}$.

Тогда $\sigma_j = 1.4826 m_j$.

Вторая медианная модификация

Математическое ожидание μ_j оценивается так же, как и в первой медианной модификации. Рассмотрим дисперсию σ_j.

Обозначим

$s_j = \sum \pi_{i,j} \left| x_i - \mu_j \right|$, тогда σ_j=1.2533 s_j.

Стохастический EM-алгоритм

Классический EM-алгоритм относится к так называемым «жадным» алгоритмам, то есть он бросается на первый попавшийся локальный максимум, который, как уже упоминалось, может сильно отличаться от глобального. Одним из способов борьбы является как бы случайное, но целенаправленное «встряхивание» выборки на каждой итерации. Такая рандомизация «выбивает» оптимизационный процесс из локальных максимумов, так что:

- SEM работает относительно быстро, и его результаты практически не зависят от начального приближения.
- Как правило, SEM находит экстремум $Q(\Theta)$, близкий к глобальному.

Пусть вся выборка X^m разбита на кластеры K_j, $j=1,...,k$: каждый элемент x_i относится к единственному кластеру K_j, то есть утверждается, что данный элемент взят из j компоненты смеси.

- **S-шаг**

На первом этапе SEM-алгоритма производится так называемое *стохастическое моделирование*. Для каждого $i=1,\ldots,m$ генерируется вектор y_i =($y_{i1},\ldots, y_{ik}$) как реализация случайного вектора из полиномиального распределения с параметрами 1 и y_{ij}, где y_{ij} – это вероятность того, что величина y_{ij} равна 1. По векторам y_i определяется разбиение выборки X^m на кластеры K_j и соответствующие численности кластеров $v_1,\ldots, v_k$.

- **E-шаг**: остается таким же, как в предыдущем разделе.
- **M-шаг**: Пересчитываются веса:

$$w_j = \frac{v_j}{m},$$

и вместо максимизации взвешенного правдоподобия:

$$\theta_j = \arg\max_{\theta} \sum_{i-1}^{m} g_{ij} \ln \varphi(x_i, \theta),$$

решается задача обычного невзвешенного правдоподобия:

$$\theta_j = \arg\max_{\theta} \sum_{x_i \in K_j} \ln \varphi(x_i, \theta).$$

Для SEM-алгоритма также существуют медианные модификации.

Классификационный EM-алгоритм

Этот алгоритм совпадает с SEM-алгоритмом, за исключением того, что вместо S-шага используется детерминированное правило, эквивалентное классификации по принципу максимума апостериорной вероятности, то есть x_i приписывается тому кластеру, номер которого совпадает с номером наибольшего из чисел $g_{i1},\ldots, g_{ik}$. Формулы для оценок параметров аналогичны SEM-алгоритму.

В тех случаях, когда максимизация функционала $Q(\Theta)$, имеющего смысл полного правдоподобия, по каким-либо причинам затруднена, применяется подход, в котором достаточно лишь сместиться в сторону

максимума значения функционала, сделав одну или несколько итераций на М-шаге. Этот алгоритм также обладает неплохой сходимостью.

EM-алгоритм с добавлением компонент

Алгоритм позволяет решить две проблемы сразу: проблему выбора числа компонент и проблему выбора начального приближения. Идея заключается в следующем. Имея некоторый набор компонент, можно выделить объекты x_i, которые хуже всего описываются смесью – это объекты с наименьшими значениями правдоподобия $p(x_i)$. По этим объектам строится ещё одна компонента. Затем она добавляется в смесь, и запускаются EM-итерации, чтобы новая компонента и старые компоненты «притерлись друг к другу». Так продолжается до тех пор, пока все объекты не окажутся покрытыми компонентами.

SEM-алгоритм с удалением компонент

Алгоритм решает проблему выбора компонент и «застоя» в локальном максимуме. До начала работы алгоритма фиксируются два числа:

- K – максимально возможное число компонент, которое определяется соображениями практической интерпретируемости полученного результата.
- v_0 – минимально допустимое число наблюдений в одном кластере, которое выбирают исходя из минимально допустимой значимости компонент. Если нецелесообразно считать значимыми компоненты, веса которых меньше некоторого заданного (малого) значения α, то $v_0=\alpha m$, где m – объем выборки.

Начальное число компонент смеси равно K. Далее это число может только уменьшаться: в результате стохастического моделирования на S-шаге SEM-алгоритма могут возникнуть кластеры, для которых $v_j \le v_0$. Тогда данный кластер аннулируется, то есть

1. Переопределяются апостериорные вероятности y_{ij}.

2. Элементы выборки, попавшие в аннулированные классы, перераспределяются по «достаточно представительным» кластерам.
3. Переопределяется число компонент смеси.

Формулы для оценок параметров аналогичны SEM-алгоритму с фиксированным числом компонент.

Все вышеперечисленные схемы не являются жесткими. Возможно изменение некоторых их частей или комбинирование нескольких схем. Существует множество методов, о которых не было рассказано в данной статье. EM-алгоритм – это довольно мощный и часто встречающийся инструмент для решения прикладных задач и использование той или иной его модификации зависит от конкретной задачи и поставленных целей.

2. Модели самообучаемых нейронных сетей

Процесс самообучения подразумевает выявление нейронной сетью из множества входных данных существенных признаков, позволяющих нейронной сети самостоятельно осуществлять классификацию входных образов.

Каждый нейрон создает локальное правило, определяемое его начальными значениями и состоянием соседних нейронов. Правило формируется в процессе процедуры самообучения для множества входных образов.

Алгоритмы анализа главных компонент позволяют уменьшить размерность данных в статистических системах распознавания образов и обработки сигналов. Анализ главных компонент лежит в основе принципа самообучения Хебба.

Алгоритм процесса самообучения включает два цикла:

- внешний цикл самообучения, повторяющий алгоритм самообучения до достижения выполнения критерия завершения обучения;
- внутренний цикл по множеству объектов самообучения, который должен удовлетворять критерию достаточной представительности классов распознаваемых образов.

В процессе выполнения алгоритма самообучения вычисляется выходной сигнал НС и корректируются по заданному правилу весовые коэффициенты НС.

Правило самообучения Хебба состоит в том, что увеличиваются те весовые коэффициенты, которые связывают большие входные сигналы с нейронами, у которых большой выходной сигнал. Процессу самообучения свойственны следующие своства:

- самоусиление весов,
- ограниченность ресурсов и конкуренция,

- кооперация весов,
- избыточность информации.

В системах зрения нейронные сети организованы в двумерные слои. Входные сигналы нейронов пространственно ограничены полями восприятия. Самообучение охватывает все слои НС.

2.1. Анализ главных компонент

Обобщающей характеристикой многомерных входных данных X, считается матрица корреляции R. Собственные вектора корреляционной матрицы R являются ортогональными векторами, задающими ортогональную систему координат. Собственные значения корреляционной матрицы R являются дисперсиями значений проекций на соответствующие направления собственных векторов.

Анализ главных компонент имеет своей задачей вычисление собственных векторов, на направлениях которых получаются наибольшие дисперсии проекций данных. Анализ минимальных (минорных, второстепенных) компонент входных данных имеет своей задачей вычисление собственных векторов, на направлениях которых обеспечиваются наименьшие дисперсии проекций данных.

Чем больше дисперсия проекций на ось собственного вектора, тем выше его уровень значимости. Множество собственных векторов и собственных чисел матрицы, имеющих высокий уровень значимости, образует ***множество главных компонент***. Оценка значимости компонент реализуется в соответствии с выбранным критерием (правилом) значимости.

Правило Кайзера (*Kaiser's rule)* оценки значимости компонент использует в качестве порога среднее значение собственных чисел матрицы:

$$\lambda_i > \frac{1}{m} tr(R),$$

где λ_i – собственные значения матрицы R. Все собственные значения, удовлетворяющие приведенному неравенству считаются главными компонентами.

2.2. Главные компоненты множества данных

Анализ главных компонентов (*principal component analysis)* (преобразование Карунена-Лоева) и правило обучения Хебба имеют общий математический базис.

Анализируется случайный вектор X размерности m, имеющий нулевое математическое ожидание: $E(X) = 0$, где E – оператор вычисления математического ожидания. Кроме того, рассматривается единичный вектор q размерности m, на который проецируется вектор X: $A = X^T q = q^T X$.

Математическое ожидание и дисперсия величины проекции A равны:

$$E(A) = q^T E(X) = 0,$$

$$\sigma^2 = E(A^2) = E\left[\left(q^T X\right)\left(X^T q\right)\right] = q^T E\left[X \cdot X^T\right] q = q^T R q, \quad (2.1)$$

где $R = E(X \cdot X^T)$ – матрица $m \times m$ корреляции случайного вектора X.

Дисперсия σ^2 проекции множества A является функцией единичного вектора q:

$$\psi(q) = \sigma^2 = q^T R q.$$

Функция $\psi(q)$ имеет экстремальные и стационарные значения, которые зависят от матрицы корреляции R. В точке экстремума для малого возмущения δq выполняются равносильные соотношения:

$$\frac{d\psi(q)}{dq} = 0,\ d\psi(q) = 0,\ \psi(q+\delta q) = \psi(q). \quad (2.2)$$

Из определения функции $\psi(q)$ от единичного вектора q следует соотношение:

$$\psi(q+\delta q) = (q+\delta q)^T R(q+\delta q) = q^T R \cdot q + 2(\delta q)^T R \cdot q + (\delta q)^T R \cdot \delta q.$$

Игнорирование слагаемого второго прядка малости позволяет упростить формулу:

$$\psi(q+\delta q) = q^T R \cdot q + 2(\delta q)^T R \cdot q.$$

Использование формулы (2.2) позволяет получить:

$$2(\delta q)^T R \cdot q = 0$$

2.3. Алгоритм вычисления наибольшего главного значения одним нейроном Хебба

Исследование решений системы дифференциальных уравнений при $t \to \infty$ дает следующий результат: единственным ненулевым решением является наибольшее собственное число и соответствующий собственный вектор матрицы корреляции R [9]. Правило обучения Хебба для одного нейрона выражается следующей формулой:

$$w_i(n+1) = w_i(n) + \eta \cdot y(n) \cdot x_i(n).$$

Приведенная зависимость имеет существенный недостаток: синаптические коэффициенты в процессе обучения быстро растут до значений, выходящих за границы величин допустимых вычислений. Для решения данной проблемы используется процедура нормализации синаптических коэффициентов [1,9]:

$$w_i(n+1) = \frac{w_i(n) + \eta \cdot y(n) \cdot x_i(n)}{\left(\sum_{i=1}^{m} \left(w_i(n) + \eta \cdot y(n) \cdot x_i(n) \right)^2 \right)^{1/2}}.$$

Приведенная формула требует выполнения значительно большего объема вычислений, по сравнению с исходной формулой. Сократить объем вычислений позволяет применение разложения в ряд нормализующей формулы с использованием фактора малости параметра η и пренебрежением слагаемыми второго порядка $O^2(\eta)$ [9]:

$$w_i(n+1) = w_i(n) + \eta \cdot y(n) \cdot \left[x_i(n) - y(n) \cdot w_i(n)\right].$$

Аналитическое следование свойств алгоритма обучения Хебба базируется на рассмотрении дифференциального уравнения:

$$\frac{d}{dt} w(t) = y(t) \cdot \left[x(t) - y(t) \cdot w(t)\right],$$

которое соответствует рекуррентному приведенному соотношению, с учетом того, что параметр η эквивалентен дифференциалу времени dt. Применение оператора вычисления математического ожидания позволяет получить нелинейное дифференциальное уравнение для весовых коэффициентов [1,9,10]:

$$\frac{d}{dt} w(t) = R \cdot w(t) \cdot w(t) - [w^T(t) \cdot R \cdot w(t)] \cdot w(t).$$

Вектор весовых коэффициентов $w(t)$ можно разложить в ортогональном базисе собственных векторов матрицы корреляции R следующим образом:

$$w(t) = \sum_{k=1}^{m} \theta_k(t) \cdot q_k .$$

Подстановка разложения в дифференциальное уравнение позволяет получить систему обыкновенных дифференциальных уравнений:

$$\frac{d\theta_k(t)}{dt} = \lambda_k \theta_k(t) - \theta_k(t) \sum_{u=1}^{1} \lambda_u \theta_u^2(t)$$

Полученный результат означает, что нейрон, обученный по правилу Хебба, формирует последовательность выходных сигналов, дисперсия которых равна наибольшему собственному числу корреляционной

матрицы R для множества входных образов. Кроме того, вектор своих весовых коэффициентов этого нейрона совпадает с собственным вектором корреляционной матрицы R, который соответствует наибольшему собственному числу этой матрицы.

2.4. Фильтр Хебба главных компонент

Обобщенный алгоритм обучения Хебба (GHA) является расширением алгоритма вычисления наибольшего главного значения одним нейроном Хебба на случай одного полносвязанного слоя нейронов.

Один слой линейных нейронов, обучаемых по правилу Хебба, позволяет извлекать главные компоненты из входного сигнала.

Предполагается, что количество выходов меньше количества входов: $l < m$. Выходной сигнал $y_j(n)$ нейрона j в момент времени n является откликом на множество входных воздействий $\{x_i(n) \mid i = 1,2,...,m\}$ и вычисляется по следующей формуле:

$$y_j(n) = \sum_{i=1}^{m} w_{ji}(n) x_i(n).$$

Синаптические веса настраиваются в соответствии с обобщенной формой правила Хебба:

$$\Delta w_{ji}(n) = \eta \left[y_j(n) \cdot x_i(n) - y_j(n) \sum_{k=1}^{j} w_{ki}(n) y_k(n) \right]$$

Получение вектора собственных значений корреляционной матрицы входных образов позволяет определить количество классов, на которые можно разделить множество входных образов.

3. Модели и методы процесса кластеризации данных

Кластеризация – это одна из форм автоматической обработки данных об объектах, позволяющая классифицировать эти объекты, то есть разбить их на непересекающиеся кластеры. Исходные данные о каждом объекте имеют одинаковый формат, обычно соответствующий числовому вектору, то есть множеству объектов сопоставляется множество векторов $\{x_i\}_{i=1..n}$.

Методы кластеризации используют следующие обозначения:

- IID (independent and identically distributed) – независимые и идентичные распределения,
- $\boldsymbol{x_i}$, – набор данных *для* $i=1, ...,n$, которые рассматриваются как результаты IID переменной $\boldsymbol{X}$.

Цель кластеризации состоит в том, чтобы найти класс множеств $C=\{C_1,. . . ,C_K\}$ такой, что каждый $\boldsymbol{x_i}$ находится в одном множестве C_k, и объединение множеств C_k является целым набором данных $x^n = (x_1, ...,x_n)$.

Ключевой момент интерпретации заключается в том, что элементы внутри C_k гораздо больше похожи друг на друга, чем на любой элемент из другого C_k . Естественное сравнение, если возможно, происходит между известной кластеризацией, которая по существу является классификационными данными, и любой кластеризацией, которую выводит процедура кластеризации.

Кластеризация представляет собой набор процедур, используемых для описания методов группировки немаркированных данных в подмножества, которые, как полагают, отражают базовую структуру генератора данных [11]. Кластеризация может рассматриваться в качестве интеллектуальной первичной обработки данных.

Если однородных данных много, то можно использовать вероятностные подходы к решению задачи кластеризации. Множество данных, в этом случае, можно рассматривать как представительную

выборку сложного распределения, являющего смесью некоторого числа более простых по структуре распределений. Каждое отдельное распределение представляет класс, и идентификация этого распределения соответствует решению задачи выделения из всего множества данных одного из классов.

Методы кластеризации многочисленны и разнообразны, частично потому, что кластеризация охватывает очень много областей применения, а частично потому, что кластеризация является своего рода предварительной обработкой, часто необходимой перед применением любого метода вывода на основе модели.

Виды кластеризации:

- Разбиение на кластеры алгоритмами формирования центроид, создания базовой модели, построения диаграмм, спектрального анализа.
- Иерархическая кластеризация, построенная на совместном использовании алгоритмов разделения и алгоритмов агломерации.
- Байсовая кластеризация на основе базовых решений или непараметрических алгоритмов.

Методы разбиения на кластеры (методы секционирования), обычно требуют, чтобы число кластеров K и начальная кластеризация были указаны в качестве входных данных для процедуры, которая затем пытается улучшить начальное назначение точек данных, то есть скорректировать их принадлежность к том или иному кластеру.

Наиболее распространенным методом кластеризации, вероятно, является ***метод K-средних (K-means)***, который является центроидным методом, изобретенным MacQueen (1967). Он основывается на присвоении точек тому центроиду, который ближе всего к ним, вычислении центроида и повторении процедуры, надеясь, что кластеризация сходится. Метод K-means начинается с первоначального назначения кластеризации, которое он уточняет. Можно утверждать, что несходство используется только как

расстояние по оси, поэтому начальная кластеризация важна для этой процедуры.

Идея иерархического метода заключается в том, что он генерирует ***вложенную*** последовательность кластеризации. Обычно нужно ***выбрать пороговое значение***, указывающее, как далеко в процедуре идти, чтобы найти лучшую кластеризацию в последовательности.

Вложенность может уменьшаться или увеличиваться. Если обработка начинается с рассмотрения каждой точки данных как одноэлементного кластера, то вложенность уменьшается. Такие процедуры называются агломеративными (собирательными), поскольку кластеры объединяются. Если начинать кластеризацию с одного кластера, охватывающего все данные, то используют разделяющие процедуры, решающие задачу построения иерархического дерева.

Иерархические методы также обычно требуют введения несходства, которое является мерой расстояния на отдельных точках данных.

Иерархическая агломеративная кластеризация базируется на ключевой процедуре, которая называется ***алгоритмом E-M***.

Байесовские методы кластеризации отличаются от первых двух видов кластеризации тем, что они пытаются создать апостериорное распределение по коллекции всех разделов данных, с указанием или без указания числа кластеров. Этот метод кластеризации позволяет осуществить оптимальную кластеризацию.

Типичными проблемами, возникающими в процессе формирования кластеров, являются: слипание кластеров и рассечение кластера. Нечёткие методы кластеризации допускают частичное членство: точка может принадлежать на 70% к одному кластеру и на 30% к другому. ***Нечеткая кластеризация*** возникает естественным образом при оценке устойчивости кластеризации.

3.1. Кластеризация данных методом K-means

Центроид кластера (centroid) можно рассматривать как простой тип, который представляет кластер, независимо от того, существует ли этот объект на самом деле или это просто математическая конструкция. Центроид – центр тяжести кластера.

Медоид (k-medoid) – объект, принадлежащий набору данных или кластеру, различие которого с другими объектами в наборе данных или кластере минимально.

Учитывая начальную кластеризацию, центроидные методы находят центроиды кластеров, переназначают точки данных новым кластерам, определенным близостью к центроидам, и затем повторяют процедуру. Сходство между двумя кластерами – это сходство между их соответствующими центроидами. Таким образом, ключевыми вариантами выбора, помимо начальной кластеризации, являются расстояние, центроиды и количество итераций процедуры. Большинство вариантов алгоритмов кластеризации ***K-средних*** допускают наглядную геометрическую интерпретацию.

Метод кластеризации K-средних заключается в следующем:

- Выбирается число кластеров K и делается первоначальное пространственное распределение центров кластеров.
- Начинается процедура обхода этих K центров, при котором каждый центр поглощает близлежащие точки на основе расстояния. Часто расстояния определяются с помощью ковариационной матрицы для определения нормы.
- Затем, в зависимости от поглощений, корректируются положения центров кластеров.
- Затем процедура повторяется: новым центрам разрешается поглощать близлежащие точки на основе нормы, корректируется положение новых центров и так далее.

Обычно кластеризация реализуется с помощью расстояния Махаланобиса между точкой x_i и текущими K центрами:

$$d(x_i, \bar{x}_k) = \left[(x_i - \bar{x}_k)' S^{-1} (x_i - \bar{x}_k)\right]^{1/2},$$

где x_k-центр текущего кластера k, S - матрица ковариации внутри кластера, определяемая как $S = (s_{j,l})$ для $j, k = 1, ..., p$ и вычисляемая по формуле:

$$s_{j,l} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{i,j} - \bar{x}_j)(x_{i,l} - \bar{x}_l)$$

С помощью расстояния Махаланобиса можно определять сходство неизвестной и известной выборки. Оно отличается от расстояния Евклида тем, что учитывает корреляции между переменными и инвариантно к масштабу.

Ускорить процесс вычисления решения задачи кластеризации методом K-средних, можно используя объединенную внутри кластера ковариационную матрицу $W = (w_{j,l})$, где для $j, l = 1, ..., p$,

$$w_{j,l} = \frac{1}{K} \sum_{k=1}^{K} \sum_{i=1}^{n_k} d_{i,k} (x_{i,j} - \bar{x}_{k,j})(x_{i,l} - \bar{x}_{k,l})$$

где n_k - число точек в кластере k, $\bar{x}_{k,l}$ - среднее значение l переменной в кластере k, а $d_{i,k} = 1$, если наблюдение i находится в кластере k и 0 в противном случае. На практике кластеризация K-средних часто является единственной процедурой кластеризации, которая является вычислительно возможной для больших p и малых n наборов данных.

Процедура K-means работает, если центры кластера меняются от итерации к итерации и сходятся к правильным центрам кластера. Если центры перемещаются слишком много или слишком много итераций, кластеры могут быть нестабильными. Фактически, нет гарантии, что существует уникальное решение для кластеризации K-means.

Идеальным решением K-средних является $C_{K-means} = \left(C_1^*, ..., C_{Kopt}^*\right)$, которое удовлетворяет соотношению:

$$C_{K-means} = \arg\left(\min_{K} \min_{m_1,\ldots,m_K} \sum_{k=1}^{K} \sum_{i:C(i)=k} \|x_i - m_k\|^2 \right),$$

где *C(i)* - функция принадлежности кластеризации с центрами *m1, ...,m_K* по Евклидову расстоянию.

Оптимизация по числу кластеров выполняется, в принципе, для каждого возможного K, чтобы найти минимизирующее значение. Однако на практике можно найти значение только для нескольких значений *K*.

Полная изменчивость кластеризации с функцией принадлежности C при несходстве d имеет вид:

$$T = \sum_{i=1}^{n} \sum_{i'=1}^{n} d(x_i, x_{i'}) = \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')\neq k} d(x_i, x_{i'}) + \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} d(x_i, x_{i'}) = .$$
$$= B(C) + W(C)$$

Таким образом, полная изменчивость *T* складывается из суммы изменчивости *B(C)* между кластерами и изменчивости *W(C)* внутри кластеров. Внутрикластерная изменчивость для данного *K* и набора центров m_k равна:

$$W(C) = \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} \|x_i - x_{i'}\|^2 = 2 \sum_{k=1}^{K} \sum_{C(i)=k} \|x_i - \bar{x}_k\|^2 .$$

Минимизация внутрикластерной изменчивости *W(C)* может потребоватьменьше вычислений и быстрее привести к искомой кластеризации.

Алгоритм K-medoid, как и K-means, пытается минимизировать критерий квадратичной ошибки, но вместо того, чтобы принимать среднее значение, ***в качестве центра кластера алгоритм выбирает одну из точек данных***. Медоид кластера – это элемент кластера, средняя непохожесть которого на все объекты кластера минимальна. Это прототипическая точка данных кластера, поскольку она расположена в самом центре. Опять же, K и начальная кластеризация должны быть даны, и основная процедура аналогична процедуре K-средних или K-медиан, но

итеративно заменяет каждый из медоидов одной из немедоидных точек для поиска меньшей ошибки. То есть, после выбора K точек medoid, каждая точка данных связана со своими ближайшими medoid. Затем немедоидная точка данных выбирается случайным образом и используется для замены одного из медоидов, чтобы увидеть, уменьшается ли ошибка. Если да, то эти два пункта взаимозаменяемы. Если нет, проверяется другая немедоидная точка. Это повторяется, пока нет никаких изменений в medoid. Очевидно, что это не очень хорошо масштабируется для больших наборов данных; однако он более устойчив к шуму и выбросам, чем K-means.

3.2. Иерархическая кластеризация данных

3.2.1. Агломеративная иерархическая кластеризация

Иерархическая агломеративная кластеризация объединяет наборы в соответствии с фиксированными правилами. Основным шаблоном алгоритма иерархической агломеративной кластеризации является следующая последовательность действий:

- Начните с выборки x_i, $i = 1, ..., n$, рассматриваемой как n одиночных кластеров. Задайте меру несходства d, определенную для всех пар разъединенных, непересекающихся подмножеств выборки. При желании можно выбрать фиксированное количество кластеров для формирования или, в противном случае, процедуру можно разрешить повторять $n-1$ раз.
- На первом шаге соедините два одиночных элемента x_i и x_j, которые имеют минимальное несходство $d(x_i, x_j)$ среди всех возможных пар точек данных.
- На втором шаге, имеется $n-1$ кластеров $C_{1,1}, ..., C_{1,n-1}$. Следует найти решение

$$\left(j^*, j'^*\right) = \arg\min d\left(C_{1,j}, C_{1,j'}\right)$$

и объединить найденные кластеры $C_{1,\,j*}$ и $C_{1,\,j'*}$. Теперь осталось *N-2* кластера, $C_{2,\,1}$, ..., $C_{2,\,n-2}$.

- Далее следует продолжать, пока *n* точек не будут объединены в нужное число кластеров или в один кластер размером *n*.

Кроме того, можно выбрать критерий ошибки. Найдите ошибки для кластеризации с различным числом кластеров. Выберите наибольшее число кластеров, для которых падение ошибки от разрешения еще одного кластера невелико.

На практике, чем больше число кластеров K, тем меньше информации приходится на дополнительный кластер, поскольку в данных содержится конечное количество информации. Интуитивно, когда K мало, существует большое количество информации на кластер, потому что информация в данных суммируется небольшим количеством центров кластера. Однако по мере увеличения K информация в данных, точно собранных с помощью дополнительных кластеров, уменьшается. Это означает, что существует K, за пределами которого разрешение еще одного кластера уменьшает ошибку на такую малую величину, что дополнительный кластер не стоит включать. В этих случаях график K по сравнению с критерием ошибки обычно имеет вмятину или угол, указывающий на внезапное выпадение информации на дополнительный кластер, указывающий на точку уменьшения отдачи. Это называется коленом или локтем кривой и обычно можно найти, просто взглянув на график. Тем не менее, есть неоднозначные случаи, и может быть трудно решить, когда прекратить агломерацию и объявить кластеризацию.

Результат иерархического агломеративного кластерного анализа часто отображается в виде дерева и называется дендрограммой (рис.3.1). Длины ребер в двоичном дереве показывают порядок, в котором случаи

или наборы были соединены вместе, и величину расстояния между ними. Если бы были найдены три кластера, естественным местом для рисования горизонтальной линии было бы чуть выше, где $\{x_2\}$ объединяется с $\{x_5, x_3\}$, так что три кластера являются $\{x_1\}$, $\{x_4\}$ и $\{x_2, x_3, x_5\}$.

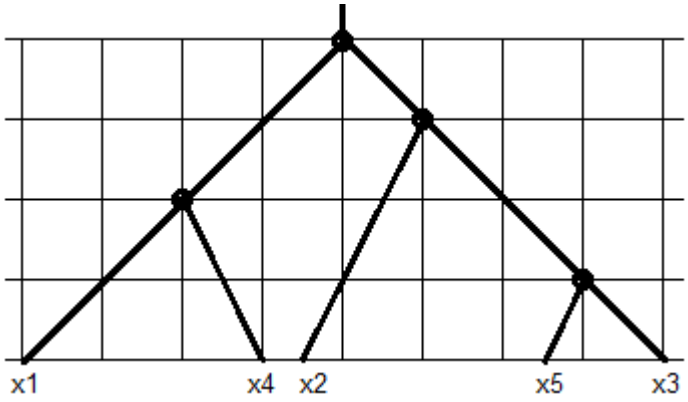

Рис.3.1. Пример агломеративной кластеризации на пяти точках

В этой иерархической процедуре агломеративной кластеризации на пяти точках сначала соединяются x_5 и x_3, а затем x_1 и x_4. Затем x_2 объединяется с первой парой, и, наконец, два набора из двух и трех элементов объединяются. В зависимости от того, где проводится горизонтальная линия, можно получить один, два, три, четыре или пять кластеров.

Выбор меры несхожести d определяет результат окончательной кластеризации. Различные варианты d благоприятствуют различным классам кластеризации так же, как различные варианты правил остановки расщепления благоприятствуют качественно различным классам деревьев в рекурсивном разбиении.

Пусть A и B – два непересекающихся подмножества $\{x_1,...,x_n\}$. Чтобы использовать шаблон агломеративной кластеризации из последнего подраздела, достаточно определить $d(A,B)$ в целом для различных вариантов d. Поскольку целью является последовательное объединение наборов точек, значения d часто называют связями, поскольку они

управляют длинами отрезков линий, соединяющих или связывающих точки x_i друг с другом.

Четыре из наиболее распространенных формулы для меры различий:

- ***Ближайший сосед или одиночная связь***: метрика d на $\mathbb{R}^p$ минимизируется, чтобы определить несходство на множествах, также обозначаемых d, заданное

$$d(A,B) = \min_{a\in A, b\in B} d(a,b).$$

- ***Полная связь***: здесь метрика d максимизируется, чтобы определить несходство на множествах, заданных

$$d(A,B) = \max_{a\in A, b\in B} d(a,b)$$

- ***Центроидная связь***: идея состоит в том, чтобы объединить на итерации $k+1$ те кластеры, центроиды которых на этапе k были ближе всего. То есть,

$$d(A,B) = d(\overline{a},\overline{b})$$

где a-центроид для A, и b-центроид для B.

- ***Средняя связь***: в этом случае метрические расстояния между точками в A и B усредняются:

$$d(A,B) = \frac{1}{\#(A)\#(B)} \sum_{i\in A,\, j\in B} d(x_i, x_j)$$

Это обратная центроидная связь, где используется расстояние между средними значениями.

Кластеризация с одной связью по существу является критерием ближайшего соседа: две точки или кластеры связаны, если они являются ближайшими точками или кластерами данных. Для их соединения требуется только одна связь. Расширение однозвенного соединения между точками до множеств гарантирует, что всегда будет существовать путь с наименьшей длиной, соединяющий все точки в кластере. Таким образом, кластеризация с одной связью также допускает теоретико-графовую

интерпретацию. Хотя у этого есть много хороших свойств, у него есть некоторые недостатки: два кластера любой формы, которые хорошо отделены друг от друга, кроме нескольких точек на линии между ними, будут соединены, если точки между ними достаточно близки друг к другу. Это называется проблемой сцепления, и она очень серьезна во многих наборах данных. Тенденция к таким «штангообразным» областям может быть довольно сильной, потому что кластеризация с одной связью имеет тенденцию формировать несколько больших кластеров в начале иерархии. Хуже того, при наличии выбросов метод может объединять два набора точек, когда большинство точек удалены. Нечетные области возникают естественным образом, поэтому в целом неясно, являются ли такие свойства функциями или ошибками. Действительно, если вы ожидаете сформировать области, которые разделены, кластеризация с одной связью может быть хорошим способом их найти. Часто кластеризация с одной связью выполняется за $O(n^2)$ время (диапазон $O(n \cdot log\ n) - O\ (n^5)$) для фиксированного p.

Кластеризация с полной связью означает, что все точки в двух кластерах соединены линиями, длина которых меньше верхней границы максимума. Это означает, что все связи между точками находятся в пределах расстояния, на котором образовался кластер. По сравнению с одной связью, полная связь имеет тенденцию формировать много меньших, более плотных кластеров. Таким образом, полные дендрограммы связей часто имеют большую структуру и поэтому могут быть более информативными, чем односвязные дендрограммы. Кроме того, из-за максимума полная связь чувствительна к выбросам самих кластеров. Возможно, как и машины опорных векторов, полная связь часто управляется небольшим набором точек, которые могут быть выбросами или входами и не особенно репрезентативны для набора данных. Это также может привести к неустойчивости кластеризации, поскольку

небольшие изменения в данных могут привести к большим изменениям в кластеризации. Часто кластеризация с полной связью выполняется за $O(n^3)$ время для фиксированного p, возможно, меньше для разреженных матриц подобия.

Кластеризация с центроидной связью вычисляет расстояние между средними кластерами. Это среднее значение может быть хорошим компромиссом между крайностями одиночной и полной связи. Ограничение заключается в том, что расстояния, на которых формируются кластеры, являются средними и не соответствуют никаким фактическим связям между точками данных. Недостатком является то, что эта связь не является монотонной: если d_j-уровень несходства для данной итерации j, он может быть выше или ниже, чем d_{j+1}. Такое обращение порядка кластеризации называется инверсией. Напротив, одиночная связь и полная связь монотонны в том, что несходство всегда увеличивается. Таким образом, кластеры из центроидной связи могут быть более сложными для интерпретации.

Кластеризация со средней связью – это еще один компромисс между одиночной и полной связью, который более требователен в вычислительном отношении из-за большого числа операций суммирования. Проблема сцепления гораздо менее распространена для этого метода по сравнению с одиночным сцеплением, и эффект выбросов уменьшается за счет усреднения. Таким образом, часто результат дает более рыхлые кластеры, чем алгоритм одной связи, но меньше более жестких кластеров, чем алгоритм полной связи. По этим причинам средняя связь довольно популярна в приложениях. Кластеризация методом средней связи обычно выполняется за $O(n^2)$ время для фиксированного p.

3.2.2. Разделительная иерархическая кластеризация

Разделительная иерархическая кластеризация рассматривает данные первоначально как одну группу, которая последовательно разбивается с использованием меры расстояния, в принципе, пока каждое подмножество не будет состоять из одного элемента. Очевидно, что делительная иерархическая кластеризация является обратной операцией агломеративной кластеризации. Фактически, каждый метод агломеративной кластеризации, в конечном счете, объединяет все точки данных в один кластер, поэтому выполнение его в обратном порядке дает метод для разделительной кластеризации. И наоборот, каждый метод для делительной кластеризации может быть выполнен в обратном порядке, чтобы дать агломеративный метод. Однако различные отправные точки для двух видов иерархической кластеризации делают различные методы более естественными, хотя они математически эквивалентны.

В своем корне иерархическая разделительная кластеризация разбивает наборы в соответствии с фиксированными правилами. Алгоритм начинается с обработки образца x_i, $i = 1, ..., n$, который рассматривается как единый кластер из n точек данных. Мера несходства d определяется для всех пар точек в выборке. Установить пороговое значение t для принятия решения о целесообразности разделения кластера.

Основными шаблонами алгоритмов для иерархической разделительной кластеризации являются:

- Сначала определите меру несхожести $d(x_i, x_j)$ – расстояние между всеми парами точек данных, и выберите пару с наибольшим расстоянием d_{max} между ними.
- Сравните величину d_{max} с величиной порога t. Если $d_{max} > t$, то разделите один кластер на два, используя выбранную пару в качестве первых элементов в двух новых кластерах. Оставшиеся $n-2$ точки данных помещаются в один из двух новых кластеров: x_l добавляется в новый

кластер, содержащий x_i, если $d(x_i, x_l) < d(x_j, x_l)$; в противном случае x_l добавляется в новый кластер, содержащий x_j.

- На втором этапе значения $d(x_i , x_j)$ находят для x_i, и x_j в пределах одного из двух новых кластеров, чтобы найти пару в кластере с наибольшим расстоянием d_{max} между ними. Если $d_{max} < t$, то разделение кластера прекращается и рассматривается другой кластер. Затем процедура повторяется на кластерах, созданных из этой итерации.

Процедура может быть выполнена до $n-1$ раз, что дает n одноэлементных кластеров, в этом случае уровень в дендрограмме должен быть выбран, чтобы указать кластеризацию, или процедура может быть выполнена до $d_{max} > t$ для всех существующих кластеров.

Методы разделения часто более требовательны к вычислениям, чем агломеративные методы, потому что решения о разделении кластера на два должны приниматься всеми возможными способами. На самом деле, шаблон алгоритма показывает, что есть две основные проблемы, которые должны быть решены для реализации процедуры разделения. Сначала необходимо выбрать кластер для разделения, а затем найти фактическое разделение.

Первая задача (выбор кластера для разбиения) имеет три стандартных варианта решения:

1) Разбить каждый кластер до получения полной двоичной дендрограммы.

2) Всегда разделять самый большой кластер.

3) Если w является центром кластера, то выбрать кластер с наибольшей изменчивостью по w, т. е. j с наибольшим значением $\sum_{i \in C_j} \|x_i - w\|^2$.

Первые два варианта просты: первый игнорирует, являются ли кластеры значимыми, и позволяет выбрать, где вдоль дендрограммы остановиться. Всегда разделение самого большого кластера имеет тенденцию генерировать дендрограммы, где все листья примерно

одинакового размера. Третий может быть лучшим, потому что он чувствителен к разбросу точек в кластерах. Третий алгоритм имеет тенденцию производить кластеры, которые являются более жесткими, чем другие.

Ещё одним вариантом решения может быть алгоритм разделение самого слабого кластера (т. е. кластера с наименьшим средним сходством или наибольшим средним несходством), который дает хорошую производительность.

3.3. Секционная иерархическая кластеризация

В отличие от иерархических методов, которые дают вложенную последовательность кластеризации, обычно основанную на несходстве, процедуры секционированной кластеризации начинаются с начальной кластеризации (разбиения данных) и уточняют ее итеративно, как правило, до тех пор, пока конечная кластеризация не удовлетворяет критерию оптимальности. В общем случае кластеризации последовательности не являются вложенными, поэтому никакая дендрограмма не возможна. С другой стороны, это означает, что секционированные кластеризации не фиксируют плохие выборы; в принципе, плохой выбор для формирования кластера в одной итерации может быть отменен на более поздней итерации. В тривиальном смысле любую иерархическую кластеризацию можно рассматривать как секционированную, игнорируя вложенность кластеров на каждой итерации. Однако, помимо вложенности, иерархические методы обычно основаны на несходстве, в то время как процедуры разделения обычно основаны на целевой функции.

Хотя целевые функции не обязательно являются частью определения методов разделения, они встречаются достаточно часто, чтобы оправдать

обсуждение. Итак, рассмотрим функцию $F=F(K_0,C(K_0))$, где K_0 - начальное число кластеров, а $C_0(K_0)$ – начальная кластеризация x_i. Учитывая F, естественный порядок кластеризации разбивается на секции, чтобы изменить точки данных и максимизировать F:

- Начните с образца x_i для $i = 1, ...,n$ и целого числа K_0. Пусть $C_0(K_0)$ записывается в виде последовательности $C_{0,1},...,C_{0,K0}$.
- Найти $F(K_0, C_0(K_0))$ и установить

$$F_0 = F(K_0,C_{0,1}, ...,C_{0,K0}).$$

- Переставьте точки, назначенные $C_{0,k}$, возможно, повторно выбрав K_0, чтобы получить новую кластеризацию, $C_1(K_1)$, с K_1 кластерами $(C_{1,1}, ...,C_{1,K1})$.
- Найти новое значение целевой функции

$$F_1 = F(K_1,C_{1,1}, ...,C_{1,K1}).$$

- Продолжить итерацию, поиск по значениям K и кластеризации $C_{j,1}$, $...,C_{j,Kj}$ до тех пор, пока F_j не будет достаточно мало и будет сформирован окончательный выбор K_{fin} и $C_{fin,1}$, $...,C_{fin,K\ fin}$ в полученных кластерах.

Начальная кластеризация $C_0(K_0)$ может быть случайной или определяться статистикой K_0, такой как кластерные средние.

4. Модели динамических нейронов

4.1. Клеточные мембраны

Клеточные мембраны обладают избирательной ионной проницаемостью. Избирательная проницаемость мембраны обусловлена специальными каналами – интегральными белками. Они пронизывают мембрану насквозь, образовывая своего рода проход. В клеточных мембранах есть отдельные каналы для ионов натрия (Na^+), калия (K^+) и хлорида (Cl^-). При раздражении клетки каналы натриевых ионов раскрываются, и происходит резкое поступление в клетку ионов натрия. При этом происходит дисбаланс мембранного потенциала. В дальнейшем мембранный потенциал восстанавливается. Каналы калия всегда открыты, через них в клетку медленно попадают ионы калия.

Ионные каналы, управляемые напряжением (Voltage-gated ion channels), содержат центральные поры, через которые ионы могут перемещаться по своим электрохимическим градиентам. При моделировании динамики нейронов обычно рассматривают проникновение через мембрану нейрона (membrane of neuron) ионов натрия (Na^+), калия (K^+) и хлорида (Cl^-). Ионные каналы, как правило, специфичны для ионов разных видов, хотя иногда через них могут проходить ионы аналогичного размера или заряженные ионы. Функциональность ионных каналов обеспечивается тремя основными дискретными блоками: датчиком напряжения, поровым или проводящим трактом (conducting pathway) и затвором (gate). Ионные каналы содержат трансмембранные домены, расположенные вокруг центральной поры. Калиевый канал выпускает катионы калия из клетки в окружающую среду (в межклеточную жидкость). Натриевый канал впускает катионы натрия внутрь клетки (рис.4.1).

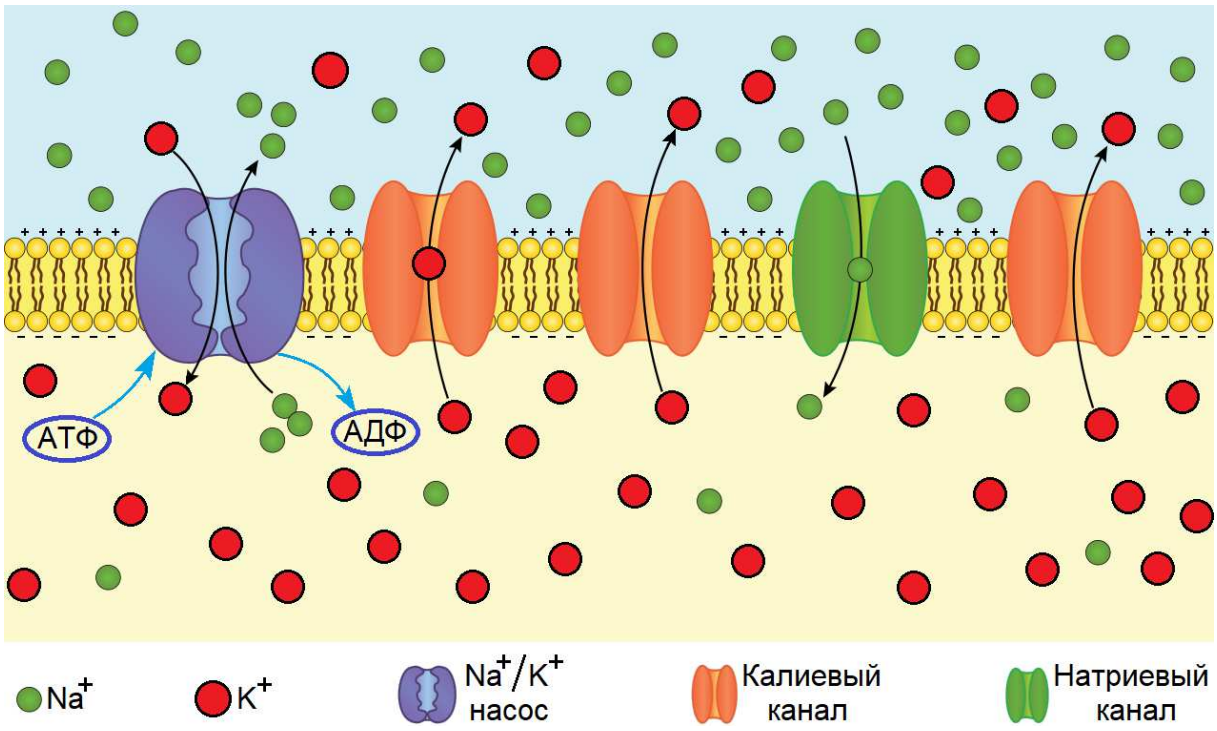

Рис.4.1. Модель фрагмента клеточной мембраны нейрона с ионными каналами и Na^+/K^+ насосом

Насос Na^+/K^+ – это фермент, который переносит ионы K^+ внутрь клетки, а ионы Na^+ выбрасывает во внешнюю среду. Катионы Na^+ и K^+ транспортируются против электрохимического градиента (градиента потенциала). Фермент присоединяет с внутренней стороны мембраны три иона Na^+. Эти ионы изменяют конформацию активного центра АТФ-азы. После этого фермент способен гидролизовать одну молекулу АТФ (аденозинтрифосфа́т). АТФ – универсальный источник энергии для всех биохимических процессов, протекающих в живых системах, в частности, для образования ферментов. В результате гидролиза АТФ образуется АДФ (аденозиндифосфат) и фосфорная кислота с выделением энергии:

$$АТФ + H_2O \rightarrow АДФ + H_3PO_4 + Энергия$$

Гидролиз макроэргических связей молекулы АТФ, сопровождаемый отщеплением 1 или 2 остатков фосфорной кислоты, приводит к выделению, по различным данным, от 40 до 60 кДж/моль.

Выделившаяся после гидролиза АТФ энергия расходуется на изменение конформации переносчика, благодаря чему три иона Na^+ и ион PO_4^{3-} (фосфат) оказываются на внешней стороне мембраны. Здесь ионы Na^+ отщепляются, и присоединяются два иона K^+. После этого фермент возвращается в исходную конформацию, а фосфат-ион и ионы K^+ оказываются на внутренней стороне мембраны. Здесь ионы K^+ отщепляются, и переносчик вновь готов к работе. В итоге во внеклеточной среде создается высокая концентрация ионов Na^+, а внутри клетки – высокая концентрация K^+. Эта разность концентраций используется в клетках при проведении нервного импульса.

Упорядоченное движение ионов образует электрический ток. Каждый фрагмент клеточной мембраны характеризуется электрической ёмкостью и проводимостью, которая может определяться для каждого вида ионов. Электрические процессы в мембране можно приближённо описать на основе электрической модели.

4.2. Модель Ходжкина-Хаксли (Hodgkin-Huxley model)

В 1940-х годах А.Л. Ходжкин и А.Ф. Хаксли прояснили фундаментальный физический механизм, с помощью которого электрические импульсы генерируются нервными клетками и передаются по аксонам у животных и людей. Они экспериментировали с отдельными кусочками аксона кальмара и обобщили свои выводы в статье, опубликованной в 1952 году [12]. Последняя из этих работ [13], возможно, является самой известной статьёй в области нейробиологии, и образует основу области математической и вычислительной нейробиологии.

Авторы провели серию экспериментов по выяснению механизма, лежащего в основе потенциалов действия в зажатом в пространстве аксоне кальмара, и обобщили этот механизм в виде системы обыкновенных дифференциальных уравнений (ОДУ). Ходжкин и Хаксли продевали серебряную проволоку вдоль аксона, тем самым устраняя пространственные вариации мембранного потенциала V. Этот метод, разработанный Джорджем Мармонтом [14], называется методом пространственного зажима (space clamp technique).

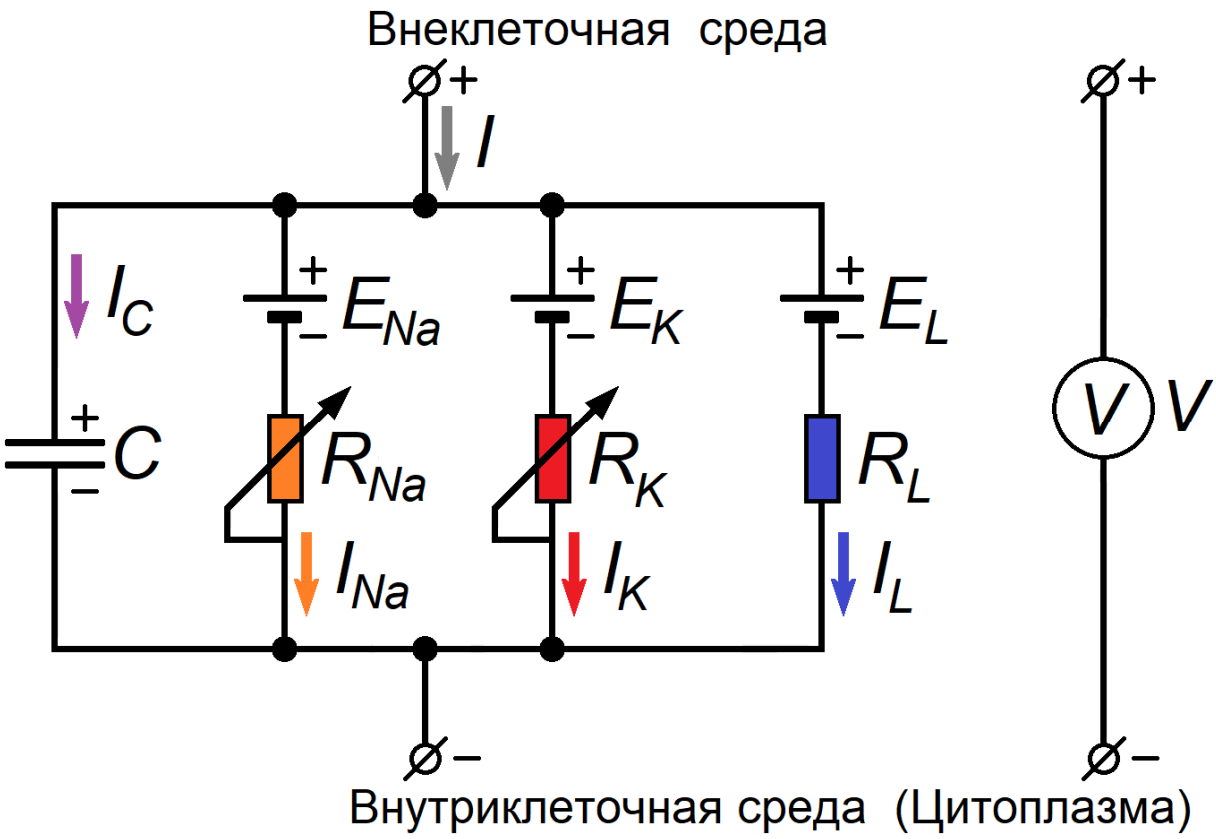

Рис.4.2. Эквивалентная электрическая схема клеточной мембраны в модели активационного потенциала нейрона (схема, соответствующая модели Ходжкина–Хаксли)

На схеме (рис.4.2) использованы следующие обозначения сопротивлений: R_{Na}; R_K; R_L – сопротивления ионных каналов натрия,

калия и канала утечки заряда (соответственно). Сопротивления R_{Na} и R_K – переменные величины, поэтому в дальнейшем будем считать их функциями времени.

ЭДС источников E_{Na} и E_K – потенциалы Нернста ионов натрия и калия (соответственно), т.е. равновесные мембранные потенциалы для ионов натрия и калия, которые определяются на основе известного уравнения Нернста; E_L – средневзвешенное значение потенциалов Нернста всех участвующих в процессе видов ионов. Физический смысл величины E_L можно объяснить другими словами: E_L – потенциал, при котором ток утечки равен нулю.

В общем случае эти потенциалы зависят от температуры T, которая может меняться во времени t, поэтому иногда нужно рассматривать функции $E_{Na}(T(t))$, $E_K(T(t))$ и $E_L(T(t))$. Будем считать температуру постоянной, тогда эти потенциалы можно использовать в виде констант.

Ток утечки переносится ионами хлорида и другими ионами. Иногда величину E_L называют потенциалом Нернста ионов хлорида, но это не совсем правильно. Если бы ток утечки создавался исключительно хлоридом, тогда E_L можно было считать потенциалом Нернста для ионов хлорида, но поскольку заряд утечки переносится смесью разных видов ионов, то E_L является средневзвешенным значением потенциалов Нернста этих видов ионов.

Клеточная мембрана может накапливать электрические заряды, т.е. она обладает емкостными свойствами конденсатора. Мембрана нервной клетки разделяет два слоя электрических зарядов с противоположными знаками. Диэлектрический слой мембраны порождает разность электрических потенциалов на её поверхностях. Эту разность называют транс-мембранным напряжением (trans-membrane voltage). Согласно фундаментальному уравнению конденсатора:

$$C \cdot V = Q, \tag{4.1}$$

где C – электрическая ёмкость мембраны (мембранная емкость – membrane capacitance) – коэффициент пропорциональности между напряжением на мембране и накопленным зарядом); V – транс-мембранное напряжение (разность электрических потенциалов на мембране); Q – разделенный заряд (заряд, на противоположных слоях мембраны равен $\pm Q$).

Если Q и V зависят от времени t, тогда можно продифференцировать обе стороны (4.1) по времени:

$$C \cdot \frac{d}{dt} V(t) = I_C(t). \tag{4.2}$$

где $I_C(t)$ – электрический ток через конденсатор; $I_C(t) = \frac{d}{dt} Q(t)$.

Уравнение (4.2) связывает функции $V(t)$ и $I_C(t)$.

Основываясь на экспериментах, Ходжкин и Хаксли предположили, что полный ток $I(t)$ через мембрану состоит из пяти компонентов: емкостного тока $I_C(t)$; натриевого ионного тока $I_{Na}(t)$; калийного ионного тока $I_K(t)$; токов, образованных другими ионами $I_L(t)$; и дополнительного тока $J(t)$, который они вводили с помощью электродов в ходе своих экспериментов. Величину $I_L(t)$ часто называют током утечки. Обозначение тока утечки заряда связано с англоязычной версией этого термина (Ток утечки – Leakage current).

Токи могут иметь разные направления в зависимости от электрических потенциалов и численных значений характеристик элементов цепи. С целью получения символьных уравнений предполагаемые направления токов выбираются произвольно. Если истинное направление какого-либо тока противоположно выбранному направлению, то в результате решения задачи по вычислению токов мы

получим отрицательную величину этого тока. Предположим, что токи направлены так, как показано на схеме (рис.4.2), тогда:

$$I(t) = I_C(t) + I_{Na}(t) + I_K(t) + I_L(t) + J(t). \qquad (4.2)$$

Выразив величину $I_C(t)$ из равенства (4.2) приведём уравнение (4.1) к виду:

$$C \cdot \frac{d}{dt} V(t) = I(t) - [I_{Na}(t) + I_K(t) + I_L(t) + J(t)]. \qquad (4.3)$$

Токи $I_{Na}(t)$, $I_K(t)$ и $I_L(t)$ меняют знак и направление, когда величина $V(t)$ становится больше или меньше величин источников ЭДС E_{Na}, E_K и E_L (соответственно).

Способность клеточной мембраны пропускать электрические (ионные) токи свидетельствует о том, что она обладает электрической проводимостью. Предполагается, что токи $I_{Na}(t)$, $I_K(t)$ и $I_L(t)$ подчиняются закону Ома:

$$I_{Na}(t) = g_{Na}(t) \cdot [V(t) - E_{Na}], \qquad (4.4)$$

$$I_K(t) = g_K(t) \cdot [V(t) - E_K], \qquad (4.5)$$

$$I_L(t) = g_L(t) \cdot [V(t) - E_L], \qquad (4.6)$$

здесь $I_L(t)$ – ток утечки заряда;

$g_{Na}(t)$ и $g_K(t)$ – проводимости ионных каналов натрия и калия;

$g_L(t)$ – проводимость ионного канала утечки заряда;

$$g_{Na}(t) = \frac{1}{R_{Na}(t)}; \quad g_K(t) = \frac{1}{R_K(t)}; \quad g_L(t) = \frac{1}{R_L(t)}.$$

Ходжкин и Хаксли на основе своих экспериментов пришли к выводу, что проводимости $g_{Na}(t)$ и $g_K(t)$ зависят от времени, поэтому соответствующие сопротивления $R_{Na}(t)$ и $R_K(t)$ обозначены на схеме как переменные сопротивления. Проводимости каналов $g_{Na}(t)$ и $g_K(t)$ представляют [12,13,15] в следующей форме:

$$g_{Na}(t) = \bar{g}_{Na} \cdot m^3(t) \cdot h(t), \qquad g_K(t) = \bar{g}_K \cdot n^4(t), \qquad (4.7)$$

где $\bar{g}_{Na}$ и $\bar{g}_K$ – постоянные проводимости;

$m(t)$, $h(t)$ и $n(t)$ – зависящие от времени безразмерные величины, изменяющиеся от 0 до 1.

Ходжкин и Хаксли предложили [13] следующую физическую интерпретацию равенств (4.7):

1) каждый натриевый канал открывается и закрывается четырьмя затворами, расположенными последовательно;
2) эти затворы открываются и закрываются независимо друг от друга;
3) все четыре затвора должны быть открыты, чтобы канал был открыт;
4) существуют три затвора одного типа, назовем их m-затворами, и один затвор другого типа, назовем его h-затвором.

Функции $m(t)$ и $h(t)$ обозначают фракции открытых m- и h-затворов (соответственно). Доля открытых натриевых каналов равна $m^3(t) \cdot h(t)$.

Канал ионов калия содержит четыре одинаковых независимых затвора, расположенных последовательно. Калиевый канал открыт только в том случае, если открыты все четыре затвора. Функция $n(t)$ обозначает долю открытых калиевых затворов. Доля открытых калиевых каналов равна $n^4(t)$. Проводимость натриевых и калиевых каналов описана уравнением (4.7) при $\bar{g}_{Na}$ и $\bar{g}_K$, равных максимально возможным натриевым и калиевым проводимостям, реализуемым при открытых каналах. Поэтому переменные m, h и n называются **стробирующими переменными**.

В отличие от проводимости натрия и калия, проводимость утечки $g_L(t)$ постоянна. По эстетическим соображениям запишем:

$$g_L(t) = \bar{g}_L. \tag{4.8}$$

В модели Ходжкина-Хаксли $m(t)$, $h(t)$ и $n(t)$ подчиняются системе обыкновенных дифференциальных уравнений (ОДУ) первого порядка:

$$\frac{d}{dt}m(t) = \frac{m_\infty(V) - m(t)}{\tau_m(V)}, \tag{4.9}$$

$$\frac{d}{dt}h(t) = \frac{h_\infty(V) - h(t)}{\tau_h(V)}, \tag{4.10}$$

$$\frac{d}{dt}n(t) = \frac{n_\infty(V) - n(t)}{\tau_n(V)}, \tag{4.11}$$

где $m_\infty(V)$, $h_\infty(V)$, $n_\infty(V)$ – установившиеся значения $m(t)$, $h(t)$, $n(t)$ для активации (соответственно); $\tau_m(V)$, $\tau_h(V)$ и $\tau_n(V)$ – постоянные времени процессов активации и деактивации каналов.

Функции $m_\infty(V)$, $h_\infty(V)$, $n_\infty(V)$ обычно представляются уравнениями Больцмана.

Вместо трёх уравнений для m, h и n рассмотрим одно:

$$\frac{d}{dt}x(t) = \frac{x_\infty(v) - x(t)}{\tau_x(v)}, \quad \text{для} \quad x = m, h, n. \tag{4.12}$$

Если бы x_∞ и τ_x были константами, независимыми от V, (4.12) было бы эквивалентно:

$$x(t) = x(0)\cdot\exp\left(-\frac{t}{\tau_x}\right) + x_\infty(v)\cdot\left[1 - \exp\left(-\frac{t}{\tau_x}\right)\right], \tag{4.13}$$

Мы всегда предполагаем, что τ_x положительно. Правая часть (4.13) является средневзвешенным значением $x(0)$ и x_∞. Умножение веса $x(0)$ начинается от 1 (при $t = 0$) и экспоненциально быстро спадает до 0. Умножение веса x_∞ начинается с 0 и экспоненциально быстро стремится к 1. Таким образом, $x(t)$ экспоненциально быстро движется от $x(0)$ к x_∞ (его предел при $T \to \infty$). Время, необходимое $x(t)$ для достижения "существенного прогресса" в направлении x_∞, т.е. время, необходимое для того, чтобы вес перед $x(0)$ уменьшился e раз, равно τ_x. Функция $x(t)$ стремится к x_∞ экспоненциально с постоянной времени τ_x. Таким образом, уравнения (4.9) – (4.11) выражают, что m, h и n – перемещения

по направлению к m_∞, h_∞ и n_∞ экспоненциально с постоянной времени τ_m, τ_h и τ_n (соответственно). Величины m_∞, h_∞ и n_∞ являются "движущимися целями", т.е. они меняются по мере изменения V. Скорость реагирования m, h и n на изменение напряжения V характеризуется величинами τ_m, τ_h и τ_n (соответственно).

Функции $x_\infty(V)$ и $\tau_x(V)$, где $x = m, h, n$, были определены Ходжкином и Хаксли так, чтобы полученная система ОДУ соответствовала их экспериментальным данным. На рис.4.3 показаны графики этих функций.

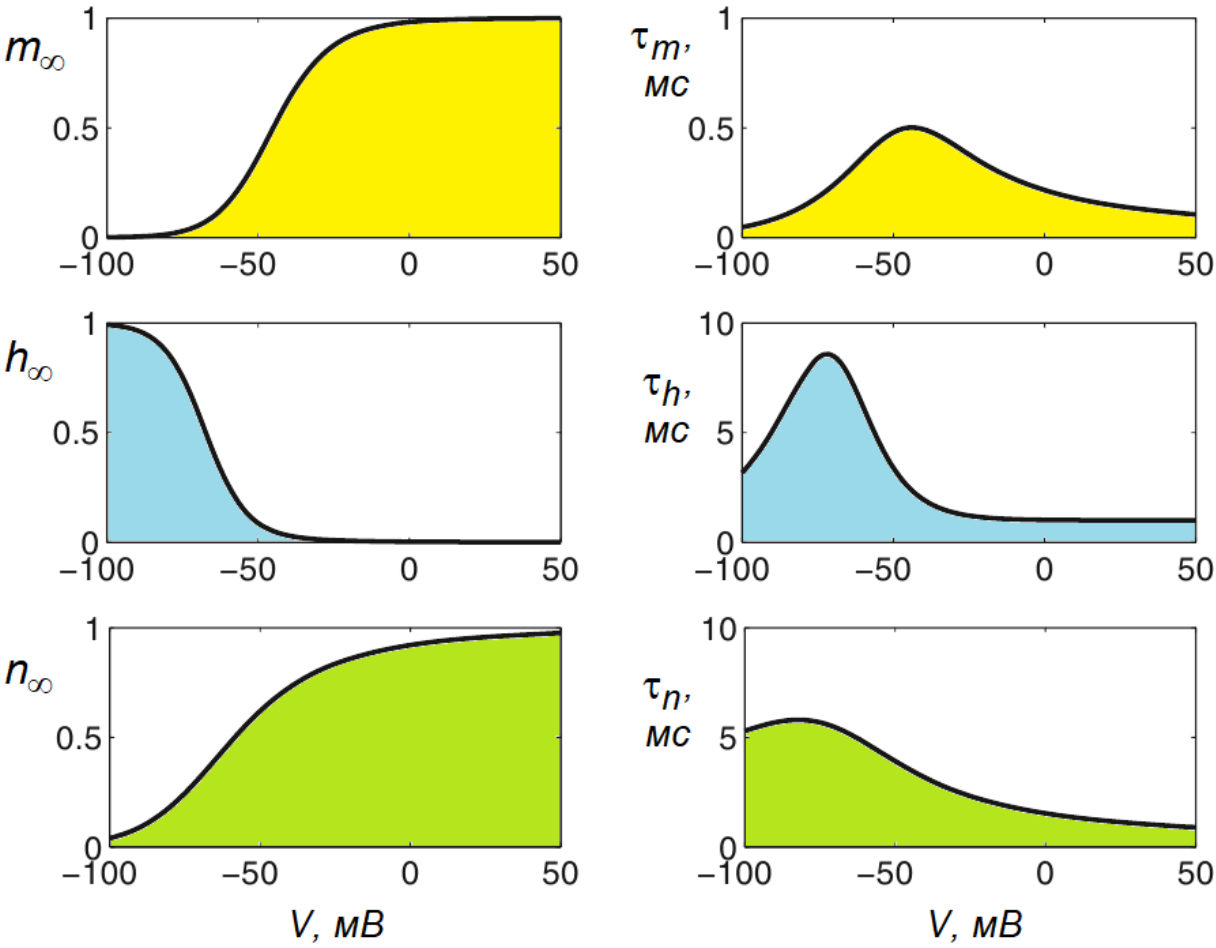

Рис.4.3. Графики функций $x_\infty(V)$ и $\tau_x(V)$, где $x = m, h, n$

Сделаем несколько наблюдений, основанных на графиках: Величина τ_m примерно в 10 раз меньше величин τ_h и τ_n, следовательно m

реагирует на изменения в V намного быстрее, чем h и n. Уравнения Ходжкина-Хаксли представляют собой так называемую «медленно-быструю систему».

Функции $m_\infty(V)$ и $n_\infty(V)$ – монотонно-возрастающие функции; m и n – **переменные активации** (activation variables). При увеличении V открываются m-затворы, а также n-затворы, хотя и в десятикратном масштабе времени. Принято, что m- и n- затворы являются затворами активации.

Функция $h_\infty(V)$ – убывающая функция V. Поэтому величина h – **переменная инактивации** (inactivation variable). Когда V увеличивается, h-затворы закрываются, но закрываются в несколько раз медленнее, чем открываются m-затворы. Иногда используют терминологию h-затвор – **инактивационный затвор** (inactivation gate).

Дифференциальное уравнение (4.3) с учётом (4.4)–(4.8) принимает вид:

$$C \cdot \frac{d}{dt}V(t) = I(t) - \left[\bar{g}_{Na} \cdot m^3(t) \cdot h(t) \cdot [V(t) - E_{Na}(t)] + \right.$$
$$\left. + \bar{g}_K \cdot n^4(t) \cdot [V(t) - E_K(t)] + \bar{g}_L \cdot [V(t) - E_L(t)] + J(t) \right]. \qquad (4.14)$$

Уравнение (4.14) содержит несколько неизвестных функций и, поэтому должно решаться совместно с другими уравнениями. Дифференциальные уравнения (4.9), (4.10), (4.11) и (4.14) образуют систему обыкновенных дифференциальных уравнений (ОДУ) Ходжкина-Хаксли. Это система из четырех ОДУ для четырех неизвестных функций $V(t)$, $m(t)$, $h(t)$ и $n(t)$. Система ОДУ (4.9), (4.10), (4.11) и (4.14) – основа математической модели динамического нейрона (модели Ходжкина-Хаксли).

В уравнении (4.14) величина C имеет физический смысл электрической ёмкости; $\bar{g}_{Na}$, $\bar{g}_K$ и $\bar{g}_L$ – проводимости; $J(t)$ – ток. Разделив обе стороны уравнения (4.14) на общую площадь мембраны, мы увидим,

что точно такое же уравнение выполняется, если в качестве величин $J(t)$, C, $\bar{g}_{Na}$, $\bar{g}_K$ и $\bar{g}_L$ принять их поверхностные плотности. Далее будем считать, что $J(t)$ – плотность дополнительного тока, вводимого с помощью электродов; C – погонная ёмкость мембраны (поверхностная плотность ёмкости мембраны); $\bar{g}_{Na}$, $\bar{g}_K$ и $\bar{g}_L$ – проводимости в расчёте на единицу площади мембраны.

Используем значения параметров, приведённые в книге [15]:

$$C = 1\ \frac{мкФ}{см^2},\quad E_{Na} = 45\ мВ,\quad E_K = -82\ мВ,\quad E_L = -59\ мВ,$$

$$\bar{g}_{Na} = 120\ \frac{мСм}{см^2},\quad \bar{g}_K = 36\ \frac{мСм}{см^2},\quad \bar{g}_L = 0.3\ \frac{мСм}{см^2}.$$

Величина $\bar{g}_L$ намного меньше, чем $\bar{g}_{Na}$ и $\bar{g}_K$. Тем не менее, проводимость утечки имеет решающее значение для поведения модели. Причина в том, что между импульсами напряжения проводимость $\bar{g}_L$ не очень мала по сравнению с $\bar{g}_{Na} \cdot m^3(t) \cdot h(t)$ и $\bar{g}_K \cdot n^4(t)$. Натриевые и калиевые каналы в основном закрыты между импульсами напряжения, в то время как каналы утечки остаются открытыми. Уравнение (4.12) можно переписать в виде:

$$\frac{d}{dt}x(t) = \alpha_x(V)\cdot[1 - x(t)] - \beta_x(V)\cdot x(t),\quad \text{для}\quad x = m,\ h,\ n. \qquad (4.15)$$

Теперь найдём функциональные зависимости $x_\infty(V)$ и $\tau_x(V)$, где $x = m,\ h,\ n$. Функции $x_\infty(V)$ и $\tau_x(V)$ в (4.12) связаны с функциями $\alpha_x(V)$ и $\beta_x(V)$ в (4.15) соотношениями:

$$x_\infty(V) = \frac{\alpha_x(V)}{\alpha_x(V) + \beta_x(V)},\qquad \tau_x(V) = \frac{1}{\alpha_x(V) + \beta_x(V)}. \qquad (4.16)$$

Выражения (4.16) эквивалентны (соответственно) равенствам:

$$\alpha_x(V) = \frac{x_\infty(V)}{\tau_x(V)},\qquad \beta_x(V) = \frac{1 - x_\infty(V)}{\tau_x(V)}. \qquad (4.17)$$

Предположим, что затворы открываются и закрываются в случайные моменты времени с мгновенными скоростями, зависящими от V. Величина $\alpha_x(V)$ – мгновенная скорость, с которой закрытые затворы открываются. Величина $\beta_x(V)$ – мгновенная скорость, с которой открытые затворы закрываются. Величины $\alpha_x(V)$ и $\beta_x(V)$ – нормы времени (time rates), измеряемые в $мс^{-1}$. В моделях нейронов величина V меньше вольта, поэтому её обычно измеряют в $мВ$.

Вероятность открытия закрытого затвора за короткое время Δt составляет приблизительно $\alpha_x(V)\cdot\Delta t$. Вероятность закрытия открытого затвора на отрезке времени Δt приблизительно равна $\beta_x(V)\cdot\Delta t$. Дифференциальное уравнение для неизвестной функции $x(t)$, может быть построено либо путем указания $x_\infty(V)$ и $\tau_x(V)$, либо путем задания $\alpha_x(V)$ и $\beta_x(V)$. Ходжкин и Хаксли измерили величины $x_\infty(V)$ и $\tau_x(V)$, а потом вычисляли по ним мгновенные скорости $\alpha_x(V)$ и $\beta_x(V)$. Такой способ проще, чем вычисление $\alpha_x(V)$ и $\beta_x(V)$ напрямую. Все функции $\alpha_x(V)$ и $\beta_x(V)$ являются монотонными функциями от V, в то время как функция $\tau_x(V)$ не является монотонной (смотрите графики). Ходжкин и Хаксли использовали следующие формулы:

$$\alpha_m(V) = \frac{(V+45)/10}{1-\exp[-(V+45)/10]}, \quad \beta_m(V) = 4\cdot\exp[-(V+70)/18],$$

$$\alpha_h(V) = 0.07\cdot\exp[-(V+70)/20], \quad \beta_h(V) = \frac{1}{\exp[-(V+40)/10]+1},$$

$$\alpha_n(V) = 0.01\cdot\frac{V+60}{1-\exp[-(V+60)/10]}, \quad \beta_n(V) = \frac{1}{8}\cdot\exp[-(V+70)/80].$$

В формулах для $\alpha_m(V)$ и $\alpha_n(V)$ знаменатели становятся нулевыми для специальных значений V: $V=-45$ для $\alpha_m(V)$, и $V=-60$ для $\alpha_n(V)$. Правило Лопиталя должно использоваться для оценки $\alpha_m(V)$ и $\alpha_n(V)$ для этих значений V.

В модели Ходжкина-Хаксли потенциалы V активации нейрона генерируются, когда m-затворы открываются в ответ на начальную деполяризацию, вызывая натриевый ток в клетку. Этот натриевый ток поднимает потенциал V дальше, и происходит своего рода цепная реакция. Рост V прекращается, когда h-затворы закрываются, тем самым прекращается приток натрия, а затем n-затворы открываются, что приводит к оттоку калия. Важно, что $\tau_m(V)$ намного меньше $\tau_h(V)$ и $\tau_n(V)$ – это видно на графиках (смотрите рис.10). Если бы все постоянные времени были равны, закрытие h-затворов и открытие n- затворов могли бы немедленно отменить открытие m- затворов, тем самым предотвращая скачок напряжения.

В модели Ходжкина-Хаксли переменную $x(t)$ обычно называют **переменной активации** (activation variable), если $x_\infty(V)$ является возрастающей функцией от V. Переменную $x(t)$ называют **переменной инактивации** (inactivation variable), если $x_\infty(V)$ представляет собой убывающую функцию от V. Как упоминалось ранее в классической модели Ходжкина-Хаксли m и n являются переменными активация, и h является переменной инактивации. Увеличение переменной активации в результате деполяризации вызывает активацию тока. Уменьшение переменной активации в результате гиперполяризации вызывает деактивацию тока. Уменьшение переменной инактивации в результате деполяризации приводит к инактивации и увеличении переменной инактивации в результате гиперполяризации вызывает деинактивацию тока. Деактивация – это не то же самое, что инактивация. Деинактивация – это не то же самое, что активация.

Система ОДУ (4.9)–(4.11) и (4.14) приводится [15] к векторному дифференциальному уравнению:

$$\frac{d}{dt}Y(t) = F(t), \tag{4.18}$$

где: $Y(t)$ и $F(t)$ – векторные функции:

$$Y(t)=\begin{bmatrix} V(t) \\ m(t) \\ h(t) \\ n(t) \end{bmatrix}, \qquad F(t)=\begin{bmatrix} \dfrac{f(t)}{C} \\ \dfrac{m_\infty(V(t))-m(t)}{\tau_m(V(t))} \\ \dfrac{h_\infty(V(t))-h(t)}{\tau_h(V(t))} \\ \dfrac{n_\infty(V(t))-n(t)}{\tau_n(V(t))} \end{bmatrix},$$

$$f(t)=I(t)-\begin{bmatrix} \overline{g}_{Na}\cdot m^3(t)\cdot h(t)\cdot[V(t)-E_{Na}(t)]+\overline{g}_K\cdot n^4(t)\cdot[V(t)-E_K(t)] \\ +\overline{g}_L\cdot[V(t)-E_L(t)]+J(t) \end{bmatrix}.$$

В результате дискретизации времени с постоянным шагом Δt сформируем последовательность моментов времени: $t_s = s\cdot\Delta t$, где $s = 0, 1, 2, ..., M$.

Неизвестную векторную функцию будем аппроксимировать последовательностью векторов $Y_s \approx Y(t_s)$ для всех $s = 1, 2, ..., M$, где M – количество расчётных точек.

Численное решение ОДУ выполним методом разностной аппроксимации. Производную в уравнении (4.18) аппроксимируем разностным отношением:

$$\left.\frac{d}{dt}Y(t)\right|_{t=t_s} = \frac{Y_{s+1}-Y_s}{\Delta t}$$

В результате дискретизации времени из дифференциального уравнения (4.18) получаем алгебраическое уравнение:

$$\frac{Y_{s+1}-Y_s}{\Delta t} = F(t_s). \tag{4.19}$$

Из формулы (4.19) можно получить следующее соотношение:

$$Y_{s+1} = Y_s + \Delta t\cdot F(t_s). \tag{4.20}$$

Таким образом, численное решение векторного ОДУ (4.18) сводится к векторной рекуррентной формуле:

$$\begin{bmatrix} V_{s+1} \\ m_{s+1} \\ h_{s+1} \\ n_{s+1} \end{bmatrix} = \begin{bmatrix} V_s \\ m_s \\ h_s \\ n_s \end{bmatrix} + \Delta t \cdot \begin{bmatrix} \dfrac{f(t_s)}{C} \\ \dfrac{m_\infty(V_s) - m_s}{\tau_m(V_s)} \\ \dfrac{h_\infty(V_s) - h_s}{\tau_h(V_s)} \\ \dfrac{n_\infty(V_s) - n_s}{\tau_n(V_s)} \end{bmatrix}. \qquad (4.21)$$

С целью запуска рекурсии необходимо задать начальные условия (initial condition). Зададим начальные условия задачи: $V_0 = -50\ мВ$; $m_0 = 0.4$; $h_0 = 0.4$; $n_0 = 0.4$.

Рекуррентная формула (4.21) позволяет найти следующие значения V_{s+1}; m_{s+1}; h_{s+1}; n_{s+1} для $s = 0, 1, 2, ..., M$.

Плотность дополнительного тока, вводимого с помощью электродов, будем считать равной $J(t) = 10\ мкА/см^2$.

Точность решения ОДУ зависит от выбранной величины Δt. Чем меньше Δt, тем выше точность вычислений, но больше циклов рекурсии нужно выполнить для расчёта динамических величин на заданном отрезке времени, и, следовательно, требуется больше вычислительных ресурсов. При моделировании процессов в нейроне обычно принимают $\Delta t = 0.01\ мс$, и здесь мы примем это значение Δt. Пусть $M = 5000$.

Решение векторного дифференциального уравнения (4.18) представлено на рис. 4.4.

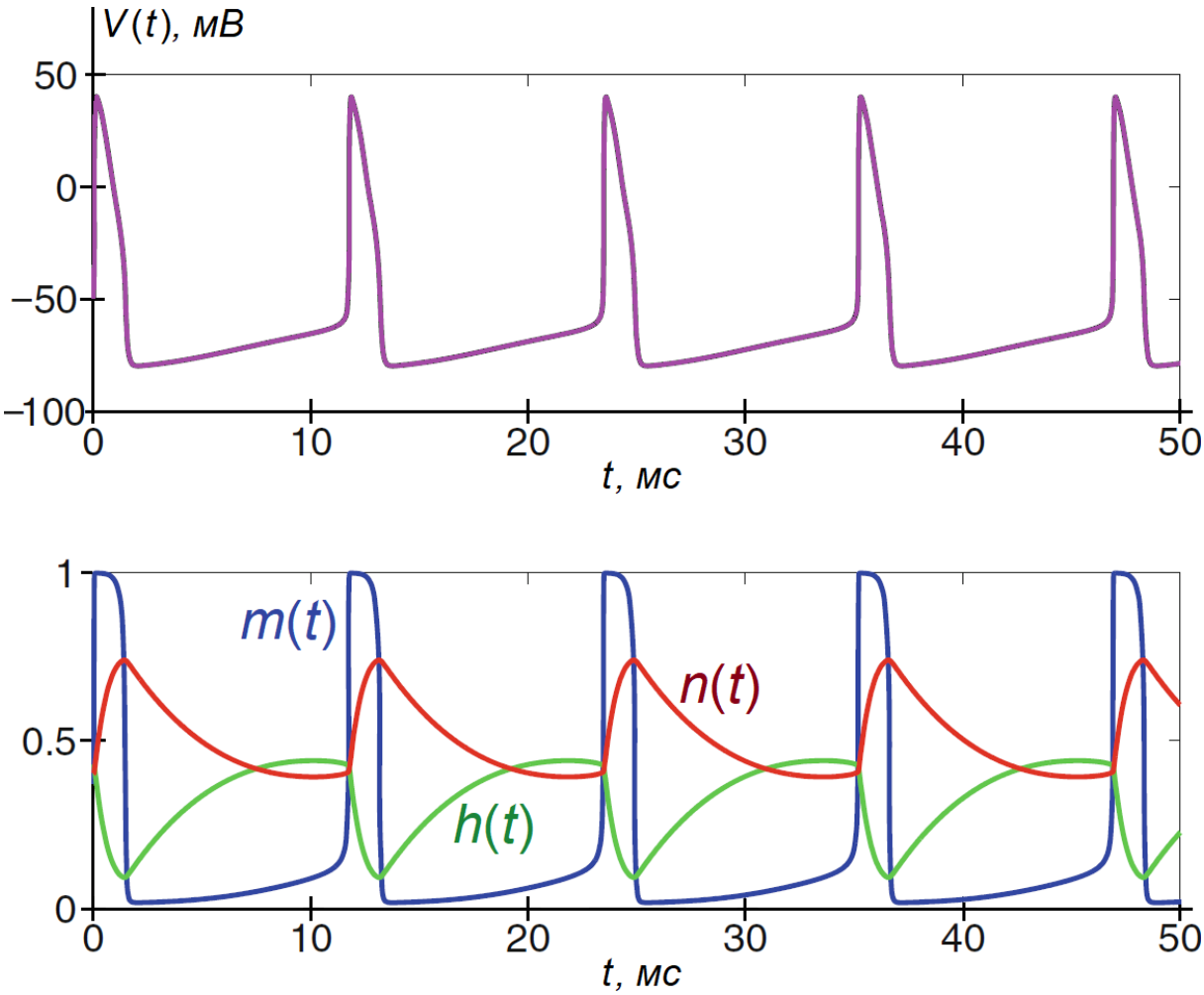

Рис.4.4. Графики функций $V(t)$, $m(t)$, $h(t)$ и $n(t)$.

Получены периодические функции $V(t)$, $m(t)$, $h(t)$ и $n(t)$. Возможны и не периодические режимы работы нейрона. Непериодические режимы также описываются уравнением (4.18) и сводятся к векторной рекуррентной формуле (4.21). Непериодические сигналы можно получить, задав непостоянную плотность дополнительного тока $J(t)$, вводимого с помощью электродов.

4.3. Модель Фитцхью-Нагумо

Фитцхью (FitzHugh R.) и Нагумо (Nagumo J.) в 1960–1962 годах предложили [16,17] новую модель нейрона. Модель описывает регенеративное самовозбуждение посредством нелинейной положительной обратной связи (рис.4.5.) напряжения на мембране, а также восстановление посредством линейной отрицательной обратной связи напряжения на затворе.

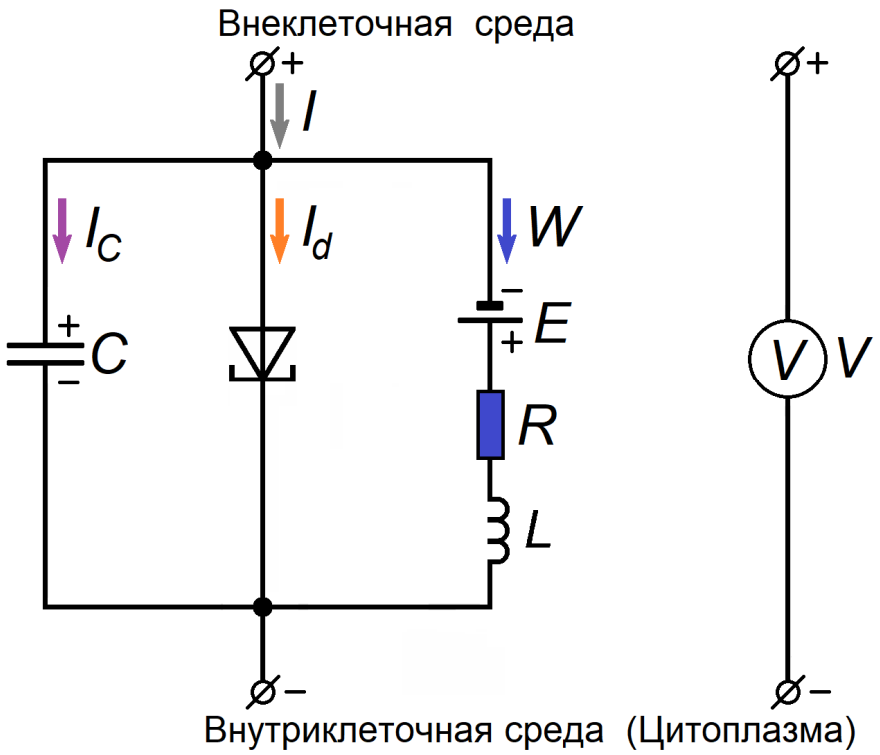

Рис.4.5. Эквивалентная схема мембраны на основе туннельного диода

Система дифференциальных уравнений Фитцхью-Нагумо в наиболее общем виде:

$$\begin{cases} C \cdot \dfrac{d}{dt} V(t) = I(t) - I_d(t) - W(t) \\ L \cdot \dfrac{d}{dt} W(t) = E + V(t) - R \cdot W(t) \end{cases}, \tag{4.22}$$

где $I(t)$ – внешний ток; $I_d(t)$ – ток через туннельный диод; $W(t)$ – ток через индуктивность и сопротивление.

Туннельный диод характеризуется нелинейной ВАХ (рис.4.6). ВАХ туннельного диода имеет область отрицательного дифференциального сопротивления (рис.4.6) в диапазоне напряжений от U_1 до U_2.

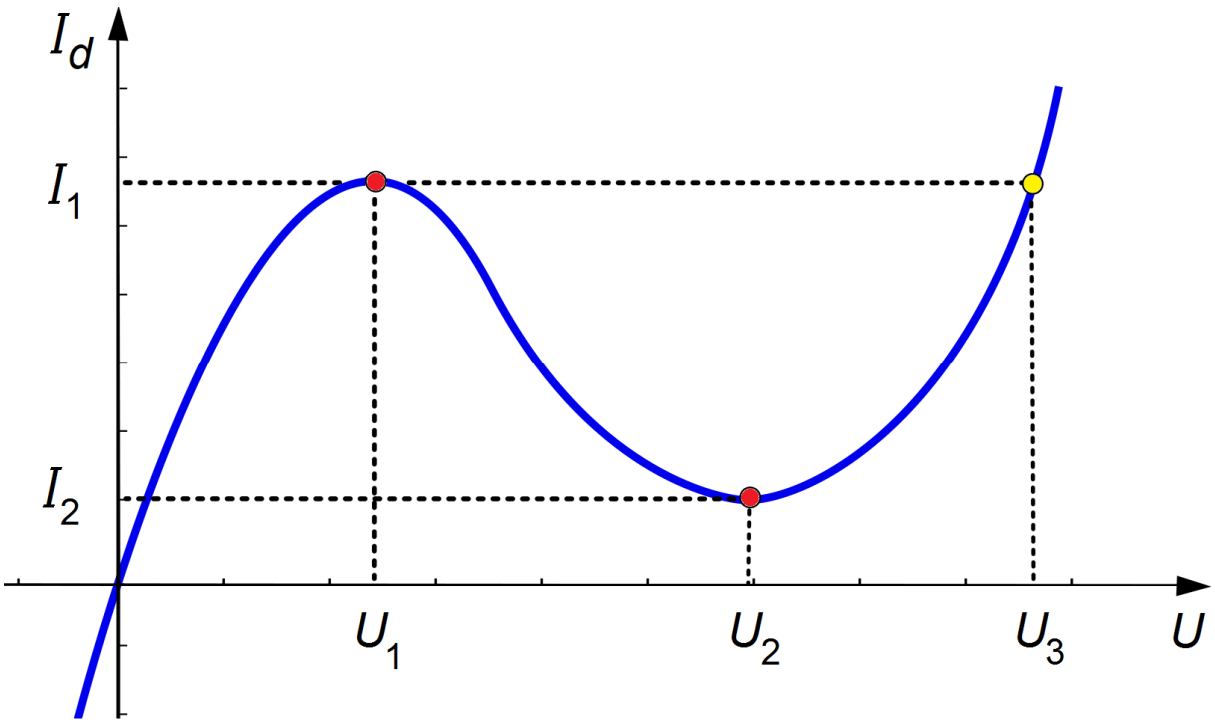

Рис.4.6. ВАХ туннельного диода

Ток через туннельный диод $I_d(t) = F(V(t))$, где нелинейная функция $F(V)$ – ВАХ туннельного диода. Функцию $F(V)$ в математической модели нейрона обычно задают в виде:

$$F(V) = k_3 \cdot \frac{V^3}{3} - k_1 \cdot V. \qquad (4.23)$$

Система уравнений (4.22) принимает вид:

$$\begin{cases} C \cdot \dfrac{d}{dt} V(t) = I(t) - \left[k_3 \cdot \dfrac{V^3(t)}{3} - k_1 \cdot V(t) \right] - W(t) \\ L \cdot \dfrac{d}{dt} W(t) = E + V(t) - R \cdot W(t) \end{cases}. \qquad (4.24)$$

Когда внешний ток $I(t)$ превышает определенное пороговое значение, система демонстрирует характерное возвратно-поступательное движение в фазовом пространстве, до тех пора переменные $V(t)$ и $W(t)$ не "релаксируют" до предыдущего состояния.

Численное решение системы уравнений (4.24) сводится к разностному уравнению:

$$\begin{bmatrix} C \cdot \dfrac{V_{s+1} - V_s}{\Delta t} \\ L \cdot \dfrac{W_{s+1} - W_s}{\Delta t} \end{bmatrix} = \begin{bmatrix} I_s - k_3 \cdot \dfrac{V_s^3}{3} + k_1 \cdot V_s - W_s \\ E + V_s - R \cdot W_s \end{bmatrix}. \qquad (4.25)$$

Из равенства (4.25) получаем векторную рекуррентную формулу:

$$\begin{bmatrix} V_{s+1} \\ W_{s+1} \end{bmatrix} = \begin{bmatrix} V_s \\ W_s \end{bmatrix} + \Delta t \cdot \begin{bmatrix} \dfrac{I_s - k_3 \cdot \dfrac{V_s^3}{3} + k_1 \cdot V_s - W_s}{C} \\ \dfrac{E + V_s - R \cdot W_s}{L} \end{bmatrix}. \qquad (4.26)$$

Пусть параметры модели имеют следующие значения: $L = 12\ мГн$;

$C = 1\ мФ$; $R = 0.2\ Ом$; $k_1 = 1\ См$; $k_3 = 1 \dfrac{A}{B^3}$; $E = 0$.

Система уравнений (4.24) чувствительна к величине внешнего тока. Решение быстро приходит к периодическому режиму. Три периодических решения системы (4.24), построенных при $\Delta t = 0.01\ мс$ и при разных I, показаны на рис.4.7. Соответствующие значения внешнего тока отмечены на графиках.

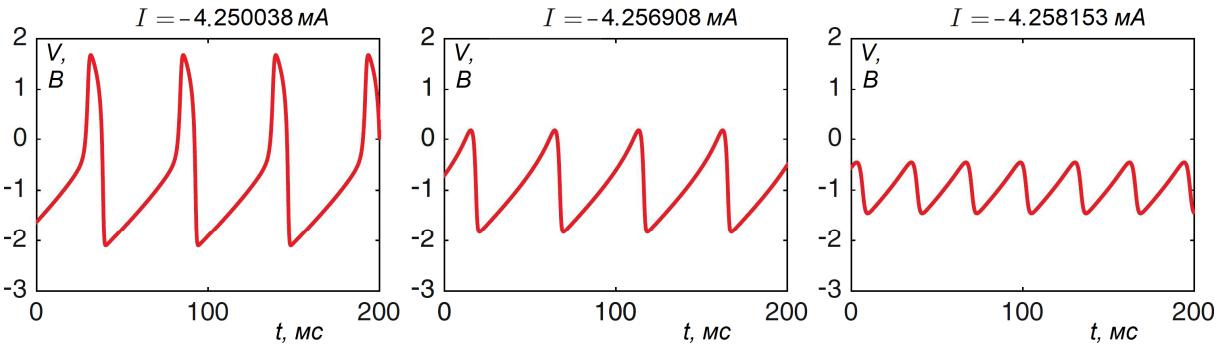

Рис.4.7. Решения системы уравнений Фитцхью-Нагумо

Модель Фитцхью–Нагумо описывает динамику активации и деактивации пульсирующего нейрона. На рис.4.8 показан предельный цикл притяжения траекторий для нескольких значений I, близких друг к другу.

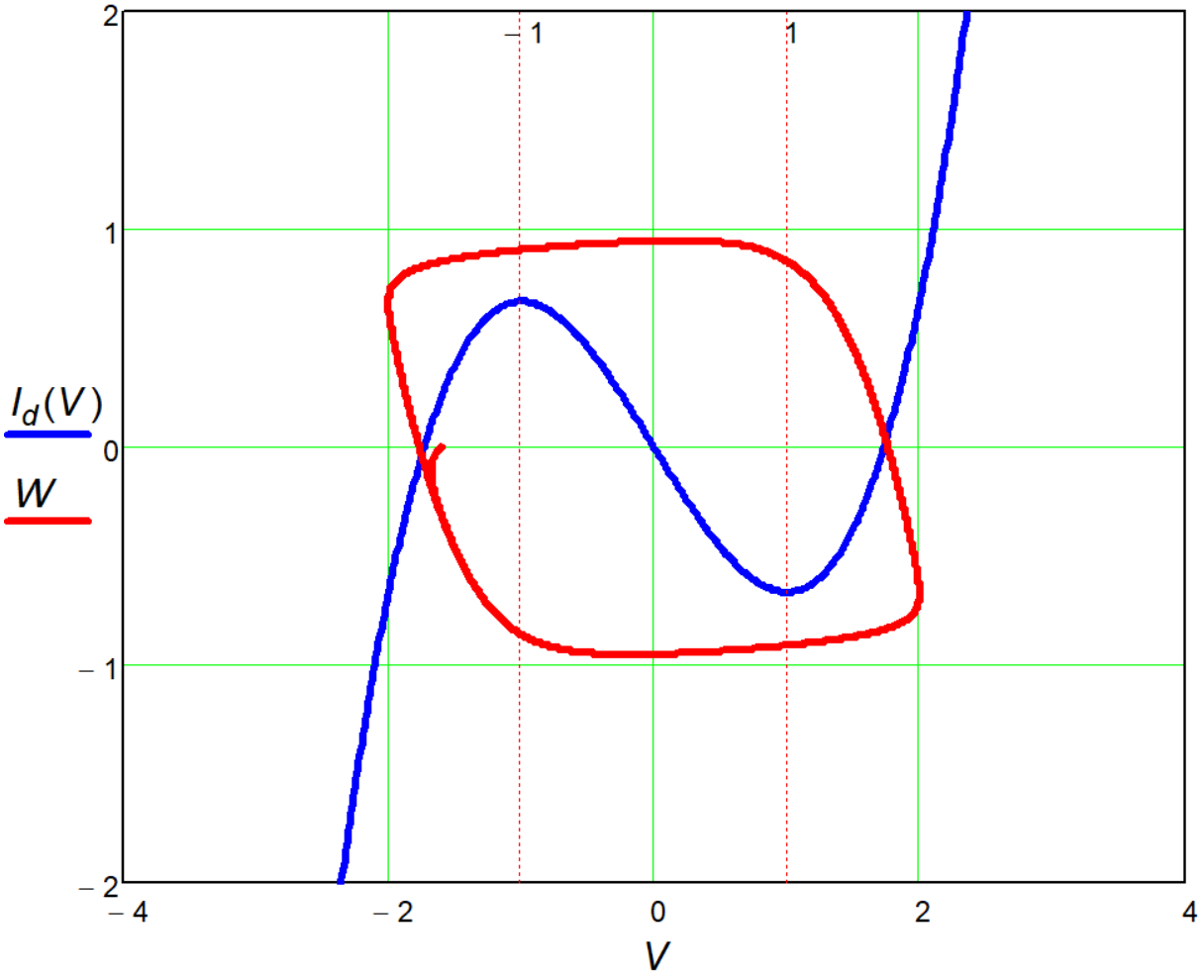

Рис.4.8. Решения системы уравнений Фитцхью-Нагумо в плоскости $V-W$ и центрированная ВАХ туннельного диода (синяя кривая)

Предельному циклу соответствует $I_c \approx$ -4.256889 *мА*. При $I \le I_c$ наблюдаются колебания потенциала активации с малой амплитудой. При $I > I_c$, включается механизм модели, генерирующий пиковые импульсы, и амплитуда колебаний потенциала активации нейрона резко увеличивается.

Рассмотренные динамические модели нейронов имеют определенную степень биологической значимости. Кроме того, описываемые нейронные модели имеют специфические свойства, такие как послеполяризация и Би-пороговая генерация спайков на фоне подпороговых колебаний. Возможно создание моделей с дискретным временем, т. е. в виде точечной карты. Такие модели предполагают только качественное описание динамики реальных нейронов. С другой стороны, дискретные модели обладают множеством динамических режимов подходят для численного моделирования, включая крупномасштабные нейронные сети.

5. Интеллектуальные приборы

Интеллектуальные приборы – это компьютеризированные устройства, отличающиеся от неинтеллектуальных включением микроконтроллеров и микропроцессоров для выполнения функций обработки сигналов, связи, обработки данных, самокалибровки и принятия решений. Например, некоторые интеллектуальные инструменты могут автоматически выполнять следующие функции: осуществлять коррекцию выхода прибора в части смещений сигналов, вызванных изменениями окружающей среды (например, температурой, влажностью); обеспечивать линейность выходных сигналов, не смотря на нелинейные характеристики чувствительных элементов датчиков [18].

В работе [19] представлена конструкция интеллектуального датчика влажности. Моделирования емкостного датчика влажности, выполненного

на основе технологий MEMS, было реализовано на базе нейронных сетей в среде моделирования Matlab. Модель учитывает точные характеристики нелинейности датчика, эффекта гистерезиса и перекрестной чувствительности выходного сигнала датчика влажности. Проведено обучение для формирования параметров аналитической модели "емкостный датчик влажности".

На первом этапе полученная модель ёмкостного датчика в среде моделирования электронных схем PSPICE, которая позволяет учитывать зависимость изменения влажности от изменения электрической ёмкости датчика.

На втором этапе осуществлена линеаризация выходной характеристики датчика с помощью программы Matlab. Создана база данных для элемента коррекции "корректор". Матрица смещения и матрица весов, формируются путем обучения нейронной сети. Обученная модель датчика и модель корректора позволяет исправлять нелинейный отклик чувствительного элемента датчика и устранять гистерезисный эффект и перекрестную чувствительность.

Модель интеллектуального датчика состоит из трёх блоков: модель датчика, модель корректора и модель преобразователя емкости в напряжение.

Этап обучения требует наличия базы данных, выбора архитектуры сети и нахождения количества слоев и нейронов в каждом слое. Численность входного слоя нейронной сети определятся числом входных параметров: влажность, температура и гистерезис. Нейронная сеть модели датчика имеет только один выход, величина сигнала на котором должна быть равна величине электрической ёмкости датчика. Поиск параметров нейронной сети осуществляется следующим образом:

1) Имеющиеся данные разделяются на обучающие, проверочные и тестовые наборы.

2) Осуществляется выбор одной из допустимых архитектур нейронной сети.

3) Выполняется процедура обучения модели с использованием обучающего набора.

4) Проводится оценка модели с использованием проверочного набора.

5) Шаги 2-4 повторяются для различных допустимых архитектур и параметров обучения.

6) Выбирается лучшая модель и проводится её проверка с использованием данных из обучающего и проверочного набора.

7) Проводится оценка окончательной модели с использованием тестового набора.

В результате поиска наилучших параметров нейронной сети был получен вывод, что наилучшее решение представляет собой многослойный персептрон с двумя скрытыми слоями. Первый слой должен состоять из 4 нейронов с логарифмической передаточной функцией. Второй слой состоит из 5 нейронов с логистической передаточной функцией. Передаточная функция выходного слоя должна быть линейной.

Применение нейросетевой модели для датчика влажности позволило улучшить воспроизводимость выходного сигнала и снизило погрешность измерений до ±1% в диапазоне влажности от 10 до 95%.

Литература

1. С. Хайкин. Нейронные сети: полный курс, 2-е издание.: Пер. с англ. М. Издательский дом "Вильямс", 2006. 1104 с.
2. Круглов В.В., Борисов В.В. Искусственные нейронные сети. Теория и практика. М.: Горячая линия-Телеком. 2002. 382с.
3. Бураков М.В. Нейронные сети и нейроконтроллеры. Учебное пособие
4. Kennedy J., Eberhart R. Swarm intelligence. San Francisco, CA: Morgan Kaufmann Publ. Inc., 2001.
5. Kennedy, J., Eberhart R. Particle swarm optimization // Proc. 1995 IEEE International Conference on Neural Networks IEEE Press. P. 1942–1948.
6. Janson S. Middendorf M. A hierarchical particle swarm optimizer and its adaptive variant // IEEE Trans. Systems Man and Cybernetics. Pt B: Cybernetics. December 2005. Vol. 35. P. 1272–1282.
7. Angeline P. J. Using selection to improve particle swarm optimization // IEEE Int. Conference on Evolutionary Computation. May 1998. P. 84–89.
8. A fuzzy adaptive turbulent particle swarm optimisation / Liu et. al. // Int. J. Innovative Computing and Applications. 2007. Vol. 1. N 1. P. 39–47.
9. Oja E. "A simplified neuron model as a principal component analyzer", Journal Of Mathematical Biology, 1982, vol. 15, p. 267-273.
10. Sanger T.D. "Optimal unsupervised learning in a single-layer linear feedforward neural network", Neural Networks, 1989, vol. 12, p. 459-473.
11. Bertrand_Clarke Principles and Theory for Data Mining and Machine Learning 2009-792/
12. Hodgkin A.L., Huxley A.F. A quantitative description of membrane current and its application to conduction and excitation in nerve. // Journal of Physiology, 1952, 117, pp. 500–544.
13. Hodgkin A.L., Huxley A.F. A quantitative description of membrane current and its application to conduction and excitation in nerve. // Bulletin of Mathematical Biology, 1990, Vol. 52, No. 1/2, pp. 25–71.
14. Marmont G., Studies on the axon membrane. I. A new method. // Journal of Cellular Physiology, 1949, 34, pp. 351–382.
15. Borgers C. An introduction to modeling neuronal dynamics. – Medford: Springer International Publishing AG, 2017, – 457 p.
16. FitzHugh R. Mathematical models of threshold phenomena in the nerve membrane. // Bulletin of Mathematical Biophysics, 1955, Vol.17, pp.257–278.
17. FitzHugh R. Impulses and physiological states in theoretical models of nerve membrane. // Biophysical Journal, 1961, Vol.1, pp. 445–466.
18. Webster J.G. Eren H. Measurement, Instrumentation, and Sensors Handbook: Spatial, Mechanical, Thermal, and Radiation Measurement. Taylor & Francis Group, LLC. 2014.
19. Kouda S., Dibi Z., Barra S., Dendouga A., Meddour F. ANN modeling of a smart MEMS-based capacitive humidity sensor. International Journal of Control Automation and Systems. 2011. Vol.9. N.1. P.197-202/

Электронное учебное издание

Виктор Иванович **Капля,**
Егор Викторович **Капля,**

МОДЕЛИ И МЕТОДЫ
ИСКУССТВЕННОГО ИНТЕЛЛЕКТА
Учебное пособие

Электронное издание сетевого распространения

Редактор Матвеева Н.И.

Темплан 2019 г. Поз. № 32.
Подписано к использованию 22.10.2019. Формат 60x84 1/16.
Гарнитура Times. Усл. печ. л. 5,0.

Волгоградский государственный технический университет.
400005, г. Волгоград, пр. Ленина, 28, корп. 1.

ВПИ (филиал) ВолгГТУ.
404121, г. Волжский, ул. Энгельса, 42а.