

Оглавление

Лекция 1. Введение в нейрокомпьютерные системы	10
История и перспективы развития нейрокомпьютеров	10
Символьная парадигма	11
Коннекционистская парадигма	11
Краткая история нейронных сетей	12
Ранние годы	12
Возрождение нейронных сетей	13
Лекция 2. Модели нейронов	16
Элементы нейронных сетей и задача разделения двух классов . .	16
Персептрон	16
Сигмоидальный нейрон	17
Аппроксимация функций	19
Нейрон типа «адалин»	19
Паде-нейрон	20
Нейрон с квадратичным сумматором	20
Сигма-Пи нейроны	21
Модель нейрона Хебба	21
Стохастическая модель нейрона	22
Нейроны типа WTA	23
Кубические модели нейронов	24
Запись активации в замкнутой форме	24
Обучение кубических нейронов	24
Лекция 3. Задача линейного разделения двух классов	26
Линейное разделение классов	26
Алгоритм обучения персептрона по отдельным примерам	26
Обучение по всему заданию	27
Промежуточный вариант: обучение по страницам	28
Геометрическая интерпретация линейного разделения классов .	28
Настройка весового вектора	29
Лекция 4. Задача нелинейного разделения двух классов	31
Метод максимума правдоподобия	31
Нейрофизиологическая аналогия	32
Реализация булевых функций нейронными сетями	32
Выделение выпуклых областей	34
Выделение невыпуклых областей	35

Лекция 5. Виды нейронных сетей и способы организации их функционирования	38
Виды сетей	38
Функционирование сетей	40
Настройка нейронных сетей для решения задач	40
Предобработка данных	41
Интерпретация ответов сети	41
Оценка способности сети решить задачу	42
Константа Липшица сигмоидальной сети	43
Лекция 6. Многослойные сети сигмоидального типа	45
Алгоритм обратного распространения ошибки	45
Одномерная оптимизация	49
Методы инициализации весов	51
Лекция 7. Градиентные алгоритмы обучения сети	53
Универсальный путь обучения	53
Особенности задачи оптимизации, возникающей при обучении нейронных сетей	53
Учет ограничений при обучении	54
Выбор направления минимизации	55
Партан-методы	55
Одношаговый квазиньютоновский метод и сопряженные градиенты	56
Лекция 8. Методы глобальной оптимизации	58
Элементы глобальной оптимизации	58
Алгоритмы имитации отжига	58
Генетические алгоритмы	60
Метод виртуальных частиц	61
Четыре типа устойчивости	63
Лекция 9. Радиальные нейронные сети	65
Математические основы радиальных сетей	66
Радиальная нейронная сеть	68
Лекция 10. Рекуррентные сети как ассоциативные запоминающие устройства	71
Введение	71
Автоассоциативная сеть Хопфилда	72
Обучение сети Хопфилда по правилу Хебба	73
Обучение сети Хопфилда методом проекций	74
Сеть Хемминга	76
Двунаправленная ассоциативная память	78

Лекция 11. Решение задач комбинаторной оптимизации рекуррентными сетями	80
Решение задачи коммивояжера сетью Хопфилда	80
Машина Больцмана	81
Функция консенсуса	82
Максимизация консенсуса	82
Синхронное и асинхронное функционирование машины Больцмана	83
Решение задачи коммивояжера машиной Больцмана	84
Лекция 12. Рекуррентные сети на базе персептрона	86
Введение	86
Персептронная сеть с обратной связью	86
Рекуррентная сеть Эльмана	88
Сеть RTRN	91
Лекция 13. Самоорганизация (самообучение) нейронных сетей	94
Классификация без учителя	94
Метод динамических ядер в классификации без учителя	94
Алгоритмы обучения сетей с самоорганизацией	96
Алгоритм Кохонена	98
Применение сетей с самоорганизацией	99
Компрессия данных	99
Прогнозирование нагрузок энергетической системы	101
Лекция 14. Адаптивная резонансная теория (АРТ)	103
Адаптивная резонансная теория (АРТ)	103
Сеть АРТ-1	103
Архитектура и работа	104
Слой сравнения	104
Слой распознавания	105
Работа сети АРТ	106
Необходимость поиска	108
Положительные качества и недостатки АРТ	109
Лекция 15. Нечеткие и гибридные нейронные сети	111
Интеллектуальные информационные системы в условиях неопределенности и риска	111
Нечеткие множества	112
Лингвистические переменные	112
Нечеткие правила вывода	113
Системы нечеткого вывода Мамдани-Заде	113
Фазификатор	116
Дефазификатор	116

Модель Мамдани-Заде как универсальный аппроксиматор . . .	117
Нечеткие сети TSK (Такаги-Сугено-Канга)	118
Гибридный алгоритм обучения нечетких сетей	119
Мягкая экспертная система	121
Определение мягкой экспертной системы. Сравнение нечеткой и мягкой экспертных систем	121
Представление знаний в мягкой экспертной системе. Содержа- ние баз знаний и данных мягкой экспертной системы . . .	123
Лекция 16. Контрастирование (редукция) нейронной сети	124
Значимость параметров и сигналов.	
Сокращение описания (контрастирование) сетей.	124
Определение значимости параметров на основании функции оценки	124
Определение значимости параметров по изменению выходных сигналов системы	126
Сокращение числа выходов в адаптивном линейном сумматоре (путь «снизу вверх»)	127
Показатели значимости для нейрона с дифференцируемым нели- нейным элементом	128
Показатели значимости для нейрона с пороговым нелинейным элементом (персептрона)	129
Сокращение описания «сверху вниз» — набор достаточного се- мейства наиболее значимых параметров	130
Рекурсивное контрастирование и бинаризация	132
Лекция 17. Методы аппаратной реализации нейрокомпьютеров . .	134
Электронная реализация нейронных сетей	134
Нейрочипы	134
Нейропроцессор NM6403	135
Оптическая реализация нейронных сетей	135
Векторно-матричные умножители	137
Голографические корреляторы	138

Предисловие

Искусственные нейронные сети приобрели широкую популярность благодаря способности сравнительно легко адаптироваться к требованиям различных практических приложений. Они реализуют одну из парадигм искусственного интеллекта, а именно, коннекционистскую. Это означает, что преобразование, выполняемое сетью, определяется значениями весовых коэффициентов и топологией межнейронных соединений. Вместо программирования в традиционных вычислительных системах здесь используется обучение сети, которое сводится к настройке весовых коэффициентов с целью оптимизации заданного критерия качества функционирования сети. Нейронные сети хорошо решают те задачи, которые с трудом поддаются алгоритмизации: распознавание образов, реализация ассоциативной памяти, комбинаторная оптимизация, прогнозирование.

Представление знаний в нейронных сетях в виде матриц весов не позволяет объяснить способ, которым были получены результаты распознавания или прогнозирования. Этот недостаток преодолевается в гибридных системах, представляющих собой нечеткие нейронные сети с генетической настройкой параметров. Такие системы реализуют так называемый «вычислительный интеллект» — новое научное направление, где решаются задачи искусственного интеллекта на основе теории нечетких систем, нейронных сетей и эволюционных (генетических) вычислений.

В книге представлены основные разделы теории искусственных нейронных сетей. Основное внимание уделяется алгоритмам обучения нейронных сетей и решаемым с их помощью задачам. Приводятся детальный обзор и описание важнейших методов обучения сетей различной архитектуры. Рассматриваются методы редукции (упрощения) сетей с целью повышения эффективности их реализации и функционирования. Описываются электронный (на основе СБИС) и оптический подходы к аппаратной реализации нейрокомпьютеров.

Книга предназначена для студентов старших курсов, аспирантов и научных работников, интересующихся методами теории искусственного интеллекта.

Лекция 1. Введение в нейрокомпьютерные системы

Рассматриваются: **символьная и коннекционистская парадигмы искусственного интеллекта, понятие искусственной нейронной сети (НС), история возникновения и перспективы развития НС, отличия НС от традиционных вычислительных систем.**

Ключевые слова: нейрон, нейронные сети, символьная и коннекционистская парадигмы обработки информации

Нейронные сети — новая модель параллельных и распределенных вычислений, один из основных архитектурных принципов построения машин 6-го поколения. В основу искусственных нейросетей положены следующие черты живых нейросетей, позволяющие им справляться с нерегулярными задачами:

- 1) простой обрабатывающий элемент — **нейрон**;
- 2) участие огромного числа нейронов в обработке информации;
- 3) каждый нейрон связан с большим числом других (глобальные связи);
- 4) изменяющиеся по весу связи между нейронами;
- 5) массовый параллелизм обработки информации.

Сети, обладающие этими свойствами, принадлежат к классу **коннекционистских моделей обработки информации**. Основная их черта — использование взвешенных связей между обрабатывающими элементами как средства запоминания информации. Обработка ведется одновременно большим числом элементов, где каждый нейрон связан с большим числом других, поэтому нейронная сеть устойчива к неисправностям и способна к быстрым вычислениям. Задать нейронную сеть для решения конкретной задачи — значит определить:

- 1) модель нейрона;
- 2) топологию связей;
- 3) веса связей.

История и перспективы развития нейрокомпьютеров

На заре вычислительной техники (конец 1940-х—начало 1950-х годов) существовало два подхода к разработке машин с «интеллектуальным» поведением.

Первый из подходов заключался в: 1) представлении знаний в виде множества атомных семантических объектов или символов; 2) манипуляциях с этим множеством символов по формальным алгоритмическим правилам. Эта **символьно-алгоритмическая парадигма** является основной так называемого *традиционного искусственного интеллекта*.

Одновременно с этим существовало другое направление исследований, использующее машины, архитектура которых моделировала мозг животных и обучалась под воздействием окружающей среды, а не программировалась каким-либо языком высокого уровня. Работы по так называемым **нейронным сетям** активно велись в 1960-х годах, затем утратили популярность в 1970-х и начале 1980-х, но во второй половине 1980-х возникла новая волна интереса к ним.

Символьная парадигма

С самого начала было очевидно, что цифровые машины хорошо подходят не только для обработки цифровой информации (собственно вычислений), но и для манипуляции символами, т.к. сами машины ничего не знают о семантике битовых строк в их памяти.

Исследования по искусственному интеллекту имели успех во многих областях, особенно в области экспертных систем. Однако они не оправдали надежд пионеров этого направления на то, что все знания могут быть формализованы и что разум можно рассматривать как устройство, обрабатывающее информацию по определенным правилам. Оказалось, что знания подразделяются на *формализуемые* и *интуитивные*. Пример формализуемых знаний — математика, интуитивных — интуитивные знания эксперта в некоторой конкретной области деятельности.

С коннекционистской точки зрения интуитивные знания не могут быть представлены в виде множества лингвистически формализованных правил, и должны быть привлечены совершенно иные стратегии.

Коннекционистская парадигма

Основная идея состоит в следующем: для того, чтобы реализовать некоторые возможности мозга, необходимо воссоздать его архитектурные особенности. Таким образом, *коннекционистская машина* или *нейронная сеть* является высокосвязной сетью простых процессоров (искусственных нейронов), каждый из которых имеет много входов и много выходов.

В биологических нейронах способность к обработке выражается в электрохимических характеристиках межнейронных соединений (синапсов). В коннекционистских системах она моделируется заданием силы связи или веса каждому входу. Биологические нейроны взаимодействуют

путем передачи электрических импульсов, которые идентичны друг другу, а информация заключена в пространственно-временных связях между ними. Нейроны непрерывно суммируют эффект всех приходящих импульсов. Если результат суммирования возбуждающий, то генерируется выходной импульс, если тормозящий, то выходной импульс не генерируется.

В искусственных нейронных сетях каждый нейрон непрерывно обновляет свое состояние порождением значения активации, которая является функцией входных сигналов и внутренних параметров нейрона. Активация используется для генерации выхода через некоторую сжимающую функцию.

Типичные характеристики искусственных нейронных сетей (НС):

1. Параметры нейрона (веса) настраиваются посредством подачи на входы обучающих векторов и изменением весов таким образом, чтобы нейрон выдавал требуемые выходные сигналы. Таким образом, нейрон является *адаптивной*, а не заранее запрограммированной системой.

2. Работу НС целесообразно представлять как эволюцию динамической системы и описывать системой дифференциальных уравнений.

3. Нейронные сети устойчивы к шумам в сигналах и отказам компонентов (нейронов и синапсов). Отказ компонента не влечет отказа всей НС в целом, а лишь несколько ухудшает ее характеристики.

4. Характерной особенностью работы нейронных сетей является то, что они способны находить статистические закономерности или особенности в обучающей выборке. Это позволяет нейронной сети отнести новый входной объект к одному из уже усвоенных сетью образов либо к новому классу.

5. Типичными приложениями нейронных сетей являются классификация образов и ассоциативная память, восстанавливающая полный образ по частичным данным.

6. Не существует простого соответствия между нейронами и семантическими объектами. Скорее, представление «концепции» или «идеи» в сети осуществляется через вектор активностей нейронов, распределенный по сети, так что каждый нейрон участвует одновременно в представлении многих объектов («идей»).

Краткая история нейронных сетей

Ранние годы

Идеи создания машин, использующих свойства нейронов, возникли одновременно с разработкой первых компьютеров общего назначения. Фактически, сопоставления вычислений и работы мозга были на пе-

реднем плане во многих ранних работах. Например, Джон фон Нейман в первом отчете по машине EDVAC указал на аналогии между используемыми в машине обрабатываемыми элементами и нейронами.

В 1942 году Норберт Винер и его коллеги сформулировали идеи кибернетики, которую определили как науку об управлении и связях в организациях животных и в машинах. Главной была идея рассмотрения биологических процессов с инженерной и математической точек зрения. Наиболее важной считалась идея обратной связи.

В те же годы, когда Винер сформулировал принципы кибернетики, Маккаллок и Питтс опубликовали формальное описание искусственных нейронных сетей. Главная их идея заключалась в том, что любая связь типа «вход-выход» может быть реализована искусственной (формальной) нейронной сетью.

Одной из ключевых черт нейронных сетей является то, что они *способны обучаться*. В 1949 году Дональд Хебб описал механизм работы мозга животного. Согласно этому механизму, синаптические силы (веса) изменяются в соответствии с уровнями активности предсинаптического и постсинаптического нейронов. На языке искусственных нейронов это означает, что вес входа должен увеличиваться, чтобы отражать корреляцию между входом и выходом нейрона.

Следующей вехой было изобретение персептрона Розенблаттом в 1957 году. Наиболее важным результатом его работы является доказательство того, что простая процедура обучения сходится к решению поставленной задачи.

В 1969 году интерес к нейронным сетям снизился в связи с публикацией Минского и Пайперта, указавших на важный класс задач, которые однослойный персептрон решать не может. При этом задача обучения многослойного персептрона в то время еще не была решена: было неясно, какой вклад в ошибку многослойной сети вносит каждый нейрон.

Возрождение нейронных сетей

В 1982 году Джон Хопфилд показал, что высокосвязная сеть нейронов с обратными связями может быть описана как динамическая система, обладающая «энергией». При ассоциативном вызове сеть, стартующая в случайном состоянии, сходится к конечному устойчивому состоянию с минимальной энергией. Новый подход к описанию сетей с обратными связями оказался очень плодотворным.

Подобный прорыв произошел и в связи с многослойными сетями без обратных связей. Для обучения этих сетей был разработан алгоритм обратного распространения ошибки.

Нейронные сети являются нелинейными динамическими системами с коллективными свойствами. Для исследования таких сложных моделей нужна большая вычислительная мощность. Новый интерес к искусственным нейронным сетям обусловлен не только новым математическим подходом, но и существенным прогрессом вычислительной техники.

В модели МакКаллока-Питса нейроны имеют состояния 0, 1 и пороговую логику перехода из состояния в состояние. Каждый нейрон в сети определяет взвешенную сумму состояний всех других нейронов и сравнивает ее с порогом, чтобы определить свое собственное состояние. Розенблатт ввел в модель МакКаллока-Питса способность связей к модификации, что сделало ее обучаемой. Эта модель была названа перцептроном. Новый виток быстрого развития нейронных сетей связан с работами ряда авторов, в особенности с работами Хопфилда, а также с успехами технологии развития СБИС.

Нейрокомпьютеры — устройства, основными компонентами которых являются нейронные сети, применяются в ряде областей:

- для решения задач искусственного интеллекта — распознавания образов, обработки изображений, чтения рукописных символов и т.п.;
- в системах управления и технического контроля;
- для создания спецвычислителей параллельного действия;
- как инструмент изучения человеческого мозга.

Нейронные сети различаются между собой меньше всего моделями нейрона, а в основном топологией связей и правилами определения весов (правилами обучения). По структуре сети делятся на однослойные и многослойные. К однослойным относятся модель Хопфилда и так называемая машина Больцмана. Многослойная сеть имеет входной, выходной и скрытые слои. На входной слой подается информация, с выходного снимается результат обработки, а скрытые слои участвуют в обработке информации.

В отличие от традиционных средств обработки информации, программирование нейронных сетей осуществляется неявно в процессе обучения. Обучение строится следующим образом: существует так называемый задачник, то есть набор примеров с заданными ответами, эти примеры предъявляются системе, нейроны получают условие примера и преобразуют их. Далее нейроны несколько раз обмениваются преобразованными сигналами и, наконец, выдают ответ в виде набора сигналов. Отклонение от правильного ответа штрафуются. Обучение заключается в минимизации штрафа как неявной функции связей.

В традиционных вычислительных системах:

1. Необходимо точное описание алгоритма (ориентация на обработку символов).

2. Данные должны быть точными. Аппаратура легко повреждается. Разрушение основных элементов памяти делает машину неисправной.

3. Каждый обрабатываемый объект явно указан в памяти.

4. Трудно построить хороший алгоритм восприятия образов и ассоциативной выборки (неясно, например, как мы распознаем рукописные символы, конкретного написания которых раньше не видели).

В нейрокомпьютере (нейронной сети):

1. Способ обработки больше похож на обработку сигналов, вместо программы — набор весов нейронов, вместо программирования — обучение нейронов (настройка весов).

2. Нейронная сеть устойчива к шумам, искажения данных не влияют существенно на результат (в том числе выход из строя отдельных нейронов).

3. Обрабатываемые объекты представлены весами нейронов неявно. В результате сеть может работать с объектами, которые ей ранее не встречались, и обобщать результаты обучения.

4. Сети хороши для задач восприятия и ассоциативной выборки.

Лекция 2. Модели нейронов

Рассматриваются структура и функции различных моделей нейрона: перцептрон, сигмоидальный нейрон, адалайн, Паде-нейрон, нейрон с квадратичным сумматором, сигма-пи нейроны, нейрон Хебба, стохастическая модель нейрона, кубические модели нейронов.

Ключевые слова: весовые коэффициенты, функция активации, обучение, градиентные методы оптимизации.

Элементы нейронных сетей и задача разделения двух классов

Перцептрон

Простой перцептрон — это нейрон МакКаллока–Питса (рис. 1). **Весовые коэффициенты** входов сумматора, на которые подаются входные сигналы x_i , $i = 1, \dots, N$ обозначаются w_i , а пороговое значение — w_0 . Нелинейная **функция активации** f перцептрона является ступенчатой, вследствие чего выходной сигнал нейрона может принимать только два значения — 0 и 1 в соответствии с правилом

$$f(u) = \begin{cases} 1 & \text{для } u \geq 0 \\ 0 & \text{для } u < 0 \end{cases} \quad (1)$$

или -1 и 1 в соответствии с правилом

$$f(u) = \begin{cases} 1 & \text{для } u \geq 0 \\ -1 & \text{для } u < 0 \end{cases} \quad (2)$$

где u обозначает выходной сигнал сумматора

$$u = \sum_{i=0}^N w_i x_i. \quad (3)$$

В формуле (3) предполагается $x_0 = 1$.

Обучение перцептрона состоит в таком подборе весов w_i , чтобы выходной сигнал y совпадал с заданным значением $d \in \{0, 1\}$ или $d \in \{-1, 1\}$.

С перцептроном связана *задача четкого разделения двух классов* по обучающей выборке, которая ставится следующим образом: имеется два

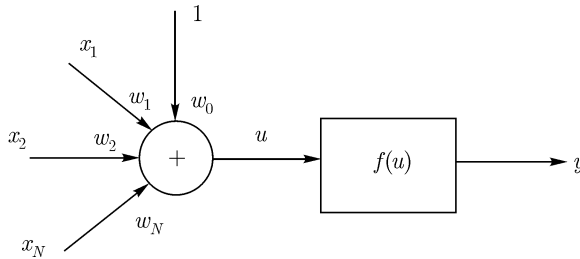


Рис. 1. Нейрон МакКаллока–Питтса

набора векторов X^1, \dots, X^n и Y^1, \dots, Y^m . Заранее известно, что X^i , $i = 1, \dots, n$ относятся к первому классу, а Y^j , $j = 1, \dots, m$ — ко второму. Требуется построить решающее правило, т.е. определить такую функцию $f(X)$, что при $f(X) > 0$ вектор X относится к первому классу, а при $f(X) < 0$ — ко второму.

Сигмоидальный нейрон

Нейрон сигмоидального типа имеет структуру, подобную модели МакКаллока-Питтса, с той разницей, что функция активации является непрерывной и может быть выражена в виде сигмоидальной униполярной или биполярной функции. Униполярная функция, как правило, представляется формулой (рис. 2)

$$f(x) = 1/(1 + \exp(-\beta x)),$$

тогда как биполярная функция задается в виде (рис. 3)

$$f(x) = \tanh(\beta x).$$

Параметр β влияет на крутизну графика функции $f(x)$. При $\beta \rightarrow \infty$ сигмоидальная функция превращается в функцию ступенчатого типа, идентичную функции активации персептрона. На практике чаще всего используется значение $\beta = 1$.

Важным свойством сигмоидальной функции является ее дифференцируемость. Для униполярной функции имеем

$$df(x)/dx = \beta f(x)(1 - f(x))$$

тогда как для биполярной функции

$$df(x)/dx = \beta(1 - f(x))^2.$$

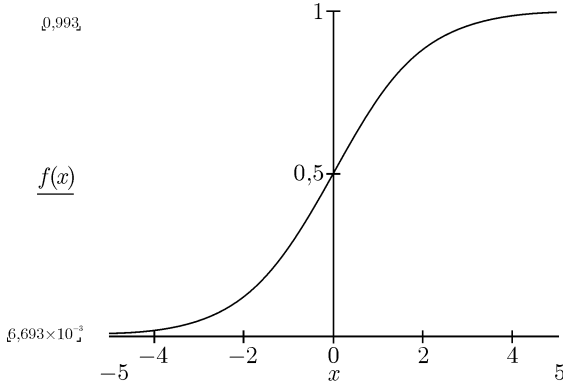


Рис. 2. Униполярная функция ($\beta = 1$)

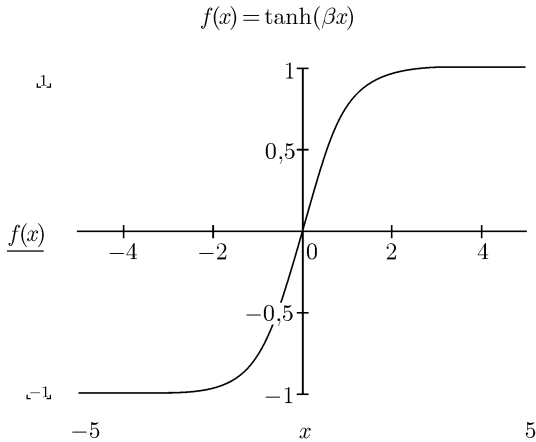


Рис. 3. Биполярная функция ($\beta = 1$)

Применение непрерывной функции активации позволяет использовать при обучении **градиентные методы оптимизации**. Проще всего реализовать метод наискорейшего спуска, в соответствии с которым уточнение вектора весов $w = [w_0, w_1, \dots, w_N]^T$ проводится в направлении отрицательного градиента целевой функции $E = (y - d)^2/2$, где

$$y = f(u) = f\left(\sum_{i=0}^N w_i x_i\right).$$

Компонента градиента имеет вид

$$\nabla_i E = dE/dw_i = ex_i df(u)/du,$$

где $e = y - d$ означает разницу между фактическим и ожидаемым значением выходного сигнала нейрона. Если ввести обозначение $\delta = e \cdot df(u)/du$, то можно получить выражение, определяющее i -ю составляющую градиента в виде

$$\nabla_i E = \delta x_i.$$

Значения весовых коэффициентов уточняются по формуле

$$w_i(t+1) = w_i(t) - \alpha \delta x_i,$$

где $\alpha \in (0, 1)$.

Применение градиентного метода для обучения нейрона гарантирует достижение только локального минимума. Для выхода из окрестности локального минимума результативным может оказаться *обучение с моментом*. В этом методе процесс уточнения весов определяется не только информацией о градиенте функции, но и предыдущим изменением весов. Подобный способ может быть задан выражением

$$\Delta w_i(t+1) = -\alpha \delta x_i + \beta \Delta w_i(t),$$

в котором первый член соответствует обычному методу наискорейшего спуска, тогда как второй член, называемый моментом, отражает последнее изменение весов и не зависит от фактического значения градиента. Значение β выбирается из интервала $(0, 1)$.

Аппроксимация функций

Нейрон типа «адалайн»

В нейроне типа «адалайн» (ADaptive LInear Neuron — адаптивный линейный нейрон) адаптивный подбор весовых коэффициентов осуществляется в процессе минимизации квадратичной ошибки, определяемой как

$$E(w) = e^2/2 = [d - (\sum_{i=0}^N w_i x_i)]^2/2.$$

В связи с выполнением условия дифференцируемости целевой функции стало возможным применение алгоритма градиентного обучения. Значения весовых коэффициентов уточняются следующим способом

$$w_i(t+1) = w_i(t) + \alpha ex_i.$$

Паде-нейрон

Паде-нейрон вычисляет произвольную дробно-линейную функцию вектора x . Так же, как и для адаптивного сумматора, числитель и знаменатель можно сделать линейными функциями x :

$$Ux/Lx, \quad Ux = \sum_{i=0}^N U_i x_i, \quad Lx = \sum_{i=0}^N L_i x_i.$$

Паде-нейрон может использоваться как обобщение нейрона типа «адаплайн» в тех случаях, когда линейных функций становится недостаточно, в частности, в задачах интерполяции эмпирических зависимостей.

В случае Паде-нейрона квадратичная ошибка определяется как

$$E(U, L) = e^2/2 = (d - Ux/Lx)^2/2$$

и значения весовых коэффициентов уточняются по следующим формулам

$$U_i(t+1) = U_i(t) + \alpha e x_i / \sum_{j=0}^N L_j x_j,$$

$$L_i(t+1) = L_i(t) - \alpha e x_i \sum_{j=0}^N U_j x_j / \left(\sum_{j=0}^N L_j x_j \right)^2.$$

Нейрон с квадратичным сумматором

Квадратичный сумматор может вычислять произвольный полином второго порядка от вектора входных сигналов

$$Q(x) = \sum_{i,j} q_{ij} x_i x_j + \sum_i p_i x_i + r.$$

Для многомерных нормальных распределений нейрон с квадратичным сумматором является наилучшим классификатором. Минимум вероятности ошибки дает квадратичная разделяющая поверхность:

если $Q(x) > 0$, то объект принадлежит первому классу;

если $Q(x) \leq 0$, то объект принадлежит второму классу (при условии правильного выбора коэффициентов $Q(x)$).

Квадратичная ошибка здесь определяется как

$$E(q_{ij}, p_i, r) = e^2/2 = (d - Q(x))^2/2.$$

Коэффициенты квадратичного сумматора уточняются по формулам

$$\begin{aligned}q_{ij}(t+1) &= q_{ij}(t) + 2\alpha e x_i x_j, \\ p_i(t+1) &= p_i(t) + \alpha e x_i, \\ r(t+1) &= r(t) + \alpha e.\end{aligned}$$

Недостаток такого классификатора — большое число настраиваемых параметров.

Сигма-Пи нейроны

Выше были рассмотрены нейроны с линейной и квадратичной функциями активации. Сигма-пи нейроны являются их обобщением на случай представления функции активации и полиномом степени N , N — число входов нейрона:

$$U = \sum_{k=1}^M w_k \prod_{i \in I_k} x_i,$$

где I_k — множество индексов, содержащее одну из возможных 2^N комбинаций первых N целых чисел, $M = 2^N$.

Модель нейрона Хебба

Структурная схема нейрона Хебба соответствует стандартной форме модели нейрона (рис. 1). Д. Хебб предложил формальное правило, в котором вес w_i нейрона изменяется пропорционально произведению его входного и выходного сигналов

$$\Delta w_i = \alpha x_i y,$$

где $\alpha \in (0, 1)$ — коэффициент обучения.

При обучении с учителем вместо выходного сигнала y используется ожидаемая от этого нейрона реакция d . В этом случае правило Хебба записывается в виде

$$\Delta w_i = \alpha x_i d,$$

В каждом цикле обучения происходит суммирование текущего значения веса и его приращения Δw_i :

$$w_i(t+1) = w_i(t) + \Delta w_i.$$

В результате применения правила Хебба веса нейрона могут принимать произвольно большие значения. Один из способов стабилизации процесса обучения по правилу Хебба состоит в учете последнего значения w_i , уменьшенного на коэффициент забывания γ . При этом правило Хебба представляется в виде

$$w_i(t+1) = w_i(t)(1 - \gamma) + \Delta w_i.$$

Значение γ выбирается из интервала $(0,1)$ и чаще всего составляет некоторый процент от коэффициента обучения α . Рекомендуемые значения коэффициента забывания — $\gamma < 0.1$, при которых нейрон сохраняет большую часть информации, накопленной в процессе обучения, и получает возможность стабилизировать значения весов на определенном уровне.

Стохастическая модель нейрона

В стохастической модели выходное состояние нейрона зависит не только от взвешенной суммы входных сигналов, но и от некоторой случайной переменной, значения которой выбираются при каждой реализации из интервала $(0,1)$.

В стохастической модели нейрона выходной сигнал y принимает значения ± 1 с вероятностью

$$\begin{aligned} P(y = 1) &= 1/(1 + \exp(-2\beta u)), \\ P(y = -1) &= 1/(1 + \exp(2\beta u)), \end{aligned}$$

где u обозначает взвешенную сумму входных сигналов нейрона, а β — положительная константа, которая чаще всего равна 1. Процесс обучения нейрона в стохастической модели состоит из следующих этапов:

1) расчет взвешенной суммы

$$u = \sum_{i=0}^N w_i x_i$$

для каждого нейрона сети.

2) расчет вероятности P того, что y принимает значение ± 1 .

3) генерация значения случайной переменной $R \in (0,1)$ и формирование выходного сигнала y , если $R < P(y)$, или $-y$ в противном случае.

При обучении с учителем по правилу Видроу-Хоффа адаптация весов проводится по формуле

$$\Delta w_i = \alpha x_i (d - y).$$

Нейроны типа WTA

Нейроны типа WTA (Winner Takes All — «Победитель получает все») имеют входной модуль в виде адаптивного сумматора. Выходной сигнал i -го сумматора определяется по формуле

$$u_i = \sum_{j=0}^N w_{ij} x_j.$$

По результатам сравнения сигналов $u_i, i = 1, 2, \dots, N$ отдельных нейронов победителем признается нейрон, у которого u_i оказался наибольшим. Нейрон-победитель вырабатывает на своем выходе состояние 1, а остальные (проигравшие) нейроны переходят в состояние 0.

Для обучения нейронов WTA учитель не требуется. На начальном этапе случайным образом выбираются весовые коэффициенты w_{ij} каждого нейрона, нормализуемые относительно 1 по формуле

$$w_{ij} \leftarrow w_{ij} / (w_{i1}^2 + w_{i2}^2 + \dots + w_{iN}^2)^{1/2}.$$

После подачи входного вектора x , компоненты которого нормализованы по формуле

$$x_{ij} \leftarrow x_{ij} / (x_{i1}^2 + x_{i2}^2 + \dots + x_{iN}^2)^{1/2},$$

определяется победитель этапа. Победитель переходит в состояние 1, что позволяет произвести уточнение весов его входных линий w_{ij} по правилу

$$w_{ij}(t+1) = w_{ij}(t) + \alpha[x - w_{ij}(t)].$$

Проигравшие нейроны формируют на своих выходах состояние 0, что блокирует процесс уточнения их весовых коэффициентов.

Выходной сигнал i -го нейрона может быть описан векторным отношением

$$u_i = w_i^T x = \|w_i\| \|x\| \cos \varphi_i.$$

Поскольку $\|w_i\| = \|x\| = 1$, значение u_i определяется углом между векторами x и $w_i, u_i = \cos \varphi_i$. Поэтому победителем оказывается нейрон, вектор весов которого оказывается наиболее близким текущему обучающему вектору x . В результате победы нейрона уточняются его весовые коэффициенты, значения которых приближаются к значениям текущего обучающего вектора x .

Следствием конкуренции нейронов становится самоорганизация процесса обучения. Нейроны уточняют свои веса таким образом, что при предъявлении группы близких по значениям входных векторов победителем всегда оказывается один и тот же нейрон. Системы такого типа чаще всего применяются для классификации векторов.

Кубические модели нейронов

Вектор x входных двоичных сигналов рассматривается как адрес ячейки памяти, содержимое которой равно 0 или 1. Для размерности N вектора x существует 2^N возможных адресов.

Можно рассматривать ячейки памяти, как вершины N -мерного гиперкуба. Ячейки памяти получают значения независимо друг от друга. Полезно рассматривать ячейки памяти как содержащие поляризованные двоичные значения ± 1 . Тогда работа кубического модуля описывается следующим образом.

Двоичный вход x используется как адрес памяти, поляризованная двоичная величина считывается и конвертируется в неполяризованную форму функцией h (см. формулу 1). Обозначим значения по адресу x через S_x , так что $y = h(S_x)$. Такие модули мы будем называть кубическими, чтобы подчеркнуть геометрическое представление множества адресов значений активации как множество вершин гиперкуба.

Запись активации в замкнутой форме

Рассмотрим двухвходовый кубический модуль. Существует 4 значения активации $\{S_{00}, S_{01}, S_{10}, S_{11}\}$. Выражение для активации будет иметь следующий вид:

$$u = S_{00}(1 - x_1)(1 - x_2) + S_{01}(1 - x_1)x_2 + S_{10}x_1(1 - x_2) + S_{11}x_1x_2,$$

$x = (x_1, x_2)$ — входной вектор. Такая запись вызвана тем, что только одно из произведений в сумме должно быть ненулевым. Для поляризованных входов x_1 и x_2 активация

$$u = [S_{00}(1 - x_1)(1 - x_2) + S_{01}(1 - x_1)(1 + x_2) + S_{10}(1 + x_1)(1 - x_2) + S_{11}(1 + x_1)(1 + x_2)]/4.$$

В случае N — входного модуля получим

$$u = \left[\sum_x S_x \prod_{i=1}^N (1 + x_i) \right] / 2^N.$$

Обучение кубических нейронов

Кубические нейроны обучаются путем изменения содержимого ячейки их памяти. Обозначим через ‘+’ операцию инкремента-установки содержимого ячейки в +1, через ‘-’ операцию декремента-установки в -1.

Пусть в начальном состоянии все ячейки кубического нейрона установлены в ноль. Обозначим ячейки, адресуемые обучающей выборкой, как центральные ячейки или центры. Ячейки, близкие к центрам в смысле расстояния Хемминга, будем настраивать на те же или близкие к ним значения, что и сами центры, т. е. должна происходить *кластеризация* значений ячейки вокруг центра. Это условие должно выполняться для сети из кубических нейронов. Алгоритм обучения строит так называемое разбиение Вороного, при котором значение в ячейке определяется значением в ближайшем центре, а ячейки, равноудаленные от центров, остаются установленными в ноль. Кубические нейроны допускают большую функциональность, чем полулинейные, и поэтому, возможно, позволяют решать те же задачи при меньшем количестве модулей.

Лекция 3. Задача линейного разделения двух классов

Рассматриваются: решение задачи линейного разделения двух классов методом центров масс, алгоритм обучения персептрона, виды обучения, геометрическая интерпретация задачи разделения двух классов.

Ключевые слова: линейное разделение классов, разделение центров масс, алгоритм обучения персептрона, разделяющая гиперплоскость, обучающая выборка (задачник), обучение с учителем, скорость обучения.

Линейное разделение классов

состоит в построении линейного решающего правила, т.е. такого вектора $w = (w_0, \dots, w_n)$, где w_0 — порог, что при $wx > 0$ вектор x относится к первому классу, а при $wx \leq 0$ — ко второму.

Разделение центров масс — простейший способ построения решающего правила. Суть этого способа заключается в вычислении вектора весов персептрона по следующей формуле

$$w = \left(\sum_{i=1}^n X^i - \sum_{j=1}^m X^j \right) / (n + m),$$

где X^i , $i = 1, \dots, n$ относятся к первому классу, а Y^j , $j = 1, \dots, m$ — ко второму.

Линейные решающие правила, построенные на основании разделения центров масс, могут ошибаться на примерах из обучающей выборки даже в тех случаях, когда существует и безошибочное линейное разделение. Однако метод центров масс полезен как средство определения начального значения вектора весов для алгоритма обучения персептрона.

Алгоритм обучения персептрона по отдельным примерам

1. При изначально заданных значениях весов w_i на вход нейрона подается обучающий вектор x и рассчитывается значение выходного сигнала y . По результатам сравнения y с d уточняются значения весов.

2. Если $y = d$, то w_i , $i = 1, \dots, N$ не изменяются.

3. Если $y = 0$, а $d = 1$, то значения весов уточняются по формуле

$$w_i(t+1) = w_i(t) + \alpha x_i, \alpha \in (0, 1),$$

где α — коэффициент обучения, t — номер предыдущего цикла.

4. Если $y = 1$, а $d = 0$, то значения весов уточняются по формуле

$$w_i(t+1) = w_i(t) - \alpha x_i.$$

В обобщенной форме обучение персептрона на векторе x выражается формулой

$$w_i(t+1) = w_i(t) + \alpha(d - y)x_i, i = 1, \dots, N.$$

По завершении уточнения весовых коэффициентов представляются очередной обучающий вектор x и связанное с ним ожидаемое значение d , и значения весов уточняются заново. Этот процесс многократно повторяется на всей обучающей выборке, пока не будут ликвидированы различия между всеми значениями y и соответствующими им ожидаемыми значениями d .

Обучение по всему задачику

Построим обучающую выборку

$$(V^1, \dots, V^n, V^{n+1}, \dots, V^{n+m}) = (X^1, \dots, X^n, -Y^1, \dots, -Y^m).$$

В обучающей выборке выделяются все $V_i, i \in (1, \dots, n, n+1, \dots, n+m)$, для которых не выполняется неравенство $(V^i, w) > 0$, где w — вектор весовых коэффициентов нейрона. Обозначим это множество через Err . Вектор w модифицируется только после проверки всей обучающей выборки:

$$w = w + \alpha \sum_{V^i \in Err} V^i.$$

Не требуется хранить все множество Err — достаточно накапливать сумму тех V^i , на которых персептрон ошибается:

$$\Delta w = w + \alpha \sum_{V^i \in Err} V^i.$$

Как показывают испытания, обучение по всему задачику, как правило, сходится быстрее, чем обучение по отдельным примерам.

Промежуточный вариант: обучение по страницам

Обучающее множество разбивается на подмножества (страницы) и задается последовательность прохождения страниц: столько-то циклов по первой странице, потом столько-то по второй и т. д. Коррекция вектора w проводится после прохождения страницы. Задачник разбивается на страницы по различным эвристическим правилам, например, по правилу «от простого к сложному». Как показывает практика, чаще всего наилучшим является обучение по всему задачнику, иногда (при большом задачнике) — обучение по страницам, размеры которых определяются объемом доступной оперативной памяти.

Геометрическая интерпретация линейного разделения классов

Пусть в нейроне в качестве функции активации используется ступенчатая функция (см. формулу (1) Лекции 2). Линейное разделяющее правило делит входное пространство на две части гиперплоскостью, классифицируя входные векторы как относящиеся к 1-му классу (выходной сигнал — 1) или 2-му классу (выходной сигнал — 0). Критическое условие классификации (уравнение **разделяющей гиперплоскости**)

$$(w, x) = \sum_{i=0}^N w_i x_i = 0$$

В N -мерном пространстве (пространстве входных сигналов) разделяющая гиперплоскость перпендикулярна вектору $w' = (w_1, \dots, w_N)$. Вектор входных сигналов $x' = (x_1, \dots, x_N)$ дает выход 1, если его проекция $x'_w = (x', w')/||w'||$ на вектор w' больше, чем расстояние $-w_0/||w'||$ от нуля до гиперплоскости. В $N + 1$ -мерном (расширенном) пространстве гиперплоскость, описываемая уравнением $(w, x) = 0$, ортогональна вектору w и проходит через начало координат пространства признаков (образов).

Пример.

В двумерном пространстве входных сигналов уравнение гиперплоскости имеет вид

$$w_0 + w_1 x_1 + w_2 x_2 = 0.$$

При $w_1 = w_2 = 1$ и $w_0 = -1.5$ получаем уравнение $x_1 + x_2 - 1,5 = 0$ гиперплоскости, которая представлена на рис. 1 пунктирной линией, пересекающей оси координат в точках $(1.5, 0)$ и $(0, 1.5)$ соответственно. Здесь: $w = (1, 1)$ — нормаль к разделяющей гиперплоскости; P — вектор, относящийся к первому классу, поскольку проекция (w, P) вектора P на нормаль w больше $-w_0/||w||$; Q — вектор, относящийся ко второму классу, поскольку $(w, Q) < -w_0/||w||$.

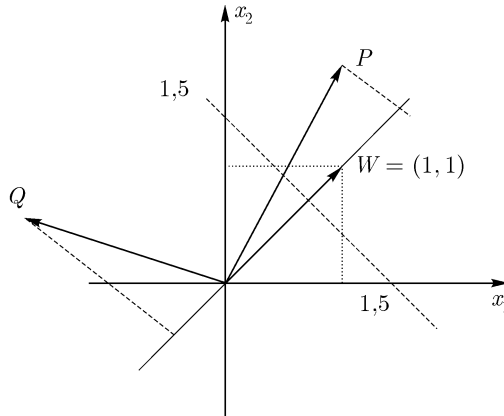


Рис. 1

Настройка весового вектора

Мы требуем, чтобы вектор весов в расширенном пространстве был ортогонален решающей гиперплоскости, и плоскость проходила через начало координат. Обучающую выборку (задачник) для нейрона можно рассматривать как множество пар (V, d) , где V — входной вектор, d — класс (выход, принимающий одно из двух значений, например, 0 или 1), которому принадлежит V . Такой тип обучения называется **обучением с учителем**, т. к. мы сообщаем сети, каким должен быть выходной сигнал для каждого вектора входных сигналов.

Пусть для некоторого V выполняется $d = 1$, но выход сети

$$y = f[(V, W)] = 0,$$

где $f(u) = 1$ при $u > 0$, и $f(u) = 0$ при $u < 0$, т.е. $(V, W) < 0$ (угол φ на рис. 2 между векторами V и W больше $\pi/2$). Чтобы исправить ситуацию, нужно повернуть вектор весов W , приближая его направление к направлению вектора V . В то же время изменение не должно быть слишком резким, чтобы не испортить уже выполненное обучение. Мы достигнем обеих целей, если добавим к вектору W часть вектора V , чтобы получить новый вектор

$$W' = W + \alpha V, \quad 0 < \alpha < 1.$$

Предположим теперь, что $d = 0$, а $y = 1$ (угол φ на рис. 2 между векторами V и W меньше $\pi/2$). Теперь нужно увеличить угол между W и V , что получается путем вычитания части V из W :

$$W' = W - \alpha V.$$

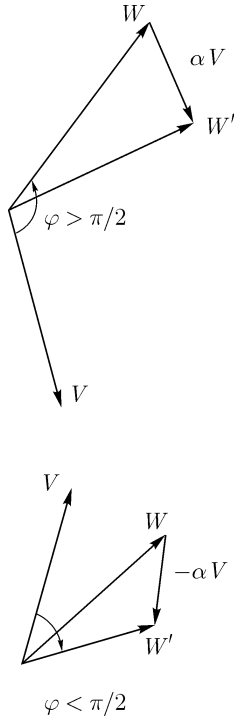


Рис. 2. Настройка вектора весов

Резльтирующая запись имеет вид:

$$W' = W + \alpha(d - y)V.$$

Параметр α называется **скоростью обучения**.

Алгоритм обучения нейрона (персептрона) будет иметь вид:

```

repeat
for  $\forall(V, d)$ 
begin
 $y = h[(W, V)];$ 
if  $y \neq d$  then  $W' = W + \alpha(d - y)V;$ 
end
until  $(y = d \text{ for } \forall(V, d))$ 
    
```

Лекция 4. Задача нелинейного разделения двух классов

Рассматриваются: решение задачи нелинейного разделения двух классов методом максимума правдоподобия и многослойной сетью, реализация булевых функций посредством нейронной сети.

Ключевые слова: формула Байеса, гиперплоскость, булевы функции И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ, выпуклые и невыпуклые области, двухслойные и трехслойные сети.

Метод максимума правдоподобия

Рассмотрим задачу разделения двух классов, с каждым из которых связано вероятностное распределение в пространстве векторов x значений признаков. Будем обозначать плотности этих распределений $P(x|C_i)$, $i = 1, 2$, C_i — событие, состоящее в том, что объект принадлежит i -му классу. Нас интересует апостериорная вероятность: $P(C_i|x)$ — вероятность принадлежности объекта к i -му классу при условии, что он характеризуется вектором признаков x . Известная из теории вероятности **формула Байеса** дает

$$P(C_i|x) = P(C_i)P(x|C_i) / \sum_j P(C_j)P(x|C_j)$$

где $P(C_i)$ — вероятность появления объектов i -го класса. Для нормальных k -мерных распределений

$$P(x|C_i) = 1/\{(2\pi)^{k/2}(\det \sum^i)^{1/2} \exp[-\frac{1}{2}(x - M^i), (\sum^i)^{-1}(x - M^i)]\},$$

где M^i — математическое ожидание x в i -м классе, \sum^i — ковариационная матрица для i -го класса. В результате обработки данных находят статистические оценки \sum^i и M^i : пусть для i -го класса имеются векторы x^1, \dots, x^r , тогда полагаем

$$M^i = (\sum_{j=1}^r x^j)/r, (\sum^i)_{pq} = \frac{1}{r} \sum_{j=1}^r (x_p^j - M_p^i)(x_q^j - M_q^i).$$

Минимизация в формуле Байеса дает простое решающее правило: x принадлежит i -му классу, если $P(C_i|x) > P(C_j|x)$ для всех $j \neq i$, т.е. выбирается такой класс, для которого вероятность $P(C_i|x)$ максимальна.

Поскольку в формуле Байеса для всех C_i знаменатель общий, то решающее правило приобретает следующий вид: выбираем то i , для которого $P(C_i)P(x|C_i)$ максимально. Для нормального распределения удобно прологарифмировать эту величину. Окончательно получаем:

x принадлежит i -му классу, если среди величин

$$P_j = \ln P(C_j) - (\ln \det \sum^j)/2 - [(x - M^j), (\sum^j)^{-1}(x - M^j)]/2$$

величина P_i — максимальная. Таким образом, разделяющей является поверхность второго порядка, а операцию деления на два класса выполняет квадратичный адаптивный сумматор в комбинации с пороговым нелинейным элементом. Пороговый элемент вычисляет ступенчатую функцию $f(P_1 - P_2)$, в результате для первого класса получим ответ 1, для второго — 0.

Нейрофизиологическая аналогия

Идея использования НС с квадратичными сумматорами для улучшения способности сети к обобщению базируется на хорошо известном факте индукции в естественных НС, когда возбуждение в одних областях мозга влияет на возбуждение в других. Простейшей формализацией этого является введение коэффициента, пропорционального сигналу от j -го нейрона, в величину веса i -го сигнала k -го нейрона. Снабдив такое произведение весом q_{ij} — «коэффициентом индукции», получим рассматриваемую архитектуру

$$y = f[Q(x) + L(x) + P],$$

где $Q(x)$ и $L(x)$ — соответственно квадратичная и линейная функция, $P = \text{const}$, f — функция активации нейрона. Коэффициенты функций Q, L и константа P являются подстроечными параметрами, определяющимися в ходе обучения.

Реализация булевых функций нейронными сетями

Простой персептрон (нейрон МакКаллока-Питса) с весовым вектором $w = (-0.5, 1, 1)$ реализует **гиперплоскость**

$$x_1 + x_2 = 0.5$$

и булеву функцию ИЛИ от двух аргументов x_1 и x_2 , каждый из которых может быть нулем или единицей. При $w = (-1.5, 1, 1)$ персептрон реализует гиперплоскость

$$x_1 + x_2 = 1.5$$

и булеву функцию И. Однако, персептрон не может воспроизвести даже такую простую функцию как ИСКЛЮЧАЮЩЕЕ ИЛИ. Она принимает значение единицы, когда один из аргументов равен единице (но не оба) (табл. 1).

Таблица 1. Булева функция ИСКЛЮЧАЮЩЕЕ ИЛИ

x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Эту функцию реализует **двухслойная нейронная сеть**, представленная на рис. 1 (сигнал $x_0 = 1$ не указан). Первый слой такой сети состоит из двух нейронов, каждый из которых реализует разделяющую гиперплоскость в двумерном пространстве входных данных. Первая гиперплоскость описывается уравнением

$$x_1 + x_2 = 0.5,$$

а вторая — уравнением

$$-x_1 - x_2 = -1.5.$$

Соответствующие векторы весов имеют вид $w^1 = (-0.5, 1, 1)$ и $w^2 = (1.5, -1, -1)$. Нейрон во втором слое реализует функцию И от двух выходных сигналов нейронов первого слоя.

$$x_1 + x_2 = 0,5$$

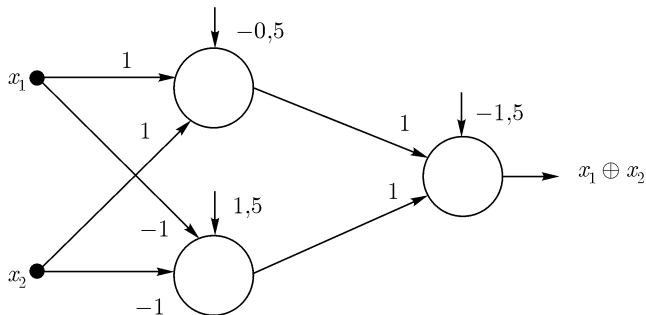


Рис. 1. Двухслойная сеть, реализующая функцию ИСКЛЮЧАЮЩЕЕ ИЛИ

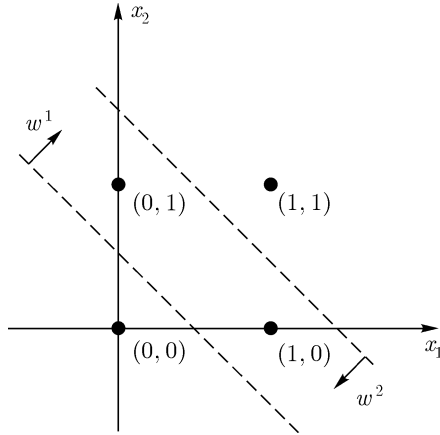


Рис. 2. Гиперплоскости, реализующие функцию ИСКЛЮЧАЮЩЕЕ ИЛИ

Выходным сигналом сети будет 1, если входные сигналы сети соответствуют точкам пространства входных сигналов, расположенным между вышеуказанными гиперплоскостями, т. е. точкам $(0, 1)$ и $(1, 0)$ (рис. 2).

Выделение выпуклых областей

Серьезное ограничение разделяющих поверхностей однослойными сетями можно преодолеть, добавив дополнительные слои. Например, двухслойные сети, получаемые каскадным соединением однослойных сетей, способны выполнять более общие классификации, отделяя точки, содержащиеся в выпуклых ограниченных и неограниченных областях. Область выпуклая, если для каждой двух её точек соединяющий их отрезок целиком лежит в области. Область ограничена, если её можно заключить в некоторый шар.

Выше приведен пример выделения **выпуклой области** двумя гиперплоскостями (реализация функции ИСКЛЮЧАЮЩЕЕ ИЛИ). Аналогично в первом слое может быть использовано 3 нейрона с дальнейшим разбиением плоскости и созданием области треугольной формы (на рис. 3, 4, $w^1 = (0, 1, 0)$, $w^2 = (0, 0, 1)$, $w^3 = (-1, -1, -1)$, входы с нулевыми весами не указаны).

Включением достаточного числа нейронов во входной слой может быть образован выпуклый многоугольник (многогранник) желаемой формы. Так как такие многогранники образованы с помощью операций И над областями, задаваемыми разделяющими линиями (гиперплоскостями единичной размерности), то все они выпуклы.

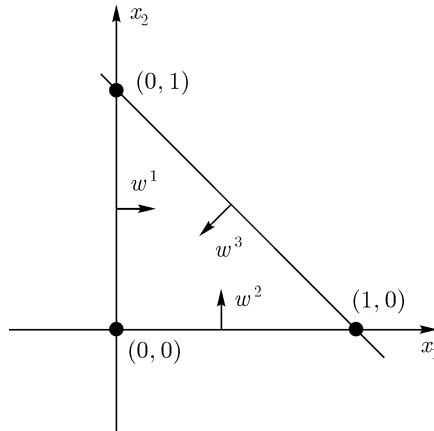


Рис. 3. Гиперплоскости, выделяющие на плоскости выпуклую (треугольную) область

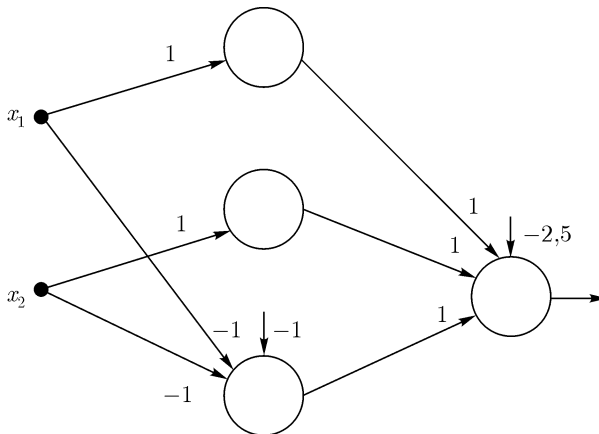


Рис. 4. Нейронная сеть, выделяющая на плоскости выпуклую (треугольную) область

Выделение невыпуклых областей

Точки, не составляющие выпуклой области, не могут быть отделены от других точек пространства двухслойной сетью. Нейрон второго слоя не ограничен функцией И. **Трехслойная сеть** является более общей. Ограничения на выпуклость отсутствуют. Нейрон третьего слоя принимает в

качестве входа набор выпуклых многоугольников, и их логическая комбинация может быть **невыпуклой**.

При добавлении нейронов число сторон многоугольников может неограниченно возрастать. Это позволяет аппроксимировать область любой формы с любой точностью. Вдобавок не все выходные области второго слоя должны пересекаться. Следовательно, возможно объединить различные области, выпуклые и невыпуклые, выдавая на выходе 1 всякий раз, когда входной вектор принадлежит одной из них.

На рис. 5 приведен пример выделения невыпуклой области, представленной в виде объединения двух треугольных областей. Пять нейронов первого слоя реализуют разделяющие гиперплоскости, два нейрона второго слоя реализуют трехходовые функции И, нейрон третьего слоя реализует функцию ИЛИ. Весовые векторы, описывающие соответствующие гиперплоскости, имеют вид:

$$w^1 = (0, 1, 0), \quad w^2 = (-1, -1, 1), \quad w^3 = (-1, -1, -1), \\ w^4 = (-1, 1, -1), \quad w^5 = (0, 0, 1).$$

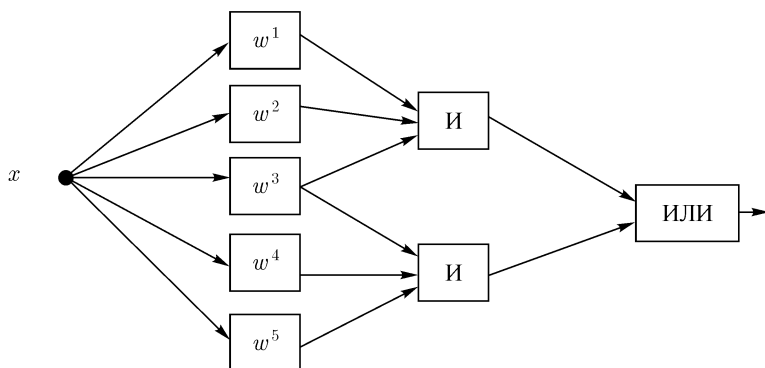
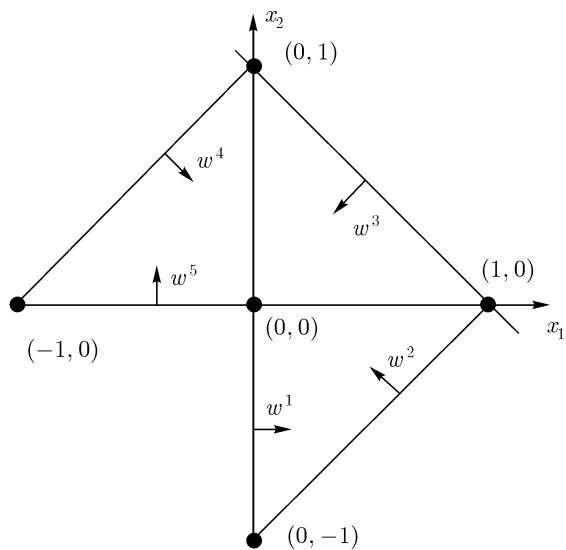


Рис. 5. Пример выделения невыпуклой области

Лекция 5. Виды нейронных сетей и способы организации их функционирования

Рассматриваются: варианты многослойных сетей, режимы функционирования сетей, предобработка входных данных сети, виды интерпретации ответов НС, задача аппроксимации функций и оценка способности сети решить эту задачу.

Ключевые слова: многослойные (слоистые) сети, монотонные слоистые сети, полносвязная сеть, слоисто-циклические сети, слоисто-полносвязные сети, поносвязно-слоистые сети, сети периодического функционирования, сети непрерывного функционирования, сети с циклами, предобработка, интерпретация, масштабирование, правило интерпретации «победитель забирает все», знаковая интерпретация, порядковая интерпретация, константа Липшица сети.

Виды сетей

В многослойных (слоистых) сетях (рис.1) нейроны первого слоя получают входные сигналы, преобразуют их и передают нейронам второго слоя. Далее срабатывает второй слой и т. д., до n -ого, который выдает выходные сигналы для интерпретатора и пользователя. Если не оговорено противное, то каждый выходной сигнал i -го слоя подается на вход всех нейронов $i + 1$ -го. Число нейронов в каждом слое может быть любым и никак заранее не связано с количеством нейронов в других слоях. Стандартный способ подачи входных сигналов: все нейроны первого слоя получают каждый входной сигнал. Наибольшее распространение получили трехслойные сети, в которых каждый слой имеет свое наименование: первый — входной, второй — скрытый, третий — выходной.

Монотонные слоистые сети — частный случай слоистых сетей с дополнительными условиями на связи и элементы. Каждый слой, кроме выходного, разбит на два блока — возбуждающий и тормозящий. Связи между слоями также подразделяются на два типа — возбуждающие (с положительными весами) и тормозящие (с отрицательными весами). Если от блока A к блоку C ведут только возбуждающие связи, то это означает, что любой выходной сигнал блока C является монотонной неубывающей функцией любого выходного сигнала блока A . Если же эти связи только тормозящие, то любой выходной сигнал блока C является монотонной невозрастающей функцией.

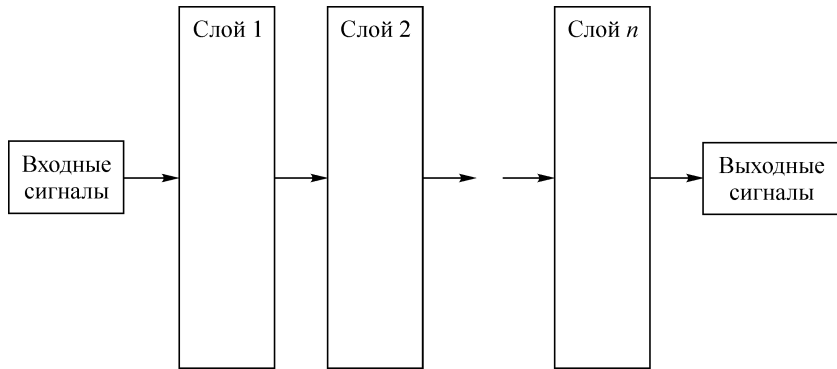


Рис. 1. Многослойная (слоистая) сеть

В полносвязной сети каждый нейрон передает свой выходной сигнал остальным нейронам, в том числе и самому себе. Выходными сигналами сети могут быть все или некоторые выходные сигналы нейронов после нескольких циклов функционирования сети. Все входные сигналы подаются всем нейронам. Для полносвязной сети входной сумматор нейрона фактически распадается на два: первый вычисляет линейную функцию от входных сигналов сети, второй - линейную функцию от выходных сигналов других нейронов, полученных на предыдущем шаге. Примером полносвязной сети является сеть Хопфилда.

Слоисто-циклические (рекуррентные) сети отличаются тем, что слои замкнуты в кольцо — последний передает свои выходные сигналы первому. Все слои равноправны и могут как получать входные сигналы, так и выдавать выходные. Такие сети до получения ответа могут функционировать неограниченно долго, так же, как и полносвязные.

Слоисто-полносвязные сети состоят из слоев, каждый из которых, в свою очередь, представляет собой полносвязную сеть. При функционировании сигналы передаются от слоя к слою, и происходит обмен сигналами внутри слоя. В каждом слое процесс протекает следующим образом: прием сигналов с предыдущего слоя (или входных сигналов сети), обмен сигналами внутри слоя, передача последующему слою (или на выход). Подобные сети до получения ответа функционируют определенное число тактов, соответствующее количеству слоев, так же, как и слоистые сети.

Полносвязно-слоистые сети по структуре такие же, как и предыдущие, но функционируют по-другому. В них не разделяются фазы обмена внутри слоя и передачи следующему: на каждом такте нейроны всех

слоев принимают сигналы от нейронов как своего, так и предыдущего, после чего передает сигналы как внутри слоя, так и последующему (или на выход). До получения ответа подобные сети могут функционировать неограниченно долго, так же, как и полносвязные.

Функционирование сетей

Сети периодического функционирования. Простейшие представления об этих сетях таковы. В начальный момент состояния всех нейронов одинаковы, выходных сигналов нет. Подаются входные сигналы, определяющие активность сети (нулевой такт). Далее входные сигналы могут подаваться на каждом такте функционирования. На каждом такте могут сниматься выходные сигналы. После k тактов цикл функционирования заканчивается, и сеть возвращается в исходное состояние, готовая к новому циклу (акту). Между актами функционирования могут вставляться акты обучения. В общем случае, в результате цикла из k тактов нейронная сеть выдает в ответ на последовательность из k наборов входных сигналов последовательность k наборов выходных сигналов. Чаще используется упрощенный вариант: входные сигналы подаются только в самом начале, выходные снимаются в самом конце.

Для слоистых и слоисто-полносвязных сетей начальные слои по мере срабатывания освобождаются и могут заниматься новой задачей, пока последние слои заканчивают работу над предыдущей. Сети периодического функционирования по характеру использования напоминают ЭВМ: на вопрос следует ответ, причем воспроизводимый. Иначе обстоит дело с **сетями непрерывного функционирования**.

Непрерывное функционирование нейронной сети более соответствует имеющимся представлениям о поведении живых существ, чем периодическое. Опыт показывает, что, чередуя циклы функционирования и обучения, для таких сетей можно получить хорошие результаты адаптации. Для непрерывного функционирования необходимы сети с циклами: полносвязные, слоисто-циклические или полносвязно-слоистые.

Настройка нейронных сетей для решения задач

Тема данного раздела — формирование нейронных сетей для решения задач. Прежде чем приступить к поиску параметров сети, нужно поставить задачу, т. е. ответить на вопросы:

1. Какие сигналы сеть будет получать?
2. Как мы будем интерпретировать сигналы, поступающие от сети?
3. Как мы будем оценивать работу сети, если сеть обучается путем минимизации ошибок (т. е. что такое вектор ошибок и как вычисляется целевая функция — оценка функционирования сети)?

Ответы на данные вопросы воплощаются в спецустройствах или программах: в преобразователе, интерпретаторе ответов, оценке.

Итак, прежде чем формировать сеть, необходимо создать её окружение. В процессе обучения, кроме того, используются:

1. Обучающая выборка (система, работающая с исходными данными);
2. Учитель, модифицирующий параметры сети;
3. Контрастер (система, упрощающая нейронную сеть).

Преобработка данных

Нормировка и центрирование данных (**преобработка**) используются почти всегда (кроме тех случаев, когда данные представляют собой бинарные векторы с координатами 0,1 или ± 1 , либо символьные последовательности). Цель этих преобразований — сделать так, чтобы каждая компонента вектора данных лежала в отрезке $[-1, 1]$ (или $[0, 1]$) или, по крайней мере, не слишком далеко выходила из этого отрезка, и её характерный разброс тоже был бы единичным.

Стандартные преобразования исходной выборки $x^p, p = 1, 2, \dots, M$:
 $x_i^p = [x_i^p - M(x_i^p)]/\sigma(x_i^p)$ или $x_i^p = [x_i^p - M(x_i^p)]/\max |x_i^p - M(x_i^p)|$,
 где x_i^p — i -я компонента вектора x^p ,

$M(x_i^p) = (\sum x_i^p)/n$ — выборочная оценка математического ожидания x_i^p ;

$\sigma(x_i^p) = \{\sum [x_i^p - M(x_i^p)]^2/n\}^{1/2}$ — выборочная оценка среднего квадратичного отклонения. Любое изменение выборки $\{x^p\}$ должно, согласно этим формулам, менять и нормировку. Нормировка и центрирование вписывают исходную выборку в куб со стороной 2, вершинами которого являются векторы с координатами ± 1 .

Интерпретация ответов сети

При **интерпретации** выходных сигналов сети необходимы аккуратность и порой изобретательность, ведь от этого истолкования зависят требования, которые мы предъявляем к работе НС. Удачная их формулировка может упростить обучение и повысить точность работы, неудачная — свести на нет предыдущие усилия.

Масштабирование является естественной операцией при обработке выходных сигналов. Стандартные (обезразмеренные) НС формируются так, чтобы их выходные сигналы лежали в интервалах $[-1, 1]$ (или $[0, 1]$). Если нам нужно получить сигнал в интервале $[a, b]$, то нужно преобразовать выходной сигнал $y \in [-1, 1]$:

$$y = (a + b)/2 + (b - a)y/2.$$

В задачах классификации наиболее распространено **правило интерпретации «победитель забирает все»**: число нейронов равно числу классов, номер нейрона с максимальным сигналом интерпретируется как номер класса. К сожалению, если классов много, то этот наглядный метод является слишком расточительным, потребляет слишком много выходных нейронов.

Знаковая интерпретация требует только $k = \log_2 m$ нейронов (m — число классов). Строится она так. Пусть y_1, \dots, y_k — совокупность выходных сигналов нейронов. Заменяем в этой последовательности положительные числа единицами, а отрицательные — нулями. Полученную последовательность нулей и единиц рассматриваем как номер класса в двоичной записи.

Порядковая интерпретация является еще более емкой, чем знаковая. В ней с помощью k нейронов можно описать принадлежность к $k!$ классам (а не 2^k как для знаковой). Пусть y_1, \dots, y_k — выходные сигналы. Проведем их сортировку и обозначим через n_i номер i -го сигнала после сортировки (1 соответствует наименьшему сигналу, k — наибольшему). Перестановку $\sigma = (n_1, n_2, \dots, n_k)$ рассмотрим как слово, кодирующее номер класса. Всего возможно $k!$ перестановок. Этим интерпретатором можно пользоваться, если характерная ошибка выходного сигнала меньше $1/k$. Даже при $k = 10$ получаем реализуемые требования к точности ($< 1/10$) и богатые возможности ($10!$ классов).

Оценка способности сети решить задачу

В данном разделе рассматриваются только сети, все элементы которых непрерывно зависят от своих аргументов. Предполагается, что все входные данные преобразованы так, чтобы все входные и выходные сигналы сети лежали в диапазоне приемлемых входных сигналов $[a, b]$.

Нейронная сеть вычисляет некоторую вектор-функцию F от входных сигналов. Эта функция зависит от параметров сети. Обучение сети состоит в подборе такого набора параметров сети, чтобы величина

$$\sum_{i,p} [F_i(x^p) - f_i^p]^2$$

была минимальной (в идеале равна нулю), здесь $\{f_i\}$ — множество аппроксимируемых функций. Для того, чтобы нейронная сеть могла хорошо приблизить заданную таблично функцию f , необходимо, чтобы реализуемая сетью функция F при изменении входных сигналов с x^i на x^j могла изменить значение с f^i на f^j . Очевидно, что наиболее трудным для сети должно быть приближение функции в точках, в которых при малом изменении входных сигналов происходит большое изменение зна-

чения функции. Таким образом, наибольшую сложность будет представлять приближение функции f в точках, в которых достигает максимума выражение $\|f^i - f^j\|/\|x^i - x^j\|$. Для аналитически заданных функций величина

$$\sup_{x,y} (\|f(x) - f(y)\|/\|x - y\|)$$

называется **константой Липшица**. Исходя из этих соображений, можно дать следующее определение сложности задачи.

Сложность аппроксимации таблично заданной функции f , которая в точках x^i принимает значения f^i , задается выборочной оценкой константы Липшица, вычисляемой по формуле:

$$\Lambda_t = \max_{i \neq j} (\|f(x^i) - f(x^j)\|/\|x^i - x^j\|) \quad (1)$$

Оценка (1) является оценкой константы Липшица аппроксимируемой функции снизу.

Константа Липшица сети вычисляется по следующей формуле:

$$\Lambda_n = \sup_{x,y} (\|F(x) - F(y)\|/\|x - y\|)$$

Для того, чтобы оценить способность сети заданной конфигурации решить задачу, необходимо оценить константу Липшица сети и сравнить ее с выборочной оценкой (1). В случае $\Lambda_n < \Lambda_t$ сеть принципиально не способна решить задачу аппроксимации функции f . Однако из $\Lambda_n \geq \Lambda_t$ еще не следует утверждение о способности сети аппроксимировать функцию f !

Константа Липшица сигмоидальной сети

Рассмотрим слоистую сигмоидальную сеть (сеть с сигмоидальными нейронами) со следующими свойствами:

1. Число входных сигналов — n_0 .
2. Число нейронов в i -м слое — n_i .
3. Каждый нейрон первого слоя получает все входные сигналы, а каждый нейрон любого другого слоя получает сигналы всех нейронов предыдущего слоя.
4. Все нейроны всех слоев имеют одинаковые функции активации $f(x) = 1/(1 + e^{-\beta x})$.
5. Все синаптические веса ограничены по модулю единицей.
6. В сети m слоев.

В этом случае оценка константы Липшица сети равна:

$$\Lambda_n \leq \beta^m (n_0 n_m)^{1/2} \prod_{i=1}^{m-1} n_i \quad (2)$$

Формула (2) подтверждает экспериментально установленный факт, что, чем круче характеристическая функция нейрона (т. е. чем больше β), тем более сложные функции (функции с большей константой Липшица) может аппроксимировать сеть с такими нейронами.

Лекция 6. Многослойные сети сигмоидального типа

Рассматриваются: многослойный перцептрон, алгоритм обратного распространения ошибки, подбор коэффициента обучения (одномерная минимизация), методы инициализации весов сети.

Ключевые слова: обратное распространение ошибки, градиент целевой функции, метод наискорейшего спуска, одномерная оптимизация, инициализация весов.

Алгоритм обратного распространения ошибки

Возьмем двухслойную сеть (рис. 1) (входной слой не рассматривается). Веса нейронов первого (скрытого) слоя пометим верхним индексом (1), а выходного слоя — верхним индексом (2). Выходные сигналы скрытого слоя обозначим $v_j, j = 1, 2, \dots, K$, а выходного слоя — $y_j, j = 1, 2, \dots, M$. Будем считать, что функция активации нейронов задана в сигмоидальной униполярной или биполярной форме. Для упрощения описания будем использовать расширенное обозначение входного вектора сети в виде $x = [x_0, x_1, \dots, x_N]^T$, где $x_0 = 1$ соответствует порогу. С вектором x связаны два выходных вектора сети: вектор фактических выходных сигналов $y = [y_0, y_1, \dots, y_M]^T$ и вектор ожидаемых выходных сигналов $d = [d_0, d_1, \dots, d_M]^T$.

Цель обучения состоит в подборе таких значений весов $w_{ij}^{(1)}$ и $w_{ij}^{(2)}$ для всех слоев сети, чтобы при заданном входном векторе x получить на выходе значения сигналов y_i , которые с требуемой точностью будут совпадать с ожидаемыми значениями d_i для $i = 1, 2, \dots, M$. Выходной сигнал i -го нейрона скрытого слоя описывается функцией

$$v_i = f\left(\sum_{j=0}^N w_{ij}^{(1)} x_j\right).$$

В выходном слое k -й нейрон вырабатывает выходной сигнал

$$y_k = f\left(\sum_{i=0}^K w_{ki}^{(2)} v_i\right) = f\left(\sum_{i=0}^K w_{ki}^{(2)} f\left(\sum_{j=0}^N w_{ij}^{(1)} x_j\right)\right)$$

Из формулы следует, что на значение выходного сигнала влияют веса обоих слоев, тогда как сигналы, вырабатываемые в скрытом слое, не зависят от весов выходного слоя.

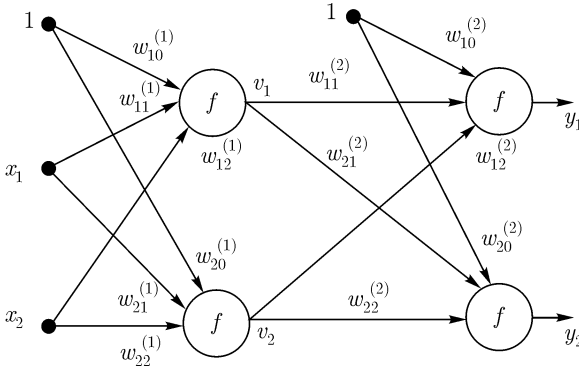


Рис. 1. Пример двухслойной нейронной сети

Основу алгоритма обратного распространения ошибки составляет целевая функция, формулируемая, как правило, в виде квадратичной суммы разностей между фактическими и ожидаемыми значениями выходных сигналов. Для обучающей выборки, состоящей из p примеров, целевая функция имеет вид

$$E(w) = \left[\sum_{j=1}^p \sum_{k=1}^M (y_k^{(j)} - d_k^{(j)})^2 \right] / 2$$

Минимизация целевой функции достигается уточнением вектора весов (обучением) по формуле

$$w(t+1) = w(t) + \Delta w,$$

где

$$\Delta w = \alpha s(w), \quad (1)$$

α — коэффициент обучения, а $s(w)$ — направление в пространстве весов w . Выбор этого направления обычно основан на определении **градиента целевой функции** относительно весов всех слоев сети. Для весов выходного слоя задача имеет очевидное решение. Для других слоев используется алгоритм обратного распространения ошибки. Рассмотрим его на примере двухслойной сети. В этом случае при $p = 1$ целевая функция определяется выражением

$$E(w) = \sum_{k=1}^M \left[f \left(\sum_{i=0}^K w_{ki}^{(2)} v_i \right) - d_k \right]^2 / 2 = \sum_{k=1}^M \left[f \left(\sum_{i=0}^K w_{ki}^{(2)} f \left(\sum_{j=0}^N w_{ij}^{(1)} x_j \right) \right) - d_k \right]^2 / 2 \quad (2)$$

Компоненты градиента рассчитываются дифференцированием зависимости (2). В первую очередь определяются веса нейронов выходного слоя. Для выходных весов получаем:

$$\partial E / \partial w_{ij}^{(2)} = (y_i - d_i) [df(u_i^{(2)}) / du_i^{(2)}] v_j,$$

где $u_i^{(2)} = \sum_{j=0}^K w_{ij}^{(2)} v_j$.

Если ввести обозначение

$$\partial_i^{(2)} = (y_i - d_i) [df(u_i^{(2)}) / du_i^{(2)}],$$

то соответствующую компоненту градиента относительно весов выходного слоя можно представить в виде

$$\partial E / \partial w_{ij}^{(2)} = \partial_i^{(2)} v_j. \quad (3)$$

Компоненты градиента относительно нейронов скрытого слоя определяются так же, но описываются более сложной зависимостью, следующей из существования функции, которая задана в виде

$$\partial E / \partial w_{ij}^{(1)} = \sum_{k=1}^M (y_k - d_k) [dy_k / dv_i] [dv_i / dw_{ij}^{(1)}].$$

Отсюда получаем

$$\partial E / \partial w_{ij}^{(1)} = \sum_{k=1}^M (y_k - d_k) [df(u_k^{(2)}) / du_k^{(2)}] w_{ki}^{(2)} [df(u_i^{(1)}) / du_i^{(1)}] x_j,$$

Если ввести обозначение

$$\partial_i^{(1)} = \sum_{k=1}^M (y_k - d_k) [df(u_k^{(2)}) / du_k^{(2)}] w_{ki}^{(2)} [df(u_i^{(1)}) / du_i^{(1)}],$$

то получим выражение, определяющее компоненты градиента относительно весов нейронов входного слоя в виде

$$\partial E / w_{ij}^{(1)} = \partial_i^{(1)} x_j. \quad (4)$$

В обоих случаях (формулы (3) и (4)) описания градиента имеют аналогичную структуру и представляются произведением двух сигналов: первый соответствует начальному узлу данной взвешенной связи, а второй — величине погрешности, перенесенной на узел, с которым эта связь установлена. Определение вектора градиента важно для последующего

процесса уточнения весов. В классическом алгоритме обратного распространения ошибки вектор $s(w)$ в выражении (1) задает направление антиградиента (**метод наискорейшего спуска**), поэтому

$$\Delta w = -\alpha \nabla E(w).$$

В соответствии с алгоритмом обратного распространения ошибки в каждом цикле обучения выделяются следующие этапы:

1. Анализ нейронной сети в прямом направлении передачи информации при генерации входных сигналов, составляющих очередной вектор x . В результате такого анализа рассчитываются значения выходных сигналов нейронов скрытых слоев и выходного слоя, а также соответствующие производные $df(u_i^{(1)})/du_i^{(1)}$, $df(u_i^{(2)})/du_i^{(2)}$, ..., $df(u_i^{(m)})/du_i^{(m)}$ функций активации каждого слоя (m — количество слоев сети).

2. Создание сети обратного распространения ошибок путем изменения направлений передачи сигналов на обратные, замена функций активации их производными и подача на бывший выход (а в настоящий момент — вход) сети сигнала в виде разности между фактическим и ожидаемым значением. Для определенной таким образом сети необходимо рассчитать значения требуемых обратных разностей.

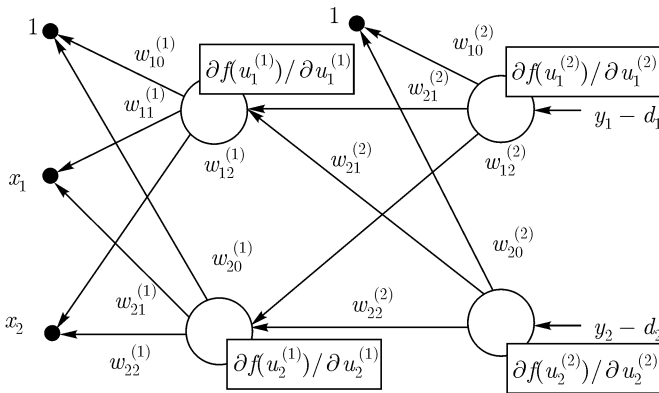


Рис. 2. Сеть обратного распространения ошибки

3. Уточнение весов (обучение сети) производится по предложенным выше формулам для оригинальной сети и для сети обратного распространения ошибки.

4. Описанный процесс следует повторить для всех обучающих примеров задачника, продолжая его вплоть до выполнения условия остано-

ки алгоритма. Действие алгоритма завершается в момент, когда норма градиента упадет ниже априори заданного значения, характеризующего точность процесса обучения.

Руководствуясь рис. 2, можно легко определить все компоненты градиента целевой функции, т. е. все частные производные функции E по весам сети. Для этого, двигаясь от входов сети (бывших выходов), нужно перемножить все встречающиеся на пути величины (кроме весов $w_{ij}^{(k)}$, для которых рассчитывается частная производная $\partial E/w_{ij}^{(k)}$). Кроме того, там, где дуги сходятся к одной вершине, нужно выполнить сложение произведений, полученных на этих дугах.

Так, например, чтобы посчитать производную $\partial E/w_{12}^{(2)}$, нужно перемножить величины $y_2 - d_2$, $df(u_2^{(2)})/du_2^{(2)}$, v_1 , а для вычисления производной $\partial E/w_{21}^{(1)}$ нужно посчитать произведения $\pi_1 = (y_1 - d_1) \times [\partial f(u_1^{(2)})/\partial u_1^{(2)}]w_{12}^{(2)}$ и $\pi_2 = (y_2 - d_2)[\partial f(u_2^{(2)})/\partial u_2^{(2)}]w_{22}^{(2)}$ и затем сложить эти произведения и результат умножить на $\partial f(u_2^{(2)})/\partial u_2^{(2)}$ и x_1 .

Таким образом, получим

$$\begin{aligned} \partial E/\partial w_{12}^{(1)} &= (\pi_1 + \pi_2)[\partial f(u_2^{(1)})/\partial u_2^{(1)}]x_1 = \\ &= [(y_1 - d_1)[\partial f(u_1^{(2)})/\partial u_1^{(2)}]w_{12}^{(2)} + (y_2 - d_2)[\partial f(u_2^{(2)})/\partial u_2^{(2)}]w_{22}^{(2)}] \times \\ &\times [\partial f(u_2^{(1)})/\partial u_2^{(1)}]x_1 = x_1 \sum_{k=1}^2 (y_k - d_k)[\partial f(u_k^{(2)})/\partial u_k^{(2)}]w_{k2}^{(2)} [\partial f(u_2^{(1)})/\partial u_2^{(1)}]. \end{aligned}$$

Одномерная оптимизация

Все пошаговые методы оптимизации состоят из двух важнейших частей:

- выбора направления,
- выбора шага в данном направлении (подбор коэффициента обучения).

Методы **одномерной оптимизации** дают эффективный способ для выбора шага.

В простейшем случае коэффициент обучения фиксируется на весь период оптимизации. Этот способ практически используется только совместно с методом наискорейшего спуска. Величина подбирается раздельно для каждого слоя сети по формуле

$$\alpha \leq \min(1/n_i),$$

где n_i обозначает количество входов i -го нейрона в слое.

Более эффективный метод основан на адаптивном подборе коэффициента α с учетом фактической динамики величины целевой функции.

Стратегия изменения значения α определяется путем сравнения суммарной погрешности ε на t -й итерации с ее предыдущим значением, причем рассчитывается по формуле

$$\varepsilon = \left[\sum_{i=1}^M (y_i - d_i)^2 \right]^{1/2}.$$

Для ускорения процесса обучения следует стремиться к непрерывному увеличению α при одновременном контроле прироста погрешности ε по сравнению с ее значением на предыдущем шаге. Незначительный рост погрешности считается допустимым.

Если погрешности на $t-1$ -й и t -й итерациях обозначить соответственно ε_{t-1} и ε_t , а коэффициенты обучения на этих же итерациях — α_{t-1} и α_t , то значение ε_{t+1} следует рассчитывать по формуле

$$\alpha_{t+1} = \alpha_t \rho_d, \text{ если } \varepsilon_t > k_w \varepsilon_{t-1},$$

$$\alpha_{t+1} = \alpha_t \rho_i, \text{ если } \varepsilon_t \leq k_w \varepsilon_{t-1}.$$

где k_w — коэффициент допустимого прироста погрешности, ρ_d — коэффициент уменьшения α , ρ_i — коэффициент увеличения α .

Наиболее эффективный, хотя и наиболее сложный, метод подбора коэффициентов обучения связан с направленной минимизацией целевой функции в выбранном направлении s_t . Необходимо так подобрать значение α_t , чтобы новое решение $w_{t+1} = w_t + \alpha_t s_t$ соответствовало минимуму целевой функции в данном направлении s_t .

Поиск минимума основан на полиномиальной аппроксимации целевой функции. Выберем для аппроксимации многочлен второго порядка

$$E(w) = P_2(\alpha) = a_2 \alpha^2 + a_1 \alpha + a_0,$$

где a_2 , a_1 и a_0 — коэффициенты, определяемые в цикле оптимизации. Для расчета этих коэффициентов используем три произвольные точки w_1, w_2, w_3 , лежащие в направлении s_t , т.е.

$$w_i = w + \alpha_i s_t, \quad i = 1, 2, 3.$$

Соответствующие этим точкам значения целевой функции $E(w)$ обозначим как

$$P_2(\alpha_i) = E_i = E(w_i), \quad i = 1, 2, 3. \quad (5)$$

Коэффициенты a_2 , a_1 и a_0 рассчитываются в соответствии с решением системы уравнений (5). Для определения минимума многочлена $P_2(\alpha)$ его производная $dP_2/d\alpha = 2a_2\alpha + a_1$ приравняется к нулю, что

позволяет получить $\alpha_{\min} = -a_1/2a_2$. После подстановки выражений для E_1, E_2, E_3 в формулу для α_{\min} получаем

$$\alpha_{\min} = \alpha_2 - [(\alpha_2 - \alpha_1)^2(E_2 - E_3) - (\alpha_2 - \alpha_3)^2(E_2 - E_1)]/2[(\alpha_2 - \alpha_1)(E_2 - E_3) - (\alpha_2 - \alpha_3)(E_2 - E_1)].$$

Методы инициализации весов

Обучение нейронных сетей представляет собой трудоемкий процесс, далеко не всегда дающий ожидаемые результаты. Проблемы возникают из-за нелинейных функций активации, образующих многочисленные локальные минимумы, к которым может сводиться процесс обучения. Применение методов глобальной оптимизации уменьшает вероятность останова процесса обучения в точке локального минимума, однако платой за это становится резкое увеличение трудоемкости и длительности обучения. Для правильного подбора управляющих параметров требуется большой опыт.

На результаты обучения огромное влияние оказывает подбор начальных значений весов сети. Выбор начальных значений, достаточно близких к оптимальным, значительно ускоряет процесс обучения. К сожалению, не существует универсального метода подбора весов, который бы гарантировал нахождение наилучшей начальной точки для любой решаемой задачи.

Неправильный выбор диапазона случайных значений весов может вызвать слишком раннее насыщение нейронов, в результате которого, несмотря на продолжающееся обучение, среднеквадратичная погрешность будет оставаться практически постоянной. Это означало бы попадание в седловую зону целевой функции вследствие слишком больших начальных значений весов. При этом взвешенная сумма входных сигналов нейрона может иметь значение, соответствующее глубокому насыщению сигмоидальной функции активации, и поляризация насыщения будет обратна ожидаемой. Значение возвратного сигнала, генерируемое в методе обратного распространения, пропорционально величине производной от функции активации и в точке насыщения близко нулю. Поэтому изменения значений весов, выводящие нейрон из состояния насыщения, происходят очень медленно. Процесс обучения надолго застревает в седловой зоне. Нейрон, остающийся в состоянии насыщения, не участвует в преобразовании данных, сокращая таким образом эффективное количество нейронов в сети. В итоге процесс обучения чрезвычайно замедляется, поэтому состояние насыщения отдельных нейронов может длиться практически непрерывно вплоть до исчерпания лимита итераций.

Удаление стартовой точки активации нейронов от зоны насыщения достигается путем ограничения диапазона случайных значений. Почти все оценки нижней и верхней границ диапазона допустимых значений лежат в интервале $(0, 1)$. Хорошие результаты дает равномерное распределение весов, нормализованное для каждого нейрона по амплитуде $w_{in} = 2/(n_{in})^{1/2}$, где n_{in} означает количество входов нейрона. Веса порогов для нейронов скрытых слоев должны принимать случайные значения из интервала $(-1/w_{in}, 1/w_{in})$, а для выходных нейронов — нулевые значения.

Лекция 7. Градиентные алгоритмы обучения сети

Рассматриваются: особенности задачи оптимизации, возникающей при обучении нейронных сетей; алгоритмы выбора направления минимизации: алгоритм наискорейшего спуска, партан-методы, одношаговый квазиньютоновский метод и сопряженные градиенты.

Ключевые слова: функция оценки, оптимизация нейронных сетей, выбор направления минимизации, наискорейший спуск, случайный выбор направления, итерационный партан-метод, модифицированный партан-метод, квазиньютоновские методы с ограниченной памятью, метод сопряженных градиентов.

Универсальный путь обучения

Существует универсальный путь обучения нейронных сетей — минимизация оценки как неявно заданной функции параметров сети. При реализации этого подхода предполагается, что:

- задана обучающая выборка, состоящая из векторов входных сигналов x^p ;
- известны требования к соответствующим выходным сигналам y^p , зафиксированные в функции оценки $E(y^p)$;
- оценка E по всей выборке или какой-либо ее части строится известным способом по значениям $E(y^p)$.

После подготовки (создание обучающей выборки, выбор функции оценки, предобработка входных данных и т. п.), предшествующей обучению, имеем способ вычисления некоторой функции E , минимизация которой как функции параметров настроит сеть для правильной работы.

Особенности задачи оптимизации, возникающей при обучении нейронных сетей

Задачи оптимизации нейронных сетей имеют ряд специфических ограничений. Они связаны с огромной размерностью задачи обучения. Число параметров может достигать 10^8 и более. В простейших программных имитаторах на персональных компьютерах подбирается 10^3 – 10^4 параметров. Из-за высокой размерности возникают два требования к алгоритму:

1. *Ограничение по памяти.* Пусть n — число параметров. Если алгоритм требует затрат памяти порядка n^2 , то он вряд ли применим для

обучения. Желательно иметь алгоритмы, которые требуют затрат памяти kn , $k = \text{const}$.

2. *Возможность параллельного вычисления* наиболее трудоемких этапов алгоритма, и желательно нейронной сетью.

3. *Обученный нейрокомпьютер* должен с приемлемой точностью решать все тестовые задачи. Поэтому задача обучения становится *многокритериальной задачей оптимизации*: нужно найти точку общего минимума большого числа функций. Обучение нейрокомпьютера исходит из гипотезы о существовании этой точки.

4. Обученный нейрокомпьютер должен иметь возможность приобретать *новые навыки без утраты старых*. Возможно более слабое требование: новые навыки могут сопровождаться потерей точности в старых, но потеря не должна быть существенной. Это означает, что в достаточно большой окрестности найденной точки общего минимума оценок их значения незначительно отличаются от минимальных. Итак, имеем четыре специфических ограничения, выделяющих обучение нейрокомпьютера из общих задач оптимизации:

- астрономическое число параметров;
- необходимость высокого параллелизма при обучении;
- многокритериальность решаемых задач;
- необходимость найти достаточно широкую область, в которой значения всех минимизируемых функций близки к минимальным.

Учет ограничений при обучении

Для параметров сети возможны ограничения простейшего вида:

$$w_{i \min} \leq w_i \leq w_{i \max}.$$

Они вводятся из различных соображений: чтобы избежать слишком крутых или, наоборот, слишком пологих характеристик нейронов, чтобы предотвратить появления слишком больших коэффициентов усиления сигнала на синапсах и т.п.

Учесть ограничения можно, например, *методом штрафных функций* либо *методом проекций*:

1. Использование метода штрафных функций означает, что в оценку E добавляется штраф за выход параметров из области ограничений. В градиент E вводятся производные штрафных функций.

2. Проективный метод означает, что если в сети предлагается изменение параметров w_i : $= W_i$ и W_i для некоторых i выходит за ограниче-

ния, то следует положить

$$w_i := \begin{cases} W_i, & \text{если } w_{i \min} \leq W_i \leq w_{i \max} \\ w_{i \max}, & \text{если } W_i > w_{i \max} \\ w_{i \min}, & \text{если } W_i < w_{i \min} \end{cases}$$

Практика показывает, что проективный метод не приводит к затруднениям. Обращение со штрафными функциями менее успешно. Далее будем считать, что ограничения учтены одним из методов, и будем говорить об обучении как о безусловной минимизации.

Выбор направления минимизации

Пусть задано начальное значение вектора параметров w^0 и вычислена функция оценки $E = E(w^0)$. Процедура одномерной оптимизации дает приближенное положение минимума $e(x) = E(w^0 + xs)$ (вообще говоря, локального).

Наиболее очевидный выбор направления s для одномерной оптимизации — направление антиградиента E :

$$s = -\nabla E.$$

Выберем на каждом шаге это направление, затем проведем одномерную оптимизацию, потом снова вычислим градиент E и т.д. Это — **метод наискорейшего спуска**, который иногда работает хорошо. Но неиспользование информации о кривизне функции оценки (целевой функции) и резкое замедление минимизации в окрестности точки оптимального решения, когда градиент принимает очень малые значения, часто делают алгоритм наискорейшего спуска низкоэффективным.

Другой способ — случайный выбор направления s для одномерной оптимизации. Он требует большого числа шагов, но зато предельно прост — ему необходимо только прямое функционирование сети с вычислением оценки.

Партан-методы

Для исправления недостатков наискорейшего спуска разработаны итерационный и модифицированный партан-методы.

Итерационный партан-метод (k -партан) строится следующим образом. В начальной точке w^0 вычисляется градиент оценки E и делается шаг наискорейшего спуска — для этого используется одномерная оптимизация. Далее снова вычисляется градиент E и выполняется спуск (т.е.

перемещение в направлении антиградиента), и описанный процесс повторяется k раз. После k шагов наискорейшего спуска получаем точку w^k и проводим одномерную оптимизацию из w^0 в направлении $s = w^k - w^0$ с начальным шагом $\alpha = 1$. После этого цикл повторяется.

Модифицированный партан-метод требует запоминания дополнительных параметров. Он строится следующим образом. Из w^0 делается два шага наискорейшего спуска. Получаем w^1 и w^2 . Далее выполняем одномерную оптимизацию в направлении $w^2 - w^0$. Получаем w^3 . Далее выполняется наискорейший спуск из w^3 . Получаем w^4 . Выполняем одномерную оптимизацию из w^2 в направлении $w^4 - w^2$. Получаем w^5 и т.д. Таким образом, четные w^{2k} получаем наискорейшим спуском из w^{2k-1} , нечетные w^{2k+1} — одномерной оптимизацией из w^{2k-2} в направлении $s = w^{2k} - w^{2k-2}$ (начальный шаг $\alpha = 1$). Как показала практика, модифицированный партан-метод в задачах обучения работает лучше, чем k -партан.

Одношаговый квазиньютоновский метод и сопряженные градиенты

В тех случаях, когда является положительно определенной матрица D_2 вторых производных оценки E , наилучшим считается ньютоновское направление

$$s = -D_2^{-1} \nabla E.$$

С использованием этой формулы квадратичные формы минимизируются за один шаг, однако, применять эту формулу трудно по следующим причинам:

1. *Время.* Поиск всех вторых производных функции E и обращение матрицы D_2 требует больших вычислительных затрат.

2. *Память.* Для решения задач большой размерности N требуется хранить N^2 элементов матрицы D_2^{-1} — это слишком много.

3. Матрица D_2 не всегда является положительно определенной.

Для преодоления этих трудностей разработана масса методов. Идея **квазиньютоновских методов с ограниченной памятью** состоит в том, что поправка к направлению наискорейшего спуска отыскивается как результат действия матрицы малого ранга. Сама матрица не хранится, а её действие на векторы строится с помощью скалярных произведений на несколько специально подобранных векторов.

Простейший и весьма эффективный метод основан на *BFGS* формуле (Брайден-Флетчер-Гольдфард-Шанно) и использует результаты предыдущего шага. Обозначим:

s_k — направление спуска на k -шаге;

α_k — величина k шага (k -й шаг — сдвиг на $\alpha_k s_k$);

g_k — градиент функции оценки в начальной точке k -го шага;

$y_k = g_k - g_{k-1}$ — изменение градиента в результате k -го шага.

BFGS — формула для направления спуска на $k + 1$ -м шаге имеет

вид:

$$s_{k+1} = -g_{k+1} + [(s_k, g_{k+1})y_k + (y_k, g_{k+1})s_k] / (y_k, s_k) - h_k s_k (s_k, g_{k+1}) / (y_k, s_k) - s_k (y_k, y_k) \cdot (s_k, g_{k+1}) / (y_k, s_k)^2,$$

где (x, y) — скалярное произведение векторов x и y .

Если одномерную оптимизацию в поиске шага проводить достаточно точно, то новый градиент g_{k+1} будет практически ортогонален предыдущему направлению спуска, т. е. $(s_k, g_{k+1}) = 0$. При этом формула для s_{k+1} упрощается:

$$s_{k+1} = -g_{k+1} + s_k (y_k, g_{k+1}) / (y_k, s_k).$$

Это — формула **метода сопряженных градиентов** (МСГ), которому требуется достаточно аккуратная одномерная оптимизация при выборе шага.

В описанных методах предполагается, что начальное направление спуска $s_0 = -g_0$. После некоторой последовательности из k шагов целесообразно возвращаться к наискорейшему спуску — проводить рестарт. Он используется в тех случаях, когда очередное s_{k+1} — плохое направление спуска, т. е. движение вдоль него приводит к слишком маленькому шагу либо вообще не дает улучшения.

Лекция 8. Методы глобальной оптимизации

Рассматриваются: алгоритм имитации отжига, генетические алгоритмы, использование случайных возмущений в обучении (метод виртуальных частиц).

Ключевые слова: метод имитации отжига, генетические алгоритмы, метод виртуальных частиц, селекция, скрещивание, мутация, принцип элитарности, принцип рулетки, функция приспособленности.

Элементы глобальной оптимизации

Все представленные ранее методы обучения нейронных сетей являются локальными. Они ведут к одному из локальных минимумов целевой функции, лежащему в окрестности точки начала обучения. Только в ситуации, когда значение глобального минимума известно, удастся оценить, находится ли найденный локальный минимум в достаточной близости от искомого решения. Если локальное решение признается неудовлетворительным, следует повторить процесс обучения при других начальных значениях весов и с другими управляющими параметрами. Можно либо проигнорировать полученное решение и начать обучение при новых (как правило, случайных) значениях весов, либо изменить случайным образом найденное локальное решение (встряхивание весов) и продолжить обучение сети.

При случайном приращении весов переход в новую точку связан с определенной вероятностью того, что возобновление процесса обучения выведет поиск из «сферы притяжения» локального минимума.

При решении реальных задач в общем случае даже приблизительная оценка глобального минимума оказывается неизвестной. По этой причине возникает необходимость применения методов глобальной оптимизации. Рассмотрим три из разработанных подходов к глобальной оптимизации: **метод имитации отжига, генетические алгоритмы и метод виртуальных частиц.**

Алгоритмы имитации отжига

Метод имитации отжига основан на идее, заимствованной из статистической механики. Он отражает поведение расплавленного материала при отвердевании с применением процедуры отжига (управляемого охлаждения) при температуре, последовательно понижаемой до нуля.

В процессе медленного управляемого охлаждения, называемого отжигом, кристаллизация расплава сопровождается глобальным уменьшением его энергии, однако допускаются ситуации, в которых она может на какое-то время возрасти (в частности, при подогреве расплава для предотвращения слишком быстрого его остывания). Благодаря допустимости кратковременного повышения энергетического уровня, возможен выход из ловушек локальных минимумов энергии, которые возникают при реализации процесса. Только понижение температуры до абсолютного нуля делает невозможным какое-либо самостоятельное повышение энергетического уровня расплава.

Метод имитации отжига представляет собой алгоритмический аналог физического процесса управляемого охлаждения. Классический алгоритм имитации отжига можно описать следующим образом:

1. Запустить процесс из начальной точки w при заданной начальной температуре $T = T_{\max}$.

2. Пока $T > 0$, повторить L раз следующие действия:

— выбрать новое решение w' из окрестности w ;

— рассчитать изменение целевой функции $\Delta = E(w') - E(w)$;

— если $\Delta \leq 0$, принять $w = w'$; в противном случае (при $\Delta > 0$) принять, что $w = w'$ с вероятностью $\exp(-\Delta/T)$ путем генерации случайного числа R из интервала $(0, 1)$ с последующим сравнением его со значением $\exp(-\Delta/T)$. Если $\exp(-\Delta/T) > R$, принять новое решение $w = w'$; в противном случае проигнорировать его.

3. Уменьшить температуру ($T := rT$) с использованием коэффициента r , выбираемого из интервала $(0, 1)$, и вернуться к п. 2.

4. После снижения температуры до нуля провести обучение сети любым из детерминированных методов локальной оптимизации вплоть до достижения минимума целевой функции.

Наибольшего ускорения имитации отжига можно достичь путем замены случайных начальных значений весов w тщательно подобранными значениями с использованием любых доступных способов предварительной обработки исходных данных.

Метод имитации отжига оказывается особенно удачным для полимодальных комбинаторных проблем с очень большим количеством возможных решений, например, для машины Больцмана, в которой каждое состояние системы считается допустимым. При решении наиболее распространенных задач обучения многослойных нейронных сетей наилучшие результаты в общем случае достигаются применением стохастически управляемого метода повторных рестартов совместно с детерминированными алгоритмами локальной оптимизации.

Генетические алгоритмы

Генетические алгоритмы имитируют процессы наследования свойств живыми организмами и генерируют последовательности новых векторов w , содержащие оптимизированные переменные: $w = [w_1, w_2, \dots, w_n]^T$. При этом выполняются операции трех видов: **селекция**, **скрещивание** и **мутация**.

На начальной стадии выполнения генетического алгоритма случайным образом инициализируется определенная популяция хромосом (векторов w). Размер популяции, как правило, пропорционален количеству оптимизируемых параметров. Слишком малая популяция хромосом приводит к замыканию в неглубоких локальных минимумах. Слишком большое их количество чрезмерно удлиняет вычислительную процедуру и также может не привести к точке глобального минимума.

Селекция хромосом для спаривания (необходимого для создания нового поколения) может основываться на разных принципах. Одним из наиболее распространенных считается **принцип элитарности**, в соответствии с которым наиболее приспособленные (в смысле целевой функции) хромосомы сохраняются, а наименее приспособленные отбраковываются и заменяются вновь созданным потомством, полученным в результате скрещивания пар родителей.

Существует огромное множество методов скрещивания, начиная с полностью случайного. При взвешенно-случайном скрещивании учитывается информация о текущем значении целевой функции. Отбор может происходить по **принципу рулетки**; при этом площадь сегмента колеса рулетки, сопоставленного конкретной хромосоме, пропорциональна величине ее **функции приспособленности** $F(w) = -E(w)$, где $E(w)$ — ее целевая функция.

Процесс скрещивания основан на расщеплении пары хромосом на две части с последующим обменом этих частей в хромосомах родителей (рис. 1). Место расщепления также выбирается случайным образом. Количество новых потомков равно количеству отбракованных в результате селекции (размер популяции остается неизменным). Признается допустимым перенос в очередное поколение некоторых случайно выбранных хромосом вообще без скрещивания.

Последняя генетическая операция — это мутация. При двоичном кодировании мутация состоит в инверсии случайно выбранных битов. При кодировании векторов десятичными числами мутация заключается в замене значения какого-либо элемента вектора другим случайно выбранным значением. Мутация обеспечивает защиту как от слишком быстрого завершения алгоритма (в случае выравнивания значений всех хромосом и целевой функции), так и от представления в какой-либо конкретной

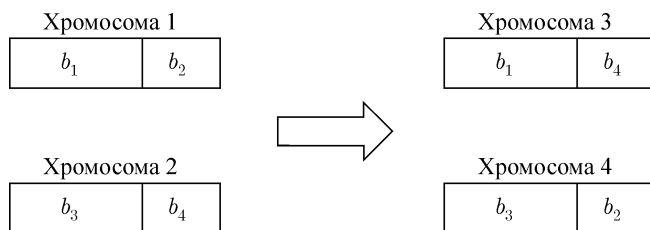


Рис. 1. Процесс скрещивания

позиции всех хромосом одного и того же значения. Однако необходимо иметь в виду, что случайные мутации приводят к повреждению уже частично приспособленных векторов. Обычно мутации подвергается не более 1–5% бит всей популяции хромосом. Элемент, подвергаемый мутации, отбирается случайным образом.

Доказано, что каждое последующее поколение, сформированное селекцией, скрещиванием и мутацией, имеет статистически лучшие средние показатели приспособленности (меньшие значения целевой функции).

В качестве окончательного решения принимается наиболее приспособленная хромосома, имеющая минимальное значение целевой функции. Генетический процесс завершается либо в момент генерации удовлетворяющего нас решения, либо при выполнении максимально допустимого количества итераций. При реализации генетического процесса отслеживается, как правило, не только минимальное значение целевой функции, но и среднее значение по всей популяции хромосом, а также их вариации. Решение об остановке алгоритма может приниматься и в случае отсутствия прогресса минимизации, определяемого по изменениям названных характеристик.

Метод виртуальных частиц

Метод виртуальных (случайных) частиц может надстраиваться почти над любым методом оптимизации. Он создан для:

- 1) повышения устойчивости обученных сетей;
- 2) вывода сетей из возникающих при обучении локальных минимумов оценки.

Основная идея метода — использование случайных сдвигов аргумента и суммирование полученных значений функции для усреднения. Ожидается, что в результате уменьшится влияние рельефа минимизируемой функции на процесс минимизации и откроется более прямой путь к её глобальному минимуму.

Метод случайных частиц состоит в том, что к оптимизируемой точке (частице) добавляется несколько других, траектории которых получаются из траектории данной частицы сдвигом на случайный вектор. Эти «виртуальные» частицы время от времени уничтожаются и рождаются новые. Спуск (минимизация) строится так, чтобы уменьшилось значение суммы значений оптимизируемой функции в указанных точках.

Рассмотрим один из вариантов алгоритма виртуальных частиц. Пусть требуется найти минимум функции $H_{pg}(w)$. Параметры сети разбиваются на группы структурно эквивалентных. Для каждой группы задается свой интервал случайных сдвигов. Определяется число виртуальных частиц n и генерируется $n - 1$ случайных векторов r_1, r_2, \dots, r_{n-1} . Их координаты независимо и равномерно распределены в заданных интервалах.

Начальное положение основной частицы - w^0 . Начальное положение i -ой виртуальной частицы $w^0 + r_i, i = 1, \dots, n - 1$. Случайный вектор для n -й виртуальной частицы строится так:

$$r_n = (r_1 + r_2 + \dots + r_{n-1})/n^{1/2} \quad (1)$$

и её положение задается вектором $w^0 + r_n$. Всем частицам, кроме n -й, присваивается вес $W, 0 < W < 1$, n -я получает вес $W_n = W/n^{1/2}$. Далее минимизируется функция

$$H_W(w) = H_{pg}(w) + W(H_{pg}(w + r_1) + \dots + H_{pg}(w + r_{n-1})) + W_n H_{pg}(w + r_n).$$

Алгоритм локальной оптимизации может быть выбран любой — от наискорейшего спуска и партан-методов до метода сопряженных градиентов. Выбор r_n в виде (1) и $W_n = W/n^{1/2}$ определяется двумя обстоятельствами:

1. для каждой координаты вектора r_n дисперсия будет совпадать с дисперсией координат векторов $r_i, i = 1, \dots, n - 1$;
2. для квадратичных $H_{pg}(w)$ точки минимума $H_{pg}(w)$ и $H_W(w)$ совпадут.

В методе виртуальных частиц возникает важный вопрос: когда уничтожать имеющиеся виртуальные частицы и породить новые?

Есть три варианта:

1. Функция $H_W(w)$ минимизируется до тех пор, пока скорость обучения не упадет ниже критической. После этого вновь производят случайные сдвиги частицы, и обучение продолжается.

2. Порождение новых частиц производится после каждого цикла базового алгоритма оптимизации — при рестартах. Например, после каждого шага метода наискорейшего спуска, после партан-шага итерационного партан-метода и т.п.

3. При каждом вычислении оценок и градиентов.

Первый способ наиболее консервативен. Он долго сохраняет все достоинства и недостатки предшествующего спуска, хотя направление движения может существенно измениться при порождении новых виртуальных частиц.

Третий способ вносит случайный процесс внутрь базового алгоритма, в результате возможны колебания даже при одномерной оптимизации. Его преимущество — экономия памяти.

Наиболее перспективным представляется второй способ. Он, с одной стороны, не разрушает базового алгоритма, а с другой — за счет многократного порождения виртуальных частиц позволяет приблизиться к глобальному множеству. Метод виртуальных частиц имеет все достоинства методов глобальной оптимизации, не использующих случайные возмущения, но лишен многих их недостатков.

Хорошие результаты обучения приносит объединение алгоритмов глобальной оптимизации с детерминированными методами локальной оптимизации. На первом этапе обучения сети применяется выбранный алгоритм глобальной оптимизации, а после достижения целевой функцией определенного уровня включается детерминированная оптимизация с использованием какого-либо локального алгоритма.

Четыре типа устойчивости

Навыки обучения нейрокомпьютера должны быть устойчивы к возмущению различных типов. Разработчики нейрокомпьютеров выделяют четыре типа устойчивости:

- 1) к случайным возмущениям входных сигналов;
- 2) к флуктуациям параметров сети;
- 3) к разрушению части элементов сети;
- 4) к обучению новым примерам.

В конкретных ситуациях необходимо доопределять возмущения, по отношению к которым нужно вырабатывать устойчивость. Например, при распознавании визуальных образцов можно выделить несколько разновидностей возмущений входного сигнала: прибавление случайного сигнала (шум фона), затенение части исходного изображения, искажение изображения некоторыми преобразованиями.

Для выработки устойчивости первых трех типов полезны генераторы случайных искажений. Для устойчивости 1-го типа генератор искажений производит возмущение входных сигналов и тем самым преобразует обучающей пример. Для устойчивости 2-го типа генератор искажений меняет случайным образом параметры сети в заданных пределах, а для

устойчивости 3-го типа — удаляет случайно выбранную часть сети, состоящую из заданного количества элементов (нейронов, синапсов).

В существенной конкретизации нуждается четвертый тип устойчивости, т.к. трудно представить себе устойчивость к обучению любому новому примеру. Если принять гипотезу, что обучение новым примерам будет действовать на старые навыки так же, как случайный сдвиг параметров, то получается, что выработка устойчивости 2-го типа является средством для обучения устойчивости 4-го типа. Другое средство — выработка устойчивости к обучению отдельным примерам, уже входящим в задачник. Это свойство устойчивости 1-го типа состоит в том, что обучение до минимума оценки по любому (одному) из обучающих примеров не разрушает навыка решения остальных. Возмущение здесь состоит в изменении процесса обучения.

Для выработки устойчивости 1-го типа примеры предъявляются сети не все сразу, а по одному, и сеть учится каждому из них до предела. Для выработки важнейшей устойчивости 4-го типа такая периодически производимая «порча» процесса обучения может быть полезной. Опыт показывает, что обучение позволяет выработать устойчивость к весьма сильным возмущениям. Так, в задачах распознавания визуальных образов уровень шума на выходе мог в несколько раз превосходить общую интенсивность сигнала, случайный сдвиг параметров — достигать 0.5–0.7 их предельного значения, разрушение — 30–50% элементов. И, тем не менее, обученная сеть делает не более 10% ошибок!

Лекция 9. Радиальные нейронные сети

Рассматриваются математические основы радиальных сетей и методы их обучения. Производится сравнение радиальных и сигмоидальных нейронных сетей.

Ключевые слова: глобальная аппроксимация, локальная аппроксимация, радиальные базисные функции.

Многослойные нейронные сети, представленные в предыдущих разделах, выполняют аппроксимацию функции нескольких переменных путем преобразования множества входных переменных $x \in R^N$ в множество выходных переменных $y \in R^M$. Сигмоидальная функция активации по своему характеру осуществляет аппроксимацию глобального типа. В результате ее нейрон, который был однажды «включен» (после превышения суммарным сигналом определенного порогового значения), остается в этом состоянии при любом значении сигнала, превышающем данный порог. Поэтому преобразование значения функции в произвольной точке пространства выполняется объединенными усилиями многих нейронов, что и объясняет название **глобальная аппроксимация**.

Другой способ отображения входного множества в выходное заключается в преобразовании путем адаптации нескольких одиночных аппроксимирующих функций к ожидаемым значениям, причем эта адаптация проводится только в ограниченной области многомерного пространства. При таком подходе отображение всего множества данных представляет собой сумму локальных преобразований. С учетом роли, которую играют скрытые нейроны, преобразования составляют множество базисных функций локального типа. Выполнение одиночных функций (при ненулевых значениях) регистрируется только в ограниченной области пространства данных — отсюда и название **локальная аппроксимация**.

Особое семейство образуют *сети с радиальной базисной функцией*, в которых нейроны реализуют функции, радиально изменяющиеся вокруг выбранного центра и принимающие ненулевые значения только в окрестности этого центра. Подобные функции, определяемые в виде $\varphi(x) = \varphi(\|x - c\|)$, будем называть **радиальными базисными функциями**. В таких сетях роль нейрона заключается в отображении радиального пространства вокруг одиночной заданной точки (центра) либо вокруг группы таких точек, образующих кластер. Суперпозиция сигналов, поступающих от всех таких нейронов, которая выполняется выходным нейроном, позволяет получить отображение всего многомерного пространства.

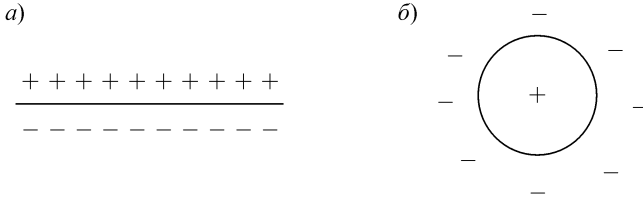


Рис. 1. Иллюстрация способов разделения пространства данных: а) сигмоидальным нейроном; б) радиальным нейроном

Сети радиального типа представляют собой естественное дополнение сигмоидальных сетей. Сигмоидальный нейрон представляется в многомерном пространстве гиперплоскостью, разделяющей это пространство на две категории (два класса), в которых выполняется одно из двух условий: либо $(w, x) > 0$, либо $(w, x) < 0$. Такой подход продемонстрирован на рис. 1а.

В свою очередь, радиальный нейрон представляет собой гиперсферу, которая осуществляет шаровое разделение пространства вокруг центральной точки (рис. 1б). Именно с этой точки зрения он является естественным дополнением сигмоидального нейрона, поскольку в случае круговой симметрии данных позволяет заметно уменьшить количество нейронов, необходимых для разделения различных классов. Поскольку нейроны могут выполнять различные функции, в радиальных сетях отсутствует необходимость использования большого количества скрытых слоев. Структура типичной радиальной сети включает входной слой, на который подаются сигналы, описываемые входным вектором x , скрытый слой с нейронами радиального типа и выходной слой, состоящий, как правило, из одного или нескольких линейных нейронов. Функция выходного нейрона сводится исключительно к взвешенному суммированию сигналов, генерируемых скрытыми нейронами.

Математические основы радиальных сетей

Математическую основу функционирования радиальных сетей составляет теорема Т. Ковера о распознаваемости образов, в соответствии с которой нелинейные проекции образов в некоторое многомерное пространство могут быть линейно разделены с большей вероятностью, чем при их проекции в пространство с меньшей размерностью.

Если вектор радиальных функций в N - мерном входном пространстве обозначить $\varphi(x)$, то это пространство является нелинейно φ - разде-

ляемым на два пространственных класса X^+ и X^- тогда, когда существует такой вектор весов w , что

$$\begin{aligned} w^T \varphi(x) &> 0, x \in X^+, \\ w^T \varphi(x) &< 0, x \in X^-. \end{aligned}$$

Граница между этими классами определяется уравнением $w^T \varphi(x) = 0$.

Доказано, что каждое множество образов, случайным образом размещенных в многомерном пространстве, является φ -разделяемым с вероятностью 1 при условии соответственно большой размерности этого пространства. На практике это означает, что применение достаточно большого количества скрытых нейронов, реализующих радиальные функции $\varphi(x)$, гарантирует решение задачи классификации при построении всего лишь двухслойной сети: скрытый слой должен реализовать вектор $\varphi(x)$, а выходной слой может состоять из единственного линейного нейрона, который выполняет суммирование выходных сигналов от скрытых нейронов с весовыми коэффициентами, заданными вектором w .

Простейшая нейронная сеть радиального типа функционирует по принципу многомерной интерполяции, состоящей в отображении p различных входных векторов $x_i, i = 1, 2, \dots, p$ из входного N -мерного пространства во множество из p чисел $d_i, i = 1, 2, \dots, p$. Для реализации этого процесса необходимо использовать p скрытых нейронов радиального типа и задать такую функцию отображения $F(x)$, для которой выполняется условие интерполяции

$$F(x_i) = d_i.$$

Использование p скрытых нейронов, соединяемых связями с весами с выходными линейными нейронами, означает формирование выходных сигналов сети путем суммирования взвешенных значений соответствующих базисных функций. Рассмотрим радиальную сеть с одним выходом и p обучающими парами (x_i, d_i) . Примем, что координаты каждого из p центров узлов сети определяются одним из векторов x_i , т.е. $c_i = x_i$. В этом случае взаимосвязь между входными и выходными сигналами сети может быть определена системой уравнений, линейных относительно весов, которая в матричной форме имеет вид:

$$\varphi \cdot w = d, \tag{1}$$

где $\varphi_{ji} = (\|x_j - x_i\|)$ определяет радиальную функцию с центром в точке x_i с вынужденным вектором x_j , $w = [w_1, w_2, \dots, w_p]^T$ и $d = [d_1, d_2, \dots, d_p]^T$.

Доказано, что для ряда радиальных функций в случае $x_1 \neq x_2 \neq \dots \neq x_p$ квадратная интерполяционная матрица φ является невырожденной и при этом неотрицательно определенной. Поэтому существует решение уравнения (1) в виде

$$W = \varphi^{-1}d, \quad (2)$$

что позволяет получить вектор весов выходного нейрона сети.

Теоретическое решение проблемы, представленное выражением (2), не может считаться абсолютно истинным по причине серьезного ограничения общих свойств сети, вытекающих из сделанных вначале допущений. При очень большом количестве обучающих выборок и равном ему количеству радиальных функций проблема с математической точки зрения становится бесконечной (плохо структурированной), поскольку количество уравнений начинает превышать число степеней свободы физического процесса, моделируемого уравнением (1). Это означает, что результатом такого чрезмерного количества весовых коэффициентов станет адаптация модели к разного рода шумам или нерегулярностям, сопровождающим обучающие выборки. Как следствие, интерполирующая эти данные гиперповерхность не будет гладкой, а обобщающие возможности останутся очень слабыми.

Чтобы их усилить, следует уменьшить количество радиальных функций и получить из избыточного объема данных дополнительную информацию для регуляризации задачи и улучшения ее обусловленности.

Радиальная нейронная сеть

Использование в разложении p базисных функций, где p — это количество обучающих выборок, недопустимо также и с практической точки зрения, поскольку обычно количество этих выборок очень велико, и в результате вычислительная сложность обучающего алгоритма становится чрезмерной. Решение системы уравнений (1) размерностью $p \times p$ при больших значениях p становится затруднительным. Так же, как и для многослойных сетей, необходимо редуцировать количество весов, что в этом случае сводится к уменьшению количества базисных функций. Поэтому отыскивается субоптимальное решение в пространстве меньшей размерности, которое с достаточной точностью аппроксимирует точное решение. Если ограничиться K базисными функциями, то аппроксимирующее решение можно представить в виде

$$F(x) = f_1 + f_2 + \dots + f_K, \quad (3)$$

где $f_i = w_i \varphi(\|x - c_i\|)$, $K < p$, а $c_i, i = 1, 2, \dots, K$ — множество центров,

которые необходимо определить. В особом случае, если принять $K = p$, можно получить точное решение $c_i = x_i$.

Чаще всего в качестве радиальной функции применяется функция Гаусса. При размещении ее центра в точке c_i она может быть определена в сокращенной форме как

$$\varphi(x) = \varphi(\|x - c_i\|) = \exp(-\|x - c_i\|^2/2\sigma_i^2). \quad (4)$$

В этом выражении σ_i — параметр, от значения которого зависит ширина функции.

Полученное решение, представляющее аппроксимирующую функцию в многомерном пространстве в виде взвешенной суммы локальных базисных радиальных функций (выражение (3)), может быть интерпретировано радиальной нейронной сетью, представленной на рис. 2 (для упрощения эта сеть имеет только один выход), в которой i определяется зависимостью (4). Это сеть с двухслойной структурой, в которой только скрытый слой выполняет нелинейное отображение, реализуемое нейронами с базисными радиальными функциями. Выходной нейрон, как правило, линеен, а его роль сводится к взвешенному суммированию сигналов, поступающих от нейронов скрытого слоя. Вес w_0 , как и при использовании сигмоидальных функций, представляет поляризацию (порог), вводящую показатель постоянного смещения функции.

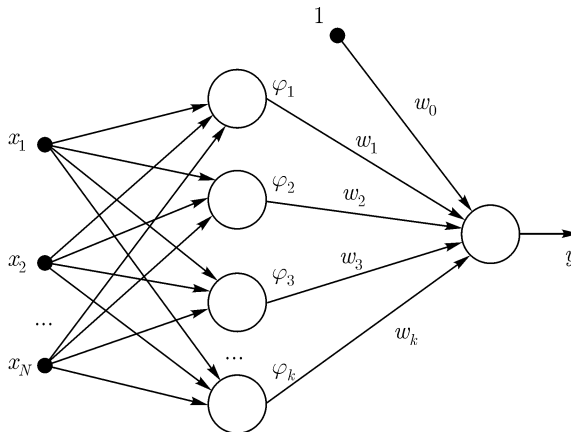


Рис. 2. Обобщенная структура радиальной сети

Полученная архитектура радиальных сетей имеет структуру, аналогичную многослойной структуре сигмоидальных сетей с одним скрытым

слоем. Роль скрытых нейронов в ней играют базисные радиальные функции, отличающиеся своей формой от сигмоидальных функций. Несмотря на отмеченное сходство, сети этих типов принципиально отличаются друг от друга. Радиальная сеть имеет фиксированную структуру с одним скрытым слоем и линейными выходными нейронами, тогда как сигмоидальная сеть может содержать различное количество слоев, а выходные нейроны бывают как линейными, так и нелинейными. У используемых радиальных функций может быть весьма разнообразная структура. Нелинейная радиальная функция каждого скрытого нейрона имеет свои значения параметров c_i и σ_i , тогда как в сигмоидальной сети применяются, как правило, стандартные функции активации с одним и тем же для всех нейронов параметром β . Аргументом радиальной функции является евклидово расстояние образца x от центра c_i , а в сигмоидальной сети — это скалярное произведение векторов $w^T x$.

Лекция 10. Рекуррентные сети как ассоциативные запоминающие устройства

Рассматриваются: нейронная сеть Хопфилда как автоассоциативная память, обучаемая с использованием метода Хебба и проекционного метода; гетероассоциативная память на основе сети Хемминга и двунаправленная ассоциативная память.

Ключевые слова: рекуррентные сети, аттракторы, автоассоциативная память, память гетероассоциативного типа, расстояние Хемминга, автосвязь, емкость памяти.

Введение

Отдельную группу нейронных сетей составляют сети с обратной связью между различными слоями нейронов. Это так называемые **рекуррентные сети**. Их общая черта состоит в передаче сигналов с выходного либо скрытого слоя на входной слой.

Благодаря обратной связи при подаче сигнала на входы сети, в ней возникает переходный процесс, который завершается формированием нового устойчивого состояния, отличающегося в общем случае от предыдущего. Если функцию активации нейрона обозначить $f(u)$, где u — взвешенная сумма его возбуждений, то состояние нейрона можно определить выходным сигналом $y = f(u) = f(w_1x_1 + \dots + w_Nx_N)$. Изменение состояния i -го нейрона можно описать системой дифференциальных уравнений

$$\tau_1(\partial u_i / \partial t) = w_{i1}f(u_1) + \dots + w_{iN}f(u_N) - u_i - b_i$$

для $i = 1, 2, \dots, N$, где b_i — пороговое значение. Рекуррентной сети можно поставить в соответствие энергетическую функцию Ляпунова

$$E = -(1/2) \sum_j \sum_{i \neq j} w_{ij} y_i y_j + \sum_{i=1, N} (1/R_i) \int_0^x f_i^{-1}(y_i) dy_i + \sum_{i=1, N} b_i y_i.$$

Изменение состояния какого-либо нейрона инициализирует изменение энергетического состояния сети в направлении минимума ее энергии вплоть до его достижения. В пространстве состояний локальные энергетические минимумы E представлены точками стабильности, называемыми **аттракторами** из-за тяготения к ним ближайшего окружения.

Благодаря наличию аттракторов, рекуррентные сети могут быть использованы как устройства ассоциативной памяти.

Ассоциативная память играет роль системы, определяющей взаимную зависимость векторов. В случае, когда на взаимозависимость исследуются компоненты одного и того же вектора, говорят об **автоассоциативной памяти**. Если же взаимозависимыми оказываются два различных вектора, можно говорить о **памяти гетероассоциативного типа**. К первому классу относится сеть Хопфилда, а ко второму — сеть Хемминга и сеть типа ВАР (Bidirectional Associative Memory — двунаправленная ассоциативная память).

Задача ассоциативной памяти сводится к запоминанию обучающих векторов, чтобы при представлении нового вектора система могла сгенерировать ответ — какой из запомненных ранее векторов наиболее близок к вновь поступившему образу. Часто в качестве меры близости отдельных множеств применяется **расстояние Хемминга**.

При использовании двоичных значений (0, 1) расстояние Хемминга между двумя векторами $y = (y_1, y_2, \dots, y_n)$ и $d = (d_1, d_2, \dots, d_n)$ определяется в виде

$$d_H(y, d) = \sum_{i=1, n} (d_i(1 - y_i) + (1 - d_i)y_i)$$

При биполярных значениях элементов обоих векторов расстояние Хемминга рассчитывается по формуле

$$d_H(y, d) = (1/2)(n - \sum_{i=1, n} d_i y_i)$$

Мера Хемминга равна числу несовпадающих компонент двух векторов. Она равна нулю, когда $y = d$.

Автоассоциативная сеть Хопфилда

Структура сети Хопфилда представляется в виде системы с непосредственной обратной связью выхода со входом (рис. 1). Выходные сигналы нейронов являются одновременно входными сигналами сети: $x_i(k) = y_i(k - 1)$. В классической сети Хопфилда отсутствует **автосвязь** (связь выхода нейрона с его собственным входом), что соответствует $w_{ii} = 0$, а матрица весов является симметричной: $W = W^T$. Отсутствие автосвязи и симметричность матрицы весов являются достаточными (но не необходимыми!) условиями сходимости итерационных (переходных) процессов в сети Хопфилда.

Далее в данной лекции предполагаем, что каждый нейрон имеет биполярную ступенчатую функцию активации со значениями ± 1 . Это означает, что выходной сигнал i -го нейрона определяется функцией

$$y_i = \operatorname{sgn}\left(\sum_{j=0, N} w_{ij}x_j + b_i\right)$$

где N обозначает количество нейронов, $N = n$.

Далее допустим, что порог срабатывания является компонентой вектора x . Тогда основную зависимость, определяющую сеть Хопфилда, можно представить в виде

$$y_i(k) = \operatorname{sgn}\left(\sum_{j=0, N} w_{ij}y_j(k-1)\right) \quad (1)$$

с начальным условием $y_j(0) = x_j$.

В процессе функционирования сети Хопфилда можно выделить два режима: обучения и классификации. В режиме обучения на основе известных векторов подбираются весовые коэффициенты сети. В режиме классификации при фиксированных значениях весов и вводе конкретного начального состояния нейронов возникает переходный процесс вида (1), завершающийся в одном из локальных минимумов, для которого $y(k) = y(k-1)$.

Обучение сети Хопфилда по правилу Хебба

Для одного обучающего вектора x значения весов могут быть вычислены по правилу Хебба

$$w_{ij} = (1/N)x_i x_j,$$

поскольку тогда

$$(1/N)\left(\sum_{j=1}^N x_i x_j x_j\right) = x_i$$

(вследствие биполярных значений элементов вектора x всегда $x_j^2 = (\pm 1)^2 = 1$).

При вводе большего количества обучающих векторов $x(k)$, $k = 1, 2, \dots, p$ веса w_{ij} подбираются согласно обобщенному правилу Хебба

$$w_{ij} = (1/N) \sum_{k=0}^p x_i^{(k)} x_j^{(k)}.$$

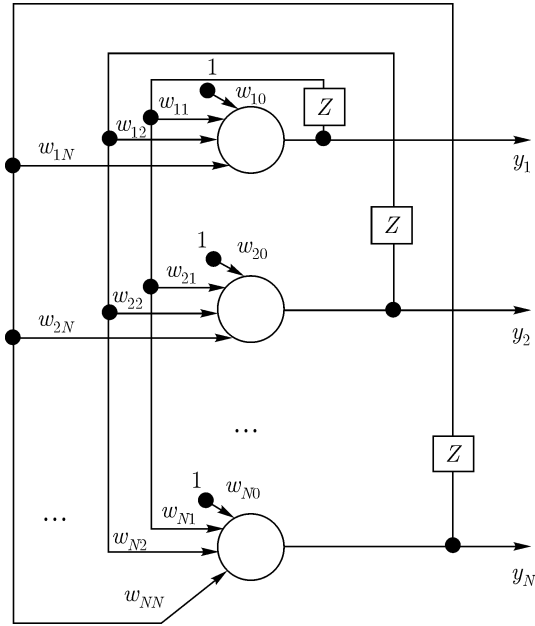


Рис. 1. Структура сети Хопфилда

Важным параметром ассоциативной памяти является ее **емкость**. Под емкостью понимается максимальное число запомненных образов, которые классифицируются с допустимой погрешностью ε_{\max} . Показано, что при использовании для обучения правила Хебба и при $\varepsilon_{\max} = 0.01$ (1% компонентов образа отличается от нормального состояния) максимальная емкость памяти составит всего лишь около 13,8% от количества нейронов, образующих ассоциативную память. Столь малая емкость обусловлена тем, что сеть Хебба хорошо запоминает только взаимно ортогональные векторы или близкие к ним.

Обучение сети Хопфилда методом проекций

Лучшие результаты, чем при использовании правила Хебба, можно получить, если для обучения использовать псевдоинверсию. В основе этого подхода лежит предположение, что при правильно подобранных весах каждый поданный на вход сети вектор вызывает генерацию самого

себя на выходе сети. В матричной форме это можно представить в виде

$$WX = X,$$

где W — матрица весов сети размерностью $N \times N$, а X — прямоугольная матрица размерностью $N \times p$, составленная из p обучающих векторов $x^{(i)}$, $i = 1, 2, \dots, p$. Решение такой линейной системы уравнений имеет вид

$$W = XX^+,$$

где знак $+$ обозначает псевдоинверсию.

Если обучающие векторы линейно независимы, последнее выражение можно упростить и представить в виде

$$W = X(X^T X)^{-1} X^T. \quad (2)$$

Здесь псевдоинверсия заменена обычной инверсией квадратной матрицы $X^T X$ размерностью $p \times p$.

Выражение (2) можно записать в итерационной форме, не требующей расчета обратной матрицы. В этом случае (2) принимает вид итерационной зависимости от последовательности обучающих векторов $x^{(i)}$, $i = 1, 2, \dots, p$:

$$y^{(i)} = (W^{(i-1)} - E)x^{(i)},$$

$$W^{(i)} = W^{(i-1)} - (y^{(i)}y^{(i)T}) / (y^{(i)T}y^{(i)})$$

при начальных условиях $W^{(0)} = 0$. В результате предъявления p векторов матрица весов сети принимает значение $W = W^{(p)}$. Описанный здесь метод называется методом проекций. Применение его увеличивает максимальную емкость сети Хопфилда до $N - 1$. Увеличение емкости обусловлено тем, что в методе проекций требование ортогональности векторов заменено гораздо менее жестким требованием их линейной независимости.

Модифицированный вариант метода проекций — метод Δ -проекции — градиентная форма алгоритма минимизации. В соответствии с этим методом веса подбираются с помощью процедуры, многократно повторяемой на всем множестве обучающих векторов:

$$W \leftarrow W + (h/N)(x^{(i)} - Wx^{(i)})x^{(i)T}, h \in (0.7, 0.9).$$

Обучающие векторы предъявляются многократно вплоть до стабилизации значений весов.

Сеть Хемминга

Сеть Хемминга включает в себя три слоя (рис.2).

Первый слой имеет однонаправленное распространение сигналов от входа к выходу и фиксированные значения весов.

Второй слой состоит из нейронов, связанных обратными связями по принципу «каждый с каждым», при этом в каждом нейроне слоя существует автосвязь (связь входа нейрона со своим собственным выходом). Разные нейроны в слое связаны отрицательной (тормозящей) обратной связью с весом $-\varepsilon$, при этом величина ε обычно обратно пропорциональна количеству образов. С собственным входом нейрон связан положительной (возбуждающей) обратной связью с весом, равным $+1$. Пороговые веса нейронов приняты равными нулю. Нейроны этого слоя функционируют в режиме *WTA*, при котором в каждой фиксированной ситуации активизируется только один нейрон, а остальные пребывают в состоянии покоя.

Выходной однонаправленный слой формирует выходной вектор, соответствующий входному вектору.

Сеть Хемминга считается гетероассоциативным запоминающим устройством с парой связанных между собой векторов (x, y) , где x и y — входной и выходной биполярные векторы сети.

Веса первого слоя соответствуют векторам $x_i, i = 1, \dots, p$, т. е.

$$w_{ij}^{(1)} = x_{ij}.$$

Аналогично, веса выходного слоя соответствуют векторам образов y_i , связанных с x_i :

$$w_{ij}^{(3)} = y_{ij}.$$

Во втором слое (*MAXNET*), функционирующем в режиме *WTA* (*Winner Takes ALL* — «Победитель забирает все»), каждый нейрон должен усиливать собственный сигнал и ослаблять сигналы остальных нейронов. Для этого принимается

$$w_{ij}^{(2)} = 1,$$

а также

$$-1/(p-1) < w_{ij}^{(2)} < 0, \quad i \neq j.$$

Для обеспечения сходимости итерационного процесса во втором слое веса

$$w_{ij}^{(2)} = -1/(p-1) + \xi,$$

где ξ — достаточно малая случайная величина, $|\xi| \ll 1/(p-1)$.

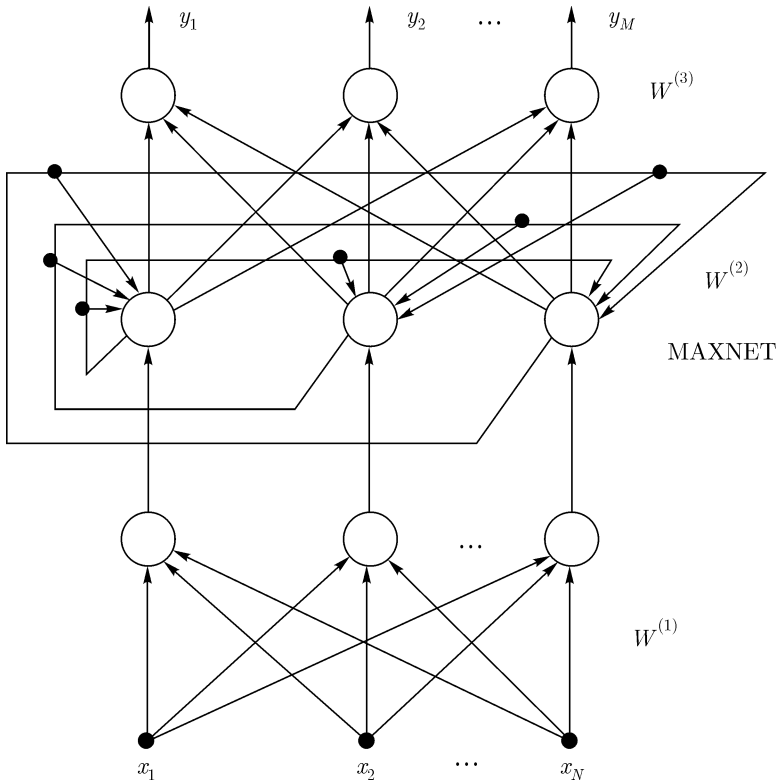


Рис. 2. Структура сети Хемминга

Нейроны первого слоя рассчитывают расстояния Хемминга $d_H(x, y)$ между поданным на вход сети вектором x и векторами весов $w_i = x_i, i = 1, 2, \dots, p$ нейронов этого слоя. Значения выходных сигналов нейронов первого слоя определяются по формуле

$$y_i^{(1)} = 1 - d_H(x, y)/N,$$

где N — число компонент вектора x .

Сигналы $y_i^{(1)}$ становятся начальными состояниями нейронов второго слоя. Этот слой определяет «победителя», т. е. нейрон, выходной сигнал которого близок к 1. Такой нейрон указывает на вектор образа с минимальным расстоянием Хемминга до входного вектора x . Функция ак-

тивации для нейронов второго слоя задается выражением

$$f(y) = \begin{cases} y, & \text{если } y > 0, \\ 0, & \text{если } y < 0. \end{cases}$$

Итерационный процесс во втором слое завершается, когда активным остается только один нейрон (победитель), тогда как остальные нейроны пребывают в нулевом состоянии. Победитель через веса $w_{ij}^{(3)}$ линейных нейронов выходного слоя представляет вектор y_i , который соответствует вектору x_i , признанному вторым слоем ближайшим к входному вектору x .

Достоинством сети Хемминга считается небольшое количество взвешенных связей между нейронами. Многочисленные эксперименты доказали, что сеть Хемминга дает лучшие результаты, чем сеть Хопфилда. Единственная проблема, связанная с сетью Хемминга, проявляется в случае, когда зашумленные образы находятся на одинаковом (в смысле Хемминга) расстоянии от двух или более эталонов. В этом случае выбор сетью Хемминга одного из эталонов становится случайным.

Двунаправленная ассоциативная память

Обобщением сети Хопфилда на случай двухслойной рекуррентной структуры, позволяющей кодировать множества двух взаимосвязанных векторов, считается двунаправленное ассоциативное запоминающее устройство, называемое ВАМ (Bidirectional Associative Memory) (рис. 3). Сигналы распространяются в двух направлениях. Если в первом цикле сигналы вначале проходят в одну сторону для задания состояний нейронов-получателей, то в следующем цикле эти нейроны сами становятся источниками, высылающими сигналы в обратную сторону. Процесс повторяется до достижения состояния равновесия.

Функция активации нейронов имеет пороговый характер. Для обеспечения лучших характеристик сети на этапе обучения используются только биполярные сигналы. Матрица весов W , связывающая обе части сети, является действительной и в общем случае несимметричной. При прямом распространении сигналов веса описываются матрицей W , а при обратном — матрицей W^T .

Пусть входные обучающие данные представляют собой множество пар $\{(x_i, y_i), i = 1, 2, \dots, m\}$ биполярных векторов. На основе этого множества формируется матрица

$$W = \sum_{i=1}^n x_i^T y_i.$$

В результате процесса двунаправленной обработки сигналов формируются два стабильных вектора x_f и y_f , удовлетворяющих уравнениям

$$\begin{aligned} y_f &= f(x_f W), \\ x_f &= f(y_f W^T) = f(W_y f^T). \end{aligned}$$

Каждой промежуточной точке (x_k, y_k) можно сопоставить энергетическую функцию

$$E_k = -x_k W y_k^T,$$

которая убывает при каждом изменении состояния вплоть до достижения локального минимума

$$E_{min} = -x_f W y_f^T, \quad f \in 1, 2, \dots, m.$$

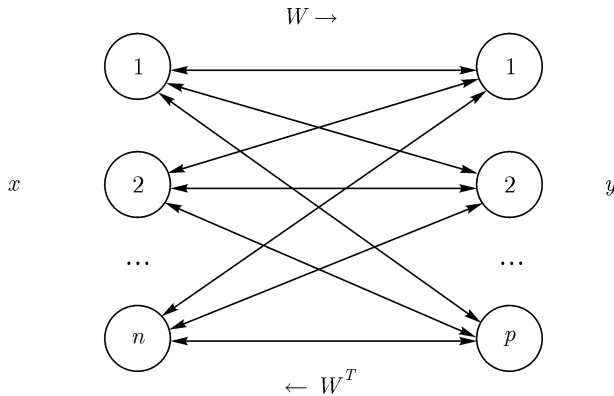


Рис. 3. Структура сети ВАМ

В режиме распознавания при начальных значениях векторов, совпадающих с использованными при обучении, сеть распознает их безошибочно. При искажении векторов x и y сеть ВАМ не всегда способна откорректировать эти векторы и распознает их с определенными погрешностями. Если размерности векторов x и y обозначить соответственно n и p , то удовлетворительное качество распознавания можно получить при выполнении зависимости

$$m < \sqrt{\min(n, p)},$$

где m — число запоминаемых в сети ВАМ пар векторов.

Лекция 11. Решение задач комбинаторной оптимизации рекуррентными сетями

Рассматривается решение задачи коммивояжера сетью Хопфилда и машиной Больцмана. Оцениваются параметры функции энергии нейронных сетей, обеспечивающие решение задачи коммивояжера.

Ключевые слова: сеть Хопфилда, задача коммивояжера, функция вычислительной энергии, машина Больцмана, консенсус.

Решение задачи коммивояжера сетью Хопфилда

Рассмотрим задачу коммивояжера для n городов. Известны расстояния d_{XY} между каждой парой городов X, Y ; коммивояжер, выходя из одного города, должен посетить $n - 1$ других городов, заходя по одному разу в каждый, и вернуться в исходный. Требуется определить порядок обхода городов, при котором общее пройденное расстояние минимально.

Пусть сеть Хопфилда состоит из $N = n^2$ нейронов, а состояние нейронов описывается двойными индексами v_{Xi} , где индекс X связан с именем города, i — с позицией города в маршруте коммивояжера. Запишем **функцию вычислительной энергии** для сети, предназначенной решать задачу коммивояжера. В ней состояние с наименьшей энергией должно соответствовать самому короткому маршруту. Функция энергии должна удовлетворять следующим требованиям:

1) должна поддерживать устойчивое состояние в форме матрицы

$$V = \{v_{Xi}\}, \quad (1)$$

в которой строки соответствуют городам, столбцы — их номерам в маршруте; в каждой строке и каждом столбце только одна единица, остальные нули;

2) из всех решений вида (1) функция энергии должна поддерживать те, которые соответствуют коротким маршрутам.

Таким требованиям удовлетворяет функция энергии в виде:

$$E = (A/2) \sum_X \sum_i \sum_{j \neq i} v_{Xi} v_{Xj} + (B/2) \sum_i \sum_X \sum_{Y \neq X} v_{Xi} v_{Xj} + \\ + (C/2) \left(\sum_X \sum_i v_{Xi} - n \right)^2 + (D/2) \sum_X \sum_{X \neq Y} \sum_i d_{XY} v_{Xi} (v_{Y,i+1} + v_{Y,i-1}), \quad (2)$$

где первые три члена поддерживают первое требование, четвертый член — второе. Первый член равен нулю, если каждая строка X содержит не более одной единицы. Второй равен нулю, если каждый столбец i содержит не более одной единицы. Третий равен нулю, если в матрице всего n единиц. Короткие маршруты поддерживает четвертый член. В нем индексы i берутся по модулю n для того, чтобы показать, что n -й город соседствует в маршруте с $(n - 1) - \text{м}$, т. е. $v_{Y,n+j} = v_{Y,j}$. Четвертый член численно равен длине маршрута. Каноническое выражение для функции вычислительной энергии имеет вид

$$E = -(1/2) \sum_X \sum_i \sum_Y \sum_j W_{X_i,Y_j} v_{X_i} v_{Y_j} - \sum_{xi} I_{X_i} v_{X_i} \quad (3)$$

Из (2) и (3) получаем веса сети Хопфилда:

$$W_{X_i,Y_j} = -A\delta_{XY}(1 - \delta_{ij}) - B\delta_{ij}(1 - \delta_{XY}) - C - Dd_{XY}(\delta_{j,i+1} + \delta_{j,i-1}), \\ I_{X_i} = Cn.$$

Здесь δ — символ Кронекера.

Моделирование работы сети Хопфилда показало, что лучшее по качеству решение дает сеть, нейроны которой имеют сигмовидную характеристику, а сеть, в которой нейроны имеют ступенчатые переходы, приходила к финальным состояниям, соответствующим маршрутам немного лучшим, чем случайные. Многочисленные исследования показывают, что качество решения задачи минимизации функции энергии (2) существенно зависит от выбора производной сигмовидной униполярной функции активации нейрона в окрестности нуля. При малой величине производной минимумы энергии оказываются в центре гиперкуба решений (некорректное решение), при большой величине производной сеть Хопфилда попадает в вершину гиперкуба, соответствующую локальному минимуму функции энергии. Кроме того, на качество решения существенное влияние оказывает выбор коэффициентов A, B, C, D . Поиск методов оптимального выбора этих коэффициентов является в настоящее время предметом интенсивных исследований.

Машина Больцмана

Математической основой для решения комбинаторных оптимизационных задач на машине Больцмана является алгоритм, моделирующий затвердевание жидкостей или расплавов (алгоритм имитации отжига). Он базируется на идеях из двух различных областей: статистической физики и комбинаторной оптимизации. Машина Больцмана (МБ) способна реализовать этот алгоритм параллельно и асинхронно. МБ задается четверкой $B = (N, E, W, V_0)$, N — число нейронов, $E = \{(i, j)\}$ —

множество связей между нейронами, при этом все автосвязи принадлежат этому множеству, т. е. $(i, i) \in E$. Каждый нейрон может иметь состояние 0 или 1. Состояние V_k МБ определяется состояниями нейронов $V_k = (v_1^k, \dots, v_N^k)$, V_0 — начальное состояние. Каждая связь (i, j) имеет вес w_{ij} — вещественное число, множество связей — W . Связь (i, j) называется активной в состоянии V_k , если $v_i^k v_j^k = 1$. Вес связи (i, j) интерпретируется как количественная мера желательности, чтобы эта связь была активной. При $w_{ij} \gg 0$ — активность очень желательна, при $w_{ij} \ll 0$ — активность очень нежелательна. Как и в модели Хопфилда, связи в МБ симметричны, т. е. $w_{ij} = w_{ji}$.

Функция консенсуса

Для состояния V_k МБ вводится понятие консенсуса

$$C_k = \sum_{i,j} w_{ij} v_i^k v_j^k.$$

Каждая связь в этой сумме учитывается один раз. Консенсус C_k интерпретируется как количественная мера желательности, чтобы все связи (i, j) в состоянии V_k были активны. Для состояния V_k определяется множество соседей $V^{(k)}$. Соседнее состояние $V_{k(i)} \in V^{(k)}$ получается из V_k при изменении состояния нейрона i ,

$$V_j^{k(i)} = \begin{cases} v_j^k & \text{если } j \neq i \\ 1 - v_j^k & \text{если } j = i \end{cases}$$

Разница консенсусов соседних состояний V_k и $V_{k(i)}$ равна

$$\Delta C_{kk(i)} = C_{k(i)} - C_k = (1 - 2v_i^k) \left(\sum_{(i,j) \in E(i)} w_{ij} v_i^k + w_{ii} \right),$$

где $E(i)$ — множество связей нейрона i . Видно, что $\Delta C_{kk(i)}$ для всех $V_{k(i)} \in V^{(k)}$ могут вычисляться параллельно.

Максимизация консенсуса

Переход МБ из одного состояния в другое с максимизацией консенсуса происходит путем выполнения пошаговой процедуры. На каждом ее шаге выполняется испытание, состоящее из двух частей:

- 1) для данного состояния V_k генерируется соседнее $V_{k(i)}$,
- 2) оценивается, может ли быть принято состояние $V_{k(i)}$, если может, то результат испытания — $V_{k(i)}$, иначе V_k .

Состояние $V_{k(i)}$ принимается с вероятностью

$$P_{kk(i)}(t) = 1/[1 + \exp(\Delta C_{kk(i)}/t)], \quad (4)$$

где $t \geq 0$ — управляющий параметр («температура»).

Процесс максимизации консенсуса начинается с высокого значения t_0 параметра t и случайно выбранного начального состояния V_0 . В течение процесса параметр t уменьшается от t_0 до 0. По мере того как t приближается к нулю, нейроны все реже изменяют свои состояния, и наконец, МБ стабилизируется в финальном состоянии. Практически, МБ стабилизируется в состоянии, соответствующем локальному максимуму консенсуса, который близок (или равен) глобальному. Сходимостью МБ управляют следующие параметры:

1. Начальное значение параметра t для каждого нейрона i

$$t_0^{(i)} = \sum_{(i,j) \in E(i)} |w_{ij}| + |w_{ii}|.$$

2. Правило понижения t

$$t_{j+1}^{(i)} = \alpha t_j^{(i)},$$

где α — положительное число, меньшее единицы, но близкое к ней.

3. Число L испытаний, которые проводятся без изменения t (L — функция от N).

4. Число M последовательных испытаний, не приводящих к изменению состояния машин (M — функция от N), как критерий завершения процесса.

Синхронное и асинхронное функционирование машины Больцмана

Для выполнения синхронного процесса все множество нейронов разбивается на непересекающиеся подмножества $\{M_1, \dots, M_m\}$, такие, что нейроны, попавшие в одно подмножество, не связаны друг с другом. Тогда на каждом такте синхронизации элементы случайно выбранного подмножества M_i могут одновременно изменять свои состояния в соответствии с заданной вероятностью.

В асинхронном параллельном процессе все нейроны могут изменять свои состояния только в зависимости от величины вероятности. Практически асинхронный параллелизм может быть выполнен следующим образом. Случайно выбирается подмножество M , содержащее $q = 2N/3$ нейронов. Для каждого нейрона из этого подмножества устанавливается

состояние в соответствии с $P_{kk(i)}(t)$. Получившееся в результате состояние есть результат одного асинхронного шага.

Решение задачи коммивояжера машиной Больцмана

Общий подход к программированию комбинаторных оптимизационных задач состоит в следующем:

каждое решение представляется набором $\{x_1, \dots, x_N\}$, $x_i \in \{0, 1\}$, N — число нейронов в сети, x_i — состояние нейрона. Структура связей и веса выбираются так, что:

R1. Все локальные максимумы функции консенсуса соответствуют приемлемым решениям задачи;

R2. Чем лучше приемлемое решение, тем больше консенсус соответствующего состояния машины Больцмана.

Перефразируем для МБ задачу коммивояжера.

R1. Состояние МБ соответствует локальному максимуму функции консенсуса, если и только если это состояние соответствует приемлемому маршруту.

R2. Чем короче маршрут, тем выше консенсус соответствующего состояния МБ.

Каждый нейрон соответствует элементу матрицы $n \times n$, состояния нейронов обозначаются v_{Xi} (n — число городов). Функция консенсуса

$$C_k = \sum_{(Xi, Yj)} w_{Xi, Yj} v_{Xi}^k v_{Yj}^k.$$

Множество связей в сети определяется как объединение трех непересекающихся подмножеств:

E_d — множество связей, несущих информацию о расстояниях между городами,

$$E_d = \{(Xi, Yj) | (X \neq Y) \wedge (i = (j + 1) \bmod n)\};$$

E_i — множество ингибиторных (запретительных) связей,

$$E_i = \{(Xi, Yj) | (i \neq j) \wedge (X = Y) \vee (i = j) \wedge (X \neq Y)\};$$

E_b — множество связей смещений,

$$E_b = \{(Xi, Yj) | (X = Y) \wedge (i = j)\}.$$

Здесь $X, Y, i, j = 1, \dots, n$. Общее число связей равно $2n^3 - n^2$.

Ингибиторные связи гарантируют, что, в конце концов, ни в одной строке и ни в одном столбце не будет более одной единицы. Связи смещений гарантируют, что хотя бы по одной единице есть в каждом столбце и в каждой строке. Таким образом, связи E_i и E_b гарантируют выполнение ограничений в задаче и веса их дают одинаковые вклады в консенсусы для всех приемлемых маршрутов.

Связь $(Xi, Yj) \in E_d$ активна только в том случае, когда в маршруте есть прямой путь из города X в город Y . Вес связи $(Xi, Yj) \in E_d$ равен расстоянию между городами X и Y с отрицательным знаком. Следовательно, для данного маршрута отрицательный вклад связи из E_d в консенсус пропорционален длине пути, поэтому максимизация функции консенсуса соответствует минимизации длины маршрута.

Доказано, что для консенсуса C_k выполняются требования $R1$ и $R2$, если и только если веса связей выбраны следующим образом:

$$\begin{aligned}\forall (Xi, Yj) \in E_d : w_{Xi, Yj} &= -d_{XY}, \\ \forall (Xi, Yj) \in E_i : w_{Xi, Yj} &< -\min(\mu_X, \mu_Y), \\ \forall (Xi, Yj) \in E_b : w_{Xi, Yj} &> \mu_X,\end{aligned}$$

где

$$\mu_X = \max\{d_{XP} + d_{XQ} | P, Q = 1, \dots, n \wedge (P \neq Q)\}.$$

При $d = 0,95, L = 10, M = 100$ было проведено 100 испытаний для $n = 10$ и 25 испытаний для $n = 30$ при различных начальных состояниях МБ. Для $n = 10$ получено оптимальное решение, для $n = 30$ получено решение на 14% хуже оптимума. Вероятностный механизм функционирования МБ дает возможность получать на ней несколько лучшие результаты, чем на модели Хопфилда.

Лекция 12. Рекуррентные сети на базе персептрона

Рассматриваются многослойные рекуррентные сети (персептронная сеть с обратной связью, рекуррентная сеть Эльмана, сеть RTRN) и их использование для идентификации динамических объектов.

Ключевые слова: обратная связь, сети RMLP, Эльмана, RTRN, моделирование временных рядов, задача прогнозирования.

Введение

Многослойные рекуррентные сети представляют собой развитие односторонних сетей персептронного типа за счет добавления в них соответствующих обратных связей. **Обратная связь** может исходить либо из выходного, либо из скрытого слоя нейронов. В каждом контуре такой связи присутствует элемент единичной задержки, благодаря которому поток сигналов может считаться односторонним (выходной сигнал предыдущего временного цикла рассматривается как априори заданный, который просто увеличивает размерность входного вектора сети). Представленная подобным образом рекуррентная сеть, с учетом способа формирования выходного сигнала, функционирует как односторонняя персептронная сеть. Тем не менее, алгоритм обучения такой сети, адаптирующий значения синаптических весов, является более сложным из-за зависимости сигналов в момент времени t от их значений в предыдущие моменты и соответственно из-за более громоздкой формулы для расчета вектора градиента.

При обсуждении рекуррентных сетей, в которых в качестве выходного элемента используется многослойный персептрон, рассмотрим наиболее известные структуры **сетей RMLP, RTRN, Эльмана**.

Персептронная сеть с обратной связью

Один из простейших способов построения рекуррентной сети на базе односторонней НС состоит во введении в персептронную сеть обратной связи. В дальнейшем мы будем сокращенно называть такую сеть RMLP (англ.: *Recurrent MultiLayer Perceptron* — рекуррентный многослойный персептрон). Ее обобщенная структура представлена на рис. 1 (z — единичные элементы запаздывания).

Это динамическая сеть, которая характеризуется запаздыванием входных и выходных сигналов, объединяемых во входной вектор сети.

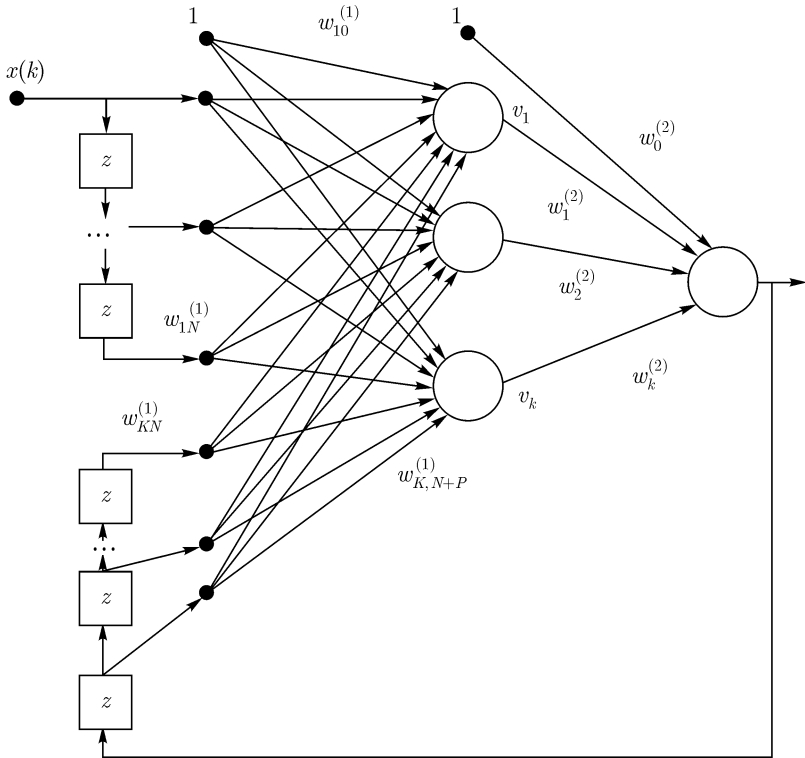


Рис. 1. Структура сети RMLP

Рассуждения будут касаться только одного входного узла $x(k)$ и одного выходного нейрона, а также одного скрытого слоя. Такая система реализует отображение:

$$y(k+1) = f(x(k), x(k-1), \dots, x(k-(N-1)), y(k-1), \dots, y(k-P)) \quad (1)$$

где $N-1$ — количество задержек входного сигнала, а P — количество задержек выходного сигнала. Обозначим K количество нейронов в скрытом слое. В этом случае сеть RMLP можно характеризовать тройкой чисел (N, P, K) . Подаваемый на вход сети вектор x имеет вид:

$$x(k) = [1, x(k), x(k-1), \dots, x(k-(N-1)), \\ y(k-P), y(k-P+1), \dots, y(k-1)]^T.$$

Допустим, что все нейроны имеют сигмоидальную функцию активи-

вазии. Обозначим u_i взвешенную сумму сигналов i -го нейрона скрытого слоя, а g — взвешенную сумму сигналов выходного нейрона. При введенных обозначениях выходные сигналы конкретных нейронов описываются зависимостями

$$\begin{aligned} u_i &= \sum_{j=0}^{N+P} w_{ij}^{(1)} x_j \\ v_i &= f(u_i) \\ g &= \sum_{i=0}^K w_i^{(2)} v_i \\ y &= f(g) \end{aligned}$$

Сеть RMLP повсеместно применяется для моделирования динамических процессов в режиме «онлайн». Типичным примером ее приложения может служить имитация нелинейных динамических объектов, для которых сеть RMLP выступает в роли модели, а алгоритм уточнения весов — в роли процедуры идентификации параметров этой модели (рис. 2). Идентифицированная модель может в последующем использоваться для управления данным объектом. Именно по этой причине сети RMLP наиболее популярны для имитации систем управления машинами, устройствами и динамическими процессами.

В результате сравнения выходного сигнала модели $y(k)$ с выходным сигналом динамического объекта $d(k)$ рассчитывается значение погрешности $e(k) = y(k) - d(k)$, управляющей процессом уточнения параметров нейронной сети. Символом M на рис. 2 обозначен коэффициент усиления модуля, масштабирующего выходной сигнал сети $y(k)$ таким образом, чтобы его динамический уровень лежал в том же диапазоне, что и уровень выходного сигнала динамического объекта $d(k)$.

Рекуррентная сеть Эльмана

Рекуррентная сеть Эльмана характеризуется частичной рекуррентностью в форме обратной связи между скрытым и входным слоем, реализуемой с помощью единичных элементов запаздывания z . Обобщенная структура этой сети представлена на рис. 3.

Каждый скрытый нейрон имеет свой аналог в контекстном слое, образуя совместно с внешними входами сети входной слой. Выходной слой состоит из нейронов, односторонне связанных только с нейронами скрытого слоя, подобно сети RMLP. Обозначим внутренний вектор возбуждения сети x (в его состав входит также единичный сигнал поляризации), состояния скрытых нейронов — $v \in R^K$, а выходные сигналы

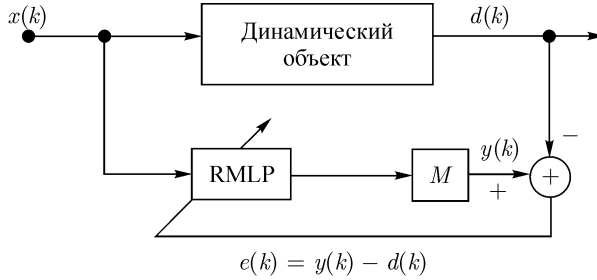


Рис. 2. Схема включения сети RMLP при решении задачи идентификации

сети — $y \in R^M$. При таких обозначениях входной вектор сети в момент t имеет форму

$$X(k) = [x_0(k), x_1(k), \dots, x_N(k), v_1(k-1), v_2(k-1), \dots, v_K(k-1)].$$

Веса синаптических связей первого (скрытого) слоя сети обозначим $w_{ij}^{(1)}$, а второго (выходного) слоя — $w_{ij}^{(2)}$. Если взвешенную сумму i -го нейрона скрытого слоя обозначить u_i , а его выходной сигнал — v_i , то

$$u_i(k) = \sum_{j=0}^{N+K} w_{ij}^{(1)} x_j(k),$$

$$v_i(k) = f_1(u_i(k)).$$

Веса $w_{ij}^{(1)}$ образуют матрицу $W^{(1)}$ синаптических связей скрытого слоя, а $f_1(u_i)$ — функция активации i -го нейрона этого слоя. Аналогично можно обозначить взвешенную сумму i -го нейрона выходного слоя g_i , а соответствующий ему выходной сигнал сети — y_i . Эти сигналы описываются формулами

$$g_i(k) = \sum_{j=0}^K w_{ij}^{(2)} v_j(k),$$

$$y_i(k) = f_2(g_i(k)).$$

В свою очередь, веса $w_{ij}^{(2)}$ образуют матрицу $W^{(2)}$, описывающую синаптические связи нейронов выходного слоя; $f_2(g_i)$ — функция активации i -го нейрона выходного слоя.

Сеть Эльмана естественным образом предназначена для **моделирования временных рядов**. В частности, она решает задачу прогнозирования

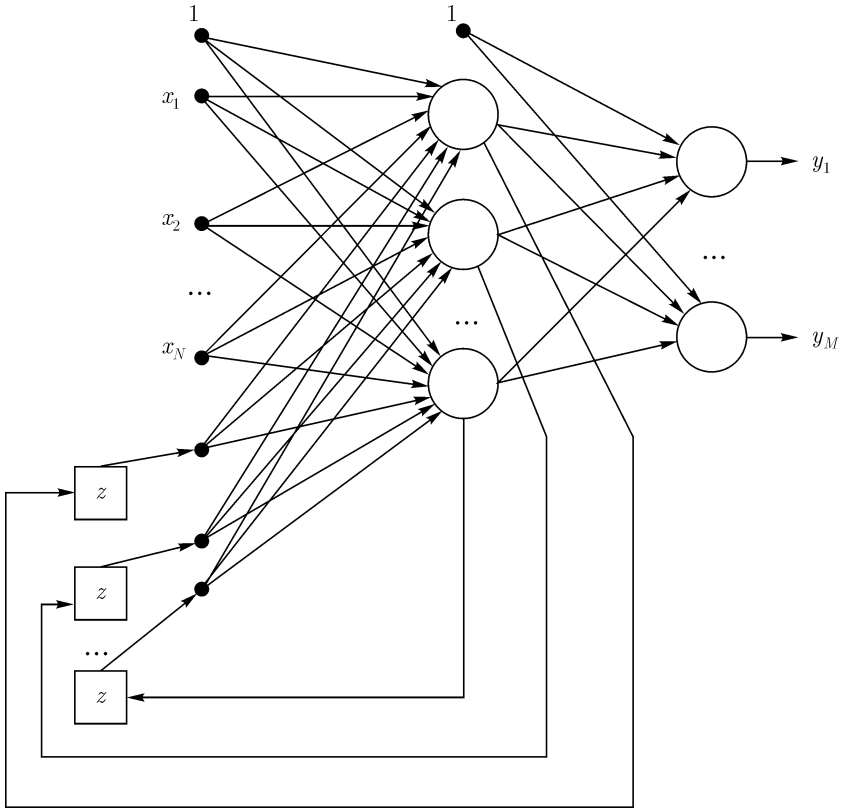


Рис. 3. Структура сети Эльмана

амплитуды сигнала на основе текущего значения входного сигнала и запомненных значений из предыдущего временного цикла. Задача прогноза временных рядов возникает в финансовой области: прогноз котировок товаров и ценных бумаг на бирже, курсов валют, показателей банковской деятельности. В экономике прогноз может быть связан, например, с анализом уровня производства в различных отраслях промышленности и сельского хозяйства, процента трудовой занятости населения, роста средней заработной платы. Проблема прогноза временных рядов возникает в многочисленных экологических задачах и самых разнообразных технических приложениях.

Для прогноза временных рядов могут применяться статистические методы. В этом случае должна быть построена динамическая модель дан-

ных (например, регрессионная модель) изучаемого явления. Для простейших задач такая модель может быть построена известными методами. Однако для практических задач, примеры которых приведены выше, построение подобной динамической модели представляет собой сложную аналитическую задачу. Эти приложения связаны обычно не со скалярными, а с векторными временными рядами. Например, в финансовой сфере прогноз котировок товара зависит от вектора динамических данных, которые включают цены открытия и закрытия торговой сессии, среднюю и максимальную цены торговой сессии, суммарный уровень заявок, валютные курсы и пр.

В том случае, когда адекватной математической модели изучаемых временных рядов не существует, удобным инструментом для решения задачи прогноза является нейросетевой экстраполятор динамических данных.

Задача прогноза векторного временного ряда ставится следующим образом:

— задана реализация временного ряда $x_j = (x_{j1}, x_{j2}, \dots, x_{jM})$, $j = 1, 2, \dots, T$, на интервале времени $[\Delta, T\Delta]$ с постоянным интервалом дискретности Δ ;

— требуется построить оценку значения временного ряда (обычно одной его координаты) в момент времени $(T + t_{pr})\Delta$, где $t_{pr}\Delta$ — заданное время прогноза.

Из логических соображений или путем статистического анализа имеющейся реализации можно установить, сколько предшествующих значений относительно произвольного текущего момента времени $j\Delta$ определяюще связаны с прогнозируемым значением. Это означает, что если представить прогнозируемое значение $y_j = x_{j+t, m}$ m -ой координаты вектора x как функцию его предшествующих измерений:

$$y_j = \Phi(x_j, x_{j-1}, \dots, x_{j-q+1}),$$

то выбор значения q устанавливает «память» экстраполятора. Значение q определяет также входной вектор для нейронной сети, которая строится для решения задачи прогноза. Размерность этого вектора равна $M * q$.

Таким образом, задача прогноза данных на нейронной сети сведена к задаче воспроизведения функции многих переменных $\Phi(x_j, x_{j-1}, \dots, x_{j-q+1})$ по данным обучающей выборки.

Сеть RTRN

Среди рекуррентных сетей особого внимания заслуживает сеть типа RTRN (англ.: *Real Time Recurrent Network*), предложенная Р. Вильямсом и

Д. Зипсером и предназначенная для обработки сигналов в реальном времени. Сеть RTRN — частный случай сети Эльмана.

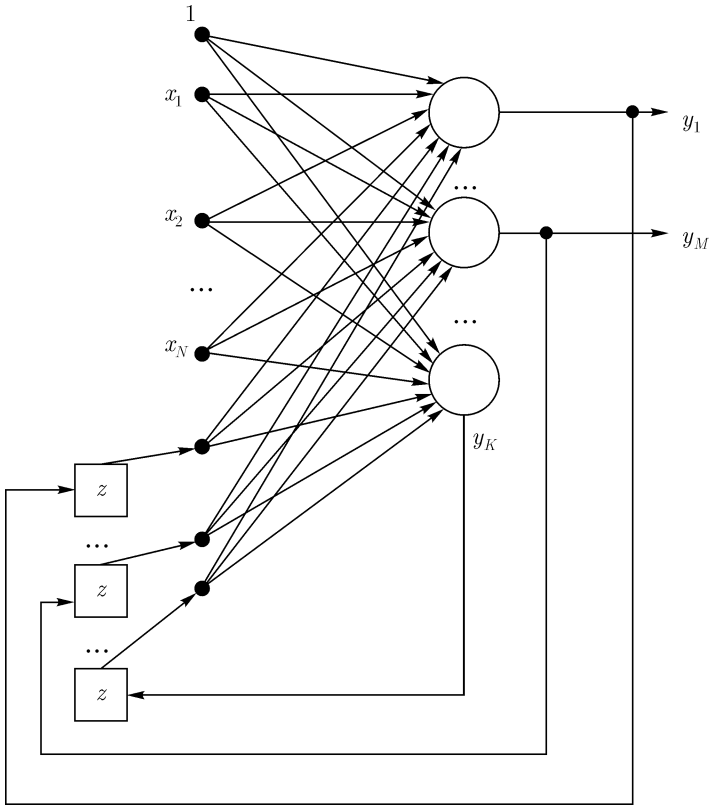


Рис. 4. Структура сети RTRN

Обобщенная структура сети представлена на рис. 4. Сеть содержит N входных узлов, K скрытых нейронов и K соответствующих им узлов контекстного слоя. Из K скрытых нейронов только M составляют выход сети. Обозначим взвешенную сумму i -го нейрона скрытого слоя u_i , а выход этого нейрона — y_i . Вектор $x(k)$ и смещенный (задержанный) на один цикл вектор $y(k-1)$ образуют расширенный вектор активации $x(k)$, возбуждающий нейроны сети:

$$x(k) = [1, x_1(k), x_2(k), \dots, x_N(k), y_1(k-1), \dots, y_K(k-1)]^T.$$

После описания входного вектора сети в момент t можно определить состояние всех нейронов согласно зависимостям:

$$u_i(k) = \sum_{j=0}^{N+K} w_{ij} x_j(k), \quad (2)$$

$$y_i(k) = f(u_i(k)). \quad (3)$$

причем $f(\)$ обозначает непрерывную функцию активации нейрона (как правило, сигмоидальную). На рис. 4 видно, что сеть RTRN представляет собой частный случай сети Эльмана, в которой веса выходного слоя постоянны и равны дельте Кронекера, т.е. $w_{ij} = \delta_{ij} = 1$ для $i = j$ или 0 для $i \neq j$. В этом случае можно применять алгоритм обучения Вильяма—Зипсера.

1. Выбрать случайные начальные значения весов сети, составляющих матрицу W и равномерно распределенных в заданном интервале (обычно в диапазоне от -1 до 1).

2. Рассчитать состояние всех K нейронов для очередного момента $t = 0, 1, 2, \dots$ с использованием формул (1) и (2). На этой основе можно определить входной вектор $x(k)$, возбуждающий нейроны в момент t .

3. Рассчитать значения

$$dy_i(k)/dw_{ab} = (df_1(u_i)/du_i)[\delta_{ja} x_b + \sum_{k=1}^K (dy_i(k-1)/dw_{ab})w_{i,k+N}]$$

4. Уточнить значения весов по алгоритму наискорейшего спуска согласно формуле

$$w_{ab}(k+1) = w_{ab}(k) - \alpha \sum_{i=1}^K [y_i(k) - d_i(k)](dy_i(k)/dw_{ab})$$

для $a = 1, 2, \dots, K$ и $b = 0, 1, 2, \dots, N + K$.

Шаги (2–4) повторять вплоть до стабилизации значений всех весов сети.

Лекция 13. Самоорганизация (самообучение) нейронных сетей

Рассматриваются: метод динамических ядер в классификации без учителя, алгоритмы обучения сетей с самоорганизацией и их применение к компрессии данных и прогнозированию.

Ключевые слова: мера близости, сеть Кохонена, погрешность квантования, векторное квантование, кодовая таблица, алгоритм K -усреднений, самоорганизующаяся карта признаков, функция соседства, компрессия данных, задача прогнозирования.

Классификация без учителя

Задан набор объектов, каждому объекту поставлен в соответствие вектор значений признаков (строка таблицы). Требуется разбить эти объекты на классы эквивалентности. Для каждого нового объекта нужно:

1. Найти класс, к которому он принадлежит.
2. Использовать новую информацию, полученную об этом объекте, для исправления (коррекции) правил классификации.

Отнесение объекта к классу проводится путем его сравнения с типичными элементами разных классов и выбора из них ближайшего.

Простейшая **мера близости** объектов — квадрат евклидова расстояния между векторами значений их признаков (чем меньше расстояние, тем ближе объекты). Соответствующее определение признаков типичного объекта — среднее арифметическое значение признаков по выборке, представляющей класс. Другая мера близости, возникающая при обработке сигналов, изображений и т. п. — квадрат коэффициента корреляции (чем он больше, тем ближе объекты). Возможны и иные варианты.

Если число классов m заранее определено, то задачу классификации без учителя можно поставить следующим образом.

Метод динамических ядер в классификации без учителя

Пусть задана выборка преобразованных векторов данных $\{x\} \subseteq E$, E — пространство векторов данных. Каждому классу будет соответствовать некоторое ядро $w \subseteq W$, W — пространство ядер.

Для любых $x \in E$ и $w \in W$ определим меру близости $d(x, w)$, а для каждого набора из k ядер w_1, \dots, w_k и любого разбиения $\{x\}$ на k классов $\{x\} = P_1 \cup P_2 \cup \dots \cup P_k$ определим критерий качества

$$D = D(w_1, \dots, w_k, P_1, \dots, P_k) = \sum_{i=1}^k \sum_{x \in P_i} d(x, w_i). \quad (1)$$

Требуется найти набор w_1, \dots, w_k и разбиение P_1, \dots, P_k , минимизирующие D . Шаг алгоритма разбиваем на 2 этапа:

1) Для фиксированного набора ядер w_1, \dots, w_k ищем минимизирующее D разбиение P_1, \dots, P_k ; оно дается следующим решающим правилом: $x \in P_i$, если $d(x, w_i) < d(x, w_j)$ при $i \neq j$ (когда для x минимум $d(x, w_i)$ достигается при нескольких значениях i , выбор между ними может быть сделан произвольно).

2) Для каждого $P_i, i \in 1, \dots, k$, полученного на первом этапе, отыскивается $w_i \in W$, минимизирующее критерий качества

$$D_i = \sum_{x \in P_i} d(x, w_i).$$

Начальные значения $w_1, \dots, w_k, P_1, \dots, P_k$ выбираются произвольно либо по какому-нибудь эвристическому правилу. Если ядру w_i ставится в соответствие элемент сети, вычисляющей по входному сигналу x функцию $d(x, w_i)$, то решающее правило для классификации дается интерпретатором «проигравший забирает все»: элемент x принадлежит классу P_i , если выходной сигнал i -го элемента $d(x, w_i)$ меньше всех остальных. Мера близости d выбирается такой, чтобы легко можно было найти ядро w_i , минимизирующее D_i для данного P_i .

В простейшем случае пространство ядер W совпадает с E , а $d(x, w_i)$ — положительно определенная квадратичная форма от $x - w_i$, например, квадрат евклидова расстояния. Тогда ядро w_i , минимизирующее D_i , есть центр масс класса P_i :

$$w_i = (1/|P_i|) \sum_{x \in P_i} x,$$

где $|P_i|$ — число элементов в P_i .

Пусть векторы пространства E нормированы. Тогда

$$(x, x) = (w_i, w_i) = 1. \quad (2)$$

Так как $d(x, w_i) = (x - w_i, x - w_i) = (x, x) - 2(x, w_i) + (w_i, w_i)$, то с учетом (2) упрощается решающее правило, разделяющее классы:

$$x \in P_i, \text{ если } (x, w_i) > (x, w_j) \text{ при } i \neq j,$$

поскольку минимум $d(x, w_i)$ достигается при максимуме (x, w_i) . Такое решающее правило реализуется с помощью k сумматоров, вычисляющих (x, w_i) , и интерпретатора, выбирающего сумматор с максимальным выходным сигналом. Номер этого сумматора и есть номер класса, к которому относится x .

Задача поиска ядра w_i для класса P_i превращается в поиск вектора w , максимизирующего

$$D_i = \sum_{x \in P_i} (x, w).$$

Этот максимум достигается в точке

$$w = \sum_{x \in P_i} x / \left\| \sum_{x \in P_i} x \right\|$$

где $\| \dots \|$ — евклидова норма.

В тех простейших случаях, когда ядро класса точно определяется как среднее арифметическое (или нормированное среднее арифметическое) элементов класса, а решающее правило основано на сравнении выходных сигналов линейных адаптивных сумматоров, нейронную сеть, реализующую метод динамических ядер, называют **сетью Кохонена**. В определение ядра w_i для сетей Кохонена входят суммы $\sum_{x \in P_i} x$. Это позволит накапливать новые динамические ядра, обрабатывая по одному примеру и пересчитывая w_i после получения в P_i нового примера.

Если число классов заранее не определено, то полезен критерий слияния классов: классы Y_i и Y_j сливаются, если расстояние между их ядрами меньше, чем среднее расстояние от элемента класса до ядра в одном из них:

$$(y^i, y^j) < \max\left[\left(\frac{1}{|Y_i|}\right) \sum_{x \in Y_i} \rho(x, y^i), \left(\frac{1}{|Y_j|}\right) \sum_{x \in Y_j} \rho(x, y^j)\right],$$

где $|Y|$ — число элементов в Y . Использовать критерий слияния классов можно так: сначала принимаем гипотезу о достаточном числе классов, строим их, минимизируя D , затем некоторые Y_i объединяем, повторяем минимизацию D с новым числом классов и т. д.

Алгоритмы обучения сетей с самоорганизацией

Целью обучения сети с самоорганизацией на основе конкуренции нейронов считается такое упорядочение нейронов (подбор значений их весов), которое минимизирует значение ожидаемого искажения, оцениваемого погрешностью аппроксимации входного вектора x значениями

весов нейрона-победителя. При p входных векторах x и применении евклидовой метрики эта погрешность, называемая также погрешностью квантования, может быть выражена в виде

$$E = (1/p) \sum_{i=1}^p \|x^i - w_{win}\|^2, \quad (3)$$

где w_{win} — вес нейрона-победителя при предъявлении вектора x^i .

Этот подход также называется **векторным квантованием** (англ. *Vector Quantization* — *VQ*) или кластеризацией. Номера нейронов-победителей при последовательном предъявлении векторов x^i образуют так называемую **кодovou таблицу**. При классическом решении задачи кодирования применяется алгоритм K -усреднений (англ. *K-means*), носящий имя обобщенного алгоритма Ллойда.

Для нейронных сетей аналогом алгоритма Ллойда считается алгоритм WTA (англ.: *Winner Takes All* — «победитель получает все»). В соответствии с ним после предъявления вектора x рассчитывается активность каждого нейрона. Победителем признается нейрон с самым сильным выходным сигналом, т. е. тот, для которого скалярное произведение (x, w) оказывается наибольшим. В предыдущем разделе было показано, что при использовании нормализованных векторов это равнозначно наименьшему евклидову расстоянию между входным вектором и вектором весов нейронов. Победитель получает право уточнить свои веса в направлении вектора x согласно правилу

$$w_{win} \leftarrow w_{win} + \alpha(x - w_{win}),$$

где α — коэффициент обучения. Веса остальных нейронов уточнению не подлежат. Алгоритм позволяет учитывать усталость нейронов путем подсчета количества побед каждого из них и поощрять элементы с наименьшей активностью для выравнивания их шансов. Такая модификация применяется чаще всего на начальной стадии обучения с последующим отключением после активизации всех нейронов. Подобный способ обучения реализован в виде режима CWTA (*Conscience Winner Takes All*) и считается одним из лучших и наиболее быстрых алгоритмов самоорганизации.

Помимо алгоритмов WTA, в которых в каждой итерации может обучаться только один нейрон, для обучения сетей с самоорганизацией широко применяются алгоритмы типа WTM (англ.: *Winner Takes Most* — «победитель получает больше»), в которых, кроме победителя, уточняют значения своих весов и нейроны из его ближайшего окружения. При этом, чем дальше какой-либо нейрон находится от победителя, тем меньше изменяются его веса. Процесс уточнения вектора весов может быть опреде-

лен обобщенной зависимостью, которая здесь представляется в виде

$$w_i \leftarrow w_i + \alpha G(i, x)[x - w_i]$$

для всех нейронов, расположенных в окрестности победителя. Если функция $G(i, x)$ определяется в форме

$$G(i, x) = \{1 \text{ для } i = I, 0 \text{ для } i \neq I\},$$

где I обозначает номер победителя, то мы получаем классический алгоритм WTA. Существует множество вариантов алгоритма WTM, отличающихся прежде всего формой функции $G(i, x)$. Для дальнейшего изучения выберем классический алгоритм Кохонена.

Алгоритм Кохонена

Алгоритм Кохонена относится к наиболее старым алгоритмам обучения сетей с самоорганизацией на основе конкуренции, и в настоящее время существуют различные его версии. В классическом алгоритме Кохонена сеть инициализируется путем приписывания нейронам определенных позиций в пространстве и связывания их с соседями на постоянной основе. Такая сеть называется **самоорганизующейся картой признаков** (сеть SOFM — Self-Organizing Feature Map). В момент выбора победителя уточняются не только его веса, но также и веса его соседей, находящихся в ближайшей окрестности. Таким образом, нейрон-победитель подвергается адаптации вместе со своими соседями. В классическом алгоритме Кохонена **функция соседства** $G(i, x)$ определяется в виде

$$G(i, x) = \{1 \text{ для } d(i, I) \leq L, 0 \text{ для } d(i, I) > L\}.$$

В этом выражении $d(i, I)$ обозначает евклидово расстояние между векторами весов нейрона-победителя I и i -го нейрона. Коэффициент L выступает в роли уровня соседства, его значение уменьшается в процессе обучения до нуля. Соседство такого рода называется прямоугольным.

Другой тип соседства, часто применяемый в картах Кохонена, — это соседство гауссовского типа, при котором функция $G(i, x)$ задается формулой

$$G(i, x) = \exp(-d^2(i, x)/2\lambda^2).$$

Степень адаптации нейронов-соседей определяется не только евклидовым расстоянием между i -м нейроном и победителем (I -м нейроном), но также и уровнем соседства λ . В отличие от соседства прямоугольного типа, где каждый нейрон, находящийся в окрестности победителя, адаптировался в равной степени, при соседстве гауссовского типа уровень адаптации различен и зависит от значения функции Гаусса. Как правило, гауссовское соседство дает лучшие результаты обучения и обеспечивает лучшую организацию сети, чем прямоугольное соседство.

Самоорганизующаяся карта признаков проходит два этапа обучения. На первом этапе элементы упорядочиваются так, чтобы отражать пространство входных элементов, а на втором происходит уточнение их позиций. Как правило, процесс представляется визуально путем использования двумерных данных и построения соответствующей поверхности. Например, входные векторы выбираются случайным образом на основе однородного распределения в некотором квадрате, и начинается обучение карты. В определенные моменты в ходе обучения строятся изображения карты путем использования соответствия, показанного на рис. 1. Элементы соединяются линиями, чтобы показать их относительное размещение. Сначала карта выглядит сильно «измятой», но постепенно в ходе обучения она разворачивается и расправляется. Конечным результатом обучения является карта, покрывающая все входное пространство и являющаяся достаточно регулярной (т. е. элементы оказываются распределенными почти равномерно). Для примера была рассмотрена карта с топологией квадрата из 49 элементов, и для 250 точек данных, взятых из единичного квадрата, было проведено ее обучение, которое начиналось со случайного набора весовых значений, задающих размещение кластерных элементов в центре входного пространства, как показано на рис. 1. На рис. 2 и 3 иллюстрируется процесс разворачивания карты с течением времени. Как и для других типов сетей, в данном случае результат обучения зависит от учебных данных и выбора параметров обучения.

Применение сетей с самоорганизацией

Главным свойством сети Кохонена считается компрессия данных, состоящая в том, что образующие кластер группы данных представляются единственным вектором весов нейрона-победителя. При разделении данных на кластеры и представлении каждого кластера одним из нейронов достигается значительное сокращение объема используемой под данные памяти, которое и называется компрессией. Это компрессия с потерей информации, которая сопровождается определенной погрешностью квантования.

Компрессия данных

Примером использования компрессионных свойств сети Кохонена может считаться сжатие изображений, предназначенное для уменьшения количества информации, представляющей конкретный образ, при сохранении погрешности восстановления на заданном уровне.

Пусть изображение разделяется на одинаковые кадры размером $n_x \times n_y$ пикселей. Образующие кадр пиксели представляют собой компоненты входного вектора x .

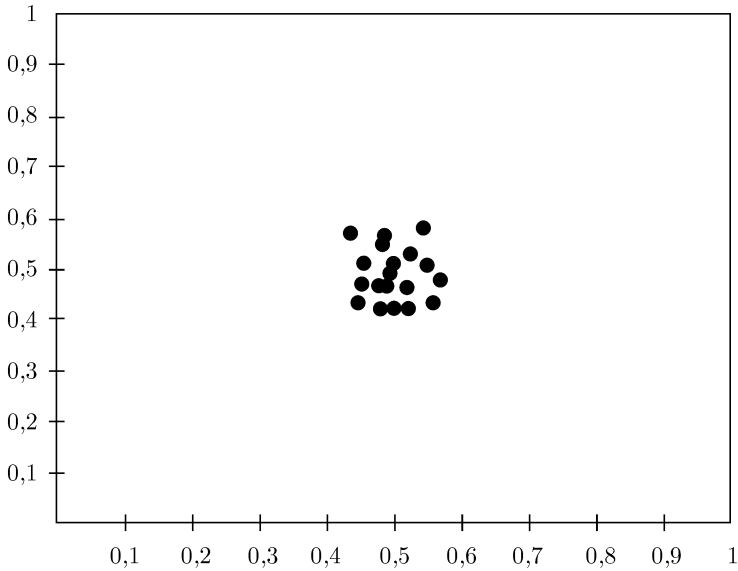


Рис. 1. Весовые векторы инициализируются случайными значениями из диапазона 0.4–0.6

Сеть с самоорганизацией содержит n нейронов, каждый из которых имеет входом вектор x . Обучение сети при помощи одного из алгоритмов самоорганизации состоит в подборе таких весов конкретных нейронов, при которых минимизируется погрешность квантования (3). В результате обучения формируется структура сети, при которой вектору x каждого кадра соответствует вектор весов нейрона победителя. В процессе предъявления очередного кадра выбирается номер нейрона-победителя. Номера нейронов-победителей образуют кодовую таблицу, а веса этих нейронов представляют средние значения, соответствующим конкретным компонентам вектора x (т.е. уровням интенсивности пикселей, составляющих кадр).

Поскольку количество нейронов обычно намного меньше количества кадров, то можно получить существенное сокращение объема данных, описывающих исходное изображение. В итоге коэффициент компрессии изображения равен

$$K = N \cdot n_x n_y T / (N \cdot \lg_2 n + n \cdot n_x n_y t),$$

где n_x и n_y — размеры кадра в осях x и y , N — количество кадров, n — количество нейронов, а T и t — количество битов для представления соот-

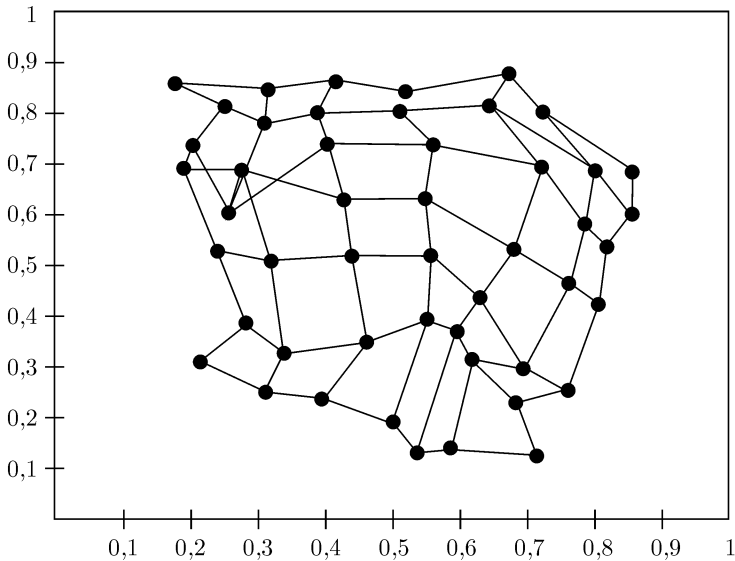


Рис. 2. Карта по прошествии 20 итераций

ответственно градаций интенсивности пиксела и значений весов. Этот подход позволяет получить степень компрессии изображений порядка 16 при значении коэффициента сигнал/шум (PSNR) около 26–28 дБ.

Прогнозирование нагрузок энергетической системы

Рассмотрим решение задачи прогнозирования часовых нагрузок в электроэнергетической системе на 24-часовом интервале. Пусть имеется база данных, содержащая векторы профильных нагрузок дня

$$p_j = [p(j, 1), p(j, 2), \dots, p(j, 24)],$$

где компонент $p(j, k)$ соответствует действительной нагрузке в k -й час суток. Множество профильных векторов подается на вход сети Кохонена, состоящей из n нейронов. Процесс самоорганизации сети приводит к автоматической кластеризации данных и к сопоставлению каждому кластеру одного из нейронов сети. Этот нейрон считается победителем, а его веса наилучшим образом адаптируются к усредненным весам профильных векторов, составляющих кластер. Характерная особенность состоит в том, что соседние векторы имеют сходные профильные характеристики.

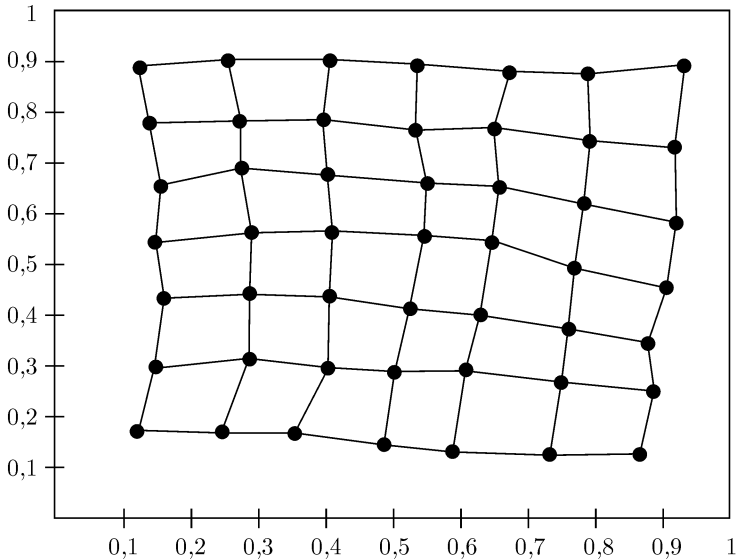


Рис. 3. Карта незадолго до окончания обучения. Элементы теперь упорядочены, и карта станет еще более регулярной по окончании финальной фазы сходимости

Близость весов нейронов, расположенных недалеко друг от друга, объясняется тем, что один и тот же день в разные годы при небольших отличиях в часовых нагрузках может возбуждать различные нейроны, которые образуют кластеры, группирующие данные сходных классов.

Знание таблицы распределения побед конкретных нейронов сети позволяет относительно легко предвидеть профили часовых нагрузок для произвольного дня года. С этой целью создаются таблицы принадлежности каждого дня года к области доминирования определенного нейрона с обозначением количества его побед для всех дней в прошлом. Для выбора прогнозируемого профиля нагрузок актуального дня в требуемом месяце рассчитываются усредненные значения весов нейронов победителей, которые указывали в прошлом на требуемый день. Если количество побед i -го нейрона, соответствующего j -му дню, обозначить k_{ji} , а соответствующие векторы весов класса — w_i , то прогнозируемый профильный вектор j -го дня рассчитывается по формуле

$$p_j = \sum_{i=1}^n k_{ji} w_i / \sum_{i=1}^n k_{ji}$$

Лекция 14. Адаптивная резонансная теория (АРТ)

Рассматриваются: вопрос о соотношении стабильности и пластичности при запоминании; архитектура, реализация, обучение и характеристики сети АРТ (адаптивной резонансной теории).

Ключевые слова: стабильность и пластичность, сеть АРТ, слой сравнения, слой распознавания, латеральное торможение, деградация и размножение классов.

Адаптивная резонансная теория (АРТ)

Серьезная проблема для нейронных сетей — правильное соотношение **стабильности и пластичности** при запоминании образов. Существуют наборы эталонов (даже состоящие всего из 4-х векторов), которые при циклическом предъявлении в обучении дают никогда не сходящиеся наборы параметров сети. Предъявление всего одного нового образа в обучающем множестве часто приводит к долгому переобучению. Если сеть работает в реальном времени, например, обрабатывает сенсорную информацию, то обучающее множество может все время меняться. Для большинства моделей нейронных сетей это приводит к отсутствию обучения вообще.

Человеческая память, напротив, эффективно хранит и корректирует запоминаемые образы. Ни предъявление нового образа, ни изменение старых не приводит к уничтожению памяти или невозможности запоминания. Даже удаление части нервной ткани чаще всего не прерывает работу сети и не стирает запомненные образы, а лишь делает их менее четкими.

Сеть АРТ — попытка приблизить механизм запоминания образов в искусственных НС к биологическому. Результатом работы АРТ является устойчивый набор запомненных образов и возможность выборки «похожего» вектора по произвольному предъявленному на входе вектору. Важное качество АРТ — динамическое запоминание новых образов без полного переобучения и отсутствие потерь уже запомненных образов при предъявлении новых.

Сеть АРТ-1

Сеть АРТ-1 предложена Карпентером и Гроссбергом в 1986 г. Она представляет собой векторный классификатор и обучается без учителя, лишь на основании предъявляемых входных векторов. АРТ-1 работает

только с двоичными векторами, состоящими из нулей и единиц. Позже было предложено много разновидностей этой модели. АРТ-2 запоминает и классифицирует непрерывные входные векторы. Группа моделей с суффиксом «MAP» (ARTMAP и др.) классифицирует и входные, и выходные вектора, а также строит связи между ними.

Архитектура и работа

Структура сети АРТ-1 (далее АРТ) представлена на рис.1. Входной вектор сети $x = x_1, \dots, x_n, \dots, x_N$ имеет N компонент. В слое распознавания запоминается M классов образов, по одному классу на каждый нейрон $m = 1, \dots, M$.

Основную работу по классификации производят **слой сравнения** и **слой распознавания**. Схемы приемников (Прм1, Прм2) и схема сброса управляют режимом работы сети и могут быть реализованы в виде обычных логических схем или в виде нейронов.

Работа блоков АРТ определяется следующими формулами:

$$\text{Прм1} : G1 = (\bigvee_n x_n) \wedge \neg(\bigvee_m R_m).$$

Выход Прм1 обеспечивает единичный сигнал для слоя сравнения, если на вход сети подан вектор x (нулевой вектор на входе недопустим) и если выход слоя распознавания равен нулю.

$$\text{Прм2} : G2 = \bigvee_n x_n$$

Если на вход подан вектор x , то блок Прм2 формирует на выходе единичный сигнал и тем самым разрешает работу слоя распознавания.

$$\text{Схема сброса} : G3 = (\sum_n C_n / \sum_n x_n) < \rho.$$

Проверяет критерий сходства для векторов x и C . Критерий состоит в сравнении количества единиц в векторах x , C . Количество единиц сравниваются в виде отношения с некоторым пороговым уровнем сходства ρ . Если порог не превышен, то сходство считается плохим и схема сброса вырабатывает сигнал торможения для нейрона в слое распознавания. Выход схемы сброса — двоичный вектор с M компонентами. Схема сброса является динамической и «помнит» свое состояние в течение одной классификации. Порог ρ является внешним параметром по отношению к сети и задается пользователем в интервале от 0 до 1. Чем меньше ρ , тем менее похожие векторы будут отнесены сетью к одному классу.

Слой сравнения

Каждый нейрон в слое сравнения имеет порог, равный двум. На вход одного нейрона в слое сравнения подаются: сигнал $G1$ с единичным ве-

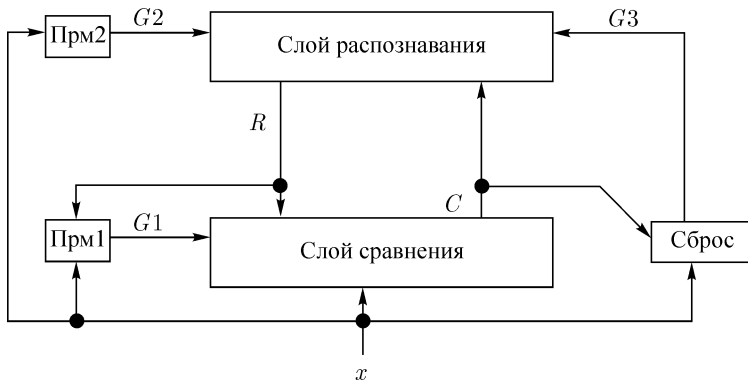


Рис. 1. Структурная схема АРТ

сом, одна компонента x^n с единичным весом и все выходы слоя распознавания, M компонент с вектором весов T^n , где n — номер нейрона в слое сравнения. Весовые коэффициенты T — двоичные. В нейроне используется нелинейность в виде жесткой ступеньки: если активация нейрона NET_n превышает порог $\Theta = 2$, то на выходе нейрона будет единица, иначе — ноль. Это «правило 2/3»: для активации нейрона достаточно два сигнала из трех.

Работа слоя определяется формулами:

$$P_n = T^n R = \sum_m T_m^n R_m$$

$$NET_n = P_n + x_n + G1$$

$$C_n = \{0, \text{ если } NET_n < 2; 1, \text{ если } NET_n \geq 2\}$$

Работой слоя управляет сигнал $G1$. Если $G1 = 1$, то x проходит без изменений на выход слоя сравнения, благодаря лишнему единичному сигналу $G1$ на входе нейрона. Если $G1 = 0$, то на выходе имеем $C = x \wedge P$, т. е. вектор C будет логическим произведением двоичных векторов x и P .

Слой распознавания

Каждый нейрон в слое распознавания имеет следующие входы: один сигнал $G2$ с единичным весом, одна компонента $G3_m$ с большим отрицательным весом (m — номер нейрона) и N сигналов со слоя сравнения с вектором весов B^m (у вектора B^m всего N компонент, B_1^m, \dots, B_N^m).

Нейроны слоя распознавания не содержат нелинейных элементов, но обладают следующей особенностью. Каждый нейрон в слое связан со

всеми остальными нейронами этого же слоя обратными тормозящими связями и положительной обратной связью — с самой собой (как во втором слое сети Хемминга, см. Лекцию 10).

Такой способ связности называется **латеральным торможением**. Это приводит к тому, что только один нейрон в слое распознавания может быть активирован. Между нейронами существует конкуренция, и нейрон с максимальным выходом «подавляет» все остальные нейроны в слое, выигрывая «состязание». Его выход становится равным единице, остальных нейронов — нулю, т. е. вектор R имеет только одну единичную компоненту, остальные — нули.

Веса B^m имеют действительные значения. Работа слоя определяется формулой:

$$R_m = f(B^m, C) \wedge G2 \wedge \neg(G3_m),$$

где $f(B^m, C)$ — выход m -го нейрона, равный нулю или единице.

Отсюда видно, что сигнал $G2$ «разрешает» работу слоя распознавания, а сигнал $G3$ позволяет выборочно затормозить любые нейроны в слое.

Работа сети АРТ

Решение задачи классификации с помощью АРТ содержит следующие этапы: инициализация, распознавание, сравнение, поиск, обучение.

1. Инициализация.

а) выбираем параметр ρ , исходя из требуемой детальности классификации;

б) создаем сеть в памяти. Количество нейронов должно быть достаточным, чтобы запомнить все ядра классов (до M). Изначально все нейроны слоя распознавания считаются «невыведенными», их веса приравниваются к одинаковым небольшим значениям:

$$B_n^m = B_{\text{нач}} < L / (L + N - 1),$$

где $L > 1$ — некоторая константа (обычно $L = 2$). Веса в слое сравнения также выбираются одинаковыми, равными единице: $T_m^n = 1$.

Такой выбор весов обеспечивает остановку поиска на невыведенном нейроне, если нет подходящих выделенных нейронов, и правильное обучение.

2. Распознавание.

а) предъявляем вектор x на входе. До этого момента $G2 = 0$ и выход слоя распознавания равен нулю: $R = 0$.

б) у вектора x есть ненулевые компоненты, поэтому $G1$ становится равным единице, т. к. $R = 0$. Сигнал $G1$ «подпитывает» нейроны слоя сравнения и x без изменений проходит через слой сравнения: $C = x$.

в) весовые коэффициенты B^m имеют смысл нормированных ядер классов. В слое распознавания активируется несколько нейронов, но благодаря латеральному торможению остается один нейрон с выходом $R_{m_0} = 1$, а остальные тормозятся. m_0 — номер выигравшего нейрона.

3. Сравнение.

а) выход $R \neq 0$ приводит к $G1 = 0$, что снимает «подкачку» нейронов в слое сравнения. Весовые коэффициенты T^n имеют смысл ненормированных двоичных ядер классов. На вход слоя сравнения передается один ненулевой выход слоя распознавания, $R_{m_0} = 1$. Эта единица умножается на весовые коэффициенты, давая в сумме сигнал

$$NET_n = x_n + T_{m_0}^n.$$

Порог всех нейронов равен 2, поэтому выход слоя сравнения равен

$$C_n = x_n \wedge T_{m_0}^n.$$

Следовательно, выход слоя сравнения на этом этапе — логическое произведение входного сигнала и двоичного ядра класса из слоя сравнения.

б) модуль сброса вычисляет второй критерий сходства (первый — максимум произведения (B^m, x) в слое распознавания). Если количества единиц в векторе C и векторе x близки, то сходство считается хорошим и выносится решение о принадлежности вектора x к классу m_0 .

4. Поиск.

а) если критерий сходства не выполняется, схема сброса вырабатывает сигнал $G3_{m_0} = 1$, который тормозит нейрон m_0 в слое распознавания. Сигнал $G3_{m_0} = 1$ остается равным 1 до окончания данной классификации. Выход нейрона m_0 становится равным 0, а, следовательно, и весь вектор $R = 0$. Сигнал $G1$ становится равным нулю и вектор x снова проходит через слой сравнения без изменений, вызывая новый цикл поиска (шаги 2в–3б), пока критерий сходства не будет удовлетворен.

При соответствующем выборе начальных значений весов B поиск всегда закончится на нераспределенном нейроне слоя распознавания. Для него будет выполнен критерий сходства, т. к. все веса T равны 1. Если все нейроны выделены и критерий сходства не выполняется, следует аварийная остановка либо расширение сети введением нового нейрона в слое распознавания и новых входов в слое сравнения.

5. Обучение.

Независимо от того, найден ли на этапе поиска распределенный нейрон или нераспределенный, обучение протекает одинаково. Коррек-

тируются лишь веса выигравшего нейрона m_0 в слое распознавания и веса $T_{m_0}^n$ для всех n в слое сравнения.

Различают быстрое и медленное обучение. При быстром обучении коррекции весов имеют вид:

$$B_n^{m_0} = LC_n / (L + \sum_n C_n - 1) \equiv \Delta B_n^{m_0},$$

где $L > 1$ — константа.

Веса в слое сравнения — двоичные: $T_{m_0}^n$.

В результате такого алгоритма обучения ядра T изменяются, несущественные компоненты обнуляются в процессе обучения. Если какая-то компонента вектора T^n стала нулевой на какой-то итерации обучения, она никогда не вернется к единице. В этом проявляется асимметрия АРТ по отношению к значениям 0 и 1. Эта асимметрия имеет серьезные отрицательные последствия для модели, приводя к деградации ядер классов в случае зашумленных входных векторов.

Медленное обучение меняет ядра малыми коррекциями:

$$\begin{aligned} B_n^{m_0} &\longrightarrow \beta \Delta B_n^{m_0} + (1 - \beta) B_n^{m_0}, \\ T_{m_0}^n &\longrightarrow \beta C_n + (1 - \beta) T_{m_0}^n, \end{aligned}$$

где β мало и характеризует скорость обучения.

В результате каждой итерации обучения ядра меняются незначительно.

Видно, что веса B в любой момент времени могут быть однозначно рассчитаны через веса T , таким образом, кодирование информации о ядрах в АРТ в рассмотренной модели является избыточным в смысле расхода памяти.

Необходимость поиска

В сети АРТ используются два критерия «похожести» векторов. Первый — максимум скалярного произведения $\max_m (B^m, x)$ при выборе «победителя» в слое распознавания. Второй — критерий сходства в блоке сброса:

$$\left(\sum_n C_n / \sum_n x_n \right) \Big|_{x, C} \geq \rho.$$

Таким образом, задача классификации в сети АРТ состоит в том, чтобы найти ядро с максимальным скалярным произведением (B^m, x) , соблюдая при этом условие выполнения критерия сходства. Эти два критерия не являются эквивалентными, поэтому и фаза поиска, и фаза распознавания являются необходимыми и не могут быть опущены.

Положительные качества и недостатки АРТ

Сеть АРТ решает дилемму стабильности-пластичности и позволяет быстро запоминать новые образы без утраты старых. Как и в случае других моделей НС, на обычных машинах фон-неймановского типа сети работают медленно и неэффективно. Для решения задачи нужно найти максимум скалярного произведения, что требует около $3NM$ операций с плавающей запятой, и вычислить в худшем случае M критериев сходства. Для этого необходимы существенные вычислительные затраты. На параллельном компьютере операции расчета скалярных произведений могут быть распараллелены, но расчет критериев сходства все равно выполняется последовательно. Таким образом, даже на параллельной машине сеть АРТ является требовательной к ресурсам.

Тем не менее, одна итерация для запоминания каждого входного вектора — редкая экономичность для нейронных сетей. Вспомним, что многослойный персептрон для запоминания нового вектора требует полного переобучения.

У сети АРТ есть несколько существенных недостатков.

1. *Чувствительность к порядку предъявления векторов.* Большинство разновидностей АРТ весьма чувствительны к порядку предъявления входных векторов x . Картины ядер классов, сформированные сетью, принципиально меняются при различных видах упорядочения.

2. *Невозможность классификации зашумленных векторов.* Пусть входные векторы содержат шум.

Если компонента незашумленного входного вектора равна x_n , то предъявленные сети значения будут определяться вероятностным законом:

$$\begin{aligned} p(x_n) &= 1 - \varepsilon, \\ p(-x_n) &= \varepsilon, \end{aligned}$$

где ε — малое положительное число, характеризующее уровень шума.

Если такие данные будут предъявлены АРТ, то будет наблюдаться **деградация и размножение классов**. Если сетью сформировано правильное ядро для класса, к которому относится вектор x , то как только компонента x_n примет нулевое значение за счет шума (если векторы предъявляются не однократно), соответствующая компонента ядра также будет обнулена. Т. к. случайное нулевое значение может принять любая компонента x , то с течением времени все компоненты ядра будут обнулены, запомненная информация об этом классе — утрачена. Если после этого предъявить незашумленный вариант вектора x , то для него будет выделен новый нейрон, т. е. сформирован новый класс. Это явление называется размножением классов. Через некоторое время в сети будет множество нейронов

с нулевыми весами, и все нейроны будут распределены. Работа сети прекратится. Это явление определяется исходной асимметрией алгоритмов АРТ относительно значений 0 и 1. Существуют методы для устранения асимметрии и предотвращения размножения классов.

Лекция 15. Нечеткие и гибридные нейронные сети

Рассматриваются: математические основы нечетких систем, преимущества и алгоритмы обучения нечетких нейронных сетей, нечеткие сети с генетической настройкой, экспертные системы на основе гибридных НС.

Ключевые слова: мягкие вычисления, вычислительный интеллект, гибридные системы, нечеткие множества, степень принадлежности, лингвистические переменные, нечеткая импликация, агрегирование, система нечеткого вывода, фазификатор, дефазификатор, нечеткие нейронные сети, мягкая экспертная система.

Интеллектуальные информационные системы в условиях неопределенности и риска

С помощью символьной обработки информации не удастся решить прикладные задачи многих предметных областей, если для них невозможно получить полную информацию и если их определение недостаточно полно. Такая ситуация характерна для:

- сложных технических систем;
- систем экономического планирования;
- социальных систем большой размерности;
- систем принятия решений и т.п.

Выходом является использование систем, основанных на **мягких вычислениях**, которые включают в себя:

- нечеткую логику и вероятностные вычисления;
- нейрокомпьютинг — обучение, адаптация, классификация, системное моделирование и идентификация;
- генетические вычисления — синтез, настройка и оптимизация с помощью систематизированного случайного поиска и эволюции.

Эти составные части не конкурируют друг с другом, а создают эффект взаимного усиления (**гибридные системы**). Наряду с термином «мягкие вычисления» используется термин **«вычислительный интеллект»** — научное направление, где решаются задачи искусственного интеллекта на основе теории нечетких систем, нейронных сетей и эволюционных (генетических) вычислений.

Нечеткие нейронные сети с генетической настройкой параметров (гибридные системы) демонстрируют взаимное усиление достоинств и нивелирование недостатков отдельных методов:

1. Представление знаний в нейронных сетях в виде матриц весов не позволяет объяснить результаты проведенного распознавания или прогнозирования, тогда как в системах вывода на базе нечетких правил результаты воспринимаются как ответы на вопросы «почему?».

2. Нейронные сети обучаются с помощью универсального алгоритма, т. е. трудоемкое извлечение знаний заменяется сбором достаточной по объему обучающей выборки. Для нечетких систем вывода извлечение знаний включает в себя сложные процессы формализации понятий, определение функций принадлежности, формирование правил вывода.

3. Нечеткие нейронные сети обучаются как нейронные сети, но их результаты объясняются как в системах нечеткого вывода.

Нечеткие множества

Понятие **нечетких множеств** (*fuzzy sets*) как обобщение обычных (четких) множеств было введено Л. Заде в 1965 г. Традиционный способ представления элемента множества A состоит в применении характеристической функции $\mu_A(x)$, которая равна 1, если элемент принадлежит множеству A , или равна 0 в противном случае. В нечетких системах элемент может частично принадлежать любому множеству. Степень принадлежности множеству A , представляющая собой обобщение характеристической функции, называется функцией принадлежности $\mu_A(x)$, причем $\mu_A(x) \in [0, 1]$, и $\mu_A(x) = 0$ означает отсутствие принадлежности x множеству A , а $\mu_A(x) = 1$ — полную принадлежность. Конкретное значение функции принадлежности называется степенью или коэффициентом принадлежности.

Лингвистические переменные

В теории нечетких множеств, помимо переменных цифрового типа, существуют **лингвистические переменные** с приписываемыми им значениями.

Пусть x обозначает температуру. Можно определить нечеткие множества «отрицательная», «близкая к нулю», «положительная», характеризуемые функциями принадлежности $\mu_{\text{отриц}}(x)$, $\mu_{\text{нул}}(x)$, $\mu_{\text{полож}}(x)$. Лингвистическая переменная «температура» может принимать значения «отрицательная», «близкая к нулю», «положительная». Функция нечеткой принадлежности является непрерывным приближением пороговой функции точной принадлежности.

Нечеткие правила вывода

Правило вывода

если x это A , то y это B

называется **нечеткой импликацией** $A \rightarrow B$, если A и B — лингвистические значения (значения лингвистической переменной), идентифицированные нечетким способом через соответствующие функции принадлежности для переменных.

Часть « x это A » называется условием (предпосылкой), а « y это B » — следствием (заключением).

Обобщение для N -мерного вектора x :

если x_1 это A_1 и x_2 это A_2 и ... и x_N это A_N , то y это B , A_1, A_2, \dots, A_N, B обозначают величины соответствующих коэффициентов принадлежности $\mu_A(x_i), i = 1, 2, \dots, N, \mu_B(y)$.

Возможна интерпретация $\mu_A(x)$

— в форме логического произведения

$$\mu_A(x) = \min_{i=1, \dots, N} \mu_A(x_i)$$

— в форме алгебраического произведения

$$\mu_A(x) = \prod_{i=1, \dots, N} \mu_A(x_i)$$

(агрегирование предпосылки).

Каждой импликации $A \rightarrow B$ можно приписать значение функции принадлежности $\mu_{A \rightarrow B}(x, y)$:

— форма логического произведения

$$\mu_{A \rightarrow B} = \min\{\mu_A(x), \mu_B(y)\}$$

— форма алгебраического произведения

$$\mu_{A \rightarrow B} = \mu_A(x)\mu_B(y)$$

(агрегирование на уровне импликации).

Системы нечеткого вывода Мамдани-Заде

Элементы теории нечетких множеств, правила импликации и нечетких рассуждений образуют **систему нечеткого вывода**. В ней можно выделить:

- множество используемых нечетких правил;
- базу данных, содержащую описания функций принадлежности;
- механизм вывода и агрегирования, который формируется применяемыми правилами импликации.

В случае технической реализации в качестве входных и выходных сигналов выступают измеряемые величины, однозначно сопоставляющие входным значениям соответствующие выходные значения.

Для обеспечения взаимодействия этих двух видов вводится нечеткая система с так называемым **фазификатором** (преобразователем множеств входных данных в нечеткое множество) на входе и **дефазификатором** (преобразователем нечетких множеств в конкретное значение выходной переменной) на выходе.

Фазификатор преобразует точное множество входных данных в нечеткое множество, определенное с помощью функции принадлежности, а дефазификатор решает обратную задачу — формирует однозначное решение относительно входной переменной на основании многих нечетких выводов, вырабатываемых исполнительным модулем нечеткой системы.

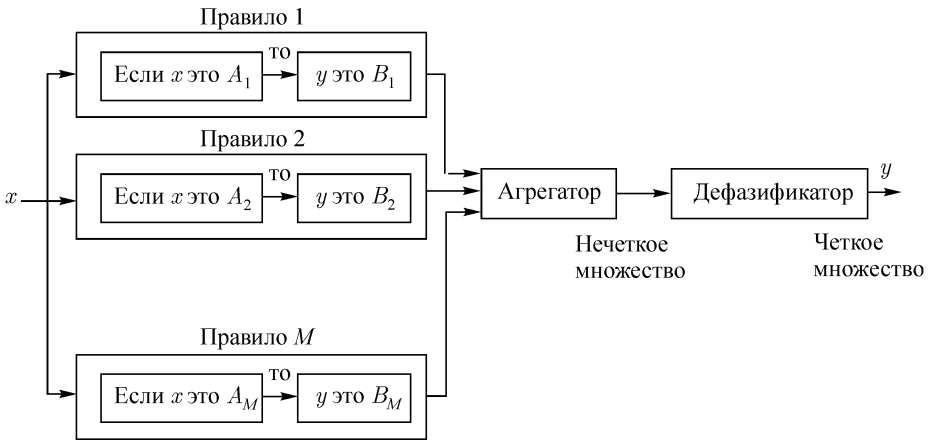


Рис. 1. Вывод в нечеткой системе при наличии M правил

Выходной сигнал модуля вывода может иметь вид M нечетких множеств, определяющих диапазон изменения выходной переменной. Дефазификатор преобразует этот диапазон в одно конкретное значение, принимаемое в качестве выходного сигнала всей системы.

В модели вывода Мамдани-Заде присутствуют следующие операторы:

- оператор логического или арифметического произведения для

определения результирующего уровня активации, в котором учитываются все компоненты вектора условия;

— оператор логического или арифметического произведения для определения значения функции принадлежности для всей импликации $A \rightarrow B$;

— оператор логической суммы как агрегатор равнозначных результатов импликации многих правил;

— оператор дефазификации, трансформирующий нечеткий результат $\mu(y)$ в четкое значение y .

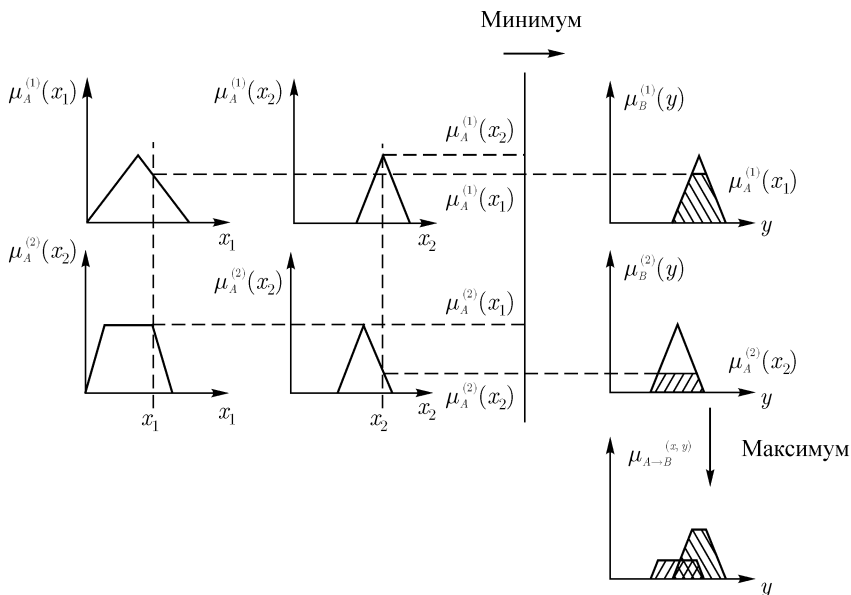


Рис. 2. Пример системы вывода Мамдани-Заде

На рис. 2 представлен способ агрегирования при двух входных переменных x_1, x_2 .

Логическое произведение (оператор \min) используется как для агрегирования нечетких правил относительно конкретных переменных x_i , $i = 1, 2$, образующих вектор x , так и на уровне импликации $A \rightarrow B$ для одиночных правил вывода. Агрегирование импликаций, касающихся правил 1 и 2, проводится с использованием логической суммы (оператор \max).

Фазификатор

Фазификатор преобразует N -мерный вектор $x = [x_1, x_2, \dots, x_N]$ в нечеткое множество A , характеризуемое функцией принадлежности $\mu_A(x)$.

Наибольшей популярностью пользуются функции гауссовского типа, треугольные и трапециевидальные функции:

1. Общая форма гауссовской функции

$$\mu_A(x) = \exp[-(x - c)^2/\sigma^2]$$

c — центр нечеткого множества,

σ — коэффициент широты.

2. Симметричная треугольная функция

$$\mu_A(x) = \{1 - |x - c|/d, \text{ при } x \in [c - d, c + d]; \\ 0, \text{ для остальных } x\},$$

c — центр,

d — ширина.

3. Трапециевидальная функция

$$\mu_A(x) = \{0, \text{ при } x > z \text{ и } x < y; \\ 1, \text{ при } c - t/2 \leq x \leq c + t/2; \\ s(z - x), \text{ при } c + t/2 \leq x \leq z; \\ s(x - y), \text{ при } y \leq x \leq c - t/2; \},$$

s — угол наклона.

При $t = 0$ получаем треугольную функцию.

Дефазификатор

Трансформировать нечеткое множество $\mu(y) = \mu_{A \rightarrow B}(y)$ в точечное решение y можно многими способами:

1. Дефазификация относительно центра области

$$y_c = \int \mu(y) \cdot y \cdot dy / \int \mu(y) dy$$

или

$$y_c = \sum_i \mu(y_i) \cdot y_i / \sum_i \mu(y_i)$$

2. Дефазификация относительно среднего центра

$$y_c = \frac{\sum_{i=1, M} \mu(c_i) \cdot c_i}{\sum_{i=1, M} \mu(c_i)}$$

где c_i — центр i -го нечеткого правила,

$\mu(c_i)$ — соответствующая функция принадлежности.

3. Дефазификация относительно среднего максимума

$$y_M = \sum_{i=1, m} y_i / m,$$

где m — количество точек, в которых $\mu(y_i)$ достигает максимального значения. Если функция $\mu(y)$ имеет максимальное значение только в одной точке, то

$$y_M = y_m ax.$$

4. выбирается минимальное из максимальных значений y :

y_s — наименьшее из y , для которых $\mu(y) = \max$.

5. выбирается максимальное из максимальных значений:

y_l — наибольшее из y , для которых $\mu(y) = \max$.

Модель Мамдани-Заде как универсальный аппроксиматор

Модели нечеткого вывода позволяют описать выходной сигнал многомерного процесса как нелинейную функцию входных переменных x_i , $i = 1, 2, \dots, N$ и параметров нечеткой системы, например, при использовании в качестве агрегатора оператора алгебраического произведения с последующей дефазификацией относительно среднего центра. В модели Мамдани-Заде каждое из M правил определяется уровнем активации условия

$$\mu(y_i) = \prod_{j=1}^M \mu_{A_i}(x_j)$$

где y_i — значение y , при котором значение $\mu(y_i)$ максимально. Пусть y_i — центр C_i нечеткого множества заключения i -го правила вывода. Тогда дефазификация относительно среднего центра дает

$$y = \left(\sum_{i=1}^M C_i \left[\prod_{j=1}^N \mu_{A_i}(x_j) \right] \right) / \sum_{i=1}^M \prod_{j=1}^N \mu_{A_i}(x_j)$$

Приведенные формулы модели Мамдани-Заде имеют модульную структуру, которая идеально подходит для системного представления в виде многослойной структуры, напоминающей структуру классических нейронных сетей. Такие сети мы будем называть **нечеткими нейронными сетями**. Характерной их особенностью является возможность использования нечетких правил вывода для расчета выходного сигнала. Обучение таких сетей сводится к расчету параметров функции фазификации.

Нечеткие сети TSK (Такаги-Сугено-Канга)

Схема вывода в модели TSK при использовании M правил и N переменных x_j имеет вид ($i = 1, 2, \dots, M$)

$$\text{if } (x_1 \text{ is } A_1^{(i)}) \& (x_2 \text{ is } A_2^{(i)}) \& \dots \& (x_N \text{ is } A_N^{(i)})$$

$$\text{then } y_i = p_{i0} + \sum_{j=1}^N p_{ij} x_j.$$

Условие ($x_i \text{ is } A_i$) реализуется функцией фазификации

$$\mu_A(x_i) = 1 / (1 + ((x_i - c_i) / \sigma_i)^{2b_i}).$$

При M правилах агрегированный выходной результат сети имеет вид

$$y(x) = \frac{\sum_{i=1}^M w_i y_i(x)}{\sum_{i=1}^M w_i}, \tag{1}$$

$$y_i(x) = p_{i0} + \sum_{j=1}^N p_{ij} x_j.$$

Веса w_i интерпретируются как значимость компонентов $\mu_A^{(i)}(x)$. Тогда формуле (1) можно поставить в соответствие многослойную нейронную сеть рис. 3.

1. Первый слой выполняет фазификацию каждой переменной. Это параметрический слой с параметрами $c_j^{(i)}, \sigma_j^{(i)}, b_j^{(k)}$, подлежащими адаптации в процессе обучения.

2. Второй слой выполняет агрегирование отдельных переменных, определяя результирующее значение коэффициента принадлежности $w_i = \mu_A^{(i)}(x)$ для вектора x (непараметрический слой).

3. Третий слой — генератор функции TSK, рассчитывает значения

$$y_i(x) = p_{i0} + \sum_{j=1}^N p_{ij} x_j.$$

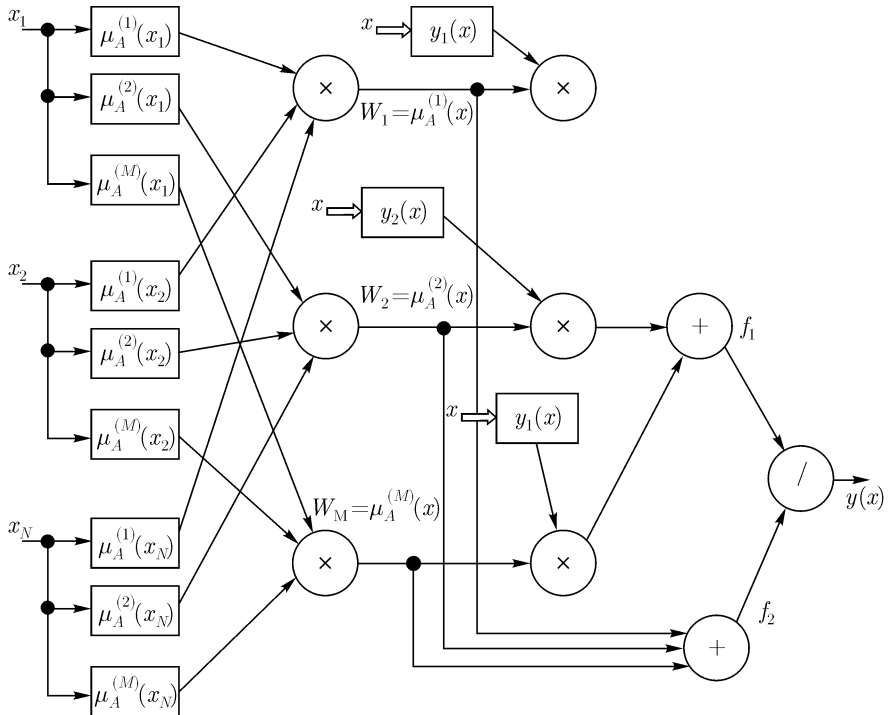


Рис. 3. Нечеткая нейронная сеть TSK

В этом слое также производится умножение $y_i(x)$ на w_i , сформированные в предыдущем слое. Здесь адаптации подлежат веса $p_{ij}, i = 1, 2, \dots, M, j = 1, 2, \dots, N$, определяющие функцию следствия модели TSK.

4. Четвертый слой составляют два нейрона-сумматора, один из которых рассчитывает взвешенную сумму сигналов $y_k(x)$, а второй — сумму весов $w_i, i = 1, 2, \dots, M$ (непараметрический слой).

5. Пятый слой из одного нейрона — это нормализующий слой, в котором выходной сигнал сети агрегируется по формуле (1).

Таким образом, в процессе обучения происходит уточнение параметров только первого (нелинейного) и третьего (линейного) слоев.

Гибридный алгоритм обучения нечетких сетей

Параметры, подлежащие адаптации, разделяются на две группы: — первая состоит из параметров p_{ij} линейного третьего слоя;

— вторая состоит из параметров нелинейной функции принадлежности первого слоя.

Уточнение параметров проводится в два этапа.

На первом этапе при фиксации определенных значений параметров функции принадлежности путем решения системы линейных уравнений рассчитываются параметры p_{ij} полинома TSK.

При известных значениях функции принадлежности преобразование, реализуемое сетью, можно представить в виде

$$y(x) = \sum_{i=1}^M w_i (p_{i0} + \sum_{j=1}^N p_{ij} x_j).$$

$$w_i = \left[\prod_{j=1}^N \mu_A^{(i)}(x_j) \right] / \sum_{k=1}^N \left[\prod_{j=1}^N \mu_A^{(k)}(x_j) \right] = \text{const.}$$

При p обучающих выборках $(x^{(l)}, d^{(l)})$, $l = 1, 2, \dots, p$ и замене выходного сигнала сети ожидаемым значением $d^{(l)}$ получим систему из p линейных уравнений вида

$$W \cdot P = d,$$

где

$$W = \left\| \begin{array}{cccccccc} w'_{11} & w'_{11}x_1^{(1)} & \dots & w'_{11}x_N^{(1)} & \dots & w'_{1M} & w'_{1M}x_1^{(1)} & \dots & w'_{1M}x_N^{(1)} \\ w'_{21} & w'_{21}x_1^{(2)} & \dots & w'_{21}x_N^{(2)} & \dots & w'_{2M} & w'_{2M}x_1^{(2)} & \dots & w'_{2M}x_N^{(2)} \\ & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ w'_{p1} & w'_{p1}x_1^{(p)} & \dots & w'_{p1}x_N^{(p)} & \dots & w'_{pM} & w'_{pM}x_1^{(p)} & \dots & w'_{pM}x_N^{(p)} \end{array} \right\|$$

$$P = \| \| p_{10} \dots p_{1N} \dots p_{M0} \dots p_{MN} \| \| ^T,$$

w'_{ki} — уровень активации (вес) i -го правила при предъявлении k -го входного вектора $x^{(k)}$.

Размерность матрицы W равна $p \times (N + 1)M$, при этом обычно количество строк (количество выборок) значительно больше количества столбцов. Решение этой системы уравнений можно получить за один шаг при помощи псевдоинверсии матрицы W :

$$P = W^+ d.$$

Псевдоинверсия матрицы заключается в решении задачи минимизации

$$\min \| W^+ W - E \|,$$

где E — единичная матрица.

На втором этапе (линейные параметры $p_{ij}, i = 1, \dots, M$ — фиксированы) рассчитываются фактические выходные сигналы $y_k, k = 1, 2, \dots, p$:

$$y = Wp,$$

вектор ошибки

$$\varepsilon = y - d,$$

и градиент целевой функции $E(n)$ по параметрам первого слоя. Если применяется метод наискорейшего спуска, то формулы адаптации принимают вид

$$\begin{aligned} c_j^{(i)}(n+1) &= c_j^{(i)}(n) - \alpha_c \partial E(n) / \partial c_j^{(i)} \\ \sigma_j^{(i)}(n+1) &= \sigma_j^{(i)}(n) - \alpha_\sigma \partial E(n) / \partial \sigma_j^{(i)} \\ b_j^{(i)}(n+1) &= b_j^{(i)}(n) - \alpha_b \partial E(n) / \partial b_j^{(i)} \end{aligned}$$

где n обозначает номер очередной итерации.

После уточнения нелинейных параметров вновь запускается процесс адаптации линейных параметров TSK (первый этап) и нелинейных параметров (второй этап). Этот цикл повторяется вплоть до стабилизации всех параметров процесса.

Мягкая экспертная система

Рассмотрим архитектуру и основные структурно-функциональные решения мягкой экспертной системы (МЭС). Для определения МЭС сопоставим понятия нечеткой и мягкой экспертных систем. В описании архитектуры МЭС будем использовать три признака: способ извлечения знаний; представление знаний; обработку знаний. Перечисленные признаки создают общую «координатную» сетку описания.

Определение мягкой экспертной системы. Сравнение нечеткой и мягкой экспертных систем

Нечеткие экспертные системы (ЭС) используют представление знаний в форме нечетких продукций и лингвистических переменных. Основу представления лингвистической переменной составляет терм с функцией принадлежности. Способ обработки знаний в нечетких ЭС — это логический вывод по нечетким продукциям. Особенностью нечеткой ЭС является способ извлечения функций принадлежности, который сводится либо к статистическим методам построения, либо к методу экспертных

оценок. *Мягкой ЭС (МЭС)* будем называть нечеткую ЭС, которая обладает следующими особенностями:

— использует статистические данные, которые интерпретирует как обучающие выборки для нечетких нейронных сетей;

— представляет знания в виде лингвистических переменных (функций принадлежности — ФП), нечетких продукций и обученных нейронных сетей. Редукция множества нечетких продукций, настройка ФП и базы правил выполняется с помощью генетических алгоритмов (ГА).

Мяжкими называют **вычисления**, сочетающие теорию нечетких систем, нейронные сети, вероятностные рассуждения и генетические алгоритмы, и обладающие синергическим эффектом; следовательно, мягкой экспертной системой называют ЭС, сочетающую перечисленные теории ради того же эффекта взаимного усиления.

Рассмотрим возможные применения МЭС в автоматизированном проектировании. Обобщенной моделью проектирования является иерархически-блочный метод, сущность которого сводится к декомпозиции функций с последующим выделением иерархий систем и подсистем. Проектируемая система формируется с помощью синтеза таких подсистем. Анализ в ходе автоматизированного проектирования обычно заключается в том, что необходимо рассмотреть условия эксплуатации будущей системы или ее окружения, которое является сложной системой (например, для экономических информационных систем окружающая среда — это социально-экономическая среда). Кроме анализа окружающей среды в ходе проектирования приходится выполнять анализ результатов физических или численных экспериментов и имитационного моделирования. Можно выделить два основных принципа экспертной деятельности в ходе проектирования.

1. Исходные данные для анализа представляются в виде качественного описания структурно-функционального решения и в виде совокупности временных рядов системных переменных окружения.

Принцип «конструктивной неопределенности» утверждает, что точность и смысл противоречат друг другу, начиная с некоторого момента анализа. Если в технике важными являются все более точные измерения, то в ходе анализа эксперт отказывается от точных цифр в пользу нечетких, но содержательных оценок, которые осмыслены и позволяют принять проектное или управленческое решение.

Мягкая экспертная система должна предоставить инструментальную и информационную среду для экспертной деятельности в ходе проектирования. Инструменты для разработки МЭС должны представлять собой совокупность различных программных продуктов, объединенных логикой работы. Покажем, что МЭС, являющаяся инструментальной

средой проектировщика, позволяет выполнить в автоматизированном режиме все этапы экспертной деятельности. Если рассматривать экспертную деятельность как управление объектом, то инструментарий экспертизы можно использовать как систему управления, а именно — нечеткий контроллер.

Представление знаний в мягкой экспертной системе. Содержание баз знаний и данных мягкой экспертной системы

Если использовать нечеткую НС на этапе извлечения знаний, то, кроме функций принадлежности и нечетких продукций, порождается совокупность обученных НС, которые входят в базу знаний МЭС. Оптимизация (редукция) множества извлеченных правил выполняется на основе генетического алгоритма.

База знаний МЭС должна содержать следующие части:

- 1) функции принадлежности;
- 2) нечеткие продукции;
- 3) обученные нечеткие нейронные сети;
- 4) процедуры интерпретации хромосом генетических алгоритмов;
- 5) функции оптимальности.

Рассмотрим проблему представления перечисленных составных частей в компьютерных интеллектуальных системах. Если функция принадлежности характеризуется такими математическими свойствами, как непрерывность, выпуклость (униmodalность), то функция принадлежности может быть представлена параметризованной функцией формы. Наибольшее распространение получили четыре вида функций формы: треугольная, трапециевидная, колоколообразная и сигмоидальная, которые определяются тройкой, четверкой и двойкой параметров соответственно. Некоторые операции нечеткой алгебры сохраняют униmodalность при использовании трапециевидного представления функций принадлежности, поэтому результаты операции также являются четверкой параметров. Представление нечетких продукций упрощается в связи с тем, что порядок обработки нечетких продукций не важен и не влияет на ход вывода результата. Представление нечеткой нейронной сети является более сложной проблемой, так как описание структуры ННС не имеет смысла без нейроимитатора соответствующей архитектуры нечетких нейронных сетей, т. е. нейроимитатор определяется как составляющая часть механизма вывода мягкой ЭС. Для организации хранения знаний МЭС можно использовать как СУБД, так и специальные форматы.

Лекция 16. Контрастирование (редукция) нейронной сети

Рассматриваются: методы оценки значимости параметров нейронной сети и сигналов в ней; сокращение числа входов в линейном сумматоре методом «снизу-вверх», метод исключения параметров «сверху-вниз» с ортогонализацией и бинаризация сумматора.

Ключевые слова: значимость параметров и сигналов, контрастирование нейронных сетей, редукция «снизу вверх», редукция «сверху вниз».

Значимость параметров и сигналов. Сокращение описания (контрастирование) сетей.

Сокращение множества параметров и входных сигналов обученной нейронной сети преследует цели:

- 1) упрощение специализированных устройств;
- 2) сокращение объема используемой памяти и увеличение быстродействия;
- 3) удешевление сбора данных;
- 4) обеспечение (или облегчение) интерпретации результатов обработки данных.

Существует два способа сокращения (редукции) описания:

- 1) **редукция «снизу вверх»** — постепенное удаление параметров от наименее значимых к более значимым;
- 2) **редукция «сверху вниз»** — выделение наиболее значимых параметров и постепенное дополнение их менее значимыми.

Способ редукции «снизу вверх»:

- 1) определяют наименее значимые параметры и устраняются вместе с соответствующими элементами системы;
- 2) оставшиеся параметры модифицируются так, чтобы наилучшим способом решить задачу;
- 3) циклически повторять пп. 1–2 до тех пор, пока задача не будет решаться с удовлетворительной точностью.

Определение значимости параметров на основании функции оценки

Есть набор x^i , $i = 1, \dots, n$ размерности N , M -мерный вектор параметров w и функция оценки $H(x, w)$, оценивающая работу системы с

параметрами w на векторе x (например, расстояние от вектора выходных сигналов системы до нужного ответа или до множества правильно интерпретируемых ответов). Требуется выделить наименее значимые параметры $w_k, k \in \{1, \dots, M\}$ и компоненты данных x_j и модифицировать систему, отбрасывая наименее значимые параметры. Процедура отбрасывания неоднозначна. Простейший вариант — обращение в ноль — не всегда лучший: он не учитывает корреляции между данными. Учитывая корреляцию, следует отбрасываемые компоненты заменять на функции остающихся компонент.

Пусть для каждого w_k определено фиксированное значение w_k^0 . Отбрасывание j -ой компоненты для i -го примера означает приравнивание $x_j := x_j^0$. В качестве простейшего варианта примем $w_k^0 = 0$ и для любого i полагаем

$$x_j^0 = (1/n) \sum_{p=1}^n x_j^p$$

(параметры обращаются в ноль, данные заменяются средним по выборке). Более тонкие методы предполагают замену отбрасываемых параметров и сигналов на некоторые функции оставшихся.

Показатели значимости вычисляются в два этапа: сначала они оцениваются для одного вектора (примера), потом для всей выборки.

1. Для данного x^p значимости w_k и x_j оцениваются как

$$\begin{aligned} \chi(w_k | x^p) &= |\partial H(x^p, w) / \partial w_k| \times |w_k - w_k^0|, \\ \chi(x_j^p | x^p) &= |\partial H(x^p, w) / \partial x_j^p| \times |x_j^p - x_j^{p0}|. \end{aligned}$$

Здесь χ — вычисленные в линейном приближении абсолютные величины изменения H при сокращении описания. Оценка на всей выборке $x^p, p = 1, \dots, n$ может проводиться по-разному. Например, может использоваться одна из следующих норм:

1. Сумма модулей:

$$\begin{aligned} \chi(w_k) &= \sum_p \chi(w_k | x^p), \\ \chi(x_j) &= \sum_p \chi(x_j | x^p). \end{aligned}$$

2. Максимум модуля

$$\begin{aligned} \chi(w_k) &= \max_p \chi(w_k | x^p), \\ \chi(x_j) &= \max_p \chi(x_j | x^p). \end{aligned}$$

Часто приходится иметь дело с системой, которая меняет свои параметры (например, в ходе обучения). Тогда к моменту принятия решения о

значимости может быть накоплена информация о частных производных H в разных точках $w \in \{w_1, \dots, w_q\}$. Ее можно использовать следующим образом.

Обозначим угловыми скобками процедуру усреднения по множеству параметров $\{w^1, \dots, w^q\}$:

$$\langle f(w, \dots) \rangle = (1/q) \sum_{i=1}^q f(w^i, \dots)$$

положим

$$\begin{aligned} \chi(w_k | x^p) &= \langle |\partial H(x, w) / \partial w_k|_{\chi=\chi^p} \rangle \cdot w_k - |w_k^0|, \\ \chi(x_j | x^p) &= \langle |\partial H(x, w) / \partial x_j|_{\chi=\chi^p} \rangle \cdot x_j^p - |x_j^p|. \end{aligned}$$

Усредняются абсолютные значения производных, а приращения берутся в тех точках, в которых будет проводиться процедура сокращения описания. Усреднение параметров w по нескольким значениям важно для нелинейных систем, в которых производные H могут сильно меняться от точки к точке.

Главная задача при сокращении описания — сохранить качество работы системы, оцениваемое с помощью H . Для этого требуется знать назначение системы и иметь способ оценки ее соответствия своему назначению.

Возможен другой подход, не предполагающий никакого знания о способах оценки. Ставится задача сохранить описание, минимально изменяя функционирование системы. В этом случае роль оценки играет изменение выходного сигнала системы после сокращения.

Определение значимости параметров по изменению выходных сигналов системы

Значимость параметров определяется практически так же, как и с помощью функции оценки. Пусть x — вектор данных, а w — вектор параметров. Пусть задан набор векторов $\{x^p\}$, на которых будет оцениваться функционирование системы, и определены значения x_j^{p0}, w_k^0 (в простейшем случае $w_k^0 = 0, x_j^{p0} = (1/n) \sum_{p=1}^n x_j^p$).

Вычислим в линейном приближении изменение вектора F при обращении w_k в w_k^0 и x_j^p в x_j^{p0} :

$$\begin{aligned} \Delta F &= \partial F(x, w) / \partial w_k |_{\chi=\chi^p} \times (w_k^0 - w_k), \\ \Delta F &= \partial F(x, w) / \partial x_j |_{\chi=\chi^p} \times (x_j^p - x_j^{p0}). \end{aligned}$$

Пусть в пространстве выходных сигналов системы задана некоторая норма (например, евклидова). Тогда положим:

$$\chi(w_k|x^p) = \|\partial F(x, w)/\partial w_k\|_{\chi=\chi^p} \times |w_k^0 - w_k|,$$

$$\chi(x_j|x^p) = \|\partial F(x, w)/\partial x_j\|_{\chi=\chi^p} \times |x_j^p - x_j^{p0}|.$$

Таким образом, для каждого w_k и любого x_j определен вектор показателей значимости. Координаты вектора соответствуют точкам x^p . Теперь нужно вычислить норму этого вектора и объявить ее показателем значимости (можно в качестве нормы взять максимум модуля или сумму модулей). При использовании евклидовой нормы в пространстве выходных сигналов бывает удобно и далее выбирать такую же норму, полагая:

$$\chi(w_k) = \left(\sum_p \chi(w_k|x^p)^2\right)^{1/2},$$

$$\chi(x_j) = \left(\sum_p \chi(x_j|x^p)^2\right)^{1/2}.$$

Подход к определению значимости через изменение выходного сигнала не имеет альтернатив в том случае, когда рассматриваемая система является лишь подсистемой в некоторой системе (например, сумматор или нейрон в нейронной сети). Тогда при изменении параметров этой подсистемы приходится ограничиться требованием: выходной сигнал подсистемы должен изменяться как можно меньше, чтобы не нарушать функционирование всей системы.

Сокращение числа выходов в адаптивном линейном сумматоре (путь «снизу вверх»)

Рассмотрим адаптивный линейный сумматор, вычисляющий линейную функцию $F(x, w) = w_0 + (x, w)$.

Решим задачу о сокращении числа выходных сигналов. Рассмотрим определение значимости по изменению выходного сигнала. Заметим, что:

$$\partial F/\partial w_0 = 1, \quad \partial F/\partial w_i|_{\chi=\chi^p} = x_i^p, \quad \partial F/\partial w_i|_{\chi=\chi^p} = w_i, \quad i = 1, \dots, N.$$

Уничтожить i -й выходной сигнал можно двумя способами:

- 1) заменой параметра w_i на 0;
- 2) заменой x_i на постоянную величину не зависящую от p .

В последнем случае получаем новую функцию

$$F_{1i} = w_0 + x_i^0 w_i + \sum_{j=1, j \neq i}^N x_j w_j$$

Такое преобразование означает, что одновременно с уничтожением i -й выходной связи w_0 приобретает новое значение:

$$w_0 := w_0 + x_i^0 w_i.$$

При этом можно добиться меньшего изменения $F(x, w)$, чем просто приравняв w_i к нулю. Поэтому остановимся на замене i -го выходного сигнала на постоянную величину x_i^0 . Значение этой постоянной определим исходя из минимизации изменения $F(x^p, w)$. Минимизация этого изменения, вычисленного в евклидовой норме, дает:

$$x_i^0 = (1/n) \sum_{p=1}^n x_i^p$$

Таким образом, оптимальной является замена x_i на его среднее значение по исходной выборке. В обозначениях теории вероятностей:

$$x_i^0 = M(x_i), \chi(x_i) = n^{1/2} w_i \sigma(x_i).$$

где $\sigma(x_i)$ — среднее квадратичное отклонение от x_i^0 на выборке $\{x^p\}$.

Значимость замены оценивается как

$$\chi(x_i) = |w_i| \sigma(x_i).$$

При исключении сигналов по одному, они сортируются в соответствии со значениями $\chi(x_i)$ и отбрасываются (заменяются средним) сначала те, что соответствуют меньшим $\chi(x_i)$. Заметим, что поэтому путь «снизу вверх» универсален, но не оптимален. В частности, для сумматоров и других элементов, линейных по параметрам (например, квадратных сумматоров), существует учитывающий все корреляции путь исключения «сверху вниз» с ортогонализацией. Далее ограничимся оценкой значимости по изменению выходного сигнала.

Показатели значимости для нейрона с дифференцируемым нелинейным элементом

Эти показатели ищутся почти так же как для сумматора. Пусть

$$F(x, w) = \varphi(w_0 + (x, w))$$

тогда

$$\partial F / \partial x_i |_{x=x^p} = \varphi'(w_0 + (x^p, w)) \cdot w_i.$$

В евклидовой норме (что соответствует методу наименьших квадратов) получаем:

$$\chi(x_i) = |w_i| \left[\sum_p (\varphi'(w_0 + (x^p, w)))^2 (x_i^p - M(x_i))^2 \right]^{1/2},$$

т. е. произведение модуля параметра w_i на среднеквадратичное отклонение с весами. Роль веса играет квадрат производной функции в точке $w_0 + (x^p, w)$.

Показатели значимости для нейрона с пороговым нелинейным элементом (персептрона)

Эти показатели требуют для своего вычисления еще одного эвристического хода, т. к. прямо воспользоваться предыдущими формулами невозможно. Пусть функция, вычисляемая нейроном, имеет вид

$$\begin{aligned} F(x, w) &= h(w_0 + (x, w)), \\ h(x) &= \{1, x > 0, \\ &0, x \leq 0\} \end{aligned}$$

Для каждого вектора данных x^p необходимо оценить значимость изменения аргумента функции h при замене x_i^p на x_i^0 . Значение $h(a)$ меняется только тогда, когда a меняет знак, поэтому естественно оценивать значимость изменения Δa переменной a в масштабе, определенном текущим значением переменной, т. е. значимость Δa оценивается, как $|\Delta a/a|$. Из этого эвристического рассуждения получаем:

$$(x_i | x^p) = |w_i| \cdot |x_i^p - x_i^0| / |w_0 + (x^p, w)|.$$

Если положить $x_i^0 = M(x_i)$ и воспользоваться евклидовой мерой, то вновь получим произведение модуля параметров w_i на среднеквадратичное отклонение с весом. В качестве весов выступают квадраты величин, обратных выходным сигналам сумматоров:

$$\chi(x_i) = |w_i| \cdot \left[\sum_p (x_i^p - M(x_i))^2 / (w_0 + (x^p, w))^2 \right]^{1/2}.$$

Полученные выражение для показателей значимости позволяют упрощать основные элементы НС «снизу вверх», начиная с исключения самых малозначимых параметров.

Сокращение описания «сверху вниз» — набор достаточного семейства наиболее значимых параметров

Метод исключения параметров «сверху вниз» с ортогонализацией применим не ко всяким функциям $F(x, w)$, а только к таким, которые имеют вид:

$$F(x, w) = \varphi\left(\sum_i w_i f_i(x)\right).$$

Достоинство метода — автоматический учет корреляции между $f_i(x)$. Рассмотрим устройства, вычисляющие функции

$$F(x, w) = \sum_i w_i f_i(x).$$

К ним относятся линейные сумматоры, квадратичные сумматоры и др.

Пусть заданы векторы данных:

$$x^p = (x_1^p, \dots, x_i^p, \dots, x_N^p), p = 1, \dots, n, i = 1, \dots, N.$$

Поставим задачу сокращения описания следующим образом: так определить некоторое наименьшее возможное множество индексов β_J и набор чисел $\beta_j, j \in J$, чтобы норма отклонения $\|\Delta F\| = \|F - F'\|$, где $F' = \sum_{j \in J} \beta_j f_j(x)$, не превышала некоторой наперед заданной величины. Все функции рассматриваются на конечном множестве $\{x^p\}$. Для любой функции $\varphi(x)$ евклидова норма:

$$\|\varphi\| = \left[\sum_p \varphi^2(x^p)\right]^{1/2}.$$

С каждой функцией $f(x)$ связан n -мерный вектор f с компонентами $f^p = f(x^p)$. Вектор F с координатами $F^p = \sum_i w_i f_i(x^p)$ является линейной комбинацией векторов f_i с координатами $f_i^p = f_i(x^p)$. Линейную оболочку семейства векторов f_i обозначим $L = L(\{f_i\})$. Построим в пространстве L ортонормированный базис с помощью последовательной ортогонализации векторов f_i . Каждый следующий шаг ортогонализации выполним так, чтобы величина проекции F на новый вектор базиса была максимальной из возможных. Процесс ортогонализации продолжим, пока $\|F\|^2 - \|F_\perp\|^2 > \xi^2$, где F_\perp — проекция F на построенную ортогональную систему. По окончании процесса полагаем $F' = F_\perp$.

Опишем вычисления более детально.

1. Проводим нормировку: для любого i полагаем $g_i^0 = f_i / \|f_i\|$.
2. Вычисляем $|(F, g_i^0)|$ (модуль проекции вектора F на вектор g_i^0), $i = 1, \dots, N$. Находим среди этих чисел максимальное (пусть его номер $i_{0,max}$), полагаем $e_1 = g_{i_{0,max}}^0$, исключаем $g_{i_{0,max}}^0$ из множества $\{g_i^0\}$, получаем $g_i^1 = g_i^0 - (g_i^0, e_1)e_1$, исключаем из множества $\{g_i^1\}$ нулевые векторы, если таковые существуют, проводим нормировку $g_i^1 = g_i^1 / \|g_i^1\|$.
3. Пусть определены векторы e_1, \dots, e_l и не более чем $n-l$ нормированных векторов g_i^l . Среди векторов g_i^l ищем такой $g_{i,max}^l$, для которого $|(F, g_i^l)|$ принимает максимальное значение, полагаем $e_{l+1} = g_{i,max}^l$, исключаем $g_{i,max}^l$ из множества векторов $\{g_i^l\}$; полагаем $g_i^{l+1} = g_i^l - (g_i^l, e_{l+1})e_{l+1}$, исключаем из этого множества нулевые векторы, если таковые существуют, нормируем: $g_i^{l+1} = g_i^{l+1} / \|g_i^{l+1}\|$.

Вычисления проводим, пока $\|F\|^2 - \sum_{i=1,l} (F, e_i)^2 > \varepsilon^2$.

После завершения вычислений имеем набор ортонормированных векторов $e_1, \dots, e_l, l \leq n$. Они являются линейными комбинациями векторов $f_{i,1max}, \dots, f_{i,lmax}$. Коэффициенты разложения e_j по набору $f_{i,1max}, \dots, f_{i,lmax}$ могут быть вычислены и сохранены в ходе ортогонализации. Полагаем $J = \{i_{1max}, \dots, i_{lmax}\}$. Тогда $F' = \sum_{j=1,l} (F, e_j)e_j = \sum_{j \in J} \beta_j f_j$. Это и есть решение задачи. Числа β_j выражаются через коэффициенты разложения векторов $e_i, i = 1, \dots, l$ по $f_j, j \in J$ и скалярные произведения (F, e_j) : если $e_i = \sum_{j \in J} q_{ij} f_j$, то $\beta_j = \sum_{i=1,l} (F, e_i) q_{ij}$.

Разложение e_i по $f_j, j \in J$ имеет рекурсивную форму:

$$e_1 = f_{i,1max} / \|f_{i,1max}\|,$$

$$e_2 = [f_{i,2max} - (f_{i,2max}, e_1)e_1] / \|f_{i,2max} - (f_{i,2max}, e_1)e_1\|,$$

...

$$e_j = [f_{i,jmax} - \sum_{r=1,j-1} (f_{i,jmax}, e_r)e_r] / \|f_{i,jmax} - \sum_{r=1,j-1} (f_{i,jmax}, e_r)e_r\|,$$

...

Для функций вида $F(x, w) = \varphi(\sum_i w_i f_i(x))$ с дифференцируемой функцией процедура аналогична с точностью до замены скалярного произведения: используется скалярное произведение с весами $(f, g) = \sum_p V_p f^p g^p$, где $V_p = ((\sum_i w_i f_i(x^p)))^2$. В этом скалярном произведении вычисляются все нормы и проводится ортогонализация.

Для функций с пороговой нелинейностью на выходе используем скалярное произведение с весами $V_p = |\sum_i w_i f_i(x^p)|^2$.

Описанная процедура сокращения «сверху вниз» с ортогонализацией особенно важна для упрощения элементов сложных сетей, в структуре которых и вектор входных сигналов элемента может быть далек от исход-

ных данных, и его выходной сигнал далек от оцениваемого выхода всей сложной системы.

Процедуры анализа значимости и сокращения описания выделяют наиболее важные параметры и связи в НС. По аналогии с обработкой изображения их называют процедурами **контрастирования** или **редукции**.

Роль контрастирования (редукции) не сводится только к сокращению описания: более общая задача — привести параметры системы к выделенному набору значений, в частности, уменьшить разрядность, что важно для удешевления специализированных устройств, экономии памяти и т. д.

Рекурсивное контрастирование и бинаризация

Рекурсивное контрастирование состоит в модификации параметров системы — одного за другим. Для этого параметры должны быть как-то линейно упорядочены w_1, \dots, w_N . При модификации w_i используются модифицированные значения w_1, \dots, w_{i-1} и немодифицированные w_{i+1}, \dots, w_N .

Пусть для сумматора задана обучающая выборка входных векторов x_1, \dots, x_n и соответствующих выходных сигналов f_1, \dots, f_n , а также известны значения параметров, которые реализуют сумматор: $f_i = w_0 + (x^i, w)$. Требуется произвести бинаризацию сумматора, т. е. найти числа a, b и вектор β с координатами 0 или 1, чтобы значения функции $\varphi(x) = a + b(x, \beta)$ на выборке $\{x^i\}$ как можно меньше отличались от f_i . Критерием такого отличия будем считать $H = \sum_{i=1, n} (f_i - \varphi(x^i))^2$. Построим координаты вектора β по порядку β_1, β_2, \dots .

Пусть построены $\beta_1, \dots, \beta_{i-1}$. Обозначим $\beta^{0i} = (\beta_1, \dots, \beta_{i-1}, 0, \dots, 0)$ (последние $N - i + 1$ координат — нули), $\beta^{1i} = (\beta_1, \dots, \beta_{i-1}, 1, 0, \dots, 0)$ (последние $N - i$ координат — нули), $\alpha^i = (0, \dots, 0, \alpha_{i+1}, \dots)$ (первые i координат — нули).

Введем функции:

$$\varphi_0^i(x) = a_{0i} + b_{0i}(x, \beta^{0i}) + (x, \alpha^i),$$

$$\varphi_1^i(x) = a_{1i} + b_{1i}(x, \beta^{1i}) + (x, \alpha^i),$$

$$H_{0i} = \sum_{j=1, n} (f_j - \varphi_0^i(x^j))^2,$$

$$H_{1i} = \sum_{j=1, n} (f_j - \varphi_1^i(x^j))^2.$$

Определим параметры $a_{0i}, b_{0i}, a_{1i}, b_{1i}$ из условий $H_{0i} \rightarrow \min, H_{1i} \rightarrow \min$, минимизируя функции H_{0i} и H_{1i} . Пусть $h_{0i} = \min H_{0i}$

и $h_{1i} = \min H_{1i}$. Если $h_{1i} \geq h_{0i}$, то полагаем $\beta_i = 0$, в противном случае $\beta_i = 1$.

После того как построены все $\beta_i, i = 1, \dots, N$ (N — размерность вектора данных), автоматически определяются a и b : если $N = 0$, то полагаем $a = a_{0N}, b = b_{0N}$, иначе $a = a_{1N}, b = b_{1N}$.

Если бинаризация проведена, а необходимая точность не достигнута, то можно построить второй бинаризованный сумматор, корректирующий ошибку первого — так, чтобы в сумме они хорошо аппроксимировали работу исходного сумматора на элементах обучающей выборки. В описанной процедуре делаем замену $f_i := f_i - \varphi(x^i)$ и для этих исходных данных вновь строим бинаризованный сумматор по алгоритму рекурсивной бинаризации. Повторяем такое построение, пока не будет достигнута удовлетворительная точность. В результате получим набор бинаризованных сумматоров, которые в совокупности (т.е. в результате суммирования выходных сигналов) достаточно точно аппроксимируют исходный. При появлении весов, определяющих значимость отдельных примеров из обучающей выборки, рекурсивная бинаризация проводится точно так же, только в функциях H появляются веса.

Если требуется тем же путем упростить любой другой элемент, линейный по параметрам, $F(x, w) = \sum_i w_i f_i(x)$, то вместо обучающей выборки $\{x^j\}$ берем семейство векторов $\{y^j\}$ с координатами $y_i^j = f_i(x^j)$. После такого $y_i^j = f_i(x^j)$ преобразования рассматриваемый элемент превращается в обычный сумматор, для которого последовательность действий уже описана.

Лекция 17. Методы аппаратной реализации нейрокомпьютеров

Рассматриваются электронные и оптические методы реализации нейрокомпьютеров.

Ключевые слова: нейрочипы, нейропроцессор NM6403, оптические системы.

Электронная реализация нейронных сетей

В качестве единицы производительности нейросетей принято «число соединений в секунду» — CPS (connections per second). Под соединением здесь понимается умножение входного сигнала на весовой коэффициент и сложение с накопленной суммой.

Анализ нейросетевых алгоритмов позволяет сделать следующие выводы:

1) При решении плохо формализованных задач моделирования, прогнозирования и распознавания, которые обычно сводятся к конструированию областей многомерного пространства, достаточно малоразрядных представлений входов и весов и операций с фиксированной точкой. Это обусловлено тем, что входные сигналы нормируются и количество их значений невелико.

2) При решении хорошо формализованных задач (например, задач комбинаторной оптимизации) существенна точность вычислений, что требует полноразрядных представлений чисел и операций с плавающей точкой.

Электронные нейронные сети обычно используются в качестве акселераторов для персональных ЭВМ при решении соответствующих классов задач (обработки сигналов и изображений, распознавания образов и т. п.).

Нейрочипы

Нейрочипы подразделяются на цифровые, аналоговые и гибридные. Они могут включать в себя схемы настройки весов при обучении или предусматривать внешнюю загрузку весов. Наибольшую проблему при создании нейрочипов представляют схемы умножения, так как именно они лимитируют скорость вычислений.

Аналоговые реализации используют простые физические эффекты для выполнения нейросетевых преобразований. Обеспечение заданной точности требует тщательного проектирования и изготовления.

Гибридные нейрочипы используют комбинацию аналогового и цифрового подходов. Например, входы могут быть аналоговыми, веса могут загружаться как цифровые и выходы могут быть цифровыми. Существуют нейрочипы, в которых используется представление данных частотой или шириной импульсов.

Нейропроцессор NM6403

Отечественный нейропроцессор NM6403, разработанный в НПО «Модуль» (www.module.ru), имеет скалярный процессор (скалярное RISC-ядро) и векторный процессор для обработки двоичных векторов произвольной разрядности в пределах 1–64 битов. Скалярный процессор выполняет всю подготовку данных для работы векторного процессора.

Модель слоя нейронов, эмулируемого процессором NM6403, показана на рис. 1. В общем случае слой имеет N входов и состоит из M нейронов. M -й нейрон выполняет операцию умножения — накопления над N данными x_1, \dots, x_N , поданными на соответствующие входы нейрона.

Нейропроцессор NM6403 имеет два встроенных линка, совместимых с линками сигнального микропроцессора TMS320C40. Кроме того, интерфейсы локальной и глобальной памяти позволяют без дополнительного оборудования подсоединять два нейропроцессора к общему блоку памяти.

Мультипроцессорная система (рис. 2) из K нейропроцессоров NM6403 эмулирует нейронную сеть в K раз быстрее, чем один нейропроцессор.

Оптическая реализация нейронных сетей

Мощность нейронной сети определяется большим количеством связей: отдельные элементы имеют относительно малые вычислительные мощности. Обеспечение требуемой связности в электронных цепях остается серьезной проблемой, особенно при реализации нейронных сетей с полным графом соединений. Электронные интегральные цепи являются существенно планарными с рельефностью, обусловленной множеством слоев.

Проблему связей можно решить при использовании **оптических систем** для реализации НС. Взаимное соединение нейронов с помощью световых лучей не требует изоляции между сигнальными путями: световые потоки могут пересекаться, не влияя друг на друга, и сигнальные пути

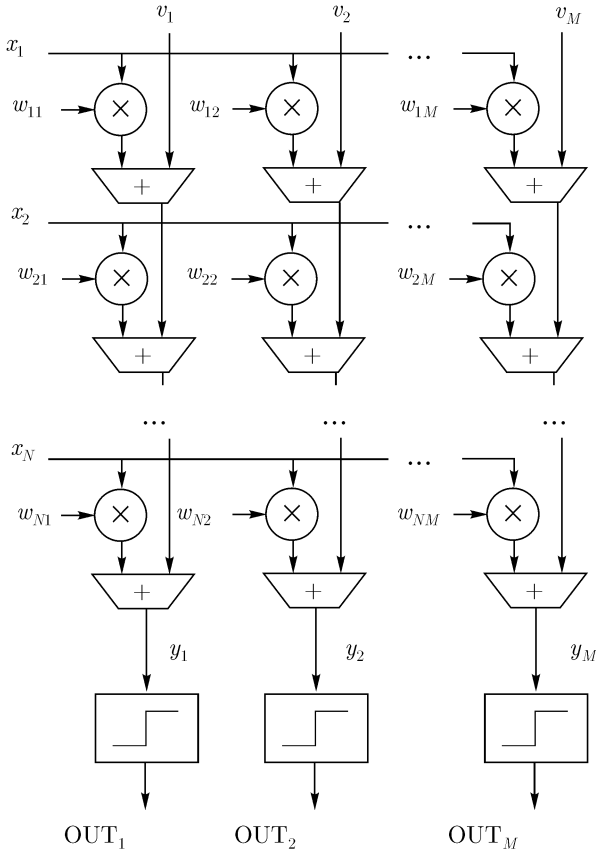


Рис. 1. Модель слоя нейронов

могут располагаться в трех измерениях. Плотность путей передачи ограничена только размерами источников и детекторов. Все сигнальные пути могут работать одновременно, тем самым обеспечивая огромную скорость передачи данных.

В оптических НС величины оптических весов могут запоминаться в голограммах с высокой степенью плотности (до 10^{12} бит на куб. см.). Веса могут модифицироваться в процессе работы сети.

К сожалению, возникает множество практических проблем при попытках оптической реализации нейронных сетей. Оптические устройства имеют собственные физические характеристики, часто не соответствующие требованиям искусственных нейронных сетей. Хотя они в действи-

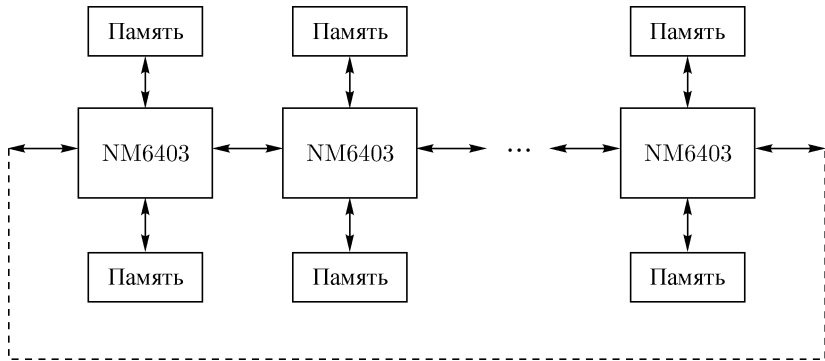


Рис. 2. Линейная (кольцевая) система из нейропроцессоров NM6403

тельности пригодны для обработки изображений, все же изображения от оптических нейронных сетей, полученные до настоящего времени, были разочаровывающе плохими. Однако достаточно взглянуть на первые пробы телевизионных передач, чтобы понять, какой огромный прогресс возможен в повышении качества изображения. Несмотря на эти трудности, а также на такие проблемы, как стоимость, размеры и критичность к ориентации, потенциальные возможности оптических систем побуждают попытки проведения интенсивных и широких исследований. В этой области происходят стремительные изменения, и в ближайшее время ожидаются важные улучшения.

Конфигурации оптических НС в основном подразделяются на две категории: векторно-матричные умножители и голографические корреляторы.

Векторно-матричные умножители

В качестве матрицы весов (рис. 3) используется фотопленка, у которой прозрачность каждого квадрата пропорциональна весу. Выход каждого фотодетектора является сверткой между входным вектором и соответствующим столбцом матрицы весов. Умножение выполняется параллельно. При использовании соответствующих высокоскоростных светодиодов и фотодетекторов умножение вектора на матрицу может быть выполнено менее, чем за наносекунду. Более того, скорость умножения практически не зависит от размерности массива. Это позволяет наращивать сети без существенного увеличения времени вычислений. Возможность менять веса основана на использовании жидкокристаллического клапана вместо фотографического негатива.

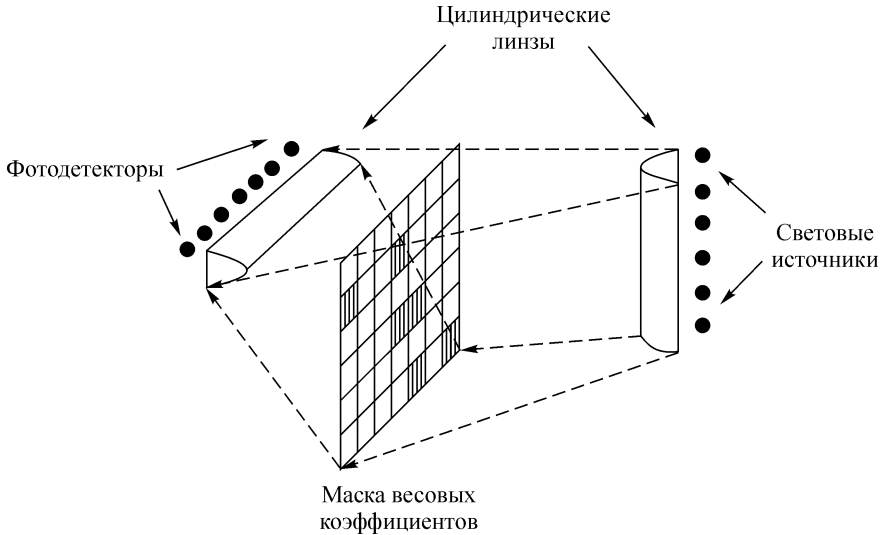


Рис. 3. Электронно-оптический векторно-матричный умножитель

Голографические корреляторы

В голографических корреляторах образцы изображений запоминаются в виде голограммы (плоской или объемной) и восстанавливаются при когерентном освещении в петле обратной связи.

Входное изображение (возможно, зашумленное или неполное) коррелируется оптически одновременно со всеми запомненными изображениями. Корреляции обрабатываются пороговой функцией и подаются на вход системы, где наиболее сильные корреляции усиливают (и, возможно, корректируют или завершают) входное изображение. Этот процесс повторяется многократно, и усиленное изображение при каждом проходе изменяется, пока система не стабилизируется на требуемом изображении.

Оптические нейронные сети предлагают огромные выгоды с точки зрения скорости и плотности внутренних связей. Они могут быть использованы (в той или иной форме) для реализации сетей фактически с любой архитектурой.

В настоящее время ограничения электронно-оптических устройств создают множество серьезных проблем, которые должны быть решены прежде, чем оптические нейронные сети получат широкое применение. Однако, учитывая, что большое количество превосходных исследователей работает над этой проблемой, а также большую поддержку со стороны военных, можно надеяться на быстрый прогресс в этой области.

Литература

- [1] Горбань А. Н., Россиев Д. А. Нейронные сети на персональном компьютере. — Новосибирск: Наука, 1996.
- [2] Осовский С. Нейронные сети для обработки информации. — М.: Финансы и статистика, 2002.
- [3] Уоссермен Ф. Нейрокомпьютерная техника. Теория и практика. — М.: Мир, 1992.
- [4] Каллан Р. Основные концепции нейронных сетей. — М.:Изд. дом «Вильямс», 2001.
- [5] Заенцев И. В. Нейронные сети. Основные модели. — Воронеж: ВГУ, 1999.
- [6] Горбань А. Н. Обучение нейронных сетей. — М.: СП Параграф, 1990.
- [7] Миркес Е. М. Нейрокомпьютер. Проект стандарта. — Новосибирск: Наука, 1999.
- [8] Нейроинформатика / А. Н. Горбань и др. — Новосибирск: Наука, 1998.
- [9] Ачасова С. М. Вычисления на нейронных сетях (обзор) // Программирование. 1991, №2. — С. 40–53.
- [10] Вороновский Г. К. и др. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности. Харьков: Основа. 1997.
- [11] Chevchenko P.A., Fomine D.V., Tchernikov V.M., and Vixne P.E., Using of microprocessor NM6403 for neural net emulation // <http://www.module.ru>.
- [12] Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория и практика. — М.: Горячая линия - Телеком, 2002.
- [13] Назаров А. В., Лоскутов А. И. Нейросетевые алгоритмы прогнозирования и оптимизации систем. — СПб.: Наука и Техника, 2003.
- [14] Нейроматематика. Кн. 6. / Под ред. А. И. Галушкина. — М.: ИПЖР, 2002 (Нейрокомпьютеры и их применение).

- [15] Ежов А. А., Шумский С. А. Нейрокомпьютинг и его применение в экономике и бизнесе. — Москва, 1998.
- [16] Минаев Ю. Н., Филимонова О. Ю., Бенамеур Лиес, Методы и алгоритмы решения задач идентификации и прогнозирования в условиях неопределенности в нейросетевом логическом базисе. — М.: Горячая линия - Телеком, 2003.
- [17] Ярушкина Н. Г. Основы теории нечетких и гибридных систем. — М.: Финансы и статистика, 2004.
- [18] Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. — М.: Горячая линия - Телеком, 2004.
- [19] Комашинский В. И., Смирнов Д. А. Нейронные сети и их применение в системах управления и связи. — М.: Горячая линия - Телеком, 2003.