Контрастер

Компонент контрастер предназначен для контрастирования нейронных сетей. Первые работы, посвященные контрастированию (скелетонизации) нейронных сетей появились в начале девяностых годов [64.]. Однако, задача контрастирования нейронных сетей не являлась центральной, поскольку упрощение сетей может принести реальную пользу только при реализации обученной нейронной сети в виде электронного (оптоэлектронного) устройства. Только в работе А.Н. Горбаня и Е.М. Миркеса «Логически прозрачные нейронные сети» [82] (более полный вариант работы см. [75]), опубликованной в 1995 году задаче контрастирования нейронных сетей был придан самостоятельный смысл – впервые появилась реальная возможность получать новые явные знания из данных. В связи с тем, что контрастирование нейронных сетей не является достаточно развитой ветвью нейроинформатики, стандарт, приведенный в данной главе, является очень общим.

Задачи для контрастера

Из анализа литературы и опыта работы группы НейроКомп можно сформулировать следующие задачи, решаемые с помощью контрастирования нейронных сетей.

1. Упрощение архитектуры нейронной сети.
2. Уменьшение числа входных сигналов.
3. Сведение параметров нейронной сети к небольшому набору выделенных значений.
4. Снижение требований к точности входных сигналов.
5. Получение явных знаний из данных.

Далее в этом разделе все перечисленные выше задачи рассмотрены более подробно.

Упрощение архитектуры нейронной сети

Стремление к упрощению архитектуры нейронных сетей возникло из попытки ответить на следующие вопрос: «Сколько нейронов нужно использовать и как они должны быть связаны друг с другом?» При ответе на этот вопрос существует две противоположные точки зрения. Одна из них утверждает, что чем больше нейронов использовать, тем более надежная сеть получится. Сторонники этой позиции ссылаются на пример человеческого мозга. Действительно, чем больше нейронов, тем больше число связей между ними, и тем более сложные задачи способна решить нейронная сеть. Кроме того, если использовать заведомо большее число нейронов, чем необходимо для решения задачи, то нейронная сеть точно обучится. Если же начинать с небольшого числа нейронов, то сеть может оказаться неспособной обучиться решению задачи, и весь процесс придется повторять сначала с большим числом нейронов. Эта точка зрения (чем больше – тем лучше) популярна среди разработчиков нейросетевого программного обеспечения. Так, многие из них как одно из основных достоинств своих программ называют возможность использования любого числа нейронов.

Вторая точка зрения опирается на такое «эмпирическое» правило: чем больше подгоночных параметров, тем хуже аппроксимация функции в тех областях, где ее значения были заранее неизвестны. С математической точки зрения задачи обучения нейронных сетей сводятся к продолжению функции заданной в конечном числе точек на всю область определения. При таком подходе входные данные сети считаются аргументами функции, а ответ сети – значением функции. На рис. 1 приведен пример аппроксимации табличной функции полиномами 3-й (рис. 1.а) и 7-й (рис. 1.б) степеней. Очевидно, что аппроксимация, полученная с помощью полинома 3-ей степени больше соответствует внутреннему представлению о «правильной» аппроксимации. Несмотря на свою простоту, этот пример достаточно наглядно демонстрирует суть проблемы.

Второй подход определяет нужное число нейронов как минимально необходимое. Основным недостатком является то, что это, минимально необходимое число, заранее неизвестно, а процедура его определения путем постепенного наращивания числа нейронов весьма трудоемка. Опираясь

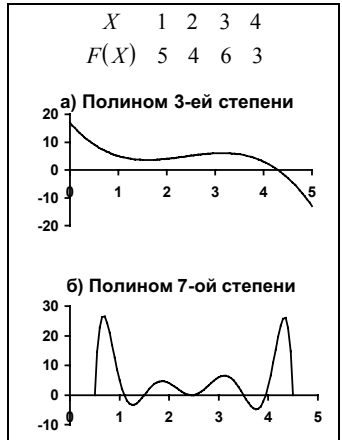

Рис. 1. Аппроксимация табличной функции

на опыт работы группы НейроКомп в области медицинской диагностики [[18, 49 – 52, 72, 90, 91, 160, 161, 165, 182 – 187, 190 – 208, 255, 295 – 298, 316, 317, 341 – 345, 351, 361]], космической навигации и психологии можно отметить, что во всех этих задачах ни разу не потребовалось более нескольких десятков нейронов.

Подводя итог анализу двух крайних позиций, можно сказать следующее: сеть с минимальным числом нейронов должна лучше («правильнее», более гладко) аппроксимировать функцию, но выяснение этого минимального числа нейронов требует больших интеллектуальных затрат и экспериментов по обучению сетей. Если число нейронов избыточно, то можно получить результат с первой попытки, но существует риск построить «плохую» аппроксимацию. Истина, как всегда бывает в таких случаях, лежит посередине: нужно выбирать число нейронов большим, чем необходимо, но не намного. Это можно осуществить путем удвоения числа нейронов в сети после каждой неудачной попытки обучения. Наиболее надежным способом оценки минимального числа нейронов является использование процедуры контрастирования. Кроме того, процедура контрастирования позволяет ответить и на второй вопрос: какова должна быть структура сети.

Уменьшение числа входных сигналов

При постановке задачи для нейронной сети не всегда удается точно определить сколько и каких входных данных нужно подавать на вход. В случае недостатка данных сеть не сможет обучиться решению задачи. Однако гораздо чаще на вход сети подается избыточный набор входных параметров. Например, при обучении сети постановке диагноза в задачах медицинской диагностики на вход сети подаются все данные, необходимые для постановки диагноза в соответствии с существующими методиками. Следует учесть, что стандартные методики постановки диагнозов разрабатываются для использования на большой территории (например, на территории России). Как правило, при диагностике заболеваний населения какого-нибудь небольшого региона (например города) можно обойтись меньшим набором исходных данных. Причем этот усеченный набор будет варьироваться от одного малого региона к другому. Требуется определить, какие данные необходимы для решения конкретной задачи, поставленной для нейронной сети. Кроме того, в ходе решения этой задачи определяются значимости входных сигналов. Следует заметить, что умение определять значимость входных сигналов представляет самостоятельную ценность.

Сведение параметров нейронной сети к выделенным значениям

При обучении нейронных сетей на универсальных компьютерах параметры сети являются действительными числами из заданного диапазона. При аппаратной реализации нейронной сети не всегда возможно реализовать веса связей с высокой точностью (в компьютерном представлении действительных чисел хранятся первые 6-7 цифр мантиссы). Опыт показывает, что в обученной сети веса многих синапсов можно изменять в довольно широком диапазоне (до полуширины интервала изменения веса) не изменяя качество решения сетью поставленной перед ней задачи. Исходя из этого, полезно уметь решать задачу замены значений параметров сети на значения из заданного набора.

Снижение требований к точности входных сигналов

При обработке экспериментальных данных полезно знать, что измерение с высокой точностью, как правило, дороже измерения с низкой точностью. Причем достаточно часто получение очередной значащей цифры измеряемого параметра стоит на несколько порядков дороже. В связи с этим задача снижения требований к точности измерения входных параметров сети приобретает смысл.

Получение явных знаний из данных

Одной из главных загадок мышления является то, как из совокупности данных об объекте, появляется знание о нем. До недавнего времени наибольшим достижением в области искусственного интеллекта являлось либо воспроизведение логики человека-эксперта (классические экспертные системы), либо построение регрессионных зависимостей и определение степени зависимости одних параметров от других.

С другой стороны, одним из основных недостатков нейронных сетей, с точки зрения многих пользователей, является то, что нейронная сеть решает задачу, но не может рассказать как. Иными словами из обученной нейронной сети нельзя извлечь алгоритм решения задачи. Таким образом нейронные сети позволяют получать неявные знания из данных.

В домашнем задании I Всесоюзной олимпиады по нейрокомпьютингу, проходившей в мае 1991 года в городе Омске, в исследовательской задаче участникам было предложено определить, как нейронная сеть решает задачу распознавания пяти первых букв латинского алфавита (полный текст задания и

наиболее интересные варианты решения приведены в [47]). Это была первая попытка извлечения алгоритма решения задачи из обученной нейронной сети.

В 1995 году была сформулирована идея логически прозрачных сетей, то есть сетей на основе структуры которых можно построить вербальное описание алгоритма получения ответа. Это достигается при помощи специальным образом построенной процедуры контрастирования.

Построение логически прозрачных сетей

Зададимся классом сетей, которые будем считать логически прозрачными (то есть такими, которые решают задачу понятным для нас способом, для которого легко сформулировать словесное описания в виде явного алгоритма). Например потребуем, чтобы все нейроны имели не более трех входных сигналов.

Зададимся нейронной сетью у которой все входные сигналы подаются на все нейроны входного слоя, а все нейроны каждого следующего слоя принимают выходные сигналы всех нейронов предыдущего слоя. Обучим сеть безошибочному решению задачи.

После этого будем производить контрастирование в несколько этапов. На первом этапе будем удалять только входные связи нейронов входного слоя. Если после этого у некоторых нейронов осталось больше трех входных сигналов, то увеличим число входных нейронов. Затем аналогичную процедуру выполним поочередно для всех остальных слоев. После завершения описанной процедуры будет получена логически прозрачная сеть. Можно произвести дополнительное контрастирование сети, чтобы получить минимальную сеть. На рис. 2 приведены восемь минимальных сетей. Если под логически прозрачными сетями понимать сети, у которых каждый нейрон имеет не более трех входов, то все сети кроме пятой и седьмой являются логически прозрачными. Пятая и седьмая сети демонстрируют тот факт, что минимальность сети не влечет за собой логической прозрачности.

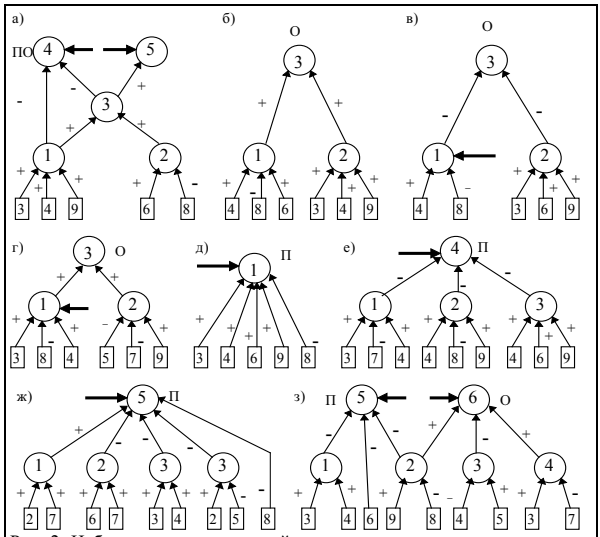

Рис. 2. Набор минимальных сетей для решения задачи о предсказании результатов выборов президента США. В рисунке использованы следующие обозначения: буквы «П» и «О» – обозначают вид ответа, выдаваемый нейроном: «П» – положительный сигнал означает победу правящей партии, а отрицательный – оппозиционной; «О» – положительный сигнал означает победу оппозиционной партии, а отрицательный – правящей;

Получение явных знаний

После получения логически прозрачной нейронной сети наступает этап построения вербального описания. Принцип построения вербального описания достаточно прост. Используемая терминология заимствована из медицины. Входные сигналы будем называть симптомами. Выходные сигналы нейронов первого слоя – синдромами первого уровня. Очевидно, что синдромы первого уровня строятся из симптомов. Выходные сигналы нейронов k-о слоя будем называть синдромами k-о уровня. Синдромы k-о первого уровня строятся из симптомов и синдромов более низких уровней. Синдром последнего уровня является ответом.

В качестве примера приведем интерпретацию алгоритма рассуждений, полученного по второй сети приведенной на рис. 2. Постановка задачи: по ответам на 12 вопросов необходимо предсказать по-

беду правящей или оппозиционной партии на выборах Президента США. Ниже приведен список вопросов.

1. Правящая партия была у власти более одного срока?
2. Правящая партия получила больше 50% голосов на прошлых выборах?
3. В год выборов была активна третья партия?
4. Была серьезная конкуренция при выдвижении от правящей партии?
5. Кандидат от правящей партии был президентом в год выборов?
6. Год выборов был временем спада или депрессии?
7. Был ли рост среднего национального валового продукта на душу населения больше 2.1%?
8. Произвел ли правящий президент существенные изменения в политике?
9. Во время правления были существенные социальные волнения?
10. Администрация правящей партии виновна в серьезной ошибке или скандале?
11. Кандидат от правящей партии – национальный герой?
12. Кандидат от оппозиционной партии – национальный герой?

Ответы на вопросы описывают ситуацию на момент, предшествующий выборам. Ответы кодировались следующим образом: «да» – единица, «нет» – минус единица. Отрицательный сигнал на выходе сети интерпретируется как предсказание победы правящей партии. В противном случае, ответом считается победа оппозиционной партии. Все нейроны реализовывали пороговую функцию, равную 1, если алгебраическая сумма входных сигналов нейрона больше либо равна 0, и -1 при сумме меньшей 0.

Проведем поэтапно построение вербального описания второй сети, приведенной на рис. 2. После автоматического построения вербального описания получим текст, приведенный на рис. 3. Заменим все симптомы на тексты соответствующих вопросов. Заменим формулировку восьмого вопроса на обратную. Подставим вместо Синдром1_Уровня2 название ответа сети при выходном сигнале 1. Текст, полученный в результате этих преобразований приведен на рис. 4.

Заметим, что все три вопроса, ответы на которые формируют Синдром1_Уровня1, относятся к оценке качества правления действующего президента. Поскольку положительный ответ на любой из этих вопросов характеризует недостатки правления, то этот синдром можно назвать синдромом плохой политики. Аналогично, три вопроса, ответы на которые формируют Синдром2_Уровня1, относятся к характеристике политической стабильности. Этот синдром назовем синдромом политической нестабильности.

Тот факт, что оба синдрома первого уровня принимают значение 1, если истинны ответы хотя бы на два из трех вопросов, позволяет избавиться от математических действий с ответами на вопросы. Окончательный ответ может быть истин-

Синдром1_Уровня1 равен 1, если выражение Симптом4 + Симптом6 – Симптом 8 больше либо равно нулю, и –1 – в противном случае.

Синдром2_Уровня1 равен 1, если выражение Симптом3 + Симптом4 + Симптом9 больше либо равно нулю, и –1 – в противном случае.

Синдром1_Уровня2 равен 1, если выражение Синдром1_Уровня1 + Синдром2_Уровня1 больше либо равно нулю, и –1 – в противном случае.

Рис. 3. Автоматически построенное вербальное описание

Синдром1_Уровня1 равен 1, если выражение «Была серьезная конкуренция при выдвижении от правящей партии?» + «Год выборов был временем спада или депрессии?» + «Правящий президент не произвел существенных изменений в политике?» больше либо равно нулю, и –1 – в противном случае.

Синдром2_Уровня1 равен 1, если выражение «В год выборов была активна третья партия?» + «Была серьезная конкуренция при выдвижении от правящей партии?» + «Во время правления были существенные социальные волнения?» больше либо равно нулю, и –1 – в противном случае.

Оппозиционная партия победит, если выражение Синдром1_Уровня1 + Синдром2_Уровня1 больше либо равно нулю.

Рис. 4. Вербальное описание после элементарных преобразований

Правление плохое, если верны хотя бы два из следующих высказываний: «Была серьезная конкуренция при выдвижении от правящей партии», «Год выборов был временем спада или депрессии», «Правящий президент не произвел существенных изменений в политике».

Ситуация политически нестабильна, если верны хотя бы два из следующих высказываний: «В год выборов была активна третья партия», «Была серьезная конкуренция при выдвижении от правящей партии», «Во время правления были существенные социальные волнения».

Оппозиционная партия победит, если правление плохое или ситуация политически нестабильна.

Рис. 5. Окончательный вариант вербального описания

ным только если оба синдрома имеют значение –1.

Используя приведенные соображения, получаем окончательный текст решения задачи о предсказании результатов выборов президента США, приведенный на рис. 5.

Таким образом, использовав идею логически прозрачных нейронных сетей и минимальные интеллектуальные затраты на этапе доводки вербального описания, был получен текст решения задачи. Причем процедура получения логически прозрачных нейронных сетей сама отобрала значимые признаки, сама привела сеть к нужному виду. Далее элементарная программа построила по структуре сети вербальное описание.

На рис. 2 приведены структуры шести логически прозрачных нейронных сетей, решающих задачу о предсказании результатов выборов президента США [299 – 301]. Все сети, приведенные на этом рисунке минимальны в том смысле, что из них нельзя удалить ни одной связи так, чтобы сеть могла обучиться правильно решать задачу. По числу нейронов минимальна пятая сеть.

Заметим, что все попытки авторов обучить нейронные сети со структурами, изображенными на рис. 2, и случайно сгенерированными начальными весами связей закончились провалом. Все сети, приведенные на рис. 2, были получены из существенно больших сетей с помощью процедуры контрастирования. Сети 1, 2, 3 и 4 были получены из трехслойных сетей с десятью нейронами во входном и скрытом слоях. Сети 5, 6, 7 и 8 были получены из двухслойных сетей с десятью нейронами во входном слое. Легко заметить, что в сетях 2, 3, 4 и 5 изменилось не только число нейронов в слоях, но и число слоев. Кроме того, почти все веса связей во всех восьми сетях равны либо 1, либо -1.

Процедура контрастирования

Существует два типа процедуры контрастирования – контрастирование по значимости параметров и не ухудшающее контрастирование. В данном разделе описаны оба типа процедуры контрастирования.

Контрастирование на основе показателей значимости

С помощью этой процедуры можно контрастировать, как входные сигналы, так и параметры сети. Далее в данном разделе будем предполагать, что контрастируются параметры сети. При контрастировании входных сигналов процедура остается той же, но вместо показателей значимости параметров сети используются показатели значимости входных сигналов. Обозначим через χ_p – показатель значимости p-о параметра; через w_p^0 – текущее значение p-о параметра; через w_p^* – ближайшее выделенное значение для p-о параметра.

Используя введенные обозначения процедуру контрастирования можно записать следующим образом:

1. Вычисляем показатели значимости.
2. Находим минимальный среди показателей значимости – χ_{p^*}.
3. Заменим соответствующий этому показателю значимости параметр $w_{p^*}^0$ на $w_{p^*}^*$, и исключаем его из процедуры обучения.
4. Предъявим сети все примеры обучающего множества. Если сеть не допустила ни одной ошибки, то переходим ко второму шагу процедуры.
5. Пытаемся обучить полученную сеть. Если сеть обучилась безошибочному решению задачи, то переходим к первому шагу процедуры, в противном случае переходим к шестому шагу.
6. Восстанавливаем сеть в состояние до последнего выполнения третьего шага. Если в ходе выполнения шагов со второго по пятый был отконтрастирован хотя бы один параметр, (число обучаемых параметров изменилось), то переходим к первому шагу. Если ни один параметр не был отконтрастирован, то получена минимальная сеть.

Возможно использование различных обобщений этой процедуры. Например, контрастировать за один шаг процедуры не один параметр, а заданное пользователем число параметров. Наиболее радикальная процедура состоит в контрастировании половины параметров связей. Если контрастирование половины параметров не удается, то пытаемся контрастировать четверть и т.д. Другие варианты обобщения процедуры контрастирования будут описаны при описании решения задач. Результаты первых работ по контрастированию нейронных сетей с помощью описанной процедуры опубликованы в [47, 302, 303].

Контрастирование без ухудшения

Пусть нам дана только обученная нейронная сеть и обучающее множество. Допустим, что вид функции оценки и процедура обучения нейронной сети неизвестны. В этом случае так же возможно контрастирование сети. Предположим, что данная сеть идеально решает задачу. В этом случае возможно контрастирование сети даже при отсутствии обучающей выборки, поскольку ее можно сгенерировать используя сеть для получения ответов. Задача не ухудшающего контрастирования ставится следующим образом: необходимо так провести контрастирование параметров, чтобы выходные сигналы сети при решении всех примеров изменились не более чем на заданную величину. Для решения задача редуцируется на отдельный адаптивный сумматор: необходимо так изменить параметры, чтобы выходной сигнал адаптивного сумматора при решении каждого примера изменился не более чем на заданную величину.

Обозначим через x_p^q p-й входной сигнал сумматора при решении q-о примера; через f^q – выходной сигнал сумматора при решении q-о примера; через w_p – вес p-о входного сигнала сумматора; через ε – требуемую точность; через n – число входных сигналов сумматора; через m – число примеров. Очевидно, что при решении примера выполняется равенство $f^q = \sum_{p=1}^{n} w_p x_p^q$. Требуется найти такой набор индексов $I = \{i_1,\ldots,i_k\}$, что $\left\| f - \sum_{p\in I} \alpha_p x_p \right\| < \varepsilon$, где α_p – новый вес p-о входного сигнала сумматора. Набор индексов будем строить по следующему алгоритму.

1. Положим $f^{(0)} = f$, $x_p^* = x_p$, $I^{(0)} = \varnothing$, $J^{(0)} = \{1,\ldots,n\}$, k=0.
2. Для всех векторов x_p^* таких, что $p \in J^{(k)}$, проделаем следующее преобразование: если $\|x_p^*\| << \varepsilon$, то исключаем p из множества обрабатываемых векторов – $J^{(k)} = J^{(k)} / \{p\}$, в противном случае нормируем вектор x_p^* на единичную длину – $x_p^{(k)} = x_p^* / \|x_p^*\|$.
3. Если $\|f^{(k)}\| < \varepsilon$ или $J^{(0)} = \varnothing$, то переходим к шагу 10.
4. Находим i_{k+1} – номер вектора, наиболее близкого к $f^{(k)}$ из условия
$$\left|\left(f^{(k)}, x_{i_{k+1}}^{(k)}\right)\right| = \min_{p\in J^{(k)}} \left|\left(f^{(k)}, x_p^{(k)}\right)\right|.$$
5. Исключаем i_{k+1} из множества индексов обрабатываемых векторов: $J^{(k+1)} = J^{(k)} / \{i_{k+1}\}$.
6. Добавляем i_{k+1} в множество индексов найденных векторов: $I^{(k+1)} = I^{(k)} \bigcup \{i_{k+1}\}$
7. Вычисляем не аппроксимированную часть (ошибку аппроксимации) вектора выходных сигналов: $f^{(k+1)} = f^{(k)} - \left(f^{(k)}, x_{i_{k+1}}^{(k)}\right) x_{i_{k+1}}^{(k)}$
8. Преобразуем обрабатываемые вектора к промежуточному представлению – ортогонализуем их к вектору $x_{i_{k+1}}^{(k)}$, для чего каждый вектор $x_p^{(k)}$, у которого $p \in J^{(k)}$ преобразуем по следующей формуле: $x_p^* = x_p^{(k)} - \left(x_p^{(k)}, x_{i_{k+1}}^{(k)}\right) x_{i_{k+1}}^{(k)}$.
9. Увеличиваем k на единицу и переходим к шагу 2.
10. Если $k = 0$, то весь сумматор удаляется из сети и работа алгоритма завершается.
11. Если $k = n+1$, то контрастирование невозможно и сумматор остается неизменным.

12. В противном случае полагаем $I = I^{(k)}$ и вычисляем новые веса связей α_p ($p \in I$) решая систему уравнений $f - f^{(k)} = \sum_{p \in I} \alpha_p x_p$.
13. Удаляем из сети связи с номерами $p \in J$, веса оставшихся связей полагаем равными α_p ($p \in I$).

Данная процедура позволяет производить контрастирование адаптивных сумматоров. Причем значения, вычисляемые каждым сумматором после контрастирования, отличаются от исходных не более чем на заданную величину. Однако, исходно была задана только максимально допустимая погрешность работы сети в целом. Способы получения допустимых погрешностей для отдельных сумматоров исходя из заданной допустимой погрешности для всей сети описаны в ряде работ [94 –96, 167, 209 – 213, 352].

Гибридная процедура контрастирования

Можно упростить процедуру контрастирования, описанную в разд. «Контрастирование без ухудшения». Предлагаемая процедура годится только для контрастирования весов связей адаптивного сумматора (см. разд. «Составные элементы»). Контрастирование весов связей производится отдельно для каждого сумматора. Адаптивный сумматор суммирует входные сигналы нейрона, умноженные на соответствующие веса связей. Для работы нейрона наименее значимым будем считать тот вес, который при решении примера даст наименьший вклад в сумму. Обозначим через x_p^q входные сигналы рассматриваемого адаптивного сумматора при решении q-го примера. Показателем значимости веса назовем следующую величину: $\chi_p^q = \left| \left(w_p - w_p^* \right) \cdot x_p^q \right|$. Усредненный по всем примерам обучающего множества показатель значимости имеет вид $\chi_p = \left| \left(w_p - w_p^* \right) \right| \cdot \max_q \left| x_p^q \right|$. Производим контрастирование по процедуре, приведенной в разд. «Контрастирование на основе показателей значимости». В самой процедуре контрастирования есть только одно отличие – вместо проверки на наличие ошибок при предъявлении всех примеров проверяется, что новые выходные сигналы сети отличаются от первоначальных не более чем на заданную величину.

Контрастирование при обучении

Существует еще один способ контрастирования нейронных сетей. Идея этого способа состоит в том, что функция оценки модернизируется таким способом, чтобы для снижения оценки было выгодно привести сеть к заданному виду. Рассмотрим решение задачи приведения параметров сети к выделенным значениям. Используя обозначения из предыдущих разделов требуемую добавку к функции оценки, являющуюся штрафом за отклонение значения параметра от ближайшего выделенного значения:, можно записать в виде $\sum_p \left(w_p - w_p^* \right)^2$.

Для решения других задач вид добавок к функции оценки много сложнее.

Определение показателей значимости

В данном разделе описан способ определения показателей значимости параметров и сигналов. . Далее будем говорить об определении значимости параметров. Показатели значимости сигналов сети определяются по тем же формулам с заменой параметров на сигналы.

Определение показателей значимости через градиент

Нейронная сеть двойственного функционирования может вычислять градиент функции оценки по входным сигналам и обучаемым параметрам сети

Показателем значимости параметра при решении q-о примера будем называть величину, которая показывает насколько изменится значение функции оценки решения сетью q-о примера если текущее значение параметра w_p заменить на выделенное значение w_p^*. Точно эту величину можно определить произведя замену и вычислив оценку сети. Однако учитывая большое число параметров сети вычисление

показателей значимости для всех параметров будет занимать много времени. Для ускорения процедуры оценки параметров значимости вместо точных значений используют различные оценки [33, 64, 90]. Рассмотрим простейшую и наиболее используемую линейную оценку показателей значимости. Разложим функцию оценки в ряд Тейлора с точностью до членов первого порядка: $H_q(w^*) = H_q^0 + \sum_p \frac{\partial H_q}{\partial w_p}(w_p - w_p^*)$,где H_q^0 – значение функции оценки решения q-о примера при $w^* = w$. Таким образом показатель значимости p-о параметра при решении q-о примера определяется по следующей формуле:

$$\chi_p^q = \left| \frac{\partial H_q}{\partial w_p}\left(w_p - w_p^*\right) \right| \tag{1}$$

Показатель значимости (1) может вычисляться для различных объектов. Наиболее часто его вычисляют для обучаемых параметров сети. Однако показатель значимости вида (1) применим и для сигналов. Как уже отмечалось в главе «Описание нейронных сетей» сеть при обратном функционировании всегда вычисляет два вектора градиента – градиент функции оценки по обучаемым параметрам сети и по всем сигналам сети. Если показатель значимости вычисляется для выявления наименее значимого нейрона, то следует вычислять показатель значимости выходного сигнала нейрона. Аналогично, в задаче определения наименее значимого входного сигнала нужно вычислять значимость этого сигнала, а не сумму значимостей весов связей, на которые этот сигнал подается.

Усреднение по обучающему множеству

Показатель значимости параметра χ_p^q зависит от точки в пространстве параметров, в которой он вычислен и от примера из обучающего множества. Существует два принципиально разных подхода для получения показателя значимости параметра, не зависящего от примера. При первом подходе считается, что в обучающей выборке заключена полная информация о всех возможных примерах. В этом случае, под показателем значимости понимают величину, которая показывает насколько изменится значение функции оценки по обучающему множеству, если текущее значение параметра w_p заменить на выделенное значение w_p^*. Эта величина вычисляется по следующей формуле:

$$\chi_p = \left| \frac{\partial H_{OM}}{\partial w_p}\left(w_p - w_p^*\right) \right|. \tag{2}$$

В рамках другого подхода обучающее множество рассматривают как случайную выборку в пространстве входных параметров. В этом случае показателем значимости по всему обучающему множеству будет служить результат некоторого усреднения по обучающей выборке.

Существует множество способов усреднения. Рассмотрим два из них. Если в результате усреднения показатель значимости должен давать среднюю значимость, то такой показатель вычисляется по следующей формуле:

$$\chi_p = \frac{1}{m}\sum_{q=1}^{m} \chi_p^q . \tag{3}$$

Если в результате усреднения показатель значимости должен давать величину, которую не превосходят показатели значимости по отдельным примерам (значимость этого параметра по отдельному примеру не больше чем χ_p), то такой показатель вычисляется по следующей формуле:

$$\chi_p = \max_q \chi_p^q . \tag{4}$$

Показатель значимости (4) хорошо зарекомендовал себя при использовании в работах группы НейроКомп.

Накопление показателей значимости

Все показатели значимости зависят от точки в пространстве параметров сети, в которой они вычислены, и могут сильно изменяться при переходе от одной точки к другой. Для показателей значимости,

вычисленных с использованием градиента эта зависимость еще сильнее, поскольку при обучении по методу наискорейшего спуска (см. раздел «Метод наискорейшего спуска») в двух соседних точках пространства параметров, в которых вычислялся градиент, градиенты ортогональны. Для снятия зависимости от точки пространства используются показатели значимости, вычисленные в нескольких точках. Далее они усредняются по формулам аналогичным (3) и (4). Вопрос о выборе точек в пространстве параметров в которых вычислять показатели значимости обычно решается просто. В ходе нескольких шагов обучения по любому из градиентных методов при каждом вычислении градиента вычисляются и показатели значимости. Число шагов обучения, в ходе которых накапливаются показатели значимости, должно быть не слишком большим, поскольку при большом числе шагов обучения первые вычисленные показатели значимости теряют смысл, особенно при использовании усреднения по формуле (4).

Стандарт первого уровня компонента контрастер

В этом разделе приводится стандарт языка описания компонента контрастер. Компонент контрастер во многом подобен компоненту учитель. Так в языке описания компонента контрастер допускается использование функций, описанных в разделе «Список стандартных функций» главы «Учитель».

Язык описания контрастера

В отличие от таких компонент как оценка, сеть и интерпретатор ответа, контрастер не является составным объектом. Однако, контрастер может состоять из множества функций, вызывающих друг друга. Собственно контрастер – это процедура, управляющая процессом контрастирования. Ключевые слова, специфические для языка описания контрастера приведены в табл. 3

Библиотеки функций контрастера

Библиотеки функций контрастера содержат описание функций, необходимых для работы одного или нескольких контрастеров. Использование библиотек позволяет избежать дублирования функций в различных контрастерах. Описание библиотеки функций аналогично описанию контрастера, но не содержит главной процедуры.

Таблица 3.

Ключевые слова специфические для языка описания контрастера

Ключевое слово	Краткое описание
1. Main	Начало главной процедуры
2. Contrast	Заголовок описания контрастера
3. ContrLib	Заголовок описания библиотеки функций
4. Used	Подключение библиотек функций
5. ContrastFunc	Глобальная переменная типа функция.

БНФ языка описания контрастера

Обозначения, принятые в данном расширении БНФ и описание ряда конструкций приведены в главе «Общий стандарт» в разделе «Описание языка описания компонент».

<Описание библиотеки> ::= <Заголовок библиотеки> <Описание глобальных переменных> <Описание функций> <Конец описания библиотеки>

<Заголовок библиотеки> ::= **ContrLib** <Имя библиотеки> [**Used** <Список имен библиотек>]

<Имя библиотеки> ::= <Идентификатор>

<Список имен библиотек> ::= <Имя используемой библиотеки> [,<Список имен библиотек>]

<Имя используемой библиотеки> ::= <Идентификатор>

<Конец описания библиотеки> ::= **End ContrLib**

<Описание контрастера> ::= <Заголовок контрастера> <Описание глобальных переменных> <Описание функций> <Главная процедура> <Конец описания контрастера>

<Заголовок контрастера> ::= **Contrast** <Имя библиотеки> [**Used** <Список имен библиотек>]

<Главная процедура> ::= **Main** <Описание статических переменных> <Описание переменных> <Тело функции>

<Конец описания контрастера> **End Contrast**

Описание языка описания контрастера

Язык описания контрастера является наиболее простым из всех языков описания компонент. Фактически все синтаксические конструкции этого языка описаны в главе «Общий стандарт». В теле функции, являющемся частью главной процедуры недопустим оператор возврата значения, поскольку главная процедура не является функцией.

Контрастер имеет одну глобальную предопределенную переменную ContrastFunc. Эта переменная должна обязательно быть определена – ей нужно присвоить адрес функции, которая будет вызывать-

ся каждый раз после того, как нейронная сеть вычислит градиент после решения одного примера. Функция, адрес которой присваивается переменной ContrastFunc должна быть объявлена следующим образом:

Function MyContrast(TheEnd : Logic) : Logic;

Значения аргумента TheEnd имеют следующий смысл: истина – обучение ведется позадачно или закончен просмотр обучающего множества; ложь – обработан еще один пример обучающего множества при обучении по всему задачнику в целом. Следует учесть, что при обучении по всему обучающему множеству в целом, нейронная сеть накапливает градиенты всех примеров, так что при первом вызове функции в сети хранится градиент функции оценки по результатам решения первого примера; при втором – результатам решения первых двух примеров и т.д. Функция возвращает значение ложь, если в ходе ее работы произошла ошибка. В противном случае она возвращает значение истина.

Значение переменной ContrastFunc присваивается оператором присваивания:

ContrastFunc = MyContrast

Если значение переменной ContrastFunc не задано, то она указывает на используемую по умолчанию функцию EmptyContrast, которая просто возвращает значение истина.

Стандарт второго уровня компонента контрастер

Компонента контрастер одновременно работает только с одним контрастером. Запросы к компоненте контрастер можно разбить на следующие группы.

1. Контрастирование сети.
2. Чтение/запись контрастера.
3. Инициация редактора контрастера.
4. Работа с параметрами контрастера.

Контрастирование сети

К данной группе относятся три запроса – контрастировать сеть (ContrastNet), прервать контрастирование (CloseContrast) и контрастировать пример (ContrastExample).

Контрастировать сеть(ContrastNet)

Описание запроса:
Pascal:

Function ContrastNet: Logic;

C:

Logic ContrastNet()

Аргументов нет.

Назначение – производит контрастирование сети.
Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если в момент получения запроса контрастер не загружен, то возникает ошибка 701 – неверное имя компонента, управление передается обработчику ошибок, а обработка запроса прекращается.
3. Выполняется главная процедура загруженного контрастера.
4. Если во время выполнения запроса возникает ошибка, а значение переменной Error равно нулю, то генерируется внутренняя ошибка 705 – ошибка исполнения контрастера, управление передается обработчику ошибок, а обработка запроса прекращается.
5. Если во время выполнения запроса возникает ошибка, а значение переменной Error не равно нулю, то обработка запроса прекращается.

Прервать контрастирование (CloseContrast)

Описание запроса:
Pascal:

Function CloseContrast: Logic;

C:

Logic CloseContrast()

Аргументов нет.

Назначение – прерывает контрастирование сети.
Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если в момент получения запроса контрастер не загружен, то возникает ошибка 701 – неверное имя компонента, управление передается обработчику ошибок, а обработка запроса прекращается.
3. Если в момент получения запроса не выполняется запрос ContrastNet, то возникает ошибка 706 – неверное использование запроса на прерывание контрастирования, управление передается обработчику ошибок, а обработка запроса прекращается.
4. Завершается выполнение текущего шага контрастирования сети.
5. Если во время выполнения запроса возникает ошибка, а значение переменной Error равно нулю, то генерируется внутренняя ошибка 705 – ошибка исполнения контрастера, управление передается обработчику ошибок, а обработка запроса прекращается.
6. Если во время выполнения запроса возникает ошибка, а значение переменной Error не равно нулю, то обработка запроса прекращается.

Контрастировать пример (ContrastExample)

Описание запроса:
Pascal:

```
Function ContrastExample( TheEnd : Logic) : Logic;
```

C:

```
Logic ContrastExample(Logic TheEnd )
```

Описание аргумента:
TheEnd – значение аргумента имеет следующий смысл: ложь – обработан еще один пример обучающего множества при обучении по всему задачнику в целом.

Назначение – извлекает из сети необходимые для вычисления показателей значимости параметры.

Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Вызывает функцию, адрес которой хранится в переменной ContrastFunc, передавая ей аргумент TheEnd в качестве аргумента.
3. Если функция ContrastFunc возвращает значение ложь, а значение переменной Error равно нулю, то генерируется внутренняя ошибка 705 – ошибка исполнения контрастера, управление передается обработчику ошибок, а обработка запроса прекращается.
4. Если функция ContrastFunc возвращает значение ложь, а значение переменной Error не равно нулю, то обработка запроса прекращается.
5. Запрос в качестве результата возвращает возвращенное функцией ContrastFunc значение.

Чтение/запись контрастера

В данном разделе описаны запросы позволяющие, загрузить контрастер с диска или из памяти, выгрузить контрастера и сохранить текущего контрастера на диске или в памяти.

Прочитать контрастера (cnAdd)

Описание запроса:
Pascal:

```
Function cnAdd( CompName : PString ) : Logic;
```

C:

```
Logic cnAdd(PString CompName)
```

Описание аргумента:
CompName – указатель на строку символов, содержащую имя файла компонента или адрес описания компонента.

Назначение – читает контрастера с диска или из памяти.

Описание исполнения.

1. Если в качестве аргумента CompName дана строка, первые четыре символа которой составляют слово File, то остальная часть строки содержит имя компонента и после пробела имя файла, содержащего компонент. В противном случае считается, что аргумент CompName содержит указатель на область памяти, содержащую описание компонента в формате для записи на диск. Если описание не вмещается в одну область памяти, то допускается включение в текст описания ком-

понента ключевого слова Continue, за которым следует четыре байта, содержащие адрес следующей области памяти.

2. Если в данный момент загружен другой контрастер, то выполняется запрос cnDelete. Контрастер считывается из файла или из памяти.
3. Если считывание завершается по ошибке, то возникает ошибка 702 – ошибка считывания контрастера, управление передается обработчику ошибок, а обработка запроса прекращается.

Удаление контрастера (cnDelete)

Описание запроса:
Pascal:

```
Function cnDelete : Logic;
```

C:

```
Logic cnDelete()
```

Аргументов нет.

Назначение – удаляет загруженного в память контрастера.
Описание исполнения.

1. Если список в момент получения запроса контрастер не загружен, то возникает ошибка 701 – неверное имя контрастера, управление передается обработчику ошибок, а обработка запроса прекращается.

Запись контрастера (cnWrite)

Описание запроса:
Pascal:

```
Function cnWrite(Var FileName : PString) : Logic;
```

C:

```
Logic cnWrite(PString* FileName)
```

Описание аргументов:
CompName – указатель на строку символов, содержащую имя контрастера.
FileName – имя файла или адрес памяти, куда надо записать контрастера.

Назначение – сохраняет контрастера в файле или в памяти.
Описание исполнения.

1. Если в момент получения запроса контрастер не загружен, то возникает ошибка 701 – неверное имя контрастера, управление передается обработчику ошибок, а обработка запроса прекращается.
2. Если в качестве аргумента FileName дана строка, первые четыре символа которой составляют слово File, то остальная часть строки содержит имя файла, для записи компонента. В противном случае FileName должен содержать пустой указатель. В этом случае запрос вернет в нем указатель на область памяти, куда будет помещено описание компонента в формате для записи на диск. Если описание не вмещается в одну область памяти, то в текст будет включено ключевое слово Continue, за которым следует четыре байта, содержащие адрес следующей области памяти.
3. Если во время сохранения компонента возникнет ошибки, то возникает ошибка 703 – ошибка сохранения контрастера, управление передается обработчику ошибок, а обработка запроса прекращается.

Инициация редактора контрастера

К этой группе запросов относится запрос, который инициирует работу не рассматриваемого в данной работе компонента – редактора контрастера.

Редактировать контрастера (cnEdit)

Описание запроса:
Pascal:

```
Procedure cnEdit(CompName : PString);
```

C:

```
void cnEdit(PString CompName)
```

Описание аргумента:

CompName – указатель на строку символов – имя файла или адрес памяти, содержащие описание контрастера.

Если в качестве аргумента CompName дана строка, первые четыре символа которой составляют слово File, то остальная часть строки содержит имя контрастера и после пробела имя файла, содержащего описание контрастера. В противном случае считается, что аргумент CompName содержит указатель на область памяти, содержащую описание контрастера в формате для записи на диск. Если описание не вмещается в одну область памяти, то допускается включение в текст описания ключевого слова Continue, за которым следует четыре байта, содержащие адрес следующей области памяти.

Если в качестве аргумента CompName передан пустой указатель или указатель на пустую строку, то редактор создает нового контрастера.

Работа с параметрами контрастера

В данном разделе описаны запросы, позволяющие изменять параметры контрастера.

Получить параметры (cnGetData)

Описание запроса:
Pascal:

```
Function cnGetData(Var Param : PRealArray ) : Logic;
```

C:

```
Logic cnGetData(PRealArray* Param)
```

Описание аргумента:
Param – адрес массива параметров.

Назначение – возвращает вектор параметров контрастера.
Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если в момент получения запроса контрастер не загружен, то возникает ошибка 701 – неверное имя компонента, управление передается обработчику ошибок, а обработка запроса прекращается.
3. В массив, адрес которого передан в аргументе Param, заносятся значения параметров. Параметры заносятся в массив в порядке описания в разделе описания статических переменных.

Получить имена параметров (cnGetName)

Описание запроса:
Pascal:

```
Function cnGetName(Var Param : PRealArray ) : Logic;
```

C:

```
Logic cnGetName(PRealArray* Param)
```

Описание аргумента:
Param – адрес массива указателей на названия параметров.

Назначение – возвращает вектор указателей на названия параметров контрастера.
Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если в момент получения запроса контрастер не загружен, то возникает ошибка 701 – неверное имя компонента, управление передается обработчику ошибок, а обработка запроса прекращается.
3. В массив, адрес которого передан в аргументе Param, заносятся адреса символьных строк, содержащих названия параметров.

Установить параметры (cnSetData)

Описание запроса:
Pascal:

```
Function cnSetData(Param : PRealArray ) : Logic;
```

C:

```
Logic cnSetData(PRealArray Param)
```

Описание аргументов:
Param – адрес массива параметров.

Назначение – заменяет значения параметров контрастера на значения, переданные, в аргументе Param.

Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если в момент получения запроса контрастер не загружен, то возникает ошибка 701 – неверное имя компонента, управление передается обработчику ошибок, а обработка запроса прекращается.
3. Параметры, значения которых хранятся в массиве, адрес которого передан в аргументе Param, передаются контрастеру.

Обработка ошибок

В табл. 4 приведен полный список ошибок, которые могут возникать при выполнении запросов компонентом контрастер, и действия стандартного обработчика ошибок.

Таблица 4.

Ошибки компонента контрастер и действия стандартного обработчика ошибок.

№Название ошибкиСтандартная обработка		
701	Несовместимость сети и контрастераЗанесение	номера в Error
702	Ошибка считывания контрастераЗанесение	номера в Error
703	Ошибка сохранения контрастераЗанесение	номера в Error
704	Некорректная работа с памятьюЗанесение	номера в Error
705	Ошибка исполнения контрастераЗанесение	номера в Error
706	Неверное использование запроса на прерывание контрастированияЗанесение	номера в Error