1. Функциональные компоненты

Эта глава посвящена выделению функциональных компонентов, составляющих универсальный нейрокомпьютер. Основные компоненты нейрокомпьютера выделяются по следующим признакам:

1. Относительная функциональная обособленность: каждый компонент имеет четкий набор функций. Его взаимодействие с другими компонентами может быть описано в виде небольшого числа запросов.
2. Возможность реализации большинства используемых алгоритмов.
3. Возможность взаимозамены различных реализаций любого компонента без изменения других компонентов.

Однако, прежде чем приступать к выделению компонент, опишем рассматриваемый набор нейронных сетей и процесс их обучения.

1.1 Краткий обзор нейронных сетей

Можно по разному описывать «зоопарк» нейронных сетей. Приведем классификацию нейронных сетей по решаемым ими задачам.

1. Классификация без учителя или поиск закономерностей в данных. Наиболее известным представителем этого класса сетей является сеть Кохонена, реализующая простейший вариант решения этой задачи. Наиболее общий вариант решения этой задачи известен как метод динамических ядер [223, 261].
2. Ассоциативная память. Наиболее известный представитель – сети Хопфилда. Эта задача также позволяет строить обобщения. Наиболее общий вариант описан в [77 – 79].
3. Аппроксимация функций, заданных в конечном числе точек. К сетям, решающим эту задачу, относятся персептроны, сети обратного распространения.

В центре нашего внимания будут сети, предназначенные для решения третьей задачи, однако предлагаемый вариант стандарта позволяет описать любую сеть. Конечно, невозможно использовать учитель, предназначенный для построения ассоциативной памяти, для решения задачи классификации без учителя и наоборот.

Среди сетей, аппроксимирующих функции, необходимо выделить еще два типа сетей – с дифференцируемой и пороговой характеристической функцией. Дифференцируемой будем называть сеть, каждый элемент которой реализует дифференцируемую функцию (точнее, непрерывно дифференцируемую). Вообще говоря, альтернативой дифференцируемой сети является недифференцируемая, а не пороговая, но на практике, как правило, все недифференцируемые сети являются пороговыми. Отметим, что для того, чтобы сеть была пороговой, достаточно вставить в нее один пороговый элемент.

Основное различие между дифференцируемыми и пороговыми сетями состоит в способе обучения. Для дифференцируемых сетей есть конструктивная процедура обучения, гарантирующая результат, если он достижим – метод двойственного обучения (обратного распространения ошибки). Для обучения пороговых сетей используют правило Хебба или его модификации. Однако, для многослойных сетей с пороговыми элементами правило Хебба не гарантирует обучения. (В случае однослойных сетей – персептронов, доказана теорема о достижении результата в случае его принципиальной достижимости). С другой стороны, в работе [145] доказано, что многослойные сети с пороговыми нейронами можно заменить эквивалентными однослойными.

1.2 Выделение компонент

Первым основным компонентом нейрокомпьютера является *нейронная сеть*. Относительно архитектуры сети принцип двойственности предполагает только одно – все элементы сети реализуют при прямом функционировании характеристические функции из класса $C^1(E)$ (непрерывно дифференцируемые на области определения E, которой, как правило, является вся числовая ось).

Для обучения нейронной сети необходимо наличие *задачника*. Однако чаще всего, обучение производится не по всему задачнику, а по некоторой его части. Ту часть задачника, по которой в данный момент производится обучение, будем называть обучающей выборкой. Для многих задач обучающая выборка имеет большие размеры (от нескольких сот до нескольких десятков тысяч примеров). При обучении с использованием скоростных методов обучения (их скорость на три-четыре порядка превышает скорость обучения по классическому методу обратного распространения ошибки) приходится быстро сменять примеры. Таким образом, скорость обработки обучающей выборки может существенно влиять на скорость обучения нейрокомпьютера. К сожалению, большинство разработчиков аппаратных средств

не предусматривает средств для быстрой смены примеров. Таким образом, задачник выделен в отдельный компонент нейрокомпьютера.

При работе с обучающей выборкой удобно использовать привычный для пользователя формат данных. Однако, этот формат чаще всего непригоден для использования нейросетью. Таким образом, между обучающей выборкой и нейросетью возникает дополнительный компонент нейрокомпьютера – *предобработчик*. Из литературных источников следует, что разработка эффективных предобработчиков для нейрокомпьютеров является новой, почти совсем не исследованной областью. Большинство разработчиков программного обеспечения для нейрокомпьютеров склонно возлагать функции предобработки входных данных на обучающую выборку или вообще перекладывают ее на пользователя. Это решение технологически неверно. Дело в том, что при постановке задачи для нейрокомпьютера трудно сразу угадать правильный способ предобработки. Для его подбора проводится серия экспериментов. В каждом из экспериментов используется одна и та же обучающая выборка и разные способы предобработки входных данных сети. Таким образом, выделен третий важный компонент нейрокомпьютера – предобработчик входных данных.

Заметим, что если привычный для человека способ представления входных данных непригоден для нейронной сети, то и формат ответов нейронной сети часто малопригоден для человека. Необходимо интерпретировать ответы нейронной сети. Интерпретация зависит от вида ответа. Так, если ответом нейронной сети является действительное число, то его, как правило, приходится масштабировать и сдвигать для попадания в нужный диапазон ответов. Если сеть используется как классификатор, то выбор интерпретаторов еще шире. Большое разнообразие интерпретаторов при невозможности решить раз и навсегда вопрос о преимуществах одного из них над другими приводит к необходимости выделения *интерпретатора ответа* нейронной сети в отдельный компонент нейрокомпьютера.

С интерпретатором ответа тесно связан еще один обязательный компонент нейрокомпьютера – *оценка*. Невнимание к этому компоненту вызвано практикой рассматривать метод обратного распространения ошибки в виде алгоритма. Доминирование такой точки зрения привело к тому, что, судя по публикациям, большинство исследователей даже не подозревает о том, что «уклонение от правильного ответа», подаваемое на вход сети при обратном функционировании, есть ни что иное, как производная функции оценки по выходному сигналу сети (если функция оценки является суммой квадратов уклонений). Возможно (и иногда очень полезно) конструировать другие оценки (см. главу «Оценка и интерпретатор ответа»). Нашей группой в ходе численных экспериментов было выяснено, что для обучения сетей-классификаторов функция оценки вида суммы квадратов, пожалуй, наиболее плоха. Использование альтернативных функций оценки позволяет в несколько раз ускорить обучение нейронной сети.

Шестым необходимым компонентом нейрокомпьютера является *учитель*. Этот компонент может меть множество реализаций. Обзор наиболее часто употребляемых и наиболее эффективных учителей приводится в главе «Учитель».

Принцип относительной функциональной обособленности требует выделения еще одного компонента, названного *исполнителем запросов учителя* или просто *исполнителем*. Назначение этого компонента не так очевидно, как всех предыдущих. Заметим, что для всех учителей, обучающих сети по методу обратного распространения ошибки, и при тестировании сети характерен следующий набор операций с каждым примером обучающей выборки:

1. Тестирование решения примера
   - 1.1. Взять пример у задачника.
   - 1.2. Предъявить его сети для решения.
   - 1.3. Предъявить результат интерпретатору ответа.
2. Оценивание решения примера
   - 2.1. Взять пример у задачника.
   - 2.2. Предъявить его сети для решения.
   - 2.3. Предъявить результат оценке.
3. Оценивание решения примера с вычислением градиента.
   - 3.1. Взять пример у задачника.
   - 3.2. Предъявить его сети для решения.
   - 3.3. Предъявить результат оценке с вычислением производных.
   - 3.4. Предъявить результат работы оценки сети для вычисления градиента.
4. Оценивание и тестирование решения примера.
   - 4.1. Взять пример у задачника.
   - 4.2. Предъявить его сети для решения.
   - 4.3. Предъявить результат оценке.
   - 4.4. Предъявить результат интерпретатору ответа.

Заметим, что все четыре варианта работы с сетью, задачником, интерпретатором ответа и оценкой легко объединить в один запрос, параметры которого позволяют указать последовательность действий. Таким образом, исполнитель исполняет всего один запрос – обработать пример. Однако выделение этого компонента позволяет исключить необходимость в прямых связях таких компонентов, как контрастер и учитель, с компонентами оценка и интерпретатор ответа, а их взаимодействие с компонентом сеть свести исключительно к запросам связанным с модификацией обучаемых параметров сети.

Последним компонентом, которого необходимо выделить, является *контрастер* нейронной сети. Этот компонент является надстройкой над учителем. Его назначение – сводить число связей сети до минимально необходимого или до «разумного» минимума (степень разумности минимума определяется пользователем). Кроме того, контрастер, как правило, позволяет свести множество величин весов связей к 2-4, реже к 8 выделенным пользователем значениям. Наиболее важным следствием применения процедуры контрастирования является получение логически прозрачных сетей – сетей, работу которых легко описать и понять на языке логики [75, 82].

Для координации работы всех компонент нейрокомпьютера вводится макрокомпонента *Нейрокомпьютер*. Основная задача этого компонента – организация интерфейса с пользователем и координация действий всех остальных компонентов.

1.3 Запросы компонентов нейрокомпьютера

В этом разделе приводится основной список запросов, которые обеспечивают функционирование нейрокомпьютера. За редким исключением приводятся только запросы, которые генерируются компонентами нейрокомпьютера (некоторые из этих запросов могут поступать в нейрокомпьютер от пользователя). Здесь рассматривается только форма запроса и его смысл. Полный список запросов каждого компонента, детали их исполнения и форматы данных рассматриваются в соответствующих главах, в разделах «Стандарт второго уровня компонента ... «.

На рис. 1. приведена схема запросов в нейрокомпьютере. При построении схемы предполагается, что на каждый запрос приходит ответ. Вид ответа описан при описании запросов. Стрелки, изображающие запросы, идут от объекта, инициирующего запрос, к объекту его исполняющему.

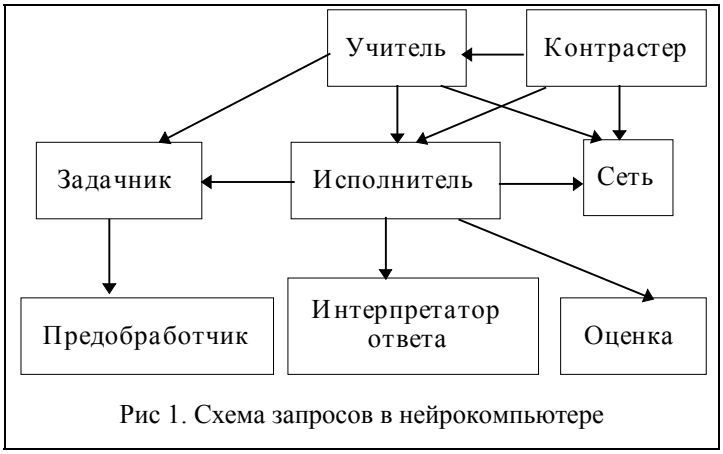

Рис 1. Схема запросов в нейрокомпьютере

1.3.1 Запросы к задачнику

Запросы к задачнику позволяют последовательно перебирать все примеры обучающей выборки, обращаться непосредственно к любому примеру задачника и изменять обучающую выборку. Обучающая выборка выделяется путем «раскрашивания» примеров задачника в различные «цвета». Понятие цвета и способ работы с цветами описаны в разделе «Переменные типа цвет и операции с цветами».

Запросы последовательного перебора обучающей выборки:

«Инициировать выдачу примеров цвета К». По этому запросу происходит инициация выдачи примеров К-го цвета.

«Дать очередной пример». По этому запросу задачник возвращает предобработанные данные очередного примера и, при необходимости, правильные ответы, уровень достоверности и другие данные этого примера.

«Следующий пример». По этому запросу задачник переходит к следующему примеру обучающей выборки. Если такого примера нет, то возвращается признак отсутствия очередного примера.

Для непосредственного доступа к примерам задачника служит запрос «Дать пример номер N». Действия задачника в этом случае аналогичны выполнению запроса «Дать очередной пример».

Для изменения обучающей выборки служит запрос «Окрасить примеры в цвет К». Этот запрос используется редко, поскольку изменение обучающей выборки, как правило, осуществляется пользователем при редактировании задачника.

1.3.2 Запрос к предобработчику

Предобработчик сам никаких запросов не генерирует. Единственный запрос к предобработчику – «Предобработать пример» может быть выдан только задачником.

1.3.3 Запрос к исполнителю

«Обработать очередной пример». Вид ответа зависит от параметров запроса.

1.3.4 Запросы к учителю

«Начать обучение сети». По этому запросу учитель начинает процесс обучения сети.

«Прервать обучение сети». Этот запрос приводит к прекращению процесса обучения сети. Этот запрос требуется в случае необходимости остановить обучение сети до того, как будет удовлетворен критерий остановки обучения, предусмотренный в учителе.

«Провести N шагов обучения» – как правило, выдается контрастером, необходим для накопления показателей чувствительности.

1.3.5 Запрос к контрастеру

«Отконтрастировать сеть». Ответом является код завершения операции контрастирования.

1.3.6 Запрос к оценке

Оценка не генерирует никаких запросов. Она выполняет только один запрос – «Оценить пример». Результатом выполнения запроса является оценка примера и, при необходимости, вектор производных оценки по выходным сигналам сети.

1.3.7 Запрос к интерпретатору ответа

Интерпретатор ответа не генерирует никаких запросов. Он выполняет только один запрос – «Интерпретировать ответ». Ответом является результат интерпретации.

1.3.8 Запросы к сети

Сеть не генерирует никаких запросов. Набор исполняемых сетью запросов можно разбить на три группы.

Запрос, обеспечивающий тестирование.

«Провести прямое функционирование». На вход сети подаются данные примера. На выходе сети вычисляется ответ сети, подлежащий оцениванию или интерпретации.

Запросы, обеспечивающие обучение сети.

«Обнулить градиент». При исполнении этого запроса градиент оценки по обучаемым параметрам сети кладется равным нулю. Этот запрос необходим, поскольку при вычислении градиента по очередному примеру сеть *добавляет* его к ранее вычисленному градиенту по сумме других примеров.

«Вычислить градиент по примеру». Проводится обратное функционирование сети. Вычисленный градиент *добавляется* к ранее вычисленному градиенту по сумме других примеров.

«Изменить карту с шагами H1 и H2». Генерируется учителем во время обучения.

Запрос, обеспечивающие контрастирование.

«Изменить карту по образцу». Генерируется контрастером при контрастировании сети.

Таким образом, выделено семь основных компонент нейрокомпьютера, определены их функции и основные исполняемые ими запросы.