4. Предобработчик

Данная глава посвящена компоненту предобработчик. В ней рассматриваются различные аспекты предобработки входных данных для нейронных сетей. Существует множество различных видов нейронных сетей (см. главу «Описание нейронных сетей»). Однако, для большинства нейронных сетей характерно наличие такого интервала входных сигналов, в пределах которого сигналы различимы. Для различных нейронных сетей эти интервалы различны. Большинство работающих с нейронными сетями прекрасно осведомлены об этом их свойстве, но до сих пор не предпринималось никаких попыток как-либо формализовать или унифицировать подходы к предобработке входных сигналов. В данной главе дан один из возможных формализмов этой задачи. За рамками рассмотрения осталась предобработка графической информации. Наиболее мощные и интересные способы обработки графической информации описаны в [90]

4.1 Описание предобработчика

В этой части главы будут описаны различные виды входных сигналов и способы их предобработки. В качестве примера будут рассмотрены сети с сигмоидными нелинейными преобразователями. Однако, описываемые способы предобработки применимы для сетей с произвольными нелинейными преобразователями. Единственным исключением является раздел «Оценка способности сети решить задачу», который применим только для сетей с нелинейными преобразователями, непрерывно зависящими от своих аргументов.

4.1.1 Нейрон

Нейроны, используемые в большинстве нейронных сетей, имеют структуру, приведенную на рис. 1. На рис. 1 использованы следующие обозначения:

x – вектор входных сигналов нейрона;
α – вектор синаптических весов нейрона;
Σ – входной сумматор нейрона;
$p=(\alpha,x)$ – выходной сигнал входного сумматора;
σ – функциональный преобразователь;
y – выходной сигнал нейрона.

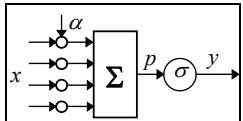

Рис. 1. Пример нейрона

Обычно нейронные сети называют по виду функции $\sigma(p)$. Хорошо известны и наиболее часто используются два вида сигмоидных сетей:

$$S_1: \quad \sigma(p)=1/(1+\exp(-cp)),$$
$$S_2: \quad \sigma(p)=p/(c+|p|),$$

где c - параметр, называемый «характеристикой нейрона». Обе функции имеют похожие графики.

Каждому типу нейрона соответствует свой интервал приемлемых входных данных. Как правило, этот диапазон либо совпадает с диапазоном выдаваемых выходных сигналов (например для сигмоидных нейронов с функцией S_1), либо является объединением диапазона выдаваемых выходных сигналов и отрезка, симметричного ему относительно нуля (например, для сигмоидных нейронов с функцией S_2), Этот диапазон будем обозначать как $[a,b]$

4.1.2 Различимость входных данных

Очевидно, что входные данные должны быть различимы. В данном разделе будут приведены соображения, исходя из которых, следует выбирать диапазон входных данных. Пусть одним из входных параметров нейронной сети является температура в градусах Кельвина. Если речь идет о температурах

Таблица 1

Величина входного сигнала	Нейрон типа S_1				Нейрон типа S_2			
	$c=0.1$	$c=0.5$	$c=1$	$c=2$	$c=0.1$	$c=0.5$	$c=1$	$c=2$
250	1.0	1.0	1.0	1.0	0.99960	0.99800	0.99602	0.99206
275	1.0	1.0	1.0	1.0	0.99964	0.99819	0.99638	0.99278
300	1.0	1.0	1.0	1.0	0.99967	0.99834	0.99668	0.99338

близких к нормальной, то входные сигналы изменяются от 250 до 300 градусов. Пусть сигнал подается прямо на нейрон (синаптический вес равен единице). Выходные сигналы нейронов с различными параметрами приведены в табл. 1.

Совершенно очевидно, что нейронная сеть просто неспособна научиться надежно различать эти сигналы (если вообще способна научиться их различать!). Если использовать нейроны с входными синапсами, не равными единице, то нейронная сеть сможет отмасштабировать входные сигналы так, чтобы они стали различимы, но при этом будет задействована только часть диапазона приемлемых входных данных - все входные сигналы будут иметь один знак. Кроме того, все подаваемые сигналы будут занимать лишь малую часть этого диапазона. Например, если мы отмасштабируем температуры так, чтобы 300 соответствовала величина суммарного входного сигнала равная 1 (величина входного синапса равна 1/300), то реально подаваемые сигналы займут лишь одну шестую часть интервала [0,1] и одну двенадцатую интервала [-1,1]. Получаемые при этом при этом величины выходных сигналов нейронов приведены в табл. 2.

Таблица 2

Величина входного сигнала	Нейрон типа S_1				Нейрон типа S_2			
	$c = 0.1$	$c = 0.5$	$c = 1$	$c = 2$	$c = 0.1$	$c = 0.5$	$c = 1$	$c = 2$
250 (0.83)	0.52074	0.60229	0.69636	0.84024	0.89286	0.62500	0.45455	0.29412
275 (0.91)	0.52273	0.61183	0.71300	0.86057	0.90164	0.64706	0.47826	0.31429
300 (1.0)	0.52498	0.62246	0.73106	0.88080	0.90909	0.66667	0.50000	0.33333

Сигналы, приведенные в табл. 2 различаются намного сильнее соответствующих сигналов из табл. 1. Таким образом, необходимо заранее позаботиться о масштабировании и сдвиге сигналов, чтобы максимально полно использовать диапазон приемлемых входных сигналов. Опыт использования нейронных сетей с входными синапсами свидетельствует о том, что в подавляющем большинстве случаев предварительное масштабирование и сдвиг входных сигналов сильно облегчает обучение нейронных сетей. Если заранее произвести операции масштабирования и сдвига входных сигналов, то величины выходных сигналов нейронов даже при отсутствии входных синапсов будут различаться еще сильнее (см. табл. 3).

Таблица 3

Величина входного сигнала	Нейрон типа S_1				Нейрон типа S_2			
	$c = 0.1$	$c = 0.5$	$c = 1$	$c = 2$	$c = 0.1$	$c = 0.5$	$c = 1$	$c = 2$
250 (-1)	0.47502	0.37754	0.26894	0.11920	-0.9091	-0.6667	-0.5000	-0.3333
275 (0)	0.50000	0.50000	0.50000	0.50000	0.0000	0.0000	0.0000	0.0000
300 (1)	0.52498	0.62246	0.73106	0.88080	0.9091	0.6667	0.5000	0.3333

Величину диапазона различимых входных сигналов можно определять различными способами. На практике в качестве диапазона различимых входных сигналов обычно используется диапазон приемлемых входных данных, исходя из того соображения, что если данные из этого интервала хороши для промежуточных нейронов, то они хороши и для входных.

Другой способ определения различимости входных сигналов приведен в разделе «Оценка способности сети решить задачу».

4.1.3 Классификация компонентов входных данных

Информация поступает к нейронной сети в виде набора ответов на некоторый список вопросов. Можно выделить три основных типа ответов (вопросов).

1. Бинарный признак (возможен только один из ответов – истина или ложь).
2. Качественный признак (принимает конечное число значений).
3. Число.

Ответ типа качественный признак - это ответ с конечным числом состояний. Причем нельзя ввести осмысленное расстояние между состояниями. Примером качественного признака может служить состояние больного - тяжелый, средний, легкий. Действительно, нельзя сказать, что расстояние от легкого больного до среднего больше, меньше или равно расстоянию от среднего больного до тяжелого.

Все качественные признаки можно в свою очередь разбить на три класса.

1. Упорядоченные признаки.
2. Неупорядоченные признаки.
3. Частично упорядоченные признаки.

Упорядоченным признаком называется такой признак, для любых двух состояний которого можно сказать, что одно из них предшествует другому. Тот факт, что состояние x предшествует состоянию y, будем обозначать следующим образом – $x < y$. Примером упорядоченного признака может служить состояние больного. Действительно, все состояния можно упорядочить по тяжести заболевания:

легкий больной < средний больной < тяжелый больной

Признак называют неупорядоченным, если никакие два состояния нельзя связать естественным в контексте задачи отношением порядка. Примером неупорядоченного признака может служить ответ на вопрос "Ваш любимый цвет?".

Признак называется частично упорядоченным, если для каждого состояния существует другое состояние, с которым оно связано отношением порядка. Примером частично упорядоченного признака является ответ на вопрос "Какой цвет Вы видите на экране монитора?", преследующий цель определение восприимчивости к интенсивностям основных цветов. Действительно, все множество из шестнадцати состояний разбивается на несколько цепочек:

Черный < Синий < Голубой < Белый;

Черный < Красный < Ярко красный < Белый;

Черный < Зеленый < Ярко зеленый < Белый;

Черный < Фиолетовый < Ярко фиолетовый < Белый

и т.д. Однако, между состояниями Синий и Красный отношения порядка нет.

Известно, что любой частично упорядоченный признак можно представить в виде комбинации нескольких упорядоченных и неупорядоченных признаков. Так, рассмотренный выше частично упорядоченный признак распадается на три упорядоченных признака: интенсивность синего, красного и зеленого цветов. Каждый из этих признаков является упорядоченным (цепочки порядка для этих признаков приведены в первых трех строчках рассмотрения примера). Каждое состояние исходного качественного признака описывается тройкой состояний полученных качественных признаков. Так, например, состояние Фиолетовый описывается в виде (Синий, Красный, Черный).

Исходя из вышесказанного, далее будет рассмотрено только кодирование упорядоченных и неупорядоченных признаков.

4.1.4 Кодирование бинарных признаков

Бинарные признаки характеризуются наличием только двух состояний – истина и ложь. Однако даже такие простые данные могут иметь два разных смысла. Значение истина означает наличие у описываемого объекта какого-либо свойства. А ответ ложь может означать либо (1) отсутствие этого свойства, либо (2) наличие другого свойства. В зависимости от смысловой нагрузки значения ложь, и учитывая заданный диапазон $[a,b]$, рекомендуемые способы кодирования бинарного признака приведены в табл. 4.

Таблица 4

Кодирование бинарного признака

Смысл значения ложьВеличина	входного сигнала для значения признака	
	ИстинаЛожь	
Отсутствие заданного свойства при $b = 0$	a	0
Отсутствие заданного свойства при $b \neq 0$	b	0
Наличие другого свойства	b	a

4.1.5 Кодирование неупорядоченных качественных признаков

Поскольку никакие два состояния неупорядоченного признака не связаны отношением порядка, то было бы неразумным кодировать их разными величинами одного входного сигнала нейронной сети. Поэтому, для кодирования качественных признаков рекомендуется использовать столько входных сигналов, сколько состояний у этого качественного признака. Каждый входной сигнал соответствует определенному состоянию. Так если набор всех состояний рассматриваемого признака обозначить через $\alpha_1, \alpha_2, \ldots, \alpha_n$, то рекомендуемая таблица кодировки имеет вид, приведенный в табл. 5.

Таблица 5.

Кодирование неупорядоченного качественного признака

СостояниеВектор	входных сигналов
α_1	$(b,a,a,\ldots,a)$
α_2	$(a,b,a,\ldots,a)$
α_n	$(a,a,\ldots,a,b)$

4.1.6 Кодирование упорядоченных частных признаков

Упорядоченные частные признаки, в отличие от неупорядоченных, имеют отношение порядка между состояниями. Однако кодирование их разными значениями одного входного сигнала неразумно из-за того, что расстояние между состояниями не определено, а такое кодирование эти расстояния задает явным образом. Поэтому, упорядоченные частные признаки рекомендуется кодировать в виде стольких входных сигналов, сколько состояний у признака. Но, в отличие от неупорядоченных признаков, накапливать число сигналов с максимальным значением. Для случая, когда все состояния обозначены через $\alpha_1 < \alpha_2 < \ldots < \alpha_n$, рекомендуемая таблица кодировки приведена в табл. 6.

Таблица 6.
Кодирование упорядоченного качественного признака

Состояние	Вектор входных сигналов
α_1	$(b,a,a,\ldots,a)$
α_2	$(b,b,a,\ldots,a)$
α_n	$(b,b,\ldots,b,b)$

4.1.7 Числовые признаки

При предобработке численных сигналов необходимо учитывать содержательное значение признака, расположение значений признака в интервале значений, точность измерения значений признака. Продемонстрируем это на примерах.

Содержательное значение признака. Если входными данными сети является угол между двумя направлениями, например, направление ветра, то ни в коем случае не следует подавать на вход сети значение угла (не важно в градусах или радианах). Такая подача приведет к необходимости "уяснения" сетью того факта, что 0 градусов и 360 градусов одно и тоже. Разумнее выглядит подача в качестве входных данных синуса и косинуса этого угла. Число входных сигналов сети увеличивается, но зато близкие значения признака кодируются близкими входными сигналами.

Точность измерения признака. Так в метеорологии используется всего восемь направлений ветра. Значит, при подаче входного сигнала сети необходимо подавать не угол, а всего лишь информацию о том, в какой из восьми секторов этот угол попадает. Но тогда имеет смысл рассматривать направление ветра не как числовой параметр, а как неупорядоченный качественный признак с восемью состояниями.

Расположение значений признака в интервале значений. Следует рассмотреть вопрос о равнозначности изменения значения признака на некоторую величину в разных частях интервала значений признака. Как правило, это связано с косвенными измерениями (вместо одной величины измеряется другая). Например, сила притяжения двух небесных тел при условии постоянства массы однозначно характеризуется расстоянием между ними. Пусть рассматриваются расстояния от 1 до 100 метров. Легко понять, что при изменении расстояния с 1 до 2 метров, сила притяжения изменится в четыре раза, а при изменении с 99 до 100 метров – в 1.02 раза. Следовательно, вместо подачи расстояния следует подавать обратный квадрат расстояния $c' = 1/c^2$.

4.1.8 Простейшая предобработка числовых признаков

Как уже отмечалось в разделе «Различимость входных данных» числовые сигналы рекомендуется масштабировать и сдвигать так, чтобы весь диапазон значений попадал в диапазон приемлемых входных сигналов. Эта предобработка проста и задается следующей формулой:

$$c' = \frac{(c - c_{\min})(b - a)}{(c_{\max} - c_{\min})} + a, \qquad (1)$$

где $[a,b]$ - диапазон приемлемых входных сигналов, $[c_{\min}, c_{\max}]$ – диапазон значений признака c, c' – предобработанный сигнал, который будет подан на вход сети. Предобработку входного сигнала по формуле (1) будем называть простейшей предобработкой.

4.1.9 Оценка способности сети решить задачу

В данном разделе рассматриваются только сети, все элементы которых непрерывно зависят от своих аргументов (см. главу «Описание нейронных сетей»). Предполагается, что все входные данные предобработаны так, что все входные сигналы сети лежат в диапазоне приемлемых входных сигналов $[a,b]$. Будем обозначать вектора входных сигналов через x^i, а требуемые ответы сети через f^i. Компоненты векторов будем обозначать нижним индексом, например, компоненты входного вектора

через x_j^i. Будем полагать, что в каждом примере ответ является вектором чисел из диапазона приемлемых сигналов $[a,b]$. В случае обучения сети задаче классификации требуемый ответ зависит от вида используемого интерпретатора ответа (см. главу «Оценка и Интерпретатор ответа»).

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 $\sum_{i,j}\left(F_j\left(x^i\right)-f_j^i\right)^2$ была минимальной (в идеале равна нулю). 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 значения функции. Таким образом, наибольшую сложность будет представлять приближение функции f в точках, в которых достигает максимума выражение $\frac{\left\|f^i-f^j\right\|}{\left\|x^i-x^j\right\|}$. Для аналитически заданных функций величина $\sup_{x,y}\frac{\left\|f(x)-f(y)\right\|}{\|x-y\|}$ называется константой Липшица. Исходя из этих соображения 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 следующей формуле:

$$\Lambda_t=\max_{i\neq j}\frac{\left\|f^i-f^j\right\|}{\left\|x^i-x^j\right\|} \tag{2}$$

Оценка (2) является оценкой константы Липшица аппроксимируемой функции снизу.

Для того, чтобы оценить способность сети заданной конфигурации решить задачу, необходимо оценить константу Липшица сети и сравнить ее с выборочной оценкой (2). Константа Липшица сети вычисляется по следующей формуле:

$$\Lambda_n=\sup_{x,y}\frac{\left\|F(x)-F(y)\right\|}{\|x-y\|} \tag{3}$$

В формулах (2) и (3) можно использовать произвольные нормы. Однако для нейронных сетей наиболее удобной является евклидова норма. Далее везде используется евклидова норма.

В следующем разделе описан способ вычисления оценки константы Липшица сети (3) сверху. Очевидно, что в случае $\Lambda_n<\Lambda_t$ сеть принципиально не способна решить задачу аппроксимации функции f.

4.1.9.1 Оценка константы Липшица сети

Оценку константы Липшица сети будем строить в соответствии с принципом иерархического устройства сети, описанным в главе «Описание нейронных сетей». При этом потребуются следующие правила.

1. Для композиции функций $f\circ g=f\left(g(x)\right)$ константа Липшица оценивается как произведение констант Липшица:
$$\Lambda_{f\circ g}\leq\Lambda_f\Lambda_g. \tag{4}$$
2. Для вектор-функции $f=\left(f_1,f_2,\ldots,f_n\right)$ константа Липшица равна:

$$\Lambda_f = \sqrt{\sum_{i=1}^{n} \Lambda_{f_i}^2} \ . \tag{5}$$

4.1.9.2 Способ вычисления константы Липшица

Для непрерывных функций константа Липшица является максимумом производной в направлении $r = (r_1, \ldots, r_n)$ по всем точкам и всем направлениям. При этом вектор направления имеет единичную длину: $\sum_{i=1}^{n} r_i^2 = 1$. Напомним формулу производной функции $f(x_1, \ldots, x_n)$ в направлении r:

$$\frac{\partial f}{\partial r} = \sum_{i=1}^{n} r_i \frac{\partial f}{\partial x_i} \tag{6}$$

4.1.9.3 Синапс

Обозначим входной сигнал синапса через x, а синаптический вес через α. Тогда выходной сигнал синапса равен αx. Поскольку синапс является функцией одной переменной, константа Липшица равна максимуму модуля производной – модулю синаптического веса:

$$\Lambda_s = |\alpha| \tag{7}.$$

4.1.9.4 Умножитель

Обозначим входные сигналы умножителя через x_1, x_2. Тогда выходной сигнал умножителя равен $f_* = x_1 x_2$. Используя (6) получаем $\Lambda_{f_*} = \sup_{x,r} |r_1 x_2 + r_2 x_1|$. Выражение $r_1 x_2 + r_2 x_1$ является скалярным произведением векторов (r_1, r_2) и, учитывая единичную длину вектора r, достигает максимума, когда эти векторы сонаправлены. То есть при векторе

$$r = \left(\frac{x_2}{\|x\|}, \frac{x_1}{\|x\|} \right), \ \|x\| = \sqrt{x_1^2 + x_1^2} \ .$$

Используя это выражение, можно записать константу Липшица для умножителя:

$$\Lambda_{f_*} = \sup_{x,r} |r_1 x_2 + r_2 x_1| = \sup_{x} \frac{\left| x_2^2 + x_1^2 \right|}{\|x\|} = \sup_{x} \|x\| \ . \tag{8}$$

Если входные сигналы умножителя принадлежат интервалу $[a, b]$, то константа Липшица для умножителя может быть записана в следующем виде:

$$\Lambda_{f_*} = \sqrt{2} \max\{|a|, |b|\} \ . \tag{9}$$

4.1.9.5 Точка ветвления

Поскольку в точке ветвления не происходит преобразования сигнала, то константа Липшица для нее равна единице.

4.1.9.6 Сумматор

Производная суммы по любому из слагаемых равна единице. В соответствии с (6) получаем:

$$\Lambda_\Sigma = \sup_{r} \left| \sum_{i=1}^{n} r_i \right| = \sqrt{n} \ , \tag{10}$$

поскольку максимум суммы при ограничении на сумму квадратов достигается при одинаковых слагаемых.

4.1.9.7 Нелинейный Паде преобразователь

Нелинейный Паде преобразователь или Паде элемент имеет два входных сигнала и один выходной. Обозначим входные сигналы через x_1, x_2. Используя (6) можно записать константу Липшица в следующем виде:

$$\Lambda_\pi = \sup_{r,x}\left|\frac{r_1 x_2}{x_2^2} - \frac{r_2 x_1}{x_2^2}\right| = \sup_{r,x}\left|\frac{r_1 x_2 - r_2 x_1}{x_2^2}\right|.$$

Знаменатель выражения под знаком модуля не зависит от направления, а числитель можно преобразовать так же, как и для умножителя. После преобразования получаем:

$$\Lambda_\pi = \sup_{x}\frac{\|x\|}{x_2^2} \tag{11}$$

4.1.9.8 Нелинейный сигмоидный преобразователь

Нелинейный сигмоидный преобразователь, как и любой другой нелинейный преобразователь, имеющий один входной сигнал x, имеет константу Липшица равную максимуму модуля производной:

$$\Lambda_\varphi = \max_x\left|\varphi'(x)\right|. \tag{12}$$

4.1.9.9 Адаптивный сумматор

Для адаптивного сумматора на n входов оценка константы Липшица, получаемая через представление его в виде суперпозиции слоя синапсов и простого сумматора, вычисляется следующим образом. Используя формулу (7) для синапсов и правило (5) для вектор-функции получаем следующую оценку константы Липшица слоя синапсов:

$$\Lambda_L = \sqrt{\sum_{i=1}^{n}\Lambda_{S_i}^2} = \sqrt{\sum_{i=1}^{n}|\alpha_i|^2} = \|\alpha\|.$$

Используя правило (4) для суперпозиции функций и оценку константы Липшица для простого сумматора (10) получаем:

$$\Lambda_A \le \Lambda_\Sigma \Lambda_L = \sqrt{n}\|\alpha\|. \tag{13}$$

Однако, если оценить константу Липшица адаптивного сумматора напрямую, то, используя (6) и тот факт, что при фиксированных длинах векторов скалярное произведение достигает максимума для сонаправленных векторов получаем:

$$\Lambda_A = \sup_{x,r}\left|\sum_{i=1}^{n} r_i\alpha_i\right| = \left|\sum_{i=1}^{n}\alpha_i\frac{\alpha_i}{\|\alpha\|}\right| = \|\alpha\|. \tag{14}$$

Очевидно, что оценка (14) точнее, чем оценка (13).

4.1.9.10 Константа Липшица сигмоидной сети

Рассмотрим слоистую сигмоидную сеть 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 вид, приведенный на рис. 1 и имеют одинаковую характеристику.
5. Все синаптические веса ограничены по модулю единицей.
6. В сети m слоев.

В этом случае, учитывая формулы (4), (5), (12) и (14) константу Липшица i-о слоя можно оценить следующей величиной:

$$\Lambda_i \le \sqrt{\sum_{j=1}^{n_i}\left(\Lambda_\varphi\Lambda_{A_j}\right)^2} = \Lambda_\varphi\sqrt{\sum_{j=1}^{n_i}\left\|\alpha^j\right\|^2} \le \Lambda_\varphi\sqrt{n_{i-1}n_i}.$$

Используя формулу (4) получаем оценку константы Липшица всей сети:

$$\Lambda_n \le \prod_{i=1}^{m} \Lambda_i \le \Lambda_\varphi^m \sqrt{n_0 n_m} \prod_{i=1}^{m-1} n_i .$$

Если используется нейроны типа S_1, то $\Lambda_\varphi = c$ и оценка константы Липшица сети равна:

$$\Lambda_{S_1} \le c^m \sqrt{n_0 n_m} \prod_{i=1}^{m-1} n_i$$

Для нейронов типа S_2, то $\Lambda_\varphi = 1/c$ и оценка константы Липшица сети равна:

$$\Lambda_{S_1} \le c^{-m} \sqrt{n_0 n_m} \prod_{i=1}^{m-1} n_i$$

Обе формулы подтверждают экспериментально установленный факт, что чем круче характеристическая функция нейрона, тем более сложные функции (функции с большей константой Липшица) может аппроксимировать сеть с такими нейронами.

4.1.10 Предобработка, облегчающая обучение

При обучении нейронных сетей иногда возникают ситуации, когда дальнейшее обучение нейронной сети невозможно. В этом случае необходимо проанализировать причины. Возможно несколько видов анализа. Одной из возможных причин является высокая сложность задачи, определяемая как выборочная оценка константы Липшица.

Для упрощения задачи необходимо уменьшить выборочную оценку константы Липшица. Наиболее простой способ добиться этого – увеличить расстояние между входными сигналами. Рассмотрим пару примеров – x^i, x^j – таких, что $\Lambda_t = \dfrac{\left\| f^i - f^j \right\|}{\left\| x^i - x^j \right\|}$. Определим среди координат векторов x^i и x^j координату, в которой достигает минимума величина $\left| x_l^i - x_l^j \right|$, исключив из рассмотрения совпадающие координаты. Очевидно, что эта координата является «узким местом», определяющим сложность задачи. Следовательно, для уменьшения сложности задачи требуется увеличить расстояние между векторами x^i и x^j, а наиболее перспективной координатой для этого является l-я. Однако увеличение расстояние между x_l^i и x_l^j не всегда осмыслено. Дело в том, что все параметры, как правило, измеряются с конечной точностью. Поэтому, если величина $\left| x_l^i - x_l^j \right|$ меньше чем точность измерения l-го параметра, значения x_l^i и x_l^j можно считать совпадающими. Таким образом, для изменения масштаба надо выбирать тот из входных параметров, для которого значение $\left| x_l^i - x_l^j \right|$ минимально, но превышает точность измерения этого параметра.

Предположим, что все входные параметры предобработаны в соответствии с формулой (1). Перенумеруем примеры обучающего множества так, чтобы были верны следующие неравенства: $x_l^1 < x_l^2 <, \ldots, < x_l^N$, где N – число примеров в обучающем множестве. При этом, возможно, придется исключить ряд пар параметр-ответ с совпадающими значениями параметра. Если в какой-либо из таких пар значения ответов различаются, то это снижает возможную полезность данной процедуры.

Наиболее простой путь – разбить диапазон l-го параметра на два. Зададимся точкой x. Будем кодировать l-й

Таблица 7.

Кодирование параметра после разбиения на два сигнала

ЗначениеП	ервый сигналВторой	сигнал
$x_l^i < x$	$\dfrac{\left(x_l^i - a\right)\left(b-a\right)}{\left(x-a\right)} + a$	a
$x_l^i > x$	b	$\dfrac{\left(x_l^i - x\right)\left(b-a\right)}{\left(b-x\right)} + a$

параметр двумя входными сигналами в соответствии с табл. 7. При таком кодировании критерий Липшица, очевидно, уменьшится. Вопрос о выборе точки x может решаться по-разному. Простейший путь – положить $x = (a-b)/2$. Более сложный, но часто более эффективный – подбор x исходя из требования минимальности критерия Липшица.

Приведенный выше способ уменьшения критерия Липшица не единственный. В следующем разделе рассмотрен ряд способов предобработки, решающих ту же задачу.

4.1.11 Другие способы предобработки числовых признаков

В данном разделе будет рассмотрено три вида предобработки числовых признаков – модулярный, позиционный и функциональный. Основная идея этих методов предобработки состоит в том, чтобы сделать значимыми малые отличия больших величин. Действительно, пусть для ответа существенно изменение величины признака на единицу при значении признака порядка миллиона. Очевидно, что простейшая предобработка (1) сделает отличие в единицу неразличимым для нейронной сети при абсолютных значениях порядка миллиона.

Все эти виды предобработки обладают одним общим свойством – за счет кодирования входного признака несколькими сигналами они уменьшают сложность задачи (критерий Липшица).

4.1.11.1 Модулярная предобработка

Зададимся некоторым набором положительных чисел $y_1,\ldots,y_k$. Определим сравнение по модулю для действительных чисел следующим образом:

$$x \mod y = x - y \cdot Int(x/y), \tag{15}$$

где $Int(x)$ – функция, вычисляющая целую часть величины x путем отбрасывания дробной части. Очевидно, что величина $x \mod y$ лежит в интервале $(-y, y)$. Кодирование входного признака x при модулярной предобработке вектором z производится по следующей формуле:

$$z_i = \frac{((x \mod y_i) + y_i)(b-a)}{2y_i} + a. \tag{16}$$

Однако модулярная предобработка обладает одним отрицательным свойством – во всех случаях, когда $y_i \neq y_1^r$, при целом r, разрушается отношение предшествования чисел. В табл. 8 приведен пример векторов. Поэтому, модульная предобработка пригодна при предобработке тех признаков, у которых важна не абсолютная величина, а взаимоотношение этой величины с величинами y_1,K,y_k. Примером такого признака может служить угол между векторами, если в качестве величин y выбрать $y_i = \pi/i$.

Таблица 8.

Пример сигналов при модулярном вводе

x	x mod 3	x mod 5	x mod 7	x mod 11
5	2	0	5	5
10	1	0	3	10
15	0	0	1	3

4.1.11.2 Функциональная предобработка

Функциональная предобработка преследует единственную цель – снижение константы Липшица задачи. В разделе «Предобработка, облегчающая обучение», был приведен пример такой предобработки. Рассмотрим общий случай функциональной предобработки, отображающих входной признак x в k-мерный вектор z. Зададимся набором из k чисел, удовлетворяющих следующим условиям: $x_{\min} < y_1 < \ldots < y_{k-1} < y_k < x_{\max}$. Пусть φ – функция, определенная на интервале $[x_{\min} - y_k, x_{\max} - y_1]$, а $\varphi_{\min}, \varphi_{\max}$ – минимальное и максимальное значения функции φ на этом интервале. Тогда i-я координата вектора z вычисляется по следующей формуле:

$$z_i = \frac{(\varphi(x - y_i) - \varphi_{\min})(b-a)}{\varphi_{\max} - \varphi_{\min}} + a \tag{17}$$

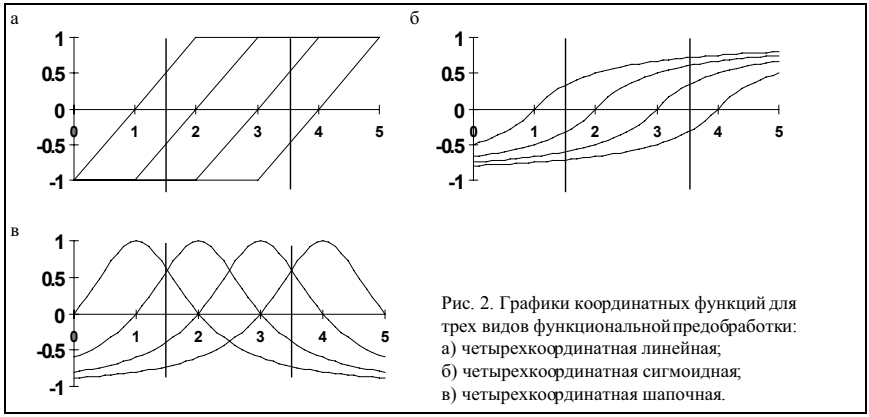

Рис. 2. Графики координатных функций для трех видов функциональной предобработки:
а) четырехкоординатная линейная;
б) четырехкоординатная сигмоидная;
в) четырехкоординатная шапочная.

Линейная предобработка. В линейной предобработке используется кусочно линейная функция:

$$z_i = \begin{cases} a, & x < a, \\ x, & a \le x \le b, \\ b, & b < x. \end{cases} \quad (18)$$

Графики функций $z_i(x)$ представлены на рис. 2а. Видно, что с увеличением значения признака x ни одна функция не убывает, а их сумма возрастает. В табл. 9 представлены значения этих функций для двух точек – $x_1 = 1.5$ и $x_2 = 3.5$.

Таблица 9

Пример функциональной предобработки числового признака $x \in [0,5]$, при условии, что сигналы нейронов принадлежат интервалу $[-1,1]$. В сигмоидной предобработке использована формула $\varphi(x) = x/(1+|x|)$, а в шапочной – $\varphi(x) = 2/(1+x^2) - 1$. Были выбраны четыре точки $y_i = i$.

x	$z_1(x)$	$z_2(x)$	$z_3(x)$	$z_4(x)$
Линейная предобработка				
1.5	0.5	-0.5	-1	-1
3.5	1	1	0.5	-0.5
Сигмоидная предобработка				
1.5	0.3333	-0.3333	-0.6	-0.7142
3.5	0.7142	0.6	0.3333	-0.3333
Шапочная предобработка				
1.5	0.6	0.6	-0.3846	-0.7241
3.5	-0.7241	-0.3846	0.6	0.6

Сигмоидная предобработка. В сигмоидной предобработке может использоваться любая сигмоидная функция. Если в качестве сигмоидной функции использовать функцию S_2, приведенную в разделе «Нейрон» этой главы, то формула (17) примет следующий вид:

$$z_i = \frac{\left(x/(c+|x|)-1\right)(b-a)}{2} + a .$$

Графики функций $z_i(x)$ представлены на рис. 2б. Видно, что с увеличением значения признака x ни одна функция не убывает, а их сумма возрастает. В табл. 9 представлены значения этих функций для двух точек $x_1 = 1.5,\ x_2 = 3.5$.

Шапочная предобработка. Для шапочной предобработки используются любые функции, имеющие график в виде «шапочки». Например, функция $\varphi(x) = 1/(1+x^2)$. Графики функций $z_i(x)$ представлены на рис. 2в. Видно, что с увеличением значения признака x ни одна из функций $z_i(x)$, ни их сумма не ведут себя монотонно. В табл. 9 представлены значения этих функций для двух точек $x_1 = 1.5,\ x_2 = 3.5$.

4.1.11.3 Позиционная предобработка

Основная идея позиционной предобработки совпадает с принципом построения позиционных систем счисления. Зададимся положительной величиной y такой, что $y^k \geq (x_{\min} - x_{\max})$. Сдвинем признак x так, чтобы он принимал только неотрицательные значения. В качестве сигналов сети будем использовать результат простейшей предобработки y-ичных цифр представления сдвинутого признака x. Формулы вычисления цифр приведены ниже:

$$\begin{aligned} z_0 &= (x - x_{\min}) \bmod y, \\ z_1 &= Int\left((x - x_{\min})/y\right) \bmod y, \\ &\dots \\ z_i &= Int\left((x - x_{\min})/y^i\right) \bmod y, \end{aligned} \tag{19}$$

где операция сравнения по модулю действительного числа определена в (15). Входные сигналы сети получаются из компонентов вектора z путем простейшей предобработки.

4.1.12 Составной предобработчик

Поскольку на вход нейронной сети обычно подается несколько входных сигналов, каждый из которых обрабатывается своим предобработчиком, то предобработчик должен быть составным. Представим предобработчик в виде совокупности независимых частных предобработчиков. Каждый частный предобработчик обрабатывает одно или несколько тесно связанных входных данных. Как уже отмечалось ранее, предобработчик может иметь один из четырех типов, приведенных в табл. 10. На входе предобработчик получает вектор входных данных (возможно, состоящий из одного элемента), а на выходе выдает вектор входных сигналов сети (так же возможно состоящий из одного элемента).

Таблица 10.

Типы предобработчиков

ТипОписание	
Number	Предобрабатывает числовые входные данные
Unordered	Предобрабатывает неупорядоченные качественные признаки
Ordered	Предобрабатывает упорядоченные качественные признаки
Binary	Обрабатывает бинарные признаки

Необходимость передачи предобработчику вектора входных данных и получения от него вектора входных сигналов связана с тем, что существуют предобработчики получающие несколько входных данных и выдающие несколько входных сигналов. Примером такого предобработчика может служить предобработчик, переводящий набор координат планеты из сферической в декартову.

Для качественных признаков принято кодирование длинными целыми числами. Первое значение равно 1, второе – 2 и т.д. Числовые признаки кодируются действительными числами.

4.2 Стандарт первого уровня компонента предобработчик

Данный раздел посвящен описанию стандарта языка описания и хранения на внешнем носителе компонента предобработчик. Поскольку крайне редко встречаются случаи, когда сеть получает один входной сигнал, предобработчик всегда является составным. Построение предобработчика происходит в редакторе предобработчика. Для описания предобработчика предлагается использовать специальный язык.

4.2.1 Неопределенные значения

В практике работы большинство таблиц данных не полны. То есть, часть данных в примерах задачника неизвестна. Задачник должен однозначно указать предобработчику неизвестные данные. Для этих целей для каждого типа входных данных определено специальное значение - неопределенное. Для передачи неизвестных значений используются следующие величины: 10^{-40} для действительных чисел и 0 для всех типов качественных признаков.

4.2.2 Стандартные предобработчики

В большинстве случаев достаточно использовать стандартные предобработчики, список которых приведен в табл. 11. Ниже в данном разделе приведено описание параметров стандартных предобработчиков.

Таблица 11

Стандартные предобработчики

Идентификатор	ПараметрыТипОписание		
BinaryPrep	MinSignals, MaxSignals : Real; Unknown: Real; Type : Logic.	Binary	Бинарный признак. Предобработка в соответствии с табл. 4.
UnOrdered	MinSignals, MaxSignals : Real; Unknown: Real; Num : Long	Unordered	Неупорядоченный качественный признак. Предобработка в соответствии с табл. 5.
Ordered	MinSignals, MaxSignals : Real; Unknown: Real; Num : Long	Ordered	Упорядоченный качественный признак. Предобработка в соответствии с табл. 6.
EmptyPrep	MinData, MaxData, Unnown, MinSignals, MaxSignals : Real	Number	Простейшая предобработка в соответствии с формулой (1).
ModPrep	MinSignals, MaxSignals : Real; Unknown: Real; Y : RealArray	Number	Модулярная предобработка в соответствии с формулой (16).
FuncPrep	MinSignals, MaxSignals, Unknown: Real; Y : RealArray; F : FuncType	Number	Функциональная предобработка в соответствии с формулой (17).
PositPrep	MinSignals, MaxSignals, Unnown, Y : Real; Num : Long	Number	Позиционная предобработка в соответствии с формулой (19).

Все стандартные предобработчики получают в качестве аргументов массивы входной информации и входных сигналов. Кроме того, они содержат различные наборы параметров. Алгоритмы выполнения стандартных предобработчиков приведены в разделе «Пример описания предобработчика». Далее описаны наборы параметров стандартных предобработчиков. Все параметры должны быть описаны как статические переменные.

Предобработка бинарного признака (BinaryPrep). Предобработка производится в соответствии с табл. 4. Принимает одно входное данное и генерирует один входной сигнал. Предобработчик содержит следующие параметры.

MinSignals, MaxSignals – значения нижней и верхней границ интервала приемлемых входных сигналов, соответственно. По умолчанию эти величины равны -1 и 1, соответственно.

Unknown– значение сигнала, который будет выдан, если значение входного признака не определено (0). По умолчанию эта величина равна $(MinSignals + MaxSignals)/2$.

Type – тип предобработки бинарного признака. Если значение параметра Type – истина, то производится предобработка по типу «Наличие другого свойства», если ложь, то по типу «Отсутствие заданного свойства». По умолчанию значение этого параметра равно истина.

Предобработка неупорядоченного качественного признака (UnOrdered). Предобработка производится в соответствии с табл. 5. Принимает одно входное данное и генерирует Num входных сигналов. Предобработчик содержит следующие параметры.

MinSignals, MaxSignals – значения нижней и верхней границ интервала приемлемых входных сигналов, соответственно. По умолчанию эти величины равны -1 и 1, соответственно.

Unknown– значение сигналов, которые будут выданы, если значение входного признака не определено (0). По умолчанию эта величина равна $(MinSignals + MaxSignals)/2$.

Num – число состояний качественного признака (число генерируемых входных сигналов). По умолчанию значение этого параметра равно 2.

Предобработка упорядоченного качественного признака (Ordered). Предобработка производится в соответствии с табл. 6. Принимает одно входное данное и генерирует Num входных сигналов. Предобработчик содержит следующие параметры.

MinSignals, MaxSignals – значения нижней и верхней границ интервала приемлемых входных сигналов, соответственно. По умолчанию эти величины равны -1 и 1, соответственно.

Unknown– значение сигналов, которые будут выданы, если значение входного признака не определено (0). По умолчанию эта величина равна $(MinSignals + MaxSignals)/2$

Num – число состояний качественного признака (число генерируемых входных сигналов). По умолчанию значение этого параметра равно 2.

Простейший предобработчик (EmptyPrep). Предобработка производится в соответствии с формулой (1). Принимает одно входное данное и генерирует один входной сигнал. Предобработчик содержит следующие параметры.

MinSignals, MaxSignals – значения нижней и верхней границ интервала приемлемых входных сигналов, соответственно. По умолчанию эти величины равны -1 и 1, соответственно.

Unknown– значение сигнала, который будет выдан, если значение входного признака не определено (10^{-40}). По умолчанию эта величина равна 0.

MinData, MaxData – значения нижней и верхней границ интервала изменения входных данных, соответственно. По умолчанию эти величины равны -1 и 1, соответственно. Эти значения могут быть определены поиском минимального и максимального значений по задачнику, однако предобработчик не может выполнить эту процедуру.

Модулярный предобработчик (ModPrep). Предобработка производится в соответствии с формулой (16). Принимает одно входное данное и генерирует столько входных сигналов, сколько элементов в массиве Y (нулевой элемент массива содержит число элементов). Предобработчик содержит следующие параметры.

MinSignals, MaxSignals – значения нижней и верхней границ интервала приемлемых входных сигналов, соответственно. По умолчанию эти величины равны -1 и 1, соответственно.

Unknown– значение сигналов, которые будут выданы, если значение входного признака не определено (10^{-40}). По умолчанию эта величина равна 0.

Y – массив величин, используемых для предобработки (см. раздел «Модулярная предобработка»).

Функциональный предобработчик (FuncPrep). Предобработка производится в соответствии с формулой (17). Принимает одно входное данное и генерирует столько входных сигналов, сколько элементов в массиве Y (нулевой элемент массива содержит число элементов). Предобработчик содержит следующие параметры.

MinSignals, MaxSignals – значения нижней и верхней границ интервала приемлемых входных сигналов, соответственно. По умолчанию эти величины равны -1 и 1, соответственно.

Unknown– значение сигналов, которые будут выданы, если значение входного признака не определено (10^{-40}). По умолчанию эта величина равна 0.

MinData, MaxData – значения нижней и верхней границ интервала изменения функции F от входных данных, соответственно. По умолчанию эти величины равны -1 и 1, соответственно. Эти значения могут быть определены поиском минимального и максимального значений функции по задачнику, однако предобработчик не может выполнить эту процедуру.

Y – массив величин, используемых для предобработки (см. раздел «Функциональная предобработка»).

F – имя однопараметрической функции действительного типа (ее адрес) используемой для предобработки.

Позиционный предобработчик (PositPrep). Предобработка производится в соответствии с формулой (19). Принимает одно входное данное и генерирует Num входных сигналов. Предобработчик содержит следующие параметры.

MinSignals, MaxSignals – значения нижней и верхней границ интервала приемлемых входных сигналов, соответственно. По умолчанию эти величины равны -1 и 1, соответственно.

Unknown– значение сигналов, которые будут выданы, если значение входного признака не определено (10^{-40}). По умолчанию эта величина равна 0.

Y – основание системы счисления (см. раздел «Функциональная предобработка»). По умолчанию эта величина равна 2.

Num – число цифр в представлении входного сигнала. По умолчанию эта величина равна 2.

4.2.3 Язык описания предобработчика

Предобработчик является составным объектом. В состав этого объекта входят частные предобработчики, правила распределения входных данных и входных сигналов сети между частными предоб-

Таблица 12

Ключевые слова языка описания предобработчика.

Идентификатор	Краткое описание
Connections	Начало блока описания распределения входных данных и сигналов.
Contents	Начало блока описания состава интерпретатора.
Data	Имя, по которому адресуются входные данные, начало блока описания входных данных
Include	Предшествует имени файла, целиком вставляемого в это место описания.
NumberOf	Функция. Возвращает число обрабатываемых частным предобработчиком входных данных или сигналов.
Prep	Начало заголовка описания частного предобработчика.
Preparator	Заголовок раздела файла, содержащий описание интерпретатора.
Signals	Имя, по которому адресуются входные сигналы; начало блока описания сигналов.

работчиками. Предобработчик при выполнении запроса на предобработку вектора входных данных получает на входе вектор исходных данных, а возвращает вектор входных сигналов сети.

Каждый частный интерпретатор ответа получает на входе вектор входных данных, которые он предобрабатывает, а на выходе дает вектор входных сигналов сети. Каждый частный интерпретатор описывается в виде процедурного блока.

В табл. 12 приведен список ключевых слов языка описания предобработчика, дополняющий список ключевых слов, приведенных в главе «Общий стандарт». Кроме того, ключевыми словами являются имена стандартных предобработчиков, приведенные в табл. 11.

4.2.3.1 БНФ языка описания предобработчика

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

<Описание предобработчика> ::= <Заголовок> [<Описание функций>] [<Описание частных предобработчиков>] <Описание состава> [<Установление параметров>] [<Описание сигналов>] [<Описание данных>] [<Описание распределения сигналов>] [<Описание распределения данных>] <Конец описания предобработчика>

<Заголовок> ::= **Preparator** <Имя предобработчика> *(* <Список формальных аргументов> *)*

<Имя предобработчика> ::= <Идентификатор>

<Описание частных предобработчиков> ::= <Описание частного предобработчика> [<Описание частных предобработчиков>]

<Описание частного предобработчика> ::= <Заголовок описания предобработчика> [<Описание статических переменных>] [<Описание переменных>] <Тело предобработчика>

<Заголовок описания предобработчика> ::= **Prep** <Имя частного предобработчика> ([(<Список формальных аргументов>)])

<Имя частного интерпретатора> ::= <Идентификатор>

<Тело предобработчика> ::= **Begin** <Составной оператор> **End**

<Описание состава> ::= **Contents** <Список имен предобработчиков> ;

<Список имен предобработчиков> ::= <Имя предобработчика> [,<Список имен предобработчиков>]

<Имя предобработчика> ::= <Псевдоним>*:* {<Имя ранее описанного интерпретатора> | <Имя стандартного интерпретатора>} [*/*<Число экземпляров >*/*] [(<Список фактических аргументов>)]

<Псевдоним> ::= <Идентификатор>

<Число экземпляров > ::= <Целое число>

<Имя ранее описанного интерпретатора> ::= <Идентификатор>

<Имя стандартного интерпретатора> ::= <Идентификатор>

<Установление параметров> ::= <Установление параметров ***Частного предобработчика***> [;<Установление параметров>]

<Описание сигналов> ::= **Signals <**Константное выражение типа ***Long*****>**

<Описание данных> ::= **Data <**Константное выражение типа ***Long*****>**

<Описание распределения сигналов> ::= <Описание распределения ***Сигналов, Предобработчика, Частного предобработчика, Signals***>

<Описание распределения данных> ::= <Описание распределения ***Данных, Предобработчика, Частного предобработчика, Data***>

<Конец описания предобработчика> ::= **End Preparator**

4.2.3.2 Описание языка описания предобработчика

Структура описания предобработчика имеет вид: заголовок; описание функций; описание частных предобработчиков; описание состава; установление параметров; описание сигналов; описание данных; описание распределения сигналов; описание распределения данных; конец описания предобработчика.

Заголовок состоит из ключевого слова **Preparator** и имени предобработчика и служит для обозначения начала описания предобработчика в файле, содержащем несколько компонентов нейрокомпьютера.

Описание функций – фрагмент описания, в котором описаны функции, необходимые для работы предобработчиков.

Описание частного предобработчика – это описание процедуры, вычисляющей входные сигналы нейронной сети по входным данным. Формальные аргументы служат для задания размерностей обрабатываемых векторов. При выполнении частный предобработчик получает в качестве аргументов два вектора – входных данных и входных сигналов. Формально, при исполнении частный предобработчик имеет описание следующего вида:

Pascal:

```
Procedure Preparator(Data, Signals : PRealArray);
```

C:

```
void Preparator(PRealArray Data; PRealArray Signals);
```

В разделе описания состава перечисляются частные предобработчики, входящие в состав предобработчика. Признаком конца раздела служит символ «;».

В необязательном разделе установления параметров производится задание значений параметров (статических переменных) частных предобработчиков. После ключевого слова SetParameters следует список значений параметров в том порядке, в каком параметры были объявлены при описании частного интерпретатора (для стандартных интерпретаторов порядок параметров соответствует порядку, приведенному в описании стандартных предобработчиков в разделе «Стандартные предобработчики»). При использовании одного оператора задания параметров для задания параметров нескольким экземплярам одного частного предобработчика после ключевого слова SetParameters указывается столько выражений, задающих значения параметров, сколько необходимо для одного экземпляра.

В необязательном разделе "описание сигналов" указывается число сигналов, вычисляемых предобработчиком. Если этот раздел опущен, то полагается, что число вычисляемых предобработчиком сигналов равно сумме сигналов, вычисляемых всеми частными предобработчиками. В константном выражении возможно использование функции NumberOf, аргументом которой является имя частного предобработчика (или его псевдоним) и ключевое слово Signals, в качестве второго аргумента.

В необязательном разделе "описание данных" указывается число входных данных, предобрабатываемых предобработчиком. Если этот раздел опущен, то полагается, что число предобрабатываемых предобработчиком данных равно сумме данных, предобрабатываемых всеми частными предобработчиками. В константном выражении возможно использование функции NumberOf, аргументом которой является имя частного предобработчика (или его псевдоним) и ключевое слово Data, в качестве второго аргумента.

В необязательном разделе описания распределения сигналов (данных) указывается для каждого частного предобработчика какие сигналы (входные данные) из общего вектора сигналов (данных) передаются ему для обработки.

Наиболее часто встречающиеся интерпретаторы объявлены стандартными. Для стандартных интерпретаторов описание частных интерпретаторов отсутствует.

Кроме того, в любом месте описания интерпретатора могут встречаться комментарии, заключенные в фигурные скобки.

4.2.3.3 Пример описания предобработчика

В этом разделе приведены два примера описания одного и того же предобработчика для метеорологической задачи. Используется следующий состав предобработчика: первый элемент вектора входных данных (температура воздуха) обрабатывается простейшим предобработчиком (EmptyPrep); второй (облачность) – бинарным предобработчиком (BinaryPrep); третий (направление ветра) – предобработчиком неупорядоченных качественных признаков (UnOrdered); четвертый (осадки) – предобработчиком неупорядоченных качественных признаков (Ordered).

В первом примере приведено описание дубликатов всех стандартных предобработчиков. Во втором – использованы стандартные предобработчики.

Пример 1.

```
Preparator Meteorology

Function Sigmoid( X Real ) : Real;
   Begin
      Sigmoid = X / (1 + Abs(X))
   End;

Prep BinaryPrep1 ()      {Предобработка бинарного признака}
   Static
      Real MinSignals Name "Нижняя граница интервала приемлемых сигналов";
      Real MaxSignals Name "Верхняя граница интервала приемлемых сигналов";
      Real Unknown Name "Значение сигнала, если значение входного признака не определено";
      Logic Type Name "Тип предобработки бинарного признака";
   Begin
      If TLong(Data[1]) = UnknownLong Then Signals[1] = Unknown
      Else Begin
```

```
        If Type Then Begin
            If TLong(Data[1]) = 1 Then Signals[1] = 0 Else Begin
                If MaxSignals =0 Then Signals[1] = MinSignals Else Signals[1] = MaxSignals
            End
        Else Begin
            If TLong(Data[1]) = 1 Then Signals[1] = MinSignals Else Signals[1] = MaxSignals
        End
    End
End

Prep UnOrdered1 ( Num : Long ) {Предобработка упорядоченного качественного признака}
    Static
        Real MinSignals Name "Нижняя граница интервала приемлемых сигналов";
        Real MaxSignals Name "Верхняя граница интервала приемлемых сигналов";
        Real Unknown Name "Значение сигнала, если значение входного признака не определено";
    Var
        Integer I;
    Begin
        If TLong(Data[1]) = UnknownLong Then Begin
            For I = 1 To Num Do
                Signals[I] = Unknown
        End Else Begin
            For I = 1 To Num Do
                Signals[I] = MinSignals
            Signals[TLong(Data[1])] = MaxSignals
        End
    End

Prep Ordered1 ( Num : Long ) {Предобработка упорядоченного качественного признака}
    Static
        Real MinSignals Name "Нижняя граница интервала приемлемых сигналов";
        Real MaxSignals Name "Верхняя граница интервала приемлемых сигналов";
        Real Unknown Name "Значение сигнала, если значение входного признака не определено";
    Var
        Integer I;
    Begin
        If TLong(Data[1]) = UnknownLong Then Begin
            For I = 1 To Num Do
                Signals[I] = Unknown
        End Else Begin
            For I = 1 To TLong(Data[1]) Do
                Signals[I] = MaxSignals
            For I = TLong(Data[1])+1 To Num Do
                Signals[I] = MinSignals
        End
    End

Prep EmptyPrep1 ()      {Предобработчик, осуществляющий масштабирование и сдвиг сигнала}
    Static
        Real MinSignals Name "Нижняя граница интервала приемлемых сигналов";
        Real MaxSignals Name "Верхняя граница интервала приемлемых сигналов";
        Real Unknown Name "Значение сигнала, если значение входного признака не определено";
        Real MinData Name "Значения нижней границы интервала изменения входных данных";
        Real MaxData Name "Значения верхней границы интервала изменения входных данных";
    Begin
        If Data[1] = UnknownReal Then Signals[1] = Unknown
        Else Signals[1] = (Data[1] – MinData) * (MaxSignals – MinSignals) / (MaxData – MinData)
            + MinSignals
    End

Prep ModPrep1 ( Num : Long )        {Модулярный предобработчик}
```

```
    Static
        Real MinSignals Name "Нижняя граница интервала приемлемых сигналов";
        Real MaxSignals Name "Верхняя граница интервала приемлемых сигналов";
        Real Unknown Name "Значение сигнала, если значение входного признака не определено";
        RealArray[Num] Y Name "Массив величин, используемых для предобработки"
    Var
        Integer I;
    Begin
        If Data[1] = UnknownReal Then Begin
            For I = 1 To Num Do
                Signals[I] = Unknown
        End Else Begin
            For I = 1 To Num Do
                Signals[I] = (Data[1] RMod Y[I] + Y[I]) * (MaxSignals – MinSignals) / (2 * Y[I])
                        + MinSignals
    End

Prep FuncPrep1(Num : Long; F : FuncType)          {Функциональный предобработчик}
    Static
        Real MinSignals Name "Нижняя граница интервала приемлемых сигналов";
        Real MaxSignals Name "Верхняя граница интервала приемлемых сигналов";
        Real Unknown Name "Значение сигнала, если значение входного признака не определено";
        Real MinData Name "Значения нижней границы интервала изменения значений функции F";
        Real MaxData Name "Значения верхней границы интервала изменения значений функции F";
        RealArray[Num] Y Name "Массив величин, используемых для предобработки"
    Var
        Integer I;
    Begin
        If Data[1] = UnknownReal Then Begin
            For I = 1 To Num Do
                Signals[I] = Unknown
        End Else Begin
            For I = 1 To Num Do
                Signals[1] = (F(Data[1] – Y[1] – MinData) * (MaxSignals – MinSignals) /
                        (MaxData – MinData) + MinSignals
    End

Prep PositPrep1( Num : Long )          {Позиционный предобработчик}
    Static
        Real MinSignals Name "Нижняя граница интервала приемлемых сигналов"
        Real MaxSignals Name "Верхняя граница интервала приемлемых сигналов"
        Real Unknown Name "Значение сигнала, если значение входного признака не определено"
        Real Y Name "Основание системы счисления"
    Var
        Integer I;
        Real W, Q;
    Begin
        If Data[1] = UnknownReal Then Begin
            For I = 1 To Num Do
                Signals[I] = Unknown
        End Else Begin
            W = Data[1];
            For I = 1 To Num Do Begin
                Q = W RMod Y;
                Signals[I] = Q * (MaxSignals – MinSignals) / Y + MinSignals;
                W = (W - Q) / Y
            End;
    End

Contents Temp : EmptyPrep1, Cloud : BinaryPrep1, Wind : UnOrdered1(8), Rain : Ordered1(3);
```

```
Temp SetParameters -1, 1, 1E-40, 273, 293;   {Для всех предобработчиков приемлемые значения вход-}
Cloud SetParameters -1, 1, 0, True;   {ных сигналов лежат в интервале от -1 до 1. В случае неопреде-}
Wind SetParameters -1, 1, 0;          {ленного значения во входных данных все сигналы данного}
Rain SetParameters -1, 1, 0           {предобработчика полагаются равными нулю. Входные данные}
                                      {первого предобработчика меняются от 273 до 293}
Signals NumberOf(Signals,Temp) + NumberOf(Signals, Cloud) + NumberOf(Signals, Wind(8)) +
        NumberOf(Signals, Rain(3))

Data NumberOf(Data,Temp) + NumberOf(Data, Cloud) + NumberOf(Data, Wind(8)) +
        NumberOf(Data, Rain(3))

Connections
    Temp.Data <=> Data[1];
    Cloud.Data <=> Data[2];
    Wind.Data <=> Data[3];
    Rain.Data <=> Data[4];
    Temp.Signals <=> Signals[1];
    Cloud.Signals <=> Signals[2];
    Wind.Signals[1..8] <=> Signals[3..10];
    Rain.Signals[1..3] <=> Signals[11..13]

End Preparator
```

Пример 2.

```
Preparator Meteorology

Contents Temp : EmptyPrep, Cloud : BinaryPrep, Wind : UnOrdered(8), Rain : Ordered(3);

Temp SetParameters -1, 1, 1E-40, 273, 293

End Preparator
```

4.3 Стандарт второго уровня компонента предобработчик

Запросы к компоненту предобработчик можно разбить на пять групп:

1. Предобработка.
2. Изменение параметров.
3. Работа со структурой.
4. Инициация редактора предобработчика.
5. Обработка ошибок.

Поскольку нейрокомпьютер может работать одновременно с несколькими сетями, то и компонент предобработчик должна иметь возможность одновременной работы с несколькими предобработчиками. Поэтому большинство запросов к предобработчику содержат явное указание имени предобработчика. Ниже приведено описание всех запросов к компоненту предобработчик. Каждый запрос является логической функцией, возвращающей значение истина, если запрос выполнен успешно, и ложь – при ошибочном завершении исполнения запроса.

В запросах второй и третьей группы при обращении к частным интерпретаторам используется следующий синтаксис:

<Полное имя частного интерпретатора> ::=
<Имя интерпретатора>.<Псевдоним частного интерпретатора> [*[*<Номер экземпляра>*]*]

Таблица 13.

Значения предопределенных констант компонента предобработчик

НазваниеВеличина	Значение
BinaryPrep	0	Стандартный предобработчик бинарных признаков
UnOrdered	1	Стандартный предобработчик неупорядоченных качественных признаков
Ordered	2	Стандартный предобработчик упорядоченных качественных признаков.
EmptyPrep	3	Стандартный простейший предобработчик
ModPrep	4	Стандартный модулярный предобработчик
FuncPrep	5	Стандартный функциональный предобработчик
PositPrep	6	Стандартный позиционный предобработчик
UserType	-1	Предобработчик, определенный пользователем.

При вызове ряда запросов используются предопределенные константы. Их значения приведены в табл. 13.

4.3.1 Запрос на предобработку

Единственный запрос первой группы выполняет основную функцию компонента предобработчик – предобрабатывает входные данные, вычисляя вектор входных сигналов.

4.3.1.1 Предобработать вектор сигналов (Prepare)

Описание запроса:

Pascal:

```
Function Prepare(CompName : PString; Data : PRealArray; Var Signals : PRealArray) : Logic;
```

C:

```
Logic Prepare(PString CompName, PRealArray Data; PRealArray* Signals)
```

Описание аргумента:

CompName – указатель на строку символов, содержащую имя предобработчика.

Data – массив входных данных.

Signals – вычисляемый массив входных сигналов.

Назначение – предобрабатывает массив входных данных Data, вычисляя массив входных сигналов Signals используя предобработчик, указанный в параметре CompName.

Описание исполнения.

1. Если Error <> 0, то выполнение запроса прекращается.
2. Если в качестве аргумента CompName дан пустой указатель, или указатель на пустую строку, то исполняющим запрос объектом является текущий предобработчик – первый в списке предобработчиков компонента предобработчик.
3. Если список предобработчиков компонента предобработчик пуст или имя предобработчика, переданное в аргументе CompName в этом списке не найдено, то возникает ошибка 201 – неверное имя предобработчика, управление передается обработчику ошибок, а обработка запроса прекращается.
4. Производится предобработка предобработчиком, имя которого было указано в аргументе CompName.
5. Если во время выполнения запроса возникает ошибка, то генерируется внутренняя ошибка 204 - ошибка предобработки. Управление передается обработчику ошибок. Выполнение запроса прекращается. В противном случае выполнение запроса успешно завершается.

4.3.2 Остальные запросы.

Ниже приведен список запросов к компоненту предобработчик, исполнение которых описано в главе «Общий стандарт»:

prSetCurrent – Сделать предобработчик текущим
prAdd – Добавление нового предобработчика
prDelete – Удаление предобработчика
prWrite – Запись предобработчика
prGetStructNames – Вернуть имена структурных единиц предобработчика
prGetType – Вернуть тип структурной единицы предобработчика
prGetData – Получить параметры предобработчика
prGetName – Получить имена параметров предобработчика
prSetData – Установить параметры предобработчика
prEdit – Редактировать предобработчик
OnError – Установить обработчик ошибок
GetError – Дать номер ошибки
FreeMemory – Освободить память

В запросе prGetType в переменной TypeId возвращается значение одной из предопределенных констант, перечисленных в табл. 13.

4.3.3 Ошибки компонента предобработчик

В табл. 14 приведен полный список ошибок, которые могут возникать при выполнении запросов компонентом предобработчик, и действия стандартного обработчика ошибок.

Таблица 14.

Ошибки компонента предобработчик и действия стандартного обработчика ошибок.

№Название ошибкиСтандартная обработка		
201	Неверное имя предобработчикаЗанесение	номера в Error
202	Ошибка считывания предобработчикаЗанесение	номера в Error
203	Ошибка сохранения предобработчикаЗанесение	номера в Error
204	Ошибка предобработкиЗанесение	номера в Error