7. Исполнитель

Компонент исполнитель является служебным. Это означает, что он универсален и невидим для пользователя. В отличие от всех других компонентов исполнитель не выполняет ни одной явной функции в обучении нейронных сетей, а является вспомогательным для компонентов учитель и контрастер. Задача этого компонента – упростить работу компонентов учитель и контрастер. Этот компонент выполняет всего несколько запросов, преобразуя каждый из них в последовательность запросов к различным компонентам. В первой части главы содержательно рассмотрены алгоритмы исполнения всех запросов исполнителя, а во второй части приведено их формальное описание. Отметим, что ввиду универсальности компонента исполнитель стандарт первого уровня отсутствует.

7.1 Описание запросов исполнителя.

Как было описано в главе «Функциональные компоненты», исполнитель выполняет четыре вида запросов.

1. Тестирование решения примера.
2. Оценивание решения примера.
1. Оценивание решения примера с вычислением градиента.
2. Оценивание и тестирование решения примера.

Все перечисленные запросы работают с текущей сетью и текущим примером задачника. Однако компоненту задачник необходимо указать, какой пример подлежит обработке. Кроме того, в главе «Оценка и интерпретатор ответа» введен класс оценок, вычисляемых по всему обучающему множеству. Такие оценки позволяют существенно улучшить обучаемость сети и ускорить ее обучение. Нет смысла возлагать перебор примеров на учителя, поскольку это снижает полезность компонента исполнитель. Таким образом, возникает еще четыре вида запросов.

5. Тестирование решения всех примеров обучающего множества.
6. Оценивание решения всех примеров обучающего множества.
5. Оценивание решения всех примеров обучающего множества с вычислением градиента.
6. Оценивание и тестирование решения всех примеров обучающего множества.

Как уже отмечалось в главе «Функциональные компоненты», каждую из приведенных четверок запросов можно объединить в один запрос с параметрами. В табл. 1 приведен полный список параметров для первой четверки запросов, а в табл. 2 – для второй.

Символ «+» означает, что в запросе, номер которого указан в первой строке колонки, возможность, задаваемая данным параметром, должна быть использована. Символ «–» – что связанная с данным параметром возможность не используется. Символы «+/–» означают, что запрос может, как использовать, так и не использовать дан-

Таблица 1

Параметры запроса для позадачной работы

Название параметра	1	2	3	4
Перейти к следующему примеру	+/–	+/–	+/–	+/–
Остановиться в конце обучающего множества	+/–	+/–	+/–	+/–
Вычислять оценку	–	+	+	+
Интерпретировать ответ	+	–	–	+
Вычислять градиент	–	–	+	–
Подготовка к контрастированию	–	–	+/–	–

Таблица 2

Параметры запроса для обучающего множества в целом

Название параметра	5	6	7	8
Вычислять оценку	–	+	+	+
Интерпретировать ответ	+	–	–	+
Вычислять градиент	–	–	+	–
Подготовка к контрастированию	–	–	+/–	–

Таблица 3

Предопределенные константы компонента исполнитель

Название	Идентификатор	Значение	
		Десят.	Шестн.
Вычислять оценку	Estimate	1	0001
Интерпретировать ответ	Interpret	2	0002
Вычислять градиент	Gradient	4	0004
Подготовка к контрастированию	Contrast	8	0008
Перейти к следующему примеру	NextExample	16	0010
Остановиться в конце обучающего множества	StopOnEnd	32	0020
Устанавливать ответы	PutAnswers	64	0040
Устанавливать оценки	PutEstimations	128	0080
Устанавливать уверенность в ответе	PutReliability	256	0100

ную возможность. Отметим, что подготовка к контрастированию может быть задействована, только если производится вычисление градиента, а вычисление градиента невозможно без вычисления оценки. Остальные параметры независимы.

Отбор примеров в обучающее множество, открытие сеанса работы с задачником должны выполняться учителем или контрастером. Исполнитель только организует перебор примеров в обучающем множестве.

7.2 Стандарт компонента исполнитель второго уровня

В данном разделе описаны запросы исполнителя с алгоритмами их исполнения. При описании запросов используется аргумент Instruct, являющийся целым числом, принимающим значение одной из предопределенных констант, приведенных в табл. 3., или суммы любого числа этих констант. Аргумент Instruct является совокупностью шести битовых флагов.

В запросах не указываются используемые сеть, оценка и интерпретатор ответа, поскольку компонент исполнитель всегда использует текущие сеть, оценку и интерпретатор ответа.

7.2.1 Позадачная обработка (TaskWork)

Описание запроса:

Pascal:

```
Function TaskWork(Instruct, Handle : Integer; Var Answers, Reliability : PRealArray; Var Estim : Real) :
                                                                                    Logic;
```

C:

```
Logic TaskWork(Integer Instruct, Integer Handle, PRealArray* Answers, PRealArray* Reliability; Real* Estim)
```

Описание аргументов:

Instruct – содержит инструкции о способе исполнения.

Handle – номер сеанса в задачнике.

Answers – указатель на массив вычисленных ответов.

Reliability – указатель на массив коэффициентов уверенности сети в ответах.

Estim – оценка решения примера.

Назначение – производит обработку одного примера.

Переменные, используемые при исполнении запроса

InArray, RelArray – адреса массивов для обменов с задачником.

Back – адрес массива для обменов с оценкой.

Описание исполнения.

Если в любой момент исполнения запроса возникает ошибка при исполнении запросов к другим компонентам, то исполнение запроса прекращается, возвращается значение ложь, ошибка компонента исполнитель не генерируется.

1. Если в аргументе Instruct установлен бит Gradient и не установлен бит Estimate, то выполнение запроса прекращается, и генерируется ошибка 001 – Некорректное сочетание флагов в аргументе Instruct.
2. Если в аргументе Instruct установлен бит Gradient, то генерируется запрос к сети NullGradient с аргументом Null.
3. Если в аргументе Instruct установлен бит NextExample, то генерируется запрос к задачнику Next с аргументом Handle. (Переход к следующему примеру)
4. Генерируется запрос к задачнику Last с аргументом Handle. (Проверка, существует ли пример)
5. Если запрос Last вернул значение истина, то
   5.1. Если в аргументе Instruct установлен бит StopOnEnd, то исполнение запроса прекращается, возвращается значение ложь. (Примера нет, переход на начало не нужен)
   5.2. Генерируется запрос к задачнику Home с аргументом Handle. (Переход на начало обучающего множества)
6. Переменной InArray присваивается значение Null и генерируется запрос к задачнику Get с аргументами Handle, InArray, tbPrepared (Получает от задачника предобработанные входные сигналы)
7. Генерируется запрос к сети Forw, с аргументами Null, InArray (выполняется прямое функционирование сети).
8. Освобождается массив InArray

9. Присваивает переменной Data значение Null и генерирует запрос к сети GetNetData с аргументами Null, OutSignals, Data (Получает от сети выходные сигналы).
10. Если в аргументе Instruct установлен бит Interpret, то
    10.1. Генерируется запрос к интерпретатору ответа Interpretate с аргументами Data, Answers, Reliability. (Производит интерпретацию ответа)
    10.2. Если в аргументе Instruct установлен бит PutAnswers, то генерируется запрос к задачнику Put с аргументами Handle, Answers, tbCalcAnswers (Передает задачнику вычисленные ответы)
    10.3. Если в аргументе Instruct установлен бит PutReliability, то генерируется запрос к задачнику Put с аргументами Handle, Reliability, tbCalcReliability (Передает задачнику вычисленные коэффициенты уверенности в ответе)
11. Если в аргументе Instruct установлен бит Gradient, то создается массив Back того же размера, что и Data. В противном случае переменной Back присваивается значение Null.
12. Если в аргументе Instruct установлен бит Estimate, то
    12.1. Переменной InArray присваивается значение Null и генерируется запрос к задачнику Get с аргументами Handle, InArray, tbAnswers (Получает от задачника правильные ответы)
    12.2. Переменной RelArray присваивается значение Null и генерируется запрос к задачнику Get с аргументами Handle, RelArray, tbCalcReliability(Получает от задачника достоверности ответов)
    12.3. Генерируется запрос к оценке Estimate с аргументами Data, Back, InArray, RelArray, Direv, Estim. Вместо Direv передается ноль, если в аргументе Instruct установлен бит Gradient, и 1 в противном случае. (Вычисляет оценку примера и, возможно, производные)
    12.4. Если в аргументе Instruct установлен бит PutEstimations, то генерируется запрос к задачнику Put с аргументами Handle, Estim, tbEstimations (Передает задачнику оценку примера)
    12.5. Освобождает массивы InArray и RelArray.
13. Если в аргументе Instruct установлен бит Gradient, то генерируется запрос к сети Back, с аргументами Null, Back. Освобождает массив Back. (Выполняется обратное функционирование сети)
14. Освобождается массив Data.
15. Если в аргументе Instruct установлен бит Contrast, то генерируется запрос к контрастеру ContrastExample с аргументом истина.
16. Завершает исполнение, возвращая значение истина

7.2.2 Обработка обучающего множества (TaskSetWork)

Описание запроса:

Pascal:

```
Function TaskSetWork(Instruct, Handle : Integer; Var Tasks : Integer; Var Correct : PRealArray; Var Estim :
                                                                    Real) : Logic;
```

C:

```
Logic TaskSetWork(Integer Instruct, Integer Handle, Integer* Tasks, PRealArray* Correct, Real* Estim)
```

Описание аргументов:

Instruct – содержит инструкции о способе исполнения.

Handle – номер сеанса в задачнике.

Tasks – число примеров в обучающем множестве.

Correct – указатель на массив, первый элемент которого равен числу правильных ответов на первую подзадачу и т.д.

Estim – средняя оценка решения всех примеров обучающего множества.

Назначение – производит обработку всех примеров обучающего множества.

Переменные, используемые при исполнении запроса

InArray, AnsArray, RelArray – адреса массивов для обменов с задачником.

Answers – указатель на массив вычисленных ответов.

Reliability – указатель на массив коэффициентов уверенности сети в ответах.

Back – адрес массива для обменов с оценкой.

Work – рабочая переменная типа Real для подсчета суммарной оценки.

Weight – рабочая переменная типа Real для веса примера.

Описание исполнения.

Если в любой момент исполнения запроса возникает ошибка при исполнении запросов к другим компонентам, то исполнение запроса прекращается, освобождаются все созданные в нем массивы, возвращается значение ложь, ошибка компонента исполнитель не генерируется.

Значение бит NextExample и StopOnEnd в аргументе Instruct игнорируются.

1. Если в аргументе Instruct установлен бит Gradient и не установлен бит Estimate, то выполнение запроса прекращается, и генерируется ошибка 001 – Некорректное сочетание флагов в аргументе Instruct.
2. Если в аргументе Instruct установлен бит Interpret, то создаются массивы Answers и Reliability того же размера, что и Correct
3. Выполняется следующий фрагмент программы (Обнуление массива количеств правильных ответов)
   3.1. For I = 1 To TLong(Correct[0]) Do
   3.2. Correct[I] = 0
4. Обнуляем счетчик числа примеров: Tasks = 0
5. Обнуляем суммарную оценку: Work = 0
6. Переменной Back присваивается значение Null.
7. Присваивает переменной Data значение Null и генерирует запрос к сети GetNetData с аргументами Null, OutSignals, Data. (Получает от сети выходные сигналы, для выяснения размерности массива Data. Сами значения сигналов не нужны)
8. Если в аргументе Instruct установлен бит Gradient, то
   8.1. Генерируется запрос к сети NullGradient с аргументом Null.
   8.2. Создается массив Back того же размера, что и Data.
9. Генерируется запрос к задачнику Home с аргументом Handle. (Переход на начало обучающего множества)
10. Переменной InArray присваивается значение Null и генерируется запрос к задачнику Get с аргументами Handle, InArray, tbPrepared (Создаем массив InArray для получения от задачника предобработанных входных сигналов)
11. Переменной AnsArray присваивается значение Null и генерируется запрос к задачнику Get с аргументами Handle, AnsArray, tbAnswers (Создаем массив AnsArray для получения от задачника правильных ответов)
12. Если в аргументе Instruct установлен бит Estimate, то создается массив RelArray того же размера, что и AnsArray.
13. Генерируется запрос к задачнику Last с аргументом Handle. (Проверка, существует ли пример)
14. Если запрос Last вернул значение ложь, то
    14.1. Tasks = Tasks + 1
    14.2. Генерируется запрос к задачнику Get с аргументами Handle, InArray, tbPrepared (Получает от задачника предобработанные входные сигналы)
    14.3. Генерируется запрос к сети Forw, с аргументами Null, InArray. (Выполняется прямое функционирование сети)
    14.4. Генерирует запрос к сети GetNetData с аргументами Null, OutSignals, Data. (Получает от сети выходные сигналы)
    14.5. Если в аргументе Instruct установлен бит Interpret, то
        14.5.1. Генерируется запрос к интерпретатору ответа Interpretate с аргументами Data, Answers, Reliability. (Производит интерпретацию ответа)
        14.5.2. Если в аргументе Instruct установлен бит PutAnswers, то генерируется запрос к задачнику Put с аргументами Handle, Answers, tbCalcAnswers (Передает задачнику вычисленные ответы)
        14.5.3. Если в аргументе Instruct установлен бит PutReliability, то генерируется запрос к задачнику Put с аргументами Handle, Reliability, tbCalcReliability (Передает задачнику вычисленные коэффициенты уверенности в ответе)
        14.5.4. Генерируется запрос к задачнику Get с аргументами Handle, AnsArray, tbAnswers (Получает от задачника правильные ответы)
        14.5.5. Выполняется следующий фрагмент программы (Подсчитываются правильно полученные ответы)
            14.5.5.1.For I = 1 To TLong(Correct[0]) Do
            14.5.5.2. If Answers[I] = AnsArray[I] Then TLong(Correct[I]) = TLong(Correct[I]) + 1
    14.6. Если в аргументе Instruct установлен бит Estimate, то
        14.6.1. Если в аргументе Instruct не установлен бит Interpret, то генерируется запрос к задачнику Get с аргументами Handle, AnsArray, tbAnswers (Получает от задачника правильные ответы)

14.6.2. Генерируется запрос к задачнику Get с аргументами Handle, RelArray, tbCalcReliability (Получает от задачника достоверности ответов)
14.6.3. Генерируется запрос к оценке Estimate с аргументами Data, Back, AnsArray, RelArray, Direv, Estim. Вместо Direv передается ноль, если в аргументе Instruct установлен бит Gradient, и 1 в противном случае. (Вычисляет оценку примера и, возможно, производные)
14.6.4. Генерируется запрос к задачнику Get с аргументами Handle, Weight, tbWeight (Получает от задачника вес примера)
14.6.5. Work = Work + Estim * Weight (Подсчитываем суммарную оценку)
14.6.6. Если в аргументе Instruct установлен бит PutEstimations, то генерируется запрос к задачнику Put с аргументами Handle, Estim, tbEstimations (Передает задачнику оценку примера)

14.7. Если в аргументе Instruct установлен бит Gradient, то генерируется запрос к сети Back, с аргументами Null, Back. (Выполняется обратное функционирование сети)
14.8. Если в аргументе Instruct установлен бит Contrast, то генерируется запрос к контрастеру ContrastExample с аргументом ложь.
14.9. Генерируется запрос к задачнику Next с аргументом Handle. (Переход к следующему примеру)
14.10. Переход к шагу 13 алгоритма.

15. Вычисляем среднюю оценку: If Tasks = 0 Then Estim = 0 Else Estim = Work / Task
16. Если в аргументе Instruct установлен бит Contrast, то генерируется запрос к контрастеру ContrastExample с аргументом истина.
17. Освобождаются массивы Data, AnsArray и InArray.
18. Если в аргументе Instruct установлен бит Estimate, то освобождается массив и RelArray.
19. Если в аргументе Instruct установлен бит Interpret, то освобождаются массивы Answers и Reliability.
20. Если Back <> Null освобождается массив Back.
21. Завершает исполнение, возвращая значение истина

7.2.3 Ошибки компонента исполнитель

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

Таблица 4.

Ошибки компонента исполнитель и действия стандартного обработчика ошибок.

№	Название ошибки	Стандартная обработка
001	Некорректное сочетание флагов в аргументе Instruct.	Занесение номера в Error