В.Н. Гридин, В.И. Солодовников, И.А. Евдокимов, С.В. Филипков

Построение деревьев решений и извлечение правил из обученных нейронных сетей[1]

Аннотация. Рассмотрены вопросы совместного использования нейросетевых технологий с методами логического вывода и поддержки принятия решений в задачах «интеллектуального» анализа данных. Проведен анализ существующих алгоритмов и методов построения деревьев решений.

Ключевые слова: нейронные сети; деревья решений, логический вывод, извлечение правил, «интеллектуальный» анализ данных.

Введение

Совершенствование технологий записи и рост объемов хранимых данных в самых различных областях человеческой деятельности предъявляет новые требования к системам обработки информации. Все большую актуальность приобретают системы, способные не просто складировать данные, но и позволяющие осуществлять их анализ, выявлять закономерности, вырабатывать стратегии принятия решений, производить прогнозирование и выдавать результат в доступном человеку виде. Особенностью алгоритмов и методов, применимых для решения задач интеллектуального анализа данных, является отсутствие ограничительных рамок априорных предположений о структуре выборки и виде распределений значений анализируемых показателей, чему наилучшим образом соответствует использование нейросетевых технологий. Это обусловлено способностью нейронных сетей к моделированию нелинейных процессов, работе с зашумленными данными, адаптивностью (обучение и самообучение), способностью обобщать и извлекать существенные особенности из поступающей информации. Однако данный подход не лишен и ряда недостатков. Например, в случае использования сетей класса многослойный персептрон возможно возникновение проблем с интерпретацией полученного результата и его предпосылок. Нейросеть, по сути, выступает «черным ящиком», на вход которого подаются исходные данные и на выходе получается некоторый результат, однако обоснования, почему было принято именно такое решение, не предоставляется. Правила содержатся в весовых коэффициентах, функциях активации и связях между нейронами, но обычно их структура слишком сложна для восприятия. Более того, в многослойной сети эти параметры могут представлять собой нелинейные, немонотонные отношения между входными и целевыми значениями. Таким образом, как правило, не представляется возможным отделить влияние определенного признака на целевое значение, потому что этот эффект может быть опосредован значениями других параметров. Другой сложностью является проблема выбора оптимальной топологии сети, значений параметров и структурных особенностей, которые бы наилучшим образом удовлетворяли решаемой задаче на имеющихся исходных данных. В связи с этим, особую важность приобретают вопросы совместного использования нейросетевых технологий с методами логического вывода и поддержки принятия решений, где в качестве основного подхода применяются деревья решений.

[1] Работа выполнена при поддержке Совета по грантам Президента РФ № МК-3702.2011.9.

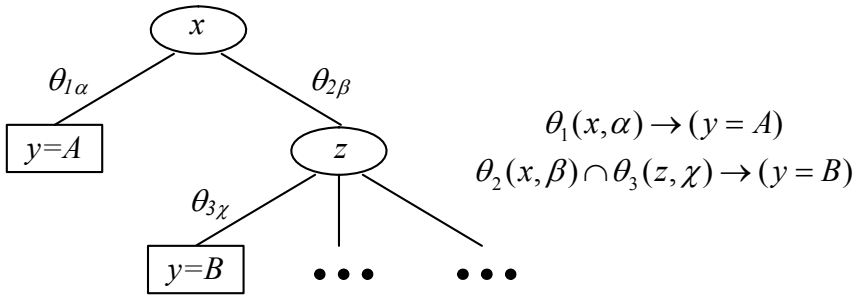


Рис. 1. Пример дерева решений и соответствующих правил вывода

1. Деревья решений

Дерево решений состоит из вершин двух типов. Вершины решений, содержащие вопросы, обозначаются окружностями. Цели или логические выводы обозначаются прямоугольниками. Вершины нумеруются и на дугах задаются условия. Каждая вершина может иметь не более одного входа. Пути движения по дереву с верхнего уровня на самые нижние определяют логические правила в виде цепочек конъюнкций.

На Рис. 1 приведен пример такого дерева решений и соответствующий логический вывод, где $\theta_1, \theta_2, \theta_3$ - предикаты, x, y, z – переменные, α, β, χ - константы.

Правила, выражающие закономерности, формулируются в виде продукций: «ЕСЛИ *A* ТО *B»* или в случае множества условий: «ЕСЛИ (условие 1) $\wedge$ (условие 2) $\wedge$ … $\wedge$ (условие N) ТО (Значение вершины вывода)». Их достоинством является простота и наглядность описания процесса поиска решения.

Построение деревьев решений обычно осуществляется на основе экспертных оценок или с использованием алгоритмов обработки примеров (CLS, ID3 - Interactive Dichotomizer, C4.5, CART - classification and regression trees и др.). Каждый из этих подходов имеет свои особенности и может использоваться для решения конкретных задач.

2. Построение деревьев решений на основе экспертных оценок

Данный подход свойственен экспертным системам, ориентированным на обработку данных с помощью некоторых правил вывода, которые предполагается извлекать у экспертов в той или иной области знаний. Такая система реализует цепочку рассуждений, имитирующую анализ ситуации экспертом-человеком, например, когда необходимо выбрать какой-либо метод для решения определенной задачи в конкретных условиях. В частности, возможно использование деревьев решений для создания средств автоматизации выбора архитектуры и вычисления параметров нейросетевых структур в зависимости от решаемой задачи и имеющихся исходных данных [6]. Это влечет за собой необходимость проведения анализа различных нейросетевых парадигм, направленного на формулировку правил построения в некотором смысле оптимальных сетей, формирование критериев их применимости для решения конкретных задач, рассмотрение вопросов предобработки, кодирования и проверки на непротиворечивость исходных данных.

3. Построение деревьев решений с использованием алгоритмов обработки примеров

Деревья решения могут использоваться как самостоятельное средство анализа многомерных данных и поиска в них логических закономерностей. Общий принцип построения дерева заключается в рекурсивном разбиении объектов из обучающей выборки на подмножества, которые содержат объекты, относящиеся к одному классу. Для обучающей выборки *T* и множества классов *C* возможны следующие три ситуации [3]:

- множество *T* содержит один или более объектов, относящихся к одному классу C_j, тогда дерево решений для *T* – это лист, определяющий этот класс;
- множество *T* не содержит ни одного объекта (пустое множество), тогда это снова лист;

класс, ассоциированный с листом, выбирается из другого множества, отличного от *T*, например, из множества ассоциированного с родителем;

- множество *T* содержит объекты, принадлежащие к разным классам. В этом случае следует разбить множество *T* на некоторые подмножества. Для этого выбирается одна из независимых переменных X, имеющая два и более различных значений $X_1,\ldots,X_l$. Множество *T* разбивается на подмножества $T_1,\ldots,T_l$, где каждое подмножество T_i содержит все объекты, имеющие значения X_i для заданного признака. Эта процедура будет продолжаться рекурсивно до тех пор, пока в конечном множестве не окажутся объекты одного класса

Общее правило для выбора переменной можно сформулировать следующим образом: выбранная переменная должна разбить множество так, чтобы получаемые в итоге подмножества состояли из объектов, принадлежащих к одному классу, или были максимально приближены к этому, т. е. чтобы количество объектов из других классов в каждом из этих подмножеств было минимально. В данном случае критерием может служить количество взаимной информации между множеством значений рассматриваемой независимой переменной и набором целевых классов. Лучшей для разделения считается переменная X, которая позволяет максимизировать информацию о классах $Gain(X)$. Ее поиск осуществляется по формуле вычисления количества информации:

$$Gain(X) = Info(T) - Info_X(T),$$

где $Info(T)$ - количество информации, необходимое для идентификации очередного примера из множества *T* и отнесения его к определенному классу из *C*, $Info_X(T)$ - аналогичное значение, но после разбиения множества *T* по *X*.

В терминах энтропии данное выражение примет вид:

$$I(T \mid X) = H(T) - H(T \mid X),$$

где $H(T)$ - энтропия множества *T*, $H(T \mid X)$ - средняя условная энтропия множества *T* при известном множестве *X*.

Значения этих величин вычисляются по формулам:

$$H(T) = -\sum_j P(C_j) \cdot \log_2 P(C_j),$$

$$H(T \mid X) = \sum_i P(X_i) \cdot H(T \mid X_i).$$

где $P(C_j)$ - вероятность того, что случайно выбранный пример из множества *T* будет принадлежать к классу C_j.

Если $|T|$ - общее число примеров множества *T*, а количество примеров из *T*, относящихся к C_j, обозначить как $freq(C_j, T)$ то

$$P(C_j) = \frac{freq(C_j, T)}{|T|}.$$

Тогда формула для вычисления количества информации для идентификации примеров из *T* примет вид:

$$Info(T) = -\sum_j \frac{freq(C_j, T)}{|T|} \cdot \log_2\left(\frac{freq(C_j, T)}{|T|}\right).$$

Рассмотрим выражение для условной энтропии $H(T \mid X)$. $P(X_i)$ - вероятность выбора примера из подмножества T_i, которое содержит объекты, имеющие значения X_i для заданного признака *X*. $H(T \mid X_i)$ - условная энтропия множества *T*, при условии, что для *X* выбрано значение X_i. Тогда для $Info_X(T)$ получаем:

$$Info_X(T) = \sum_i \frac{|T_i|}{|T|} \cdot Info(T_i)$$

Критерий $Gain(X)$ считается для всех независимых переменных X и выбирается та, которая максимизирует данное выражение. Этот признак будет являться проверкой в текущем узле дерева, и дальнейшее движение будет производиться в зависимости от полученного ответа. Такие же рассуждения можно применить к полученным подмножествам $T_1,\ldots,T_l$ и продолжить рекурсивно процесс построения дерева, до тех пор, пока в узле не окажутся примеры из одного класса.

Методы выделения закономерностей с помощью деревьев решений обладают свойством наглядности и позволяют находить такие связи,

которые заключены не только в отдельных признаках, но и в сочетании признаков. Они предоставляют возможность прогнозировать и связывать различные параметры изучаемого явления в единое целое, что во многих случаях дает этим методам значительное преимущество по сравнению с классическими методами многомерного анализа.

Однако данный подход не лишен и ряда недостатков, так как в задаче поиска логических закономерностей не способен находить наиболее полные и точные правила в данных, реализует принцип последовательного просмотра признаков и формирует лишь фрагменты закономерностей. Причем большинство таких процедур являются «жадными алгоритмами», а это значит, что если одна переменная была выбрана, и по ней было проведено разбиение на подмножества, то алгоритм не может вернуться назад и выбрать другую переменную, которая давала бы лучшее разбиение. При больших объемах многомерных данных алгоритмы построения деревьев могут выдавать очень сложную структуру, которая имеет много узлов и ветвей. Такие деревья бывает трудно анализировать, так как происходит разбиение обучающего множества на большое количество подмножеств, состоящих из малого количества объектов. Тогда как гораздо предпочтительнее иметь дерево, состоящее из малого количества узлов, которым соответствует большое количество объектов из обучающей выборки. Для решения данной проблемы часто применяются методы оценки целесообразности дальнейшего разбиения, дополнительные ограничения глубины дерева или алгоритмы отсечения ветвей, однако, они не всегда могут привести к желаемому результату.

4. Извлечение правил из обученных нейронных сетей

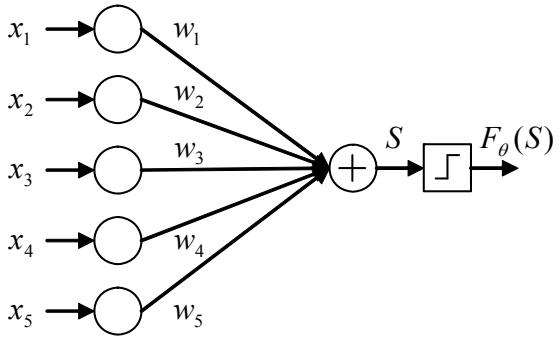
Еще одним направлением использования деревьев решений для интеллектуального анализа данных является их применение для извлечения правил из нейронных сетей. Совместное использование нейросетевых технологий с методами логического вывода способно улучшить понимание структуры изучаемого явления за счет предоставления результата, полученного в ходе обучения нейронной сети, в виде иерархической, последовательной структуры правил типа «если-то». Пусть задача состоит в классификации некоторого набора данных с помощью персептрона и последующего анализа полученной сети с целью нахождения классифицирующих правил, характеризующих каждый из классов.

Сначала рассмотрим данную задачу на примере однослойного персептрона, в котором пять булевых нейронов на входе и один на выходе (Рис.2). Данная сеть может быть в точности интерпретирована конечным числом правил «если-то», так как для нее определено конечное число возможных входных векторов.

Пусть веса принимают значения: $w_1 = 6$, $w_2 = 4$, $w_3 = 4$, $w_4 = 0$, $w_5 = -4$, а порог $\theta = 9$. В этом случае из сети можно извлечь следующий набор правил:

$$x_1 \wedge x_2 \wedge x_3 \rightarrow y; \quad x_1 \wedge x_2 \wedge \neg x_5 \rightarrow y;$$
$$x_1 \wedge x_3 \wedge \neg x_5 \rightarrow y.$$

Таким образом, процедура принятия решений заключается в предсказании значения $y = true$, если активация выходного нейрона равна 1, и $y = false$, когда активация равна 0.



$$S = \sum_i w_i \cdot x_i$$

$$y = F_\theta(S) = \begin{cases} 1 & if\ S > \theta \\ 0 & else \end{cases}$$

Рис. 2. Однослойный персептрон с пятью булевыми входами и одним выходом

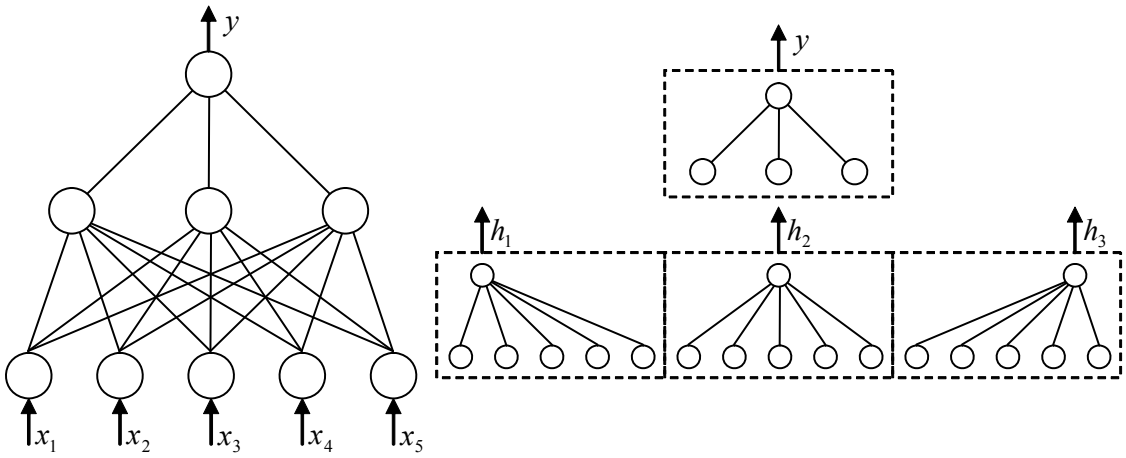


Рис. 3. Локальный подход к извлечению правил

Если для выходного нейрона вместо пороговой функции активации использовать логистическую функцию (сигмоиду), тогда решение $y = true$ будет приниматься в случае, когда значение активации нейрона превышает определенное значение, например 0,5. При решении задачи классификации возможно использование отдельного выходного нейрона для каждого класса. В этом случае решение принимается в пользу нейрона с наибольшей активацией. Таким образом, полученное правило должно характеризовать набор входных параметров, при которых обученная нейросеть в сочетании с процедурой вывода предсказывает появление определенного класса.

Вообще говоря, можно выделить два подхода к извлечению правил из многослойных нейронных сетей. Первый подход заключается в извлечении набора глобальных правил, которые характеризуют классы на выходе непосредственно через значения входных параметров. Альтернативой является извлечение локальных правил, разделяя многослойную сеть на совокупность однослойных сетей. Каждое извлекаемое локальное правило характеризует отдельный скрытый или выходной нейрон с учетом элементов, которые имеют с ним взвешенные соединения. Затем правила объединяются в набор, который определяет поведение всей сети в целом. Локальный подход проиллюстрирован Рис. 3.

Рассмотрим задачу извлечения правил в более общем виде. Пусть X обозначает набор из n свойств $X_1, X_2, ..., X_n$, а $\{x_i\}$- множество возможных значений, которое может принимать свойство X_i. Обозначим через C множество классов $c_1, c_2, \ldots, c_m$. Для обучающей выборки известны ассоциированные пары векторов входных и выходных значений $(x_1, ..., x_n, c_j)$, где $c_j \in C$.

Одним из алгоритмов извлечения правил из нейронных сетей, обученных решению задачи классификации, является метод NeuroRule [2]. Данный алгоритм основан на извлечении локальных правил и включает три основных этапа.

Этап 1. Обучение нейронной сети, когда двухслойный персептрон обучается вплоть до получения достаточной точности классификации. В первоначальный момент выбирается большое число промежуточных нейронов, и после обучения излишние нейроны и связи отбрасываются.

Этап 2. Прореживание нейронной сети. Обученная нейронная сеть содержит все возможные связи между входными нейронами и нейронами скрытого слоя, а также между скрытыми и выходными нейронами. Полное число этих связей обычно столь велико, что из анализа их значений невозможно извлечь доступные для понимания пользователем классифицирующие правила. Прореживание заключается в удалении излишних связей и нейронов, не приводящем к увеличению ошибки классификации сетью. Результирующая сеть обычно содержит немного нейронов и связей между ними, и функционирование такой сети поддается исследованию.

Этап 3. Извлечение правил. На этом этапе из прореженной нейронной сети извлекаются правила, имеющие форму *«если* $(x_1 \Theta q_1)$ *и* $(x_2 \Theta q_2)$ *и ... и* $(x_n \Theta q_n)$, *то* c_j*»*, где $q_1,...,q_n$ - константы, Θ - оператор отношения $(=, \geq, \leq, ><)$. Для этого проводят подготовку к извлечению правил, которая заключается в кодировании непрерывных величин, как на входе, так и внутри сети. Осуществляется кодирование признаков классифицируемых объектов, если они представляют собой непрерывные величины. Для их представления можно использовать бинарные нейроны и принцип кодирования типа «термометр».

Значения, которые принимают нейроны скрытого слоя кластеризуются и заменяются значениями, определяющими центры кластеров. Число таких кластеров выбирается небольшим. После такой дискретизации активностей промежуточных нейронов производится проверка точности классификации объектов сетью. Если она остается приемлемой, то подготовка к извлечению правил заканчивается. Далее осуществляется извлечение правил, при этом движение по сети происходит от классифицирующих выходных нейронов к входам сети. Предполагается, что эти правила достаточно очевидны при проверке и легко применяются к большим базам данных.

Однако данный алгоритм устанавливает довольно жесткие ограничения на архитектуру нейросети, числу элементов, связей и вид функций активации. Так для промежуточных нейронов используется гиперболический тангенс и их состояния изменяются в интервале [-1,1], а для выходных нейронов применяется функция Ферми с интервалом состояний [0,1].

К недостаткам большинства алгоритмов извлечения правил можно отнести отсутствие универсальности и масштабируемости. В связи с этим, наибольший интерес представляет алгоритм TREPAN [5] и его модификации, которые лишены этих недостатков и не предъявляют никаких требований к архитектуре сети, входным и выходным значениям, алгоритму обучения и т.д. Данный подход осуществляет построение дерева решений на основе знаний, заложенных в обученную нейросеть, причем достаточно того, что сеть является неким «черным ящиком» или «Оракулом», которому можно задавать вопросы и получать от него ответы. Более того, алгоритм является достаточно универсальным и может применяться к широкому кругу других обученных классификаторов. Он также хорошо масштабируется и не чувствителен к размерности пространства входных признаков и размеру сети.

Алгоритм построения дерева решения, аппроксимирующего работу обученной нейронной сети, состоит из двух этапов.

Предварительный этап:

1. Построить и обучить нейронную сеть, которая в дальнейшем будет выступать в роли «Оракула».

2. Инициализировать корень дерева *R* в виде листа.

3. Использовать все обучающее множество примеров *S* для конструирования модели M_R распределения входных векторов, достигающих узла *R*. Вычислить значение *q = max(0, minSamples - |S|)*, где *minSamples* – минимальное число обучающих примеров, используемое в каждом узле дерева, *S* – текущая обучающая выборка (|*S*|-объем обучающей выборки). Таким образом, *q* – количество дополнительных примеров, которые необходимо сгенерировать.

4. На основе оценки распределения признаков из *S* случайным образом генерируются *q* новых обучающих примеров, $query_R$ - множество из q примеров, генерируемых моделью M_R.

5. Использовать нейронную сеть «Оракула» для классификации, как новых $query_R$, так и старых примеров из множества *S*. Для каждого вектора признаков $x \in (S \cup query_R)$ выставить метку класса $c = Oracle(x)$.

6. Инициализировать очередь $Queue$, поместив в нее набор $\langle R, S, query_R, \{empty\_constr\} \rangle$, где $\{empty\_constr\}$ - пустое множество ограничений для корня дерева *R*.

Основной этап:

7. Взять очередной набор $\langle N, S_N, query_N, constr_N \rangle$ из начала очереди $Queue$, где N - узел дерева, S_N - обучающая выборка в узле N, $query_N$ - множество сгенерированных примеров моделью M_N, $constr_N$ - набор ограничений на определенные признаки обучающих примеров для достижения узла N.

Стоит отметить, что наборы в очереди $Queue$ должны быть упорядочены по убыва-

нию в соответствии со значением функции *F*, которая имеет вид:

$$F(N) = R(N) \cdot (1 - f(N)),$$

где $R(N)$ - доля примеров, которые достигают узла *N*, а $f(N)$ - оценка вероятности правильной обработки этих примеров деревом.

Таким образом, выбирается наилучший узел, разветвление которого оказывает наибольшее влияние на точность классификации генерируемого дерева.

8. Использовать $S_N \cup query_N$ для конструирования в узле N разветвления T. Разделение множества примеров, достигающих данный внутренний узел дерева, осуществляется в зависимости от $m - of - n$ теста [4,5]. Такой тест считается пройденным, когда выполняются, по меньшей мере, m из n условий. С другой стороны, возможно расщепление множества $S_N \cup query_N$ тем же способом, что и в приведенном выше алгоритме построения деревьев решений.

9. Для каждой дуги *t* разветвления T создать узлы следующего поколения:

- Создать D - новый дочерний узел по отношению к N.

- $constr_D = constr_N \cup \{T = t\}$ - добавить во множество ограничений для узла D дополнительное ограничение с дуги *t*.

- Сформировать множество S_D, в которое войдут примеры из S_N, удовлетворяющие условию на дуге *t*.

- Сконструировать модель M_D распределения примеров, достигающих узла D.

Подсчитать значение $q = max(0, minSamples - |S_D|)$, т.е. количество примеров, которые необходимо сгенерировать.

- На основе оценки распределения признаков из S_D и значения ограничений $constr_D$, случайным образом, сгенерировать q новых обучающих примеров, $query_D$ – множество из q примеров, сгенерированных моделью M_D и ограничением $constr_D$.

- Использовать нейронную сеть «Оракула» для классификации новых примеров $x \in query_D$ и выставить метку класса $c = Oracle(x)$.

- Изначально предполагается, что узел D является листом. Использовать S_D и $query_D$ для определения метки класса для узла D.

- Проверить необходимость дальнейшего расщепления узла D. Если локальный критерий остановки не удовлетворен, то поместить набор $< D, S_D, query_D, constr_D >$ в очередь $Queue$ с учетом значения функции $F(D)$. Локальным критерием в данном случае выступает величина, которая характеризует вероятность, что в данном узле встречаются экземпляры одного класса.

10. Если очередь $Queue$ не пуста и не выполнен глобальный критерий остановки, то перейти к шагу 7, иначе вернуть дерево с корнем R.

В качестве глобального критерия завершения алгоритма могут быть использованы максимальный размер дерева и общая оценка качества классификации примеров деревом.

Основное преимуществом данного подхода заключается в обобщающей способности искусственных нейронных сетей, что позволяет получать более простые деревья решений. Возможность генерации дополнительных примеров и использование «Оракула» для отнесения их к тому или иному классу позволяет компенсировать недостаток данных, часто наблюдающийся при построении деревьев на нижних уровнях. Таким образом, алгоритм позволяет извлекать структурированных знаний не только из чрезвычайно упрощенных нейронных сетей, но и из произвольных классификаторов, что делает возможным его применение в широком круге практических задач.

Заключение

Внедрение средств автоматизации в системы интеллектуального анализа данных способно сократить сроки, повысить качество и эффективность принимаемых решений. Совместное использование нейросетевых технологий и деревьев решений позволяет улучшить понимание структуры изучаемого явления за счет предоставления результата, полученного в ходе обучения сети, в виде иерархической, последовательной структуры правил типа «если-то». Представленные алгоритмы могут использоваться для анализа данных в информационных хранилищах с целью выявления в них скрытых

закономерностей, на основе которых можно осуществлять построение систем правил и деревьев решений.

Литература

1. Рассел С., Норвиг П. Искусственный интеллект: современный подход. М.: Издательский дом «Вильямс», 2006.
2. Ежов А., Шумский С., Нейрокомпьютинг и его применение в экономике и бизнесе, 1998.
3. Сайт компании BaseGroup Labs 1995-2012 // Деревья решений [Электронный ресурс]. - Режим доступа: http://www.basegroup.ru/library/analysis/tree/.
4. Murphy P.M., Pazzani M.J. ID2-of-3: Constructive induction of M-of-N concepts for discriminators in decision trees. In Proceedings of the Eighth International Machine Learning Workshop, Evanston, IL. Morgan Kaufmann.1991.Pp.183-187.
5. Craven M.W., Shavlik J.W. Extracting tree-structured representations of trained networks. In Touretzky, D., Mozer, M., & Hasselmo, M., editors, Advances in Neural Information Processing Systems (volume 8). MIT Press, Cambridge MA.1996.
6. Солодовников И.В., Солодовников В.И. Подход к созданию подсистемы автоматизации проектирования нейросетевых структур обработки данных с использованием деревьев решений // Информационные технологии в проектировании и производстве. 2006. № 2.С. 62-66.

Гридин Владимир Николаевич. Директор Центра информационных технологий в проектировании РАН. Окончил Московский авиационный институт в 1972 году. Доктор технических наук, профессор. Автор 244 печатных работ и девяти монографий. Область научных интересов: информационные технологии в автоматизации проектирования, методы искусственного интеллекта. E-mail: info@ditc.ras.ru

Солодовников Владимир Игоревич. Заведующий лабораторией Центра информационных технологий в проектировании РАН. Окончил Московский государственный институт электроники и математики (Технический университет) в 2000 году. Кандидат технических наук. Автор 32 печатных работ и монографий. Область научных интересов: методы искусственного интеллекта, самоорганизующиеся системы поддержки принятия решений, нейросетевой анализ данных. E-mail: info@ditc.ras.ru

Евдокимов Иван Александрович. Младший научный сотрудник Центра информационных технологий в проектировании РАН. Окончил Московский государственный технический университет им. Н.Э. Баумана в 2009 году. Автор 11 печатных работ. Область научных интересов: методы искусственного интеллекта, нейронные сети, методы криптографической защиты информации. E-mail: info@ditc.ras.ru

Филипков Сергей Владимирович. Инженер-исследователь Центра информационных технологий в проектировании РАН. Окончил Московский государственный технический университет им. Н.Э. Баумана в 2009 году. Автор четырех печатных работ. Область научных интересов: методы искусственного интеллекта, методы защиты информации.
E-mail: info@ditc.ras.ru