

Извлечение знаний с помощью нейронных сетей

Искусственный интеллект, экспертные системы и нейронные сети. Извлечение правил из нейронных сетей. Алгоритм NeuroRule. Прореживание нейронных сетей. Обучение нейронных сетей с одновременным исправлением данных. Алгоритм TREPAN для извлечения деревьев решений с использованием нейронных сетей.

 Ковбойский язык очень легко понять, - заметил один ковбой. - Нужно просто заранее знать, что хочет сказать твой собеседник, и не обращать внимание на его слова.

Д.Бурстин, Американцы: демократический опыт

До сих пор нейросети рассматривались нами лишь как инструмент предсказания, но не понимания. Действительно, классический нейросетевой подход - метод черного ящика - предполагает создание *имитационной модели*, без явной формулировки правил принятия решений нейросетью. Вернее, эти правила содержатся в весах обученной нейросети, но понять их, переформулировав на язык “**если ... - то ...**” не представлялось возможным. В этой главе мы продемонстрируем методику, позволяющую строить подобные правила, *объясняющие* нейросетевые решения. Нейросети, таким образом, можно использовать не только для предсказаний, но и для *извлечения знаний из баз данных*.

Традиционно построение правил вывода и баз знаний считается прерогативой *экспертных систем* - направления *искусственного интеллекта*, которое претендовало в начале семидесятых годов заменить собою искусственные нейронные сети в задачах обработки информации. Экспертные системы были ориентированы именно на обработку данных с помощью некоторых правил вывода, которые предполагалось извлекать у экспертов в той или иной области знаний. Экспертные системы были призваны реализовывать цепочки рассуждений, имитирующих анализ ситуации экспертом-человеком. По сути в 70-е годы сам термин “искусственный интеллект” был синонимом разработки экспертных систем, или *инженерии знаний*.

Это направление, однако, столкнулось с рядом принципиальных трудностей. В частности, инженеры знаний должны были извлекать их у *очень* квалифицированных экспертов, которые, вообще говоря, не стремились поделиться информацией. Знания - большая ценность, и передавать их, чтобы помочь создать себе легко тиражируемую замену и, в конечном счете, обесценить себя как специалиста, стремился далеко не каждый. Но даже и при наличии соответствующего желания, эксперт не всегда мог внятно сформулировать те правила, которыми он пользуется при подготовке экспертного заключения. Очень многое в его работе связано с интуитивными качественными оценками, распознаванием ситуации в целом, то есть с не формализуемыми процедурами (мы знаем, что это как раз та ситуация, в которой особенно отчетливо проявляются преимущества нейросетевого подхода). Но даже если все трудности оказывались преодоленными, достоинства построенной экспертной системы оказывались не абсолютными, поскольку именно явная формализация правил вывода, а не компьютерная система сама по себе представляла основную ценность. В этом смысле весьма показателен

опыт создания в 70-е годы в Стэнфордском университете экспертной системы **MYCIN**, с помощью которой врачи должны были повысить надежность диагностики септического шока. Септический шок, дававший в случае развития 50% летальных исходов у прооперированных больных вовремя диагностировался врачами лишь в половине случаев. Экспертная система MYCIN позволила повысить качество диагностики почти до 100%. Однако, после того, как врачи познакомились с ее работой, они очень быстро сами научились правильно ставить соответствующий диагноз. Необходимость в MYCIN отпала и она превратилась в учебную систему. Таким образом, основная польза проекта состояла именно в *извлечении* знаний в понятном для человека виде.

По мнению Стаббса, известного американского специалиста в области нейросетевых приложений, экспертные системы “пошли” только в кардиологии. Они эффективно заменили объемистые руководства по анализу электрокардиограмм, содержащие множество достаточно ясно сформулированных правил оценки их многообразных особенностей.

Нейронные сети выглядят предпочтительнее экспертных систем, позволяя одновременно анализировать множество в общем случае неточных и неполных параметров и не требуя при этом явной формализации правил вывода. Однако, *объяснение* тех или иных рекомендаций, полученных с помощью нейросетевого анализа, является требованием, которое обычно предъявляют специалисты, желающие использовать нейросетевые технологии. На первый взгляд здесь-то и находится их слабое место. Действительно, в такой области обработки информации, как извлечение знаний, нейронные сети стали применяться только относительно недавно. Это еще одна сфера, в которой доселе господствовал только традиционный искусственный интеллект. Рассмотрим ее более подробно.

Извлечение знаний

В последние годы созданы огромные базы данных, в которых хранится информация научного, экономического, делового и политического характера. В качестве примера можно привести **GenBank**, содержащий терабайты данных о последовательностях ДНК живых организмов. Для работы с подобными базами разработаны компьютерные технологии, позволяющие хранить, сортировать и визуализировать данные, осуществлять быстрый доступ к ним, осуществлять их статистическую обработку. Значительно меньшими являются, однако, достижения в разработке методов и программ, способных обнаружить в данных важную, но скрытую информацию. Можно сказать, что информация находится к данным в таком же отношении, как чистое золото к бедной золотоносной руде. Извлечение этой информации может дать критический толчок в бизнесе, в научных исследованиях и других областях. Подобное *нетривиальное* извлечение *неявной, прежде неизвестной и потенциально полезной информации* из больших баз данных и называется Разработкой Данных (**Data Mining**) или же Открытием Знаний (**Knowledge Discovery**). Мы будем использовать далее для описания этой области информатики более явный синтетический термин - извлечение знаний. Извлечение знаний использует концепции, разработанные в таких областях как машинное обучение (Machine Learning), технология баз данных (Database Technology), статистика и других.

Главными требованиями, предъявляемыми к методам извлечения знаний, являются эффективность и масштабируемость. Работа с очень большими базами данных требует эффективности алгоритмов, а неточность и, зачастую, неполнота данных порождают дополнительные проблемы для извлечения знаний. Нейронные сети имеют здесь неоспоримое преимущество, поскольку именно они являются наиболее эффективным средством работы с зашумленными данными. Действительно, заполнение пропусков в базах данных - одна из прототипических задач, решаемых нейросетями. Однако, главной претензией к нейронным сетям всегда было отсутствие объяснения. Демонстрация того, что нейронные сети

действительно можно использовать для получения наглядно сформулированных правил было важным событием конца 80-х годов. В 1989 году один из авторов настоящего курса поинтересовался у Роберта Хехт-Нильсена, главы одной из наиболее известных американских нейрокомпьютерных фирм **Hecht-Nielsen Neurocomputers**, где можно узнать подробности о нейроэкспертных системах, информация о которых тогда носила только рекламный характер. Хехт-Нильсен ответил в том смысле, что она не доступна. Но уже через 2-3 месяца после этого в журнале Artificial Intelligence Expert была опубликована информация о том, что после долгих и трудных переговоров Хехт-Нильсен и крупнейший авторитет в области экспертных систем Гэллант запатентовали метод извлечения правил из обученных нейронных сетей и метод автоматической нейросетевой генерации экспертных систем.

Извлечение правил из нейронных сетей подразумевает их предварительное обучение. Поскольку эта процедура требует много времени для больших баз данных, то естественна та критика, которой подвергается использование нейротехнологии для извлечения знаний. Другим поводом для такой критики является трудность инкорпорации в нейронные сети некоторых имеющихся априорных знаний. Тем не менее, главным является артикуляция правил на основе анализа структуры нейронной сети. Если эта задача решается, то низкая ошибка классификации и робастность нейронных сетей дают им преимущества перед другими методами извлечения знаний.

Извлечение правил из нейронных сетей

Рассмотрим один из методов извлечения правил из нейронных сетей, обученных решению задачи классификации (Lu, Setiono and Liu, 1995). Этот метод носит название **NeuroRule**.

Задача состоит в классификации некоторого набора данных с помощью многослойного персептрона и последующего анализа полученной сети с целью нахождения *классифицирующих правил*, описывающих каждый из классов.

Пусть A обозначает набор из N свойств $A_1, A_2, \dots, A_N$, а $\{a_i\}$ - множество возможных значений, которое может принимать свойство A_i . Обозначим через C множество классов $c_1, c_2, \dots, c_m$. Для обучающей выборки известны ассоциированные пары векторов входных и выходных значений $(a_1, \dots, a_m, c_k)$, где $c_k \in C$.

Алгоритм извлечения классифицирующих правил включает три этапа:

1. **Обучение нейронной сети.** На этом первом шаге двухслойный персептрон тренируется на обучающем наборе вплоть до получения достаточной точности классификации.
2. **Прореживание (pruning) нейронной сети.** Обученная нейронная сеть содержит все возможные связи между входными нейронами и нейронами скрытого слоя, а также между последними и выходными нейронами. Полное число этих связей обычно столь велико, что из анализа их значений невозможно извлечь обозримые для пользователя 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 ее функционирование поддается исследованию.
3. **Извлечение правил.** На этом этапе из прореженной нейронной сети извлекаются правила, имеющие форму *если $(a_1 \Theta q_1)$ и $(a_2 \Theta q_2)$ и ... и $(a_N \Theta q_N)$, то c_j* , где $q_1, \dots, q_N$ -

константы, Θ - оператор отношения ($=, \geq, \leq, >, <$). Предполагается, что эти правила достаточно очевидны при проверке и легко применяются к большим базам данных.

Рассмотрим все эти шаги более подробно

Обучение нейронной сети

Предположим, что обучающий набор данных необходимо расклассифицировать на два класса A и B . В этом случае сеть должна содержать N входных и 2 выходных нейрона. Каждому из классов будут соответствовать следующие активности выходных нейронов (1,0) и (0,1). Подходящее количество нейронов в промежуточном слое, вообще говоря, невозможно определить заранее - слишком большое их число ведет к переобучению, в то время как малое не обеспечивает достаточной точности обучения. Тем не менее, как уже отмечалось ранее, все методы адаптивного поиска числа нейронов в промежуточном слое делятся на два класса, в соответствии с тем, с малого или большого числа промежуточных нейронов стартует алгоритм. В первом случае по мере обучения в сеть добавляются дополнительные нейроны, в противоположном - после обучения происходит уничтожение излишних нейронов и связей. **NeuroRule** использует последний подход, так что число промежуточных нейронов выбирается достаточно большим. Заметим, что **NeuroRule** уничтожает также и избыточные входные нейроны, влияние которых на классификацию мало.

В качестве функции активации промежуточных нейронов используется гиперболический тангенс, так что их состояния изменяются в интервале $[-1, 1]$. В то же время, функцией активации выходных нейронов является функция Ферми (состояния в интервале $[0, 1]$). Обозначим через $o^k, (i = 1, 2)$ - состояния выходных нейронов при предъявлении на вход сети вектора признаков k -го объекта $\mathbf{x}^k$. Будем считать, что этот объект правильно классифицирован сетью, если

$$\max_i |o_i^k - t_i^k| \leq \eta_1,$$

где: $t_1^k = 1$ если $\mathbf{x}^k \in A$ и $t_2^k = 1$ если $\mathbf{x}^k \in B$, а $0 < \eta_1 < 0.5$. В остальных случаях $t^k = 0$.

Минимизируемая функция ошибки должна не только направлять процесс обучения в сторону правильной классификации всех объектов обучающей выборки, но и делать малыми значения многих связей в сети, чтобы облегчить процесс их прореживания. Подобную технологию - путем добавления к функции ошибки специально подобранных штрафных членов - мы уже разбирали в Главе 3. В методе **NeuroRule** функция ошибки включает два слагаемых

$$E = E_0 + \varepsilon E_1,$$

где

$$E_0 = - \sum_k \sum_i (t_i^k \log o_i^k + (1 - t_i^k) \log(1 - o_i^k))$$

функция взаимной энтропии, минимизация которой происходит быстрее, чем минимизация среднеквадратичной ошибки. Штрафная функция

$$E_1 = \sum_{j=1}^N \sum_{l=1}^{N_h} \frac{(w_{lj}^h)^2}{1 + (w_{lj}^h)^2} + \sum_{l=1}^{N_h} \sum_{i=1}^2 \frac{(w_{il}^o)^2}{1 + (w_{il}^o)^2}$$

уже фигурировала в Главе 3.

Здесь N_h - число нейронов в скрытом слое, w_{lj}^h - величина связи, между j -м входным и l -м скрытым нейронами w_{il}^o - вес связи между l -м скрытым и i -м выходным нейронами.

Использование регуляризующего члена E_1 приводит к дифференциации весов по величинам, уменьшая большинство, но сохраняя значения некоторых из них. Обучение сети производится методом обратного распространения ошибки.

Прореживание нейронной сети

Полное число связей в обученной сети составляет $(N + N_o)N_h$. Можно показать, что связь между входным и промежуточным нейроном w_{lj}^h можно удалить без снижения точности классификации сетью при выполнении условий $\max_i |w_{i,l}^o w_{l,j}^h| \leq 4\eta_2$ и $\eta_1 + \eta_2 < 0.5$.

Аналогичным образом, удаление связи $w_{i,l}^o$ не влияет на качество классификации если $|w_{i,l}^o| \leq 4\eta_2$.

Извлечение правил

Даже если параметры, описывающие признаки классифицируемых объектов, представляют собой непрерывные величины, для их представления можно использовать бинарные нейроны и принцип кодирования типа *термометра*. При таком способе кодирования область изменения параметра делится на конечное число M интервалов и для представления всех значений, лежащих в m -м интервале используется следующее состояние M бинарных нейронов: $(0, \dots, \underbrace{1, \dots, 1}_m)$.

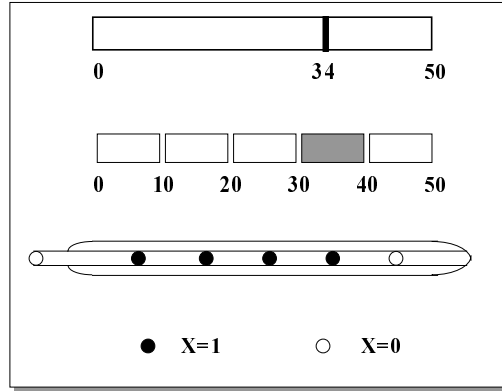


Рисунок 1. Пример кодировки непрерывной величины с помощью бинарных нейронов и принципа термометра. Интервал (0, 50) разбит на 5 равных частей. Значение 34.0 попадает в 4-й интервал. При этом состояния первых 4 из 5 кодирующих бинарных нейронов равно единице, а 5-го - нулю.

При наличии многих непрерывных входов число заменяющих их бинарных нейронов может стать весьма большим. Однако, прореживание связей приводит к получению относительно компактной сети. Но и для нее выделение классификационных правил представляет проблему. Если нейрон имеет d входов, то число различных бинарных векторов, которые он может обработать составляет 2^d , а это большая величина даже при малом d . Далее, состояния нейрона скрытого слоя являются непрерывными, что также является препятствием для извлечения правил. Для его устранения все 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. Приведем формальное описание алгоритма дискретизации значений активности нейронов скрытого слоя

Алгоритм дискретизации

1. Выбирается значение параметра $\varepsilon \in (0, 1)$, управляющего числом кластеров активности нейрона скрытого слоя. Пусть h_1 - активность этого нейрона при предъявлении сети первого вектора обучающего набора. Положим число кластеров $N_{clust} = 1$, положение кластера $A_{clust}(1) = h_1$, $count(1)=1$, $sum(1)= h_1$.
2. Для всех векторов обучающего набора $k = 1, \dots, K$
 - определяется активность нейрона скрытого слоя h
 - **если** существует индекс $\bar{j}$ такой что $|h - A_{clust}(\bar{j})| = \min_{j \in \{1, \dots, N_{clust}\}} |h - A_{clust}(j)|$ и $|h - A_{clust}(\bar{j})| \leq \varepsilon$, **то** $count(\bar{j}) := count(\bar{j}) + 1$, $sum(N_{clust}) := sum(N_{clust}) + h$
иначе $N_{clust} = N_{clust} + 1$, $A_{clust}(N_{clust}) = h$, $count(N_{clust}) = 1$, $sum(N_{clust}) = h$.

3. Заменить A_{clust} на среднее значение активаций нейрона, объединенных в один и тот же кластер: $A_{clust}(j) := \text{sum}(j) / \text{count}(j)$, $j = 1, \dots, N_{clust}$.
4. Проверить точность классификации объектов сетью при замене истинных значений активации нейрона скрытого слоя на $A_{clust}(j)$.
5. Если точность классификации оказалась ниже заданного значения, то уменьшить значение ε и вернуться к шагу 1.

Рассмотрим приведенный в (Lu, Setiono and Liu, 1995) пример, в котором прореженная сеть содержала три нейрона скрытого слоя, дискретизация активности которых была проведена при значении параметра $\varepsilon = 0.6$. Ее результаты отражены в Таблице 1.

Таблица 1. Дискретизация состояний нейронов скрытого слоя

нейрон скрытого слоя	число кластеров	дискретное значение активности
1	3	(-1, 0, 1)
2	2	(0, 1)
3	3	(-1, 0.24, 1)

В этой работе решалась задача разбиения объектов на два класса. На ее примере мы и рассмотрим последовательность извлечения правил. После дискретизации значений активности нейронов скрытого слоя, передача их воздействий выходным классифицирующим нейронам описывалась параметрами, приведенными в Таблице 2.

Таблица 2. Связь дискретных значений активности нейронов скрытого (h_i) и выходного (o_i) слоев.

h_1	h_2	h_3	o	o_2
-1	1	-1	0.92	0.08
-1	1	1	0.00	1.00
-1	1	0.24	0.01	0.99
-1	0	-1	1.00	0.00

-1	0	1	0.11	0.89
-1	0	0.24	0.93	0.07
1	1	-1	0.00	1.00
1	1	1	0.00	1.00
1	1	0.24	0.00	1.00
1	0	-1	0.89	0.11
1	0	1	0.00	1.00
1	0	0.24	0.00	1.00
0	1	-1	0.18	0.82
0	1	1	0.00	1.00
0	1	0.24	0.00	1.00
0	0	-1	1.00	0.00
0	0	1	0.00	1.00
0	0	0.24	0.18	0.82

Исходя из значений, приведенных в этой таблице, после замены значений выходных нейронов ближайшими к ним нулями или единицами, легко получить следующие правила, связывающие активности нейронов скрытого слоя с активностями классифицирующих нейронов

- правило 1 **если** $h_2 = 0, h_3 = -1$, **то** $o_1 = 1, o_2 = 0$ (объект класса А)
- правило 2 **если** $h_1 = -1, h_2 = 1, h_3 = -1$, **то** $o_1 = 1, o_2 = 0$ (объект класса А)
- правило 3 **если** $h_1 = -1, h_2 = 0, h_3 = 0.24$, **то** $o_1 = 1, o_2 = 0$ (объект класса А)
- правило 4 **в остальных случаях** $o_1 = 0, o_2 = 1$ (объект класса В)

Эти правила являются вспомогательными, поскольку нам необходимо связать значения состояний классифицирующих выходных нейронов со входами нейронной сети. Структура данной сети после прореживания связей и нейронов изображена на следующем рисунке.

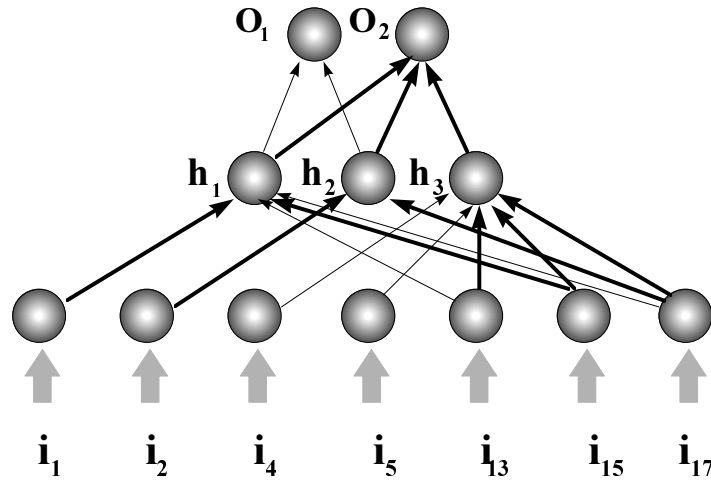


Рисунок 2. Двухслойная сеть после прореживания связей и входных нейронов. Положительные связи выделены.

Связь между активностями входных бинарных нейронов и нейронов скрытого слоя для данной сети определяется следующими правилами:

Для первого нейрона скрытого слоя: $i_{13} = 1 \Rightarrow h_1 = -1$

$$i_1 = i_{13} = i_{15} = 0, i_{17} = 1 \Rightarrow h_1 = -1$$

Для второго нейрона скрытого слоя: $i_2 = 1 \Rightarrow h_2 = 1$

$$i_{17} = 1 \Rightarrow h_2 = 1$$

$$i_2 = i_{17} = 0 \Rightarrow h_2 = 0$$

Для третьего нейрона скрытого слоя: $i_{13} = 0 \Rightarrow h_3 = -1$

$$i_5 = i_{15} = 1 \Rightarrow h_3 = -1$$

$$i_4 = i_{13} = 1, i_{17} = 0 \Rightarrow h_3 = 0.24$$

$$i_5 = 0, i_{13} = i_{15} = 1 \Rightarrow h_3 = 0.24$$

Комбинируя эти связи с правилами, связывающими активности нейронов скрытого слоя с активностями выходных нейронов, получим окончательные классифицирующие правила.

$$i_2 = i_{13} = i_{17} = 0 \Rightarrow o_1 = 1, o_2 = 0$$

$$i_2 = i_{17} = 0, i_5 = i_{15} = 1 \Rightarrow o_1 = 1, o_2 = 0$$

$$i_5 = i_{13} = i_{15} = 1 \Rightarrow o_1 = 1, o_2 = 0$$

$$i_1 = i_{13} = i_{15} = 0, i_{17} = 1 \Rightarrow o_1 = 1, o_2 = 0$$

$$i_2 = i_{17} = 0, i_4 = i_{13} = 1 \Rightarrow o_1 = 1, o_2 = 0$$

Приведенные выше правила определяют принадлежность объекта первому классу (А). Некоторые из них могут оказаться нереализуемыми, если учесть, что состояния бинарных нейронов кодируют соответствующие непрерывные величины с помощью принципа термометра.

Количество правил, полученных в данном случае, невелико. Однако, иногда даже после процедуры прореживания некоторые нейроны скрытого слоя могут иметь слишком много связей с входными нейронами. В этом случае извлечение правил становится нетривиальным, а если оно и осуществлено, то полученные правила не так просто понять. Для выхода из этой ситуации для каждого из “проблемных” нейронов скрытого слоя можно использовать вспомогательные двухслойные нейронные сети. Во вспомогательной сети количество выходных нейронов равно числу дискретных значений соответствующего “проблемного” нейрона скрытого слоя, а входными нейронами являются те, которые в исходной прореженной сети связаны с данным нейроном скрытого слоя.

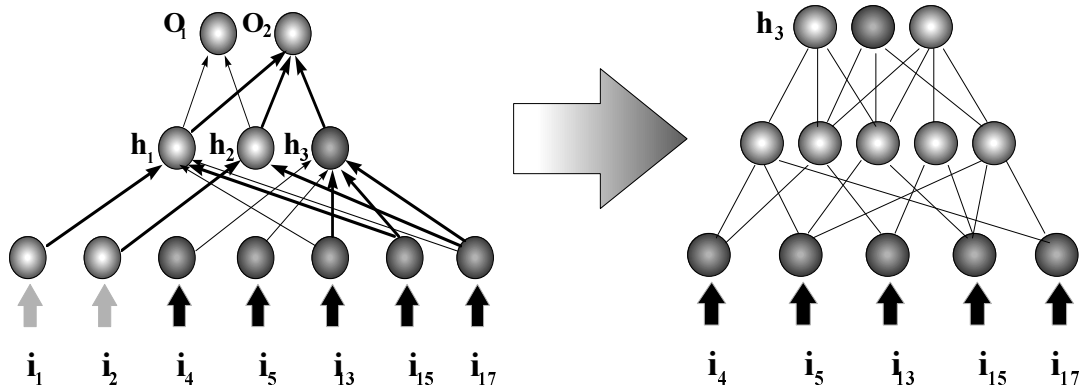


Рисунок 3. Третий нейрон скрытого слоя (h_3) связан с максимальным числом входов. Число дискретных значений его активности равно 3. Для облегчения процедуры выделения классифицирующих правил этот нейрон может быть заменен вспомогательной сетью с тремя выходными нейронами, кодирующими дискретные значения активности.

Обучающие примеры для вспомогательной сети группируются согласно их дискретизованным значениям активации “проблемного” нейрона. Для d дискретных значений $D_1, D_2, \dots, D_d$ всем обучающим примерам, соответствующим уровню активации D_j , ставится в соответствие d -мерный целевой вектор, состоящий из нулей и одной единицы в j -й позиции. Вспомогательная сеть содержит свой слой скрытых нейронов. Она обучается и прореживается тем же способом, что и основная нейронная сеть. Метод извлечения правил применяется к каждой вспомогательной сети, для того чтобы связать значения входов с дискретными значениями активации проблемных нейронов скрытого слоя оригинальной сети. Подобный процесс осуществляется рекурсивно для всех скрытых нейронов с большим числом входов до тех пор пока это число не станет достаточно малым или же новая вспомогательная сеть уже не сможет быть далее упрощена.

Исправление данных

Итак, перед извлечением правил из нейронной сети производится ее обучение и прореживание. Упомянем еще об одной процедуре, которая иногда осуществляется при извлечении знаний из нейронных сетей - *исправление (очищении)*. Подобная операция была предложена Вайгендом и коллегами и по сути используется параллельно с обучением (Weigend, Zimmermann, & Neuneier 1996). Гибридное использование обучения и исправления данных носит название **CLEARNING** (CLEARING+LEARNING). Данная процедура включает восходящий процесс обучения, при котором *данные изменяют связи в нейронной сети* и нисходящий процесс, в котором *нейронная сеть изменяет данные, на которых производится обучение*. Ее достоинствами являются выявление и удаление информационных записей, выпадающих из общей структуры обучающей выборки, а также замена искаженных данных и данных с лагунами на исправленные величины. При использовании данной процедуры происходит торг между доверием к данным и доверием к нейросетевой модели, обучаемой на этих данных. Эта конкуренция составляет существо так называемой *дилеммы наблюдателя и наблюдаемых*.

Способность работать с неточными данными является одним из главных достоинств нейронных сетей. Но она же парадоксальным образом является и их недостатком. Действительно, если данные не точны, то сеть в силу своей гибкости и адаптируемости будет подстраиваться к ним, ухудшая свои свойства обобщения. Эта ситуация особенно важна при работе с финансовыми данными. В последнем случае существует множество источников погрешности. Это и ошибки при вводе числовых значений или неправильная оценка времени действия ценных бумаг (например, они уже не продаются). Кроме того, если даже данные и введены правильно, они могут быть слабыми индикаторами основополагающих экономических процессов, таких как промышленное производство или занятость. Наконец, возможно, что многие важные параметры не учитываются при обучении сети, что эффективно может рассматриваться как введение дополнительного шума. Данные, далеко выпадающие из общей тенденции, забирают ресурсы нейронной сети. Некоторые из нейронов скрытого слоя могут настраиваться на них. При этом ресурсов для описания регулярных слабо зашумленных областей может и не хватить. Множество попыток применения нейронных сетей к решению финансовых задач выявило важное обстоятельство: *контроль гибкости нейросетевой модели является центральной проблемой*. Изложим кратко существо процедуры обучения сети, объединенной с исправлением данных. Для простоты рассмотрим сеть с одним входом и одним выходом. В этом случае минимизируемой величиной является сумма двух слагаемых (Weigend & Zimmermann, 1996):

$$E = \frac{1}{2} \eta (y - y^d)^2 + \frac{1}{2} k (x - x^d)^2.$$

Первый член описывает обычно минимизируемое в методе обратного распространения ошибки квадратичное отклонение выхода нейронной сети $y = y(x, \mathbf{w})$ от желаемого значения y^d . Второе слагаемое представляет собой квадратичное отклонение исправленного входного значения x от реального его значения x^d . Соответственно, для весов сети $\mathbf{w}$ и для исправленных входных значений x получаются два правила их модификации. Для весов оно такое же, как и в стандартном методе обратного распространения ошибки, а для исправленного входа имеет вид

$$x_{i+1} = x_i - \frac{\partial E}{\partial x},$$

где индекс i определяет номер итерации данного входа. Представляя x_i в виде суммы подлинного начального входного значения x^d и поправки Δ_i , получим для последней следующее уравнение итерационного изменения

$$\Delta_{i+1} = (1 - k)\Delta_i - \eta(y - y^d) \frac{\partial y}{\partial x}.$$

Это уравнение включает

- экспоненциальное затухание Δ : в отсутствие нового входа Δ стремится к нулю со скоростью пропорциональной $(1 - k)$ $k \in [0, 1]$.
- член, пропорциональный ошибке выходного значения $(y - y^d)$: аналогичная пропорциональность свойственна и обычному соотношению для модификации весов - чем больше ошибка, тем больше ее влияние на исправление входного значения. Этот член также пропорционален чувствительности выхода ко входу - $\partial y / \partial x$.

Вайгенд и его коллеги предложили наглядную механическую интерпретацию минимизируемой функции, а также отношению скоростей обучения и исправления (см. Рисунок 4).

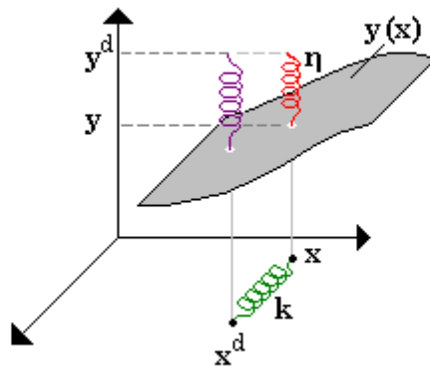


Рисунок 4. Механическая аналогия конкуренции между обучением и исправлением данных. К реальному входу x^d присоединяется пружина и растягивается другим концом до точки x , что сопровождается увеличением энергии в пружине на $\frac{1}{2} k(x^d - x)^2$. Но при этом энергия, запасенная в пружине, связывающей реальное и желаемое значения выхода сети, может уменьшиться (растяжение правой пружины меньше, чем левой) так, что суммарная энергия двух пружин уменьшается.

При обычном обучении (без исправления входного вектора) данные располагаются в пространстве вход-выход. Наблюдаемое выходное значение состояния выходного нейрона может рассматриваться как поверхность над пространством входов. Точки, изображающие данные обучающего набора вертикально прижимаются к этой поверхности пружинами, которые запасают некоторую энергию сжатия. Сложность нейронной сети определяется в конкуренции между жесткостью поверхности и жесткостью пружин. В одном из предельных случаев, бесконечно мягкая сеть (поверхность) пройдет как раз через все точки, определяемые данными. В противоположном случае, чрезмерно эластичные пружины не будут оказывать воздействия на поверхность и менять нейронную сеть.

Введение механизма исправления данных соответствует добавлению пружин в пространстве входов - между каждой точкой данных x^d и исправленным значением x . Энергия, запасенная в этих пружинах составляет $k\Delta^2 / 2$. Минимизация суммарной функции ошибки соответствует минимизации полной энергии, запасенной в обеих типах пружин. Отношение η / k описывает конкуренцию между важностью ошибок выхода и важностью ошибок входа.

Понимание закономерностей временных последовательностей

Исправление данных является важной компонентой подхода, позволяющего извлекать из нейронных сетей знания, касающиеся воспроизводимых ими временных закономерностей. Если, например, нейронная сеть обучена и используется для предсказания курса рубля по отношению к доллару, то естественно попытаться осмыслить связь большего или меньшего падения этого курса с теми или иными параметрами, подаваемыми на вход нейронной сети.

Кравен и Шавлик (Craven & Shavlik, 1996) разработали алгоритм **TREPAN**, порождающий дерево решений, аппроксимирующее поведение обученной нейронной сети. Важным достоинством алгоритма является то, что он *не предъявляет никаких требований к архитектуре сети, числу ее элементов и связей* (вспомним как важно было упростить структуру сети при использовании правила **NeuroRule**). Для него вполне достаточно того, что нейронная сеть является черным ящиком или *Оракулом*, которому можно задавать вопросы и получать от него ответы. Точность предсказания, даваемое сгенерированным деревом решений, близка к точности нейросетевого предсказания.

Приведем формальную схему алгоритма

TREPAN

Исходные данные обученная нейронная сеть (*Оракул*); обучающая выборка - S ; множество признаков - F , min_sample - минимальное количество вопросов для каждого узла дерева, baum_width - число ветвей.

Инициализируем *корень* дерева R в виде *листа*.

<Выборка векторов признаков>

Используем все обучающее множество примеров S для конструирования модели M_R распределения входных векторов, достигающих узла R .

$$q := \max(0, \text{min_sample} - |S|)$$

$$\text{query}_R := \text{множество из } q \text{ примеров, генерируемых моделью } M_R.$$

<Используем нейронную сеть для классификации всех векторов признаков>

Для каждого вектора признаков $x \in (S \cup \text{query}_R)$ узнаем у Оракула принадлежность x тому или иному классу - ставим метку класса $x := \text{Oracle}(x)$

<Осуществляем наилучшее первое расширение дерева>

Инициализируем очередь *Queue*, составленную из наборов $\langle R, S, \text{query}_R, \{\} \rangle$

До тех пор пока очередь *Queue* не пуста и глобальный критерий останова не выполнен

<создаем узел в начале очереди *Queue* >

удаляем $\langle \text{узел } N, S_N, query_N, constr_N \rangle$ из начала очереди *Queue*.

Используем $F, S_N, query_N$ и *beam_width* для конструирования в узле *N* разветвления *T*.

<создаем узлы следующего поколения>

Для каждого ответвления *t* разветвления *T*

создаем *C* - новый дочерний узел *N*

$constr_C := constr_N \cup \{T = t\}$

<выборка векторов для узла *C*>

$S_C :=$ члены S_N с ответвлением *t*.

Конструируем модель *M* распределения примеров, покрываемых узлом *C*

$q := \max(0, \min_sample - |S|)$

$query_C :=$ множество из *q* примеров, сгенерированных моделью M_C и ограничением $constr_C$

Для каждого вектора признаков $x \in query_C$ ставим метку класса $x := Oracle(x)$

<временно принимаем, что узел *C* является листом>

Используем S_C и $query_C$ для определения метки класса для *C*.

<Определяем должен ли узел *C* расширяться>

если локальный критерий останова не удовлетворен то

поместить $\langle C, S_C, query_C, constr_C \rangle$ в очередь *Queue*.

Вернуть дерево с корнем *R*.

TREPAN поддерживает очередь листьев, которые раскрываются и порождают поддеревья. В каждом узле очереди **TREPAN** сохраняет: (i) подмножество примеров, (ii) еще одно множество векторов, который называется набором вопросов (*query*) и (iii) набор ограничений (*constr*). Подмножество примеров включает просто те векторы обучающего набора, которые достигают данного узла дерева. Дополнительный набор вопросов Оракулу используется для выбора теста на разветвление в узле и определения класса примеров, если узел является листом. Алгоритм всегда требует, чтобы число примеров, на основе которых оценивается узел, было бы не меньше заданного (*min_sample*). Если же до данного узла доходит меньшее число примеров, **TREPAN** генерирует новые искусственные примеры, используя набор ограничений в данном

узле. Множество ограничений определяет условия, которым должны удовлетворять примеры, чтобы достичь данного узла - эта информация используется при формировании набора вопросов для создаваемого нового узла. Для завершения процедуры построения дерева **TREPAN** использует локальный критерий - он оценивает состояние данного узла и решает, превратить ли его в лист, и глобальные критерии - максимальный размер дерева и общую оценку качества классификации примеров деревом.

Возникает естественный вопрос: "А зачем вообще нужна нейронная сеть для данного алгоритма?" Ведь он может просто использовать обучающую выборку - известно же, какому классу принадлежит каждый пример. Более того, как бы хорошо ни была обучена сеть, она все равно будет делать ошибки, неправильно классифицируя некоторые примеры. Дело в том, что именно использование нейросетей в качестве Оракула дает возможность получать деревья решений, имеющих более простую структуру, чем у деревьев, обученных на исходных примерах. Это является следствием как хорошего обобщения информации нейронными сетями, так и использования при их обучении операции исправления данных (**CLEARNING**). Кроме того, алгоритмы построения деревьев, исходя из тренировочного набора данных, действительно разработаны и с их помощью такие деревья строятся путем рекурсивного разбиения пространства признаков. Каждый внутренний узел подобных деревьев представляет критерий расщепления некоторой части этого пространства, а каждый лист дерева - соответствует классу векторов признаков. Но в отличие от них **TREPAN** конструирует дерево признаков методом *первого наилучшего расширения*. При этом вводится понятие наилучшего узла, рост которого оказывает наибольшее влияние на точность классификации генерируемым деревом. Функция, оценивающая узел n , имеет вид $F(n) = r(n)(1 - f(n))$, где $r(n)$ - вероятность достижения узла n примером, а $f(n)$ - оценка правильности обработки этих примеров деревом. **TREPAN** очень интересно осуществляет разделение примеров, достигающих данный внутренний узел дерева, а именно, использует так называемый $m - of - n$ тест. Такой тест считается выполненным, когда выполняются по меньшей мере m из n условий. Если, например, имеется 3 булевых переменных x_1, x_2, x_3 , то использование выражения $2 - of - \{x_1, \neg x_2, x_3\}$ будет эквивалентно использованию логической функции $(x_1 \cap \neg x_2) \cup (x_1 \cap x_3) \cup (\neg x_2 \cap x_3)$. Подобная формулировка правил делает деревья вывода более компактными и четкими. Приведем пример дерева решений, полученного алгоритмом **TREPAN**, Оракулом в которой являлась нейронная сеть, обученная предсказывать курс обмена немецкой марки на доллар (Weigend et al., 1996). Заметим, что для обучения сети использовался рекомендуемый для финансовых приложений метод **CLEARNING**, с которым мы уже познакомились. Сеть обучалась на данных, охватывающих период с 1985 по 1994 гг. и предсказывала рост или падение курса обмена на следующий день в течение всего 1995 г.

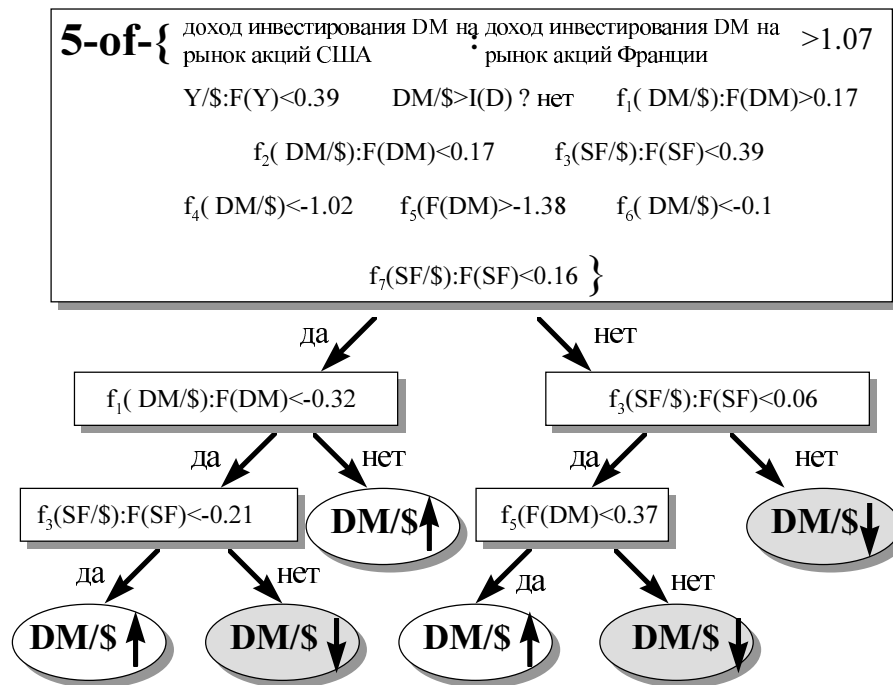


Рисунок 5. Дерево решений, построенной с использованием нейронной сети, обученной предсказывать рост или падение курса немецкой марки по отношению к доллару США. Обозначения: DM - немецкая марка; Y - иена; SF - швейцарский франк; \$ - доллар США; F - фьючерс; f - мера; I(D) - германский интерес. DM/\$ - курс марки по отношению к доллару - другие курсы обозначены аналогичным образом.

Таким образом, нейронные сети могут эффективно использоваться в практически важных задачах извлечения хорошо сформулированных знаний не только в случае, если их структура достаточно проста, но и в общем случае.

ЛИТЕРАТУРА

Craven, M.,W., & Shavlik, J.,W. "Extracting tree-structured representations of trained networks". In Touretzky, D., Mozer, M. and Hasselmo, M., eds. *Advances in Neural Information Processing Systems* (volume 8). MIT Press, Cambridge MA/

Lu Hongjun, Setiono, R. and Liu Huan (1995). "NeuroRule: A connectionist approach to Data Mining". *Proc.of the 21st VLDB Conference*, Zurich, Switzerland

Weigend, A.,S. and Zimmerman H.,G. "The observer-observation dilemma in Neuro-Forecasting: Reliable models from unreliable data through CLEARNING". <http://www.cs.colorado.edu/~andreas/Home.html>

Weigend, A.,S., Zimmermann, H.,G., and Neuneier, R. (1996) "Clearning. In *Neural Networks in Financial Engineering*", World Scientific, Singapore.