

Общезвестна истина, что чем раньше раскрывается в человеке талант, тем больше пользы он приносит обществу. Опытные педагоги предсказывают почти наверняка, получится ли из студента консерватории скрипач-виртуоз, а из студента университета настоящий математик. Можно ли, по Вашему мнению, заранее определить у человека талант конструктора с тем, чтобы помочь ему быстрее и полнее раскрыться?

— Никто, наверное, не станет спорить с тем, что талант конструктора существует. Однако членораздельно объяснить, что скрывается за этим понятием, не так-то просто. Я уже говорил, что современный конструктор — это инженер с обширными познаниями в различных областях техники. Но одной инженерной эрудиции конструктору мало — у него должно быть развито чувство взаимосвязи действующих на деталь, узел и машину нагрузок с их формой и размерами, психологическое и эстетическое «чувство формы» детали и машины, постоянная неудовлетворенность сделанным и неистребимое, страстное желание искать новое, лучшее.

Конструктору приходится нередко выступать в роли исследователя. Познания конструкторского труда — это тоже езда в неизнаемое.

Самой специфической профессиональной чертой, особенностью мышления конструктора является, пожалуй, развитое пространственное воображение. Оставляю в стороне интуицию: этим словом обычно обозначают неизвестное, некий икс в «уравнении» таланта.

В последнее время многие крупные ученые, в особенности математики, немало усилий прикладывают для выявления талантливых школьников. Организуются олимпиады, созданы школы со специальным уклоном, и т. д. Думаю, что нам, инженерам, есть смысл поучиться у математиков заботе о будущем своей науки.

Однако найти способных людей — полдела. Талант, как известно, нуждается в долгой шлифовке. Пора наконец наладить подготовку конструкторов в институтах, вести обучение по специальным программам, с упором на такие дисциплины, как математика, механика, теория упругости, материаловедение.

Валы отбора мощности дают возможность использовать двигатель самоходного шасси для приведения в действие пил, насосов, ленточных транспортеров и т. д.

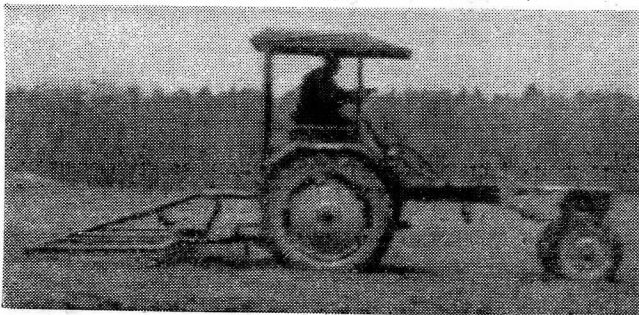
С Э В В Д Е Й С Т В И И

САМОХОДНОЕ ШАССИ РС-09

В соответствии с принципами международного сотрудничества и специализации в рамках Совета экономической взаимопомощи стран социалистического лагеря Советский Союз с 1963 года покупает у Германской Демократической Республики самоходные шасси РС-09.

Эти машины широко используются у нас в прибалтийских республиках и в Белоруссии. Самоходное шасси РС-09 — универсальная машина. Она может быть использована в соединении с сеялками, погрузчиками, различными уборочными машинами и машинами, предназначенными для других видов сельскохозяйственных работ.

Боронование всходов озимых на полях эстонского колхоза имени Эдуарда Вильде.



МОДЕЛИРОВАНИЕ ПРОЦЕССА УЗНАВАНИЯ

Кандидат физико-математических наук М. БОНГАРД.

ТРУДНО ФОРМАЛИЗУЕМЫЕ ЗАДАЧИ И УЗНАВАНИЕ

С появлением универсальных цифровых вычислительных машин человечество оказалось в необычном положении, которое можно описать словами: «Все можем, но не знаем как!» Действительно, известно, что при достаточно большой памяти и за достаточное большое время такая машина в состоянии решить любую задачу, которую способно решить вообще любое устройство. В частности, все задачи, которые может решить мозг, принципиально доступны универсальной машине с достаточно большой памятью. В нее нужно только заложить соответствующую программу.

Появились такие высказывания некоторых ученых: «Мы уже имеем машины с памятью на миллион двоичных единиц. Вероятно, недалеко время, когда появятся машины с памятью на миллиард единиц. А мозг человека содержит «только» пятнадцать миллиардов клеток. Скоро наши машины по своим возможностям начнут догонять мозг». Если бы эти люди были правы и останков была бы только за объемом памяти, то судьба «думающих» машин находилась бы в руках инженеров, конструирующих магнитные ленты и барабаны.

Однако при таком подходе упущено одно обстоятельство. Сама машина, даже имеющая очень большую память, еще ничего не умеет. В нее обязательно нужно ввести соответствующую программу, для составления которой задача должна быть полностью формализована, то есть предельно четко сформулирована — описана очень строгим и сухим математическим языком.

Человек же решает множество жизненных задач, вовсе не требуя их аккуратной постановки. Ему часто достаточно лишь наблюдения нескольких примеров. Вряд ли кто-нибудь в 6—8 лет слышал строгое определение различий между курами и утками. Но это не мешает ребятам в таком возрасте прекрасно решать задачу узнавания домашних птиц.

Человек выбирает «линию поведения» (даже при решении математических задач, не говоря уже о других случаях) в значительной степени на основе «предыдущего

опыта», «интуиции», «неформального подхода» и т. п. Поэтому, если мы хотим создать машину, которая принимала бы целесообразные решения в сложной и изменчивой обстановке, нужно суметь промоделировать эти стороны человеческой деятельности.

Казалось бы, вопрос об умении машины использовать предыдущий опыт сводится все к той же большой памяти. Однако это не совсем так (хотя память тоже, разумеется, нужна).

Представим себе, что ребенок обучался бы различать лошадей и коров, запоминая «портреты» встреченных им животных. Увидев некое животное, он должен был бы сравнить его со всеми хранящимися в памяти картинками. В случае совпадения с картинкой, называющейся «лошадь», ребенок говорил бы «лошадка».

Легко заметить, что такой способ использования жизненного опыта» принципиально лишил бы ребенка возможности узнавать новые экземпляры коров и лошадей или даже уже встречавшиеся, но в новом ракурсе, или с другого расстояния, или при ином освещении и т. п. Не говоря уже о том, что сравнение со всеми хранимыми в памяти картинками занимало бы немалое время (вспомните, как долго приходится решать, например, задачу типа «найдите на рисунке два совершенно одинаковых абажура»).

Ясно, что ребенок не помнит всех виденных им лошадей и коров, а помнит некоторые обобщенные признаки животных. Замечательно то, что в процессе обучения ребенка никто ему не рассказывал об этих признаках. Он нашел их сам. При этом ребенок и даже взрослый человек часто не могут рассказать, какими именно признаками они пользуются в том или другом случае.

Мы сказали, что опыт запасается в виде «признаков», однако это же можно описать и иначе. Ребенок обучается не **обращать внимания** на несущественные обстоятельства. (В нашем примере: каким боком к нему повернуто животное, лежит оно или стоит и т. п.) В этом умении «отвлечься от пустяков» великая сила. В самом деле, обстановка, в которую мы попадаем, может быть чрезвычайно разнообразной. Плохо было бы нашему бедному мозгу,

если бы все хоть сколько-нибудь различающиеся обстановки требовали разного поведения. К счастью, это не так. Целые группы ситуаций часто требуют одинакового поведения. Например, если мы хотим попригласить вошедшего в комнату человека, то мы произносим: «Здравствуйте, Иван Иванович!» — независимо от того, вошел ли он с правой или с левой ноги, держит голову ровно или чуть набок и т. д. Все эти ситуации требуют от нас одной и той же реакции. Но это не значит, что нам вообще безразлично происходящее. Если войдет не Иван Иванович, а некто другой, мы должны будем произвести другое имя.

Итак, узнавание (человека, животного, математической закономерности, вида болезни и т. д.) — это разделение всех ситуаций на группы, требующие одинакового поведения, или, другими словами, умение пренебречь обстоятельствами, не существенными для данной задачи.

«Узнавание» — это лишь другое название «рассуждения по аналогии», «обобщения», «обнаружения сходства», то есть именно тех процессов, которые помогают человеку решать не строго поставленные, не формализованные задачи. Ясно, что операция узнавания обязательно должна входить в набор операций думающей машины.

Вот почему проблема узнавания привлекает сейчас внимание и инженеров, и математиков, и физиологов, и психологов.

В ЧЕМ ТРУДНОСТЬ МОДЕЛИРОВАНИЯ УЗНАВАНИЯ!

Если все дело в том, чтобы пренебречь несущественными обстоятельствами, то что мешает конструктору машины с самого начала избавиться ее от поступления излишней информации? После этого вроде и проблемы не останется. И действительно, так можно поступить в случае, если мы хотим сделать машину, которая решает какую-нибудь одну задачу или настолько узкую совокупность задач, что для всех них одни и те же обстоятельства являются несущественными.

Трудность начинается тогда, когда мы хотим создать машину, которая могла бы решать много весьма разнообразных задач, целесообразно вести себя в обстановке, изменяющейся в широких пределах. В этом случае обстоятельства, которые были несущественными для одной задачи, могут оказаться главными для другой задачи. Поэтому конструктор лишен возможности «выбросить несущественное». Эта операция ложится на плечи самой машины. Она должна в период обучения выяснить, что несущественно для данной задачи, и перестать обращать на это внимание (но только на время ее решения).

А не обратиться ли за помощью к физиологам? Многие из них ведь специально занимаются выяснением того, каким образом животные используют предыдущий опыт. Почему бы не применить от-

крытые ими принципы при построении искусственных систем?

В течение некоторого времени всем казалось, что принцип работы высших отделов мозга, по крайней мере в общих чертах, разгадан. Действительно, после классических исследований И. П. Павлова в области условных рефлексов создалось впечатление, что с помощью комбинации большого числа условных рефлексов различных порядков можно организовать сколь угодно сложное поведение. Еще пятнадцать лет тому назад многие физиологи, подражая Архимеду, рискнули бы сказать: «Дайте нам достаточное количество устройств, способных вырабатывать условный рефлекс, и мы построим из них систему, которая путем обучения будет приспособляться к внешней обстановке и целесообразно вести себя в условиях, изменяющихся в очень широких пределах».

Не случайно на заре кибернетики первые попытки моделирования функций нервной системы относились именно к моделям условного рефлекса.

Оказалось, что весьма просто (и множеством способов) можно сделать устройство, умеющее вырабатывать условный рефлекс (замыкать и размыкать условные связи).

И вот тут-то выяснилось, что умение строить блоки типа «условный рефлекс» очень мало приблизило нас к возможности моделирования (а значит, и пониманию!) сложного поведения. Исследователям только казалось, что они сумеют построить сложное поведение из условных рефлексов. Самообман раскрылся в тот момент, когда появилась возможность практически проэкспериментировать самих себя.

Оказалось, что никто не знает, как нужно соединить между собой такие блоки, чтобы получить систему, способную в зависимости от ситуации отличать, например, лошадей от коров, или кур от уток, или Ивана Ивановича от Петра Петровича, или большие предметы от маленьких, и т. д., и т. п.

Но если в решении этой проблемы не помогли физиологи, изучающие высшую нервную деятельность, то, может быть, это в состоянии сделать те физиологи, которые изучают свойства отдельных нервных клеток? В самом деле, переработка информации в нервной системе происходит с помощью нервных клеток (нейронов). Если мы сделаем очень много блоков, обладающих свойствами одиночного нейрона, и соединим их между собой, то не получим ли нужную нам систему? И снова эксперименты с моделями нейронных сетей показали, что главная трудность не в выборе свойств, которыми следует наделять модель отдельного нейрона, а в том, как их соединить между собой. В результате случайных соединений интересующие нас системы не получаются. Придумать хороший способ соединения «нейронов» между собой ничуть не легче, чем найти способ соединения блоков «условный рефлекс», а как соединены нейроны в мозгу, никто не знает.

Попробуем подойти к вопросу с другой стороны. В настоящее время на вычислительных машинах решается множество сложнейших задач. Для каждой из них написана программа. Что, если воспользоваться опытом, накопленным программистами, писавшими эти программы? Попробуем «заказать» опытному программисту «узнающую» программу. При «приеме заказа» он обязательно потребует предельно четко сформулировать задачу. Спросит, с помощью каких признаков машина должна узнавать объекты.

Мы будем вынуждены ответить, что сам по себе такой вопрос нарушает «правила игры». Никто ведь не объясняет ребенку, по каким признакам надо отличать, например, кур от уток. Если ему показывают буквы, то говорят: это «а», и это «а», а это «б», — и ребенок сам находит способ различать их (рис. 1 и 2).

Поэтому, если мы хотим моделировать работу, производимую мозгом при узнавании, то должны научить машину классифицировать объекты по примерам, показанным ей при обучении, а не по формальному правилу. Признаки она должна отыскивать сама.

Вероятнее всего, программист разведет руками и скажет, что не имеет опыта в программировании таких нечетко поставленных задач. «А нельзя ли, — спросит он, — на основании физиологических данных хотя бы приблизительно сказать, как мозг решает подобные задачи? Знание этого помогло бы составить программу». Увы, круг замкнулся!

В настоящее время практически ничего не известно о законах обработки сигналов в мозгу. И одна из задач моделирования процесса узнавания как раз и состоит в том, чтобы помочь физиологам более сознательно и целеустремленно исследовать работу нервной системы. Сегодня мы еще лишены возможности «подглядеть в природу» программы узнавания. Приходится придумывать их наново, не рассчитывая ни на опыт, накопленный при написании «обычных» (расчетных) программ, ни на знания физиологов. Обнадешивает лишь уверенность в существовании хороших программ узнавания, опирающаяся на наблюдение поведения человека и животных.

КАКИЕ ЗАДАЧИ И СИСТЕМЫ НАС ИНТЕРЕСУЮТ!

Конечная цель работы всякой узнающей системы — классификация объектов, которые будут ей показаны после «обучения» — при «экзамене». С этой узкой точки зрения обыкновенный дверной замок — тоже узнающая система, так как он делит всевозможные ключи на два класса: «открывающие» и «не открывающие». Вспомним, однако, что мы заинтересовались проблемой узнавания в связи с задачей построения систем, которые могли бы решать множество разнообразных (и притом неформализованных) задач, ориентироваться в изменяющейся обстановке.

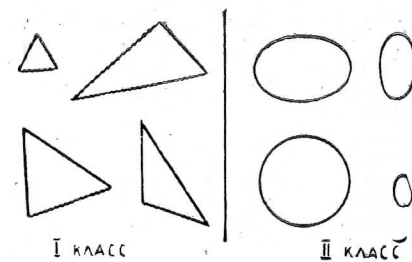


Рис. 1.

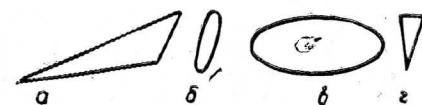


Рис. 2.

Если показать ребенку набор картинок, разбитых на два класса (рис. 1), а затем спросить, в какую «кучу» надо положить каждую из картинок, изображенных на рис. 2, то уже в 3—4 года ребенок говорит, что фигуры а и г относятся к первой «куче», б и в — ко второй.

Ясно, что с такой точки зрения замок — система неинтересная. Он всегда классифицирует ключи по одному и тому же принципу, который в готовом, формализованном виде заложен в него конструктором.

Нас же интересуют системы, способные решать много разных задач, производить разбиение объектов на классы многими способами. При этом сведения о том, по какому принципу надо в данной задаче разбивать объекты на классы, система должна получать не в виде формального правила, а в виде примеров.

Далее, нас не будут интересовать задачи, которые можно научиться решать простым запоминанием показанных при обучении объектов. Это значит, что в каждой задаче (из тех, которые решает система) каждый класс должен содержать много разных объектов. При обучении показывают небольшую часть их, а экзамен происходит на новых объектах.

И, наконец, мы хотим решать такие совокупности задач, в отношении которых конструктор не имеет возможности заранее определять, какие обстоятельства окажутся важными, а какими можно пренебречь. Это требование связано с решением реальных задач, когда заранее неизвестно, какие обстоятельства окажутся существенными. Внешним признаком этого будет то, что объекты, которые в некоторых задачах попадают в один класс, в других задачах будут попадать в разные классы. Примером совокупности задач (по понятным причинам не слишком большой), иллюстрирующим последнее требование, могут служить рисунки 3, 4 и 5.

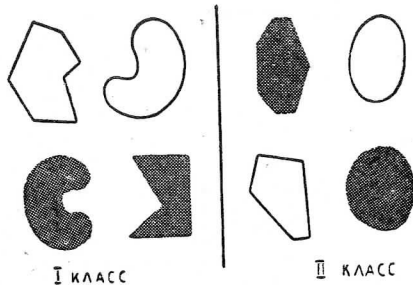


Рис. 3.



Рис. 4.



Рис. 5.

Рис. 3, 4 и 5. Задачи, являющиеся разными разбиениями на классы одних и тех же объектов. Все нарисованные объекты должны быть различимы для системы, способной решать эту совокупность задач, так как любая пара объектов, по крайней мере в одной из задач, попадает в разные классы. Устройство, которое мы бы хотели иметь, должно после обучения на материале, показанном на рис. 3, выработать абстрактные понятия «выпуклая фигура» и «невыпуклая фигура» и в дальнейшем при экзамене делить объекты по этому принципу. После обучения по рис. 4 должны образоваться понятия «многоугольник» и «криволинейная фигура», а после обучения по рис. 5 — «белая фигура» и «черная фигура».

Возникает вопрос: а какую же совокупность задач должна решать узнающая система? Первое, что приходит в голову, — это пожелание: хорошо бы она решала все задачи (все разбиения на классы данного множества объектов, например, картинок с данным числом элементов)! Однако это невыполнимо. Нетрудно доказать, что система, экзаменуемая не на тех

объектах, которые ей показывали при обучении, заведомо не может решать всех задач.

Итак, обе крайности не годятся. Нельзя сделать систему, решающую все задачи, и неинтересно делать систему, которая будет решать слишком маленькую совокупность задач. Однако между этими крайностями лежит много совокупностей задач, для которых могут существовать узнающие системы, но мы еще не умеем их строить. Примером служат те задачи по разделению геометрических фигур на классы, которые может решить человек.

Создание методов построения узнающих систем для подобных совокупностей задач и является целью ученых, занимающихся проблемой узнавания.

ПЕРЦЕПТРОН

Одна из первых систем, предназначенных для узнавания геометрических объектов, — перцептрон Розенблатта. Как он работает, поясняет схема, изображенная на рис. 6 (с ней следует познакомиться для понимания дальнейшего).

При обучении перцептрон «показывают» по очереди картинки разных классов. При показе каждой картинке (например, треугольника, как на рис. 6) некоторые A -элементы возбуждаются, а некоторые нет. Процесс обучения заключается в том, что после каждого показа картинке изменяются коэффициенты усиления K тех усилителей, которые связаны с возбужденными A -элементами. При этом, если показали картинку I класса, то соответствующие K увеличиваются, а если II класса — то уменьшаются. (В зависимости от закона увеличения и уменьшения K принято различать несколько типов перцептронов.) В результате обучения формируются коэффициенты K , которые будут использоваться при экзамене.

Во время экзамена перцептрон показывает неизвестный объект. Возбужденные A -элементы посылают в сумматор сигналы, равные соответствующим K . Если сумма всех этих сигналов положительна, — принимается решение, что показанный объект принадлежит к первому классу, если отрицательна, — то ко второму. Таким образом, поведение перцептрона во время экзамена определяется тем набором коэффициентов K , который выработался во время обучения.

Что же может и что не может узнавать перцептрон? После достаточно большого обучения он отличает фигуры, находящиеся в левой половине его «поля зрения», от фигур, находящихся в правой половине. Можно обучить перцептрон отличать большие фигуры (возбуждающие много рецепторов) от маленьких фигур.

А как обстоит дело с более сложными задачами? Опыты¹ показали, что даже

¹ Проводились А. П. Петровым в Институте проблем передачи информации Академии наук СССР.

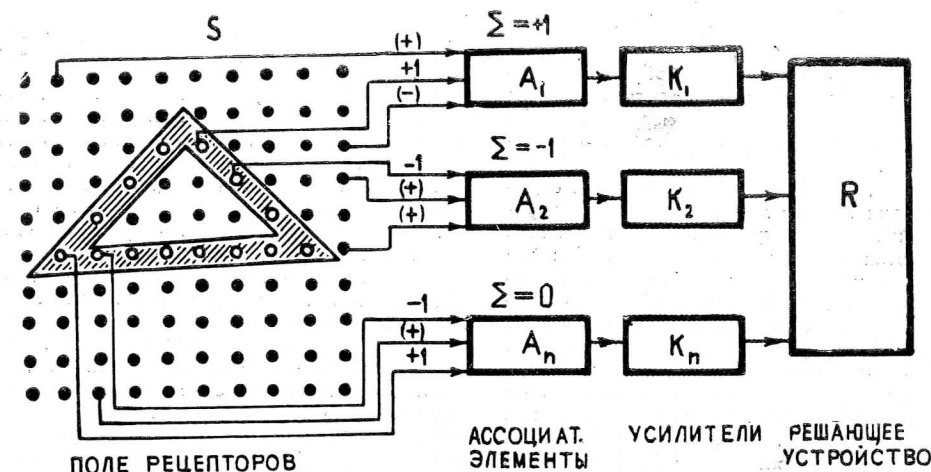


Рис. 6. Схема перцептрона. Изображение объекта при обучении и экзамене проектируется на поле рецепторов S . Каждый рецептор в зависимости от того, попало на него белое или черное место картинке, может перейти в состояние 0 или 1. Все рецепторы соединены (для простоты на схеме показано лишь несколько соединений) с ассоциативными элементами (A_1, \dots, A_n) связями, которые могут быть «возбуждающими» (+) и «тормозящими» (-). Возбуждающая связь в зависимости от состояния рецептора передает сигнал, равный 0 или +1, а тормозящая — 0 или -1. Каждый A -элемент суммирует все приходящие к нему сигналы. Если эта сумма превосходит некоторый порог, то A -элемент возбуждается. Система связей A -элементов и их пороги возбуждения выбираются заранее (например, с помощью жребия). Они не меняются ни при решениях какой-либо задачи, ни при переходе от одной задачи к другой. Если A -элемент не возбужден, то он не посылает никакого сигнала. Возбужденные A -элементы посылают в решающее устройство (сумматор) R сигналы через усилители, которые могут изменять как величину, так и полярность (знак) сигналов. До начала обучения коэффициенты усиления K всех усилителей равны нулю. На рисунке черные кружки условно изображают невозбужденные рецепторы, а белые — возбужденные. Сигналы в ассоциативные элементы приходят только от возбужденных рецепторов (обозначены +1 и -1). Знаки (+) и (-) соответствуют «возбуждающим» и «тормозящим» связям, по которым в данный момент сигналы не идут.

К ЧЕМУ НАДО СТРЕМИТЬСЯ

Как же должна быть устроена система, которая сможет создавать много разных обобщений и благодаря этому имитировать (пусть вначале и в слабой степени) работу мозга человека?

Рассмотрим действия перцептрона при обучении. Начинает он с некоторых операций над сигналами от рецепторов. Эти операции производят ассоциативные элементы. Причем какую бы задачу ни решал перцептрон, операции, производимые каждым A -элементом, не меняются. В то же время очевидно, что для решения одних задач нужны операции, результат которых не зависит от поворота изображения, для других задач — операции, результат которых не зависит от преобразования подобия, для третьих задач — от изменения взаимного расположения некоторых частей фигуры и т. д. Для ре-

² Это верно лишь для перцептронов, имеющих «не слишком большое» число A -элементов. Если же их число приблизится к 2^m (где m — число рецепторов), положение изменится. Для такого перцептрона существуют коэффициенты K , отделяющие треугольники от четырехугольников. Но в этом случае перцептрон для обучения придется показать все уместающиеся в его «поле зрения» треугольники и четырехугольники. Перцептрон с большим числом A -элементов может все запомнить, но и он не выра-

батывает абстрактных понятий «треугольник» и «четыреугольник».

Поясним теперь, что такое «большое» число A -элементов. Пусть перцептрон имеет сравнительно скромный растр 30×30 рецепторов. Тогда «большое число» — 2^{900} . Это во много раз превосходит число электронов во всей Солнечной системе. Таким образом, под понятие «не слишком большое» попадают перцептроны, имеющие миллионы и даже миллиарды A -элементов.

шения большой совокупности задач понадобится много очень разнообразных операторов (оператором мы здесь называем устройство, способное производить некоторые преобразования сигналов). По существу, речь идет о тех самых обобщенных признаках, которые в разных задачах отделяют один класс от другого.

Для каждой конкретной задачи большая часть этих признаков не нужна. Как же найти признаки, полезные для решения данной задачи? В принципе это можно сделать, например, методом проб и ошибок во время обучения: система по очереди испытывает различные операторы на показанных картинках. Операторы, которые не разделяют картинки на классы так, как нужно, бракуются. Если в арсенале системы имеется оператор, необходимый для данной задачи, то он рано или поздно будет найден и обучение закончится. При экзамене система применит найденные при обучении операторы.

Каковы главные трудности на пути к построению такой системы?

Во-первых, поскольку совокупность задач строго не определена, трудно указать набор признаков, **необходимый и достаточный** для решения всех задач. Придется придумывать набор признаков («с запасом», что приведет к дополнительному увеличению их количества. Но и в таком виде проблема создания набора признаков остается очень трудной.

Во-вторых, для большой совокупности задач нужно очень много разных операторов. Это, в свою очередь, создает две трудности: хранение признаков потребует слишком большой памяти, а перебор при обучении — слишком много времени.

Как же примирить, с одной стороны, требование большого числа операторов в наборе и, с другой стороны, — малого объема занятой ими памяти и малого времени обучения?

По-видимому, общая структура системы для узнавания геометрических объектов должна быть примерно такой.

Система имеет сравнительно небольшой набор стандартных блоков, из которых можно, как из кирпичей, построить много разных операторов. Такими элементарными операторами могут быть, например, программы, разделяющие фигуры на части, выделяющие границы фигур и их частей, измеряющие площади и длины, считающие разные объекты, строящие диаметры, находящие кривизну линий, соотношение чисел, взаимное расположение фигур и частей и т. п.

Способы соединения «кирпичей» должны предусматривать возможность построения операторов, производящих сложную, многоступенчатую обработку картинки («последовательное соединение «кирпичей»).

Пусть, например, встретилась задача, в которой нужно отличать фигуры человечков от чертиков. При ее решении поможет (но не всегда!) оператор, который выделит те части фигуры, у которых центр тяжести выше общего центра тяжести; построит на-

ибольший и наименьший диаметры для каждой такой части; выберет ту часть, у которой разница между этими диаметрами относительно мала; выделит контур этой части и установит, есть ли на нем точки с очень большой кривизной, находящиеся в верхней части контура. Нетрудно догадаться, что этот составленный из «кирпичей» оператор должен проверять, есть ли у фигурки рога. Признак получился громоздким потому, что нужно отличать рога от растопыренных пальцев руки (которая может быть поднята).

Помимо операторов, производящих последовательную обработку картинки, могут понадобиться операторы, одновременно учитывающие много разных свойств картинки («параллельное соединение «кирпичей»). В нашем примере, очевидно, есть смысл обращать внимание не только на рога, но и на хвост, копыта и т. д. Это, по-видимому, лучше всего делать с помощью блока, умеющего строить правила примерно такого типа: «Если имеет место то-то и одновременно то-то, но не имеет места то-то, значит, картинка относится к первому классу». Естественно, этот «логический блок» может оказаться полезным не только на последнем этапе построения оператора (как в нашем примере), но и на промежуточных этапах. При этом будут получаться сложные, «смешанные» соединения «кирпичей».

«Кирпичи» и законы их соединения между собой выбираются конструктором так, чтобы среди операторов, которые система сможет построить, нашлись необходимые для разделения классов во всех задачах из выбранной совокупности.

Благодаря малому количеству «кирпичей» для их хранения понадобится небольшой объем памяти. А как быть со временем обучения? Ведь если строить и проверять все операторы подряд, то это время окажется слишком большим. Нельзя ли и здесь воспользоваться тем, что признаки не хранятся в памяти в готовом виде, а строятся последовательным добавлением новых «кирпичей» к более простым комбинациям? Оказывается, это действительно может помочь в сокращении числа проб. Но, чтобы воспользоваться им, необходимо иметь критерии перспективности промежуточных (более простых) комбинаций «кирпичей». В отношении таких «полуфабрикатов» надо уметь решать: стоит их достраивать или мало надежды, что из них получится оператор, хороший для данной задачи, и поэтому лучше не терять времени. Какой критерий будет полезен при определении «перспективных» комбинаций, разумеется, зависит от решаемой совокупности задач.

Нахождение хороших критериев для данной совокупности, как и хороших «кирпичей», вообще говоря, дело трудное. Однако, по-видимому, существуют критерии, полезные для весьма широкого круга совокупностей задач, например, для всех задач, которым **можно обучиться на малом числе примеров**. Теория этого вопроса еще почти не разработана. Поэтому большой интерес

представляют опыты с обучающимися системами, пользующимися подобными критериями. Такие опыты сейчас начинаются.

НЕКОТОРЫЕ РЕЗУЛЬТАТЫ

Итак, сегодня главная трудность построения узнающей системы не поиск элементов, из которых нужно ее делать, а отыскание логики совместной работы большого количества элементов. При этом можно практически совсем не интересоваться свойствами самих элементов.

В аналогичном положении находится программист, решающий математическую задачу на универсальной вычислительной машине. Он может не знать, собрана машина на лампах или на транзисторах, с помощью каких схем она складывает и перемножает числа. Ему достаточно знать, какую команду нужно отдать машине, чтобы она совершила то или иное действие. Поэтому все внимание программиста сосредоточено на тех преобразованиях, которым нужно подвергнуть исходный материал, чтобы получить требуемый результат. Последовательность команд для машины — программа — вот плод труда программиста.

Узнающие системы, на которых велись описываемые ниже опыты, также моделировались с помощью универсальной машины, в которую вводилась соответствующая программа. Конечно, каждая программа учитывает специфику той машины, на которой ведется работа, однако путем несложной переделки программы можно приспособить к любой универсальной машине. Особенности той или другой узнающей системы определяются свойствами программы, а не машины. Поэтому мы иногда будем отождествлять программу с узнающей системой. В этом смысле и следует понимать выражения «программа учится», «программе показали», «программа ответила» и т. п. Последнее выражение, например, означает, что машина, управляемая программой, напечатала ответ.

Первую программу, которая строила признаки из небольшого количества «кирпичей», сделали для узнавания не картинок, а некоторых искусственных объектов — числовых таблиц. Они состояли из нескольких строк по 3 числа в каждой.

Табл. 1

A	B	C
2	5	-30
7	3	84
1	4	-12
2	6	-48
5	1	20

Табл. 2

A	B	C
3	4	11
6	3	21
1	5	1
8	3	29
7	2	19

В качестве примера приведены две таблицы. В первой таблице во всех строках числа C получены из A и B с помощью од-

ного и того же закона. Нетрудно убедиться, что для всех строчек подходит закон $AB(A - B) = C$. Ясно, что, используя разные числа A и B, можно по одному и тому же закону построить очень много разных таблиц. Все таблицы, построенные по одному и тому же закону, будем считать объектами одного класса. Вторая таблица принадлежит к другому классу — она построена по закону $A - B + AB = C$.

Задача заключалась в следующем: пусть программе показали несколько объектов первого класса (таблиц, построенных по одному и тому же закону), несколько объектов второго класса, третьего и т. д. При этом программе не сообщают, по каким законам построены таблицы. Для нее классы просто занумерованы. Затем при экзамене показывают новую (не показанную при обучении) таблицу, принадлежащую одному из классов, участвовавших в обучении. Программа должна «сказать», к какому классу относится эта таблица.

Из 25 «кирпичей» программа могла строить более 5 тысяч признаков. Сокращение перебора в этой работе не применялось. Проверялись все возможные признаки и отбирались полезные при данном наборе законов. Полезными считались признаки, одинаково характеризующие все таблицы одного класса и достаточно разнообразно характеризующие таблицы разных классов.

Если требовалось различать между собой объекты 9 разных классов, то программа после обучения узнавала таблицы безошибочно. Если различать нужно было 24 класса, то появлялись ошибки. Однако число ошибок и в этом случае составляло только 3 процента от количества ошибок, которые сделала бы система, определяющая номер класса случайным гаданием.

Когда об этой программе (ее устройстве и эксперименты с ней описаны в журнале «Биофизика» № 2, 1961 год) рассказали некоторым скептикам, им захотелось придумать задачу, которая оказалась бы для программы своеобразной «ловушкой». Ход их рассуждения был примерно таким: «Автор, составивший программу, рассчитывал на то, что все строчки каждой таблицы будут построены по одному и тому же арифметическому закону. Поэтому такой программе, вероятно, будет трудно, если построить разные строчки одной и той же таблицы по разным арифметическим законам, объединенным некоторым дополнительным логическим условием. Таким образом, вся таблица будет построена по единому закону, но он не будет сводиться к достаточно простой арифметической зависимости».

Примеры таких «каверзных» законов приведены в таблицах 3 и 4. Строчки таблицы 3 построены по закону:

$$C = A + B, \text{ если } A \geq 0 \text{ и } B \geq 0$$

$$C = A - B, \text{ если } A \geq 0 \text{ и } B < 0$$

$$C = A : B, \text{ если } A < 0 \text{ и } B > 0$$

$$C = A \times B, \text{ если } A < 0 \text{ и } B \leq 0$$

Таблица 4 построена по другому закону похожего типа.

Табл. 3

A	B	C
5	3	8
-7	2	-3,5
6	-3	9
-3	-8	24
-4	8	-0,5
2	-5	7

Табл. 4

A	B	C
5	3	2
-7	2	-14
6	-3	-2
-3	-8	5
-4	8	-32
2	-5	-0,4

Было проведено обучение по 10 различным «каверзным» законам. Когда на экзамене предъявили 10 новых таблиц, построенных по этим же законам, все они были узнаны правильно! Таким образом, логических возможностей программы хватило для того, чтобы не попасться в ловушку. Программа сама в ходе обучения приспособилась к законам, о которых автор вовсе и не думал, когда составлял программу.

Именно эта гибкость узнающих программ, умение самостоятельно приспособиться к новой задаче и дают надежду успешно использовать такие программы при решении некоторых практических вопросов. Сейчас имеется уже много попыток применить узнающие программы в самых различных областях. Здесь мы расскажем только про одну задачу.

ПОИСК НЕФТИ

При бурении скважины для добычи нефти возникает вопрос: на какой глубине находится нефтеносный пласт? С первого взгляда может показаться, что вопрос этот праздный. Действительно, почему бы не бурить до тех пор, пока из скважины не ударит нефтяной фонтан? Однако так поступать нельзя. Неожиданно ударившая нефть часто ломает буровое оборудование, возникают пожары и т. п. Поэтому для предотвращения выброса нефти приходится при бурении подавать в скважину тяжелый глинистый раствор. Когда скважина пробурена, в нее вставляют стальную трубу. Пространство вокруг трубы цементируют. Подготавливают устройства для приема нефти. И только после этого стальную трубу вместе с цементной оболочкой с помощью специального устройства простреливают на уровне нефтеносного пласта. Если по ошибке прострелить скважину не там, где нужно, может хлынуть не нефть, а вода. Такая ошибка причинит немалый убыток.

Как же сейчас определяют глубину, на которой нужно прострелить скважину? Исследуют целый ряд физических параметров пластов, через которые проходит скважина. Геофизики измеряют их электропроводность, естественную радиоактивность, радиоактивность, вызываемую искусственным облучением породы, и т. п. Ни один из этих параметров сам по себе

не дает возможности отличить нефтеносный пласт, например, от водоносного. Однако опытные геофизики, сопоставляя сразу несколько параметров, в общем, умеют узнавать нефтеносные пласты. Конечно, это требует очень высокой квалификации работников. Но даже самые опытные люди все-таки допускают немало ошибок.

Если мы напишем программу, точно воспроизводящую принципы работы опытного геофизика, то она будет делать столько же ошибок, сколько человек. Другое дело — обучающаяся узнающая программа. Она может в принципе найти признаки нефтеносных пластов, ускользнувшие от внимания геофизиков, и делать меньше ошибок, чем люди.

Целый ряд научных коллективов попробовал использовать обучающиеся программы для узнавания нефтеносных пластов. В частности, была испробована и программа, очень похожая на «логический блок» из описанной выше арифметической программы. Для обучения программы отобрали 45 примеров нефтеносных и 45 примеров водоносных пластов. На основании этих примеров программа отбирала из 100 000 имеющихся в ее распоряжении разных признаков те, которые оказались полезными для данной задачи. Затем, пользуясь отобранными признаками (их было около 100), программа экзаменовалась на других (не участвовавших в обучении) пластах. При экзамене на 180 «незнакомых» пластах программа сделала 3 ошибки. Это в 5—6 раз меньше, чем сделали опытные люди на том же материале! Следует отметить, что очень хорошие результаты были получены и с помощью других обучающихся программ (группой ленинградских исследователей под руководством А. Г. Француз и сотрудниками Института автоматизации и телемеханики В. Н. Вапником и А. Я. Червоненкисом).

ОСОБЕННОСТИ ОБУЧАЮЩИХСЯ ПРОГРАММ

Философы и журналисты, пишущие о кибернетике, любят заканчивать статью примерно таким «заклинанием»: раз человек написал программу для машины, значит, он передал ей только часть своих знаний; поэтому программа никогда не будет умнее человека, который ее создал.

Мы не станем спорить с буквой этого высказывания, так как слово «умнее» не имеет здесь точного смысла. А вот дух его полностью опровергается уже имеющимся опытом работы обучающихся программ. Про программы, узнававшие нефтеносные пласты, нельзя говорить, что программисты передали им свои знания по геологии и геофизике. Дело в том, что все эти программы писали люди, ничего не понимающие в геологии! Более того, когда они писали программы, то даже не знали, что с их помощью будут делать: отличать нефтеносные пласты от водоносных или туберкулез от рака легкого. (Программы, о которых идет речь, были написаны до зна-

комства их авторов с геологами, поставившими задачу!) Таким образом, в деле узнавания нефтеносных пластов программы намного превзошли своих создателей. Откуда же программы получили необходимые для этого сведения по геофизике? Только за счет наблюдения и, если хотите, «творческой обработки» тех примеров, которые были им показаны при обучении. Других источников информации о законах геологии эти программы принципиально не могли иметь.

Становится понятным значение хорошего подбора примеров, используемых для обучения. Эта работа была проделана Ш. А. Губерманом и М. Л. Извековой — специалистами-геофизиками. Именно их знания и интуиция обеспечили наличие **необходимой информации** в материале обучения.

До появления универсальных вычислительных машин существовали лишь узкоспециализированные произведения инженерного искусства. Локомотив «умеет» только тянуть состав по рельсам, а электрическая лампочка — давать свет. Существенно, что любое подобное устройство выпускается вполне готовым к выполнению своей функции.

Универсальные машины могут в принципе решать чрезвычайно широкий класс задач. Свообразная «расплата» за это состоит в том, что такая выпущенная с завода машина не умеет решать ни одной задачи. Нужна еще работа программиста, который «вдохнет душу» в «безжизненное тело», созданное инженерами.

С появлением обучающихся программ мы становимся свидетелями следующего этапа этого процесса. Инженер сконструировал машину, программист написал программу, ввел ее в машину — и... машина все равно не может решить ни одной задачи! Ей еще нужно учиться. Дело в том, что программист написал лишь часть программы, работающей во время обучения и экзамена. Другая же часть создается самой программой во время обучения. Какой эта часть получится, зависит от того, какие примеры покажут при обучении. Поэтому, кроме **инженера и программиста**, в создании устройства, решающего данную задачу, косвенно участвует и **учитель**. Программист лишь завершил создание устройства, умеющего обучаться решению большого числа разных задач. (Эти же программы могут обучиться и ставить диагноз при различных заболеваниях и обнаруживать скрытый дефект в изделиях и т. п.) По существу, программист создал «универсальную программу». Как и «универсальная машина», сама она еще ничего не может. Нужен следующий этап — «специализация».

«Ага, — слышим мы торжествующий возглас Философа, — вот вы и попались: сравнивали «знания» машины со знаниями **программистов**, и оказалось, что машина «умнее»! Но ведь вы сами говорите, что нужно учитывать и **работу учителей** машины. А в описанном случае учителями были геофизики, от них-то машина и получила сведения по геологии!»

Да, конечно, машину учили геофизики, но в узнавании нефтеносных пластов (при некотором ограничении геологических условий) машина превзошла не только **программистов**, но и **учителей!**

Никакого чуда в этом нет. Источником всех наших знаний являются в конечном счете наблюдения за природой и эксперимент. Во время обучения программа имела возможность экспериментировать с описаниями некоторых природных объектов. При этом она обнаружила закономерности, ускользнувшие от внимания людей.

Между прочим, после того, как найденные признаки машина сообщила людям, они приобрели возможность сами (без помощи машины) лучше узнавать нефтеносные пласты. Ученик и учитель поменялись местами!

Заметим, что человек совсем не обязательно должен быть посредником между обучающимися программами и природой. В принципе машина может непосредственно получать сведения от различных измерительных устройств. В этом случае «учителем» машины будет сама природа.

Таким образом, эксперименты с обучающимися узнающими программами позволяют ответить на вопрос: может ли машина знать о законах природы больше, чем создавшие ее люди? Мы подразумеваем, что критерием знания является целесообразное поведение (например, поставить правильный диагноз, прострелить скважину так, чтобы пошла нефть, а не вода, и т. п.). Удивляться этому надо не более, чем, например, тому, что следующее поколение физиков может знать больше предыдущего. Для этого они должны не только читать книги, но и, во-первых, ставить новые эксперименты, во-вторых, по-новому обдумывать старые опыты. И то и другое вполне доступно машинам.

Иногда излишне восторженные люди впадают в противоположную крайность. Часто можно услышать: «конструктор не заложил в систему никаких сведений; всю информацию она собрала сама в ходе обучения».

Если бы в систему и в самом деле не заложил никаких сведений о **совокупности задач**, то она одинаково решала бы все задачи, а как уже упоминалось, система, решающая все задачи, существовать не может. Таким образом, вольно или невольно какие-то сведения, в удовлетворительно работающую систему заложены. В действительности же конструктор должен стремиться заложить в систему как **можно больше сведений**, но не об отдельной задаче, а о той большой совокупности задач, которые могут встретиться. Информацию же о том, какая задача из совокупности подлежит в данный момент решению, система должна извлечь уже сама в ходе обучения.

Создание эффективно работающих обучающихся узнающих систем для широких классов задач сейчас, по-видимому, одно из основных направлений кибернетики.