

8. ОБУЧЕНИЕ КОМПЬЮТЕРА

Machine Learning – Машинное обучение

Обучение очень близко связано с интеллектом. Фактически, интеллект не может существовать без способности обучаться потому, что основная польза обучения состоит в том, что оно является средством получения новых знаний.

8.1. Два вида обучения

Имеет место удивительный парадокс, состоящий в том, что обучение компьютера как очень простой, так и чрезвычайно трудный процесс. Причина в том, что существуют два принципиально различных типа обучения: обучение "заучиванием наизусть" и когнитивное обучение (**cognitive** ['kognitiv] познавательный).

8.1.1. Обучение "заучиванием наизусть"

Многое из того, что мы изучаем, есть просто собранные факты, которые запоминаются (или заучиваются), а затем извлекаются из памяти. Этот процесс называют обучением "заучиванием наизусть". Вот несколько фактов, которым можно обучиться посредством этого метода: "Киев - столица Украины", " $2+1$ есть 3", "расстояние от Харькова до Киева 450 км".

Однако, обучение заучиванием наизусть также может быть применено к обучению последовательности действий. Например, обучение детей делению или умножению: ребенок заучивает определенную последовательность действий и затем применяет ее при необходимости.

Ключом к обучению заучиванием наизусть является конкретизация. Все, что вы можете выучить наизусть, есть конкретные единицы информации. Столица Украины является конкретным городом, расстояние от Харькова до Киева является конкретным расстоянием, метод деления является конкретной последовательностью конкретных действий. Хотя такие последовательности действий можно применять к широкому кругу задач, как например, деление может быть выполнено для любых двух чисел, но действительная последовательность шагов решения каждой конкретной задачи не генерируется. Таким образом, последовательности действий, заучиваемые наизусть, более правильно называть процедурами. ЭВМ может

легко изучить все, что вы можете выучить наизусть. Фактически, это естественный способ, которым ЭВМ работает: это программирование!

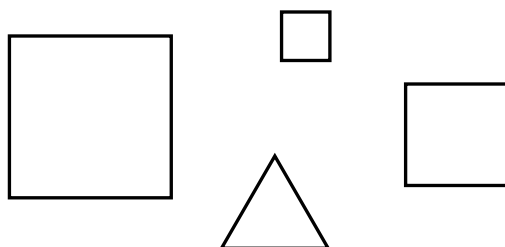
Поскольку компьютеры очень хорошо следуют инструкциям, они могут легко следовать процедуре или сохранять несколько единиц информации в базе данных. Например, учитель может сказать ученикам, что вода является жидкостью между 0 и 100 градусами. Вы можете ввести такую же информацию в базу данных. Если вы спросите, когда вода является жидкостью, то и ученик, и компьютер ответят одинаково.

Обучение заучиванием наизусть включает запоминание конкретных фактов или процедур. Оно не требует какой-либо генерации для получения информации.

8.1.2. Когнитивное обучение

Когнитивное обучение - это наиболее важный способ, которым обучаются люди и наиболее трудный для реализации на ЭВМ. При этой форме обучения люди используют свой разум для анализа, организации и установления соотношений между конкретными кусочками знаний. Результатом этих умственных усилий является создание описаний класса. Описание класса является обобщением, оно выводится на основании изучения нескольких конкретных примеров. Например, по вашим знаниям нескольких конкретных собак, вы можете создать обобщенное (сгенерированное вами) понятие собаки, которое позволяет вам распознать фактически любую собаку как собаку. В общем случае, описание класса определяет все объекты этого класса.

Человек может очень быстро выполнять обобщения. Рассмотрим 4 объекта:



Вопрос: Какая из фигур неподходящая? Ясно, что треугольник не той классификации, что квадраты, даже хотя квадраты трех различных размеров. Вы знали, что треугольник не подходит ни к одному из квадратов, потому что вы смогли легко создать описание класса, которое было бы

пригодно всем трем квадратам, но не подходило к треугольнику. В простой ситуации как эта, вы создаете общую категорию, не думая даже об этом.

Человек также может формировать описания класса и для процедур. Затем он может адаптировать эти общие процедуры ко множеству похожих ситуаций. Например, вы знаете общую процедуру для подметания пола - любого пола любым веником. Водитель может легко управлять автомобилем по любой дороге, по любому маршруту, даже если он не видел этой дороги раньше. Человек может выполнять эти задачи, потому что он может обобщать основные процедурные элементы, которые их определяют.

Способность составлять описания классов является фундаментальной в создании компьютеров, думающих так же как человек.

8.2. Как создаются описания классов в процессе обучения

Как правило, люди не имеют четких представлений о том, как они сами создают описания классов для конкретных объектов. Однако, постоянные попытки найти способ вложения когнитивного обучения в компьютер, могут открыть некоторое понимание процессов человеческого мышления. Проф. Патрик Уинстон сделал большой прорыв в понимании когнитивного обучения. Он предложил метод построения описаний классов, который называется **"Hit-and-Near-Miss"** (совпадений и небольших отклонений).

Для лучшего понимания метода Уинстона, рассмотрим традиционный пример, иллюстрирующий ключевые моменты процедуры. Представьте себе, что вы обучаете кого-то распознаванию арки, состоящей из блоков.

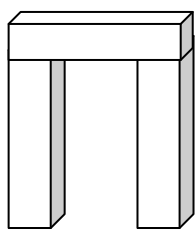


Рис. А

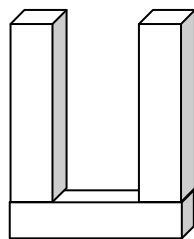


Рис. В

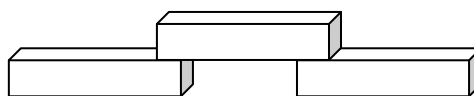


Рис. С

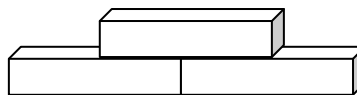


Рис. D

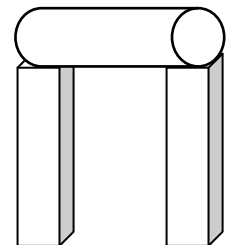


Рис. E

Сначала вы стоите арку, показанную на рис. А. И затем говорите обучаемому, что это есть арка. Далее вы перемещаете верхний блок и помещаете его по отношению к двум другим блокам, как показано на рис.В. И сообщаете обучаемому, что это не арка. Этот шаг означает, что арка

должна иметь перекладину (блок наверху). Далее вы строите арку, как показано на рис. С. И говорите обучаемому, что это арка. Затем вы конструируете структуру, показанную на рис. D. И говорите, что это не арка. Это подразумевает, что опоры арки не соприкасаются. Наконец, вы строите арку, как показано на рис. Е, которая имеет цилиндр наверху вместо бруска, и говорите, что это тоже арка.

На этот момент обучаемый будет иметь следующее описание класса для понятия "арка": она должна иметь брусок или цилиндр наверху двух других брусков, которые не касаются друг друга и поддерживают верхний блок, опорные бруски могут быть как в вертикальном, так и в горизонтальном положении.

Этот пример иллюстрирует ключевое положение: описание класса может быть создано посредством использования специально подобранных примеров объектов (или понятий, или идей, или процедур), которые либо относятся к этому классу, либо отличаются только по нескольким чертам, параметрам.

По мере того, как обучаемому рассказывают, какие объекты относятся к классу, а какие нет, обучаемый может конструировать описание класса, определяя, что общего в объектах, относящихся к классу, и чем отличаются соответствующие примеры-отклонения.

При формировании описания класса роль, которую играют правильные примеры, отличается от той, которую играют примеры-отклонения. Каждый правильный пример делает текущее описание класса более широким и общим. Например, в случае с аркой, когда вы показываете обучаемому арку с лежащими опорами, обучаемый должен обобщить описание того, что является аркой, допуская факт, что эта новая структура есть тоже арка. Описание также обобщается, когда обучаемому показывают арку с цилиндром.

А каждый пример-отклонение вносит ограничение в формирующееся описание. Когда представляется объект, показанный на рис. В, и сообщается, что он не арка, то обучаемый должен определить разницу между объектом и текущим описанием класса. В результате чего обучаемый добавит в описание класса ограничение, что все арки должны иметь блок наверху. Следующий пример-отклонение, в котором опоры касаются друг друга, вносит ограничение, состоящее в том, что опоры не касаются. И это ограничение добавится к описанию класса.

Описание класса эволюционирует, то есть прежние ограничения могут быть обобщены и прежние обобщения могут быть ограничены. Например, когда вы покажете обучаемому арку с цилиндром наверху, он должен модифицировать ограничение, которое требовало, что наверху должен быть брусок, так, что теперь оно требует, что наверху может быть брусок или цилиндр. Если показать несколько арок, каждая из которых имеет наверху блоки разного типа, то обучаемый будет изменять ограничение так, что в конечном счете оно будет требовать, чтобы арка имела что-либо наверху. Формально (то есть в ЭВМ) это можно представить длинным ИЛИ-списком.

Таким образом, каждый правильный пример предоставляет обучаемому серию атрибутов, которые объект этого класса может иметь. Поскольку каждый конкретный объект есть только часть класса, то описание класса, сделанное обучаемым, должно включать все эти объекты. Примеры-отклонения позволяют определить что именно объект данного класса должен иметь или не должен иметь.

Основным моментом в этом процессе обучения является то, что примеры-отклонения должны отличаться от правильных только по малому количеству признаков - предпочтительнее, по одному. Иначе, обучаемый не будет иметь возможности выделить ограничивающий фактор. Например, если вы показали сначала арку, а затем указали на птицу, летающую за окном, и сказали, что птица - это не арка, обучаемый не сможет добавить значимую информацию к описанию класса арка!

8.3. Алгоритм когнитивного обучения

Формальная процедура когнитивного обучения (алгоритм) строится на двух предположениях:

- есть учитель, который представляет примеры и никогда не лжет относительно примеров;
- первый пример должен быть ярким представителем класса, так как он формирует начальную модель.

УКРУПНЕННЫЙ АЛГОРИТМ:

осматриваем пример и формируем первичную модель

ПОВТОРЯТЬ

осматриваем пример

ЕСЛИ (принадлежит классу) ТОГДА обобщение
 ИНАЧЕ ограничение
 ДО ТЕХ ПОР ПОКА (есть примеры)

ПРОЦЕДУРА "ОГРАНИЧЕНИЕ":

установить (определить) различие между примером-отклонением и эволюционирующей моделью (текущим описанием класса)

ЕСЛИ (модель имеет атрибуты не найденные в примере-отклонении)

ТО этот атрибут требуется

ЕСЛИ (пример-отклонение имеет атрибут не найденный в модели)

ТО запретить этот атрибут

ПРОЦЕДУРА "ОБОБЩЕНИЕ":

установить различие между примером и эволюционирующей моделью.
 согласовать различие расширением модели.




8.4. Описание версии программы

Программа использует два списка для каждой классификации. Первый список содержит те атрибуты, которые эта классификация должна иметь (must have), второй список содержит атрибуты, которые она может иметь (may have). Хотя для описания класса необходим только must-have список, но may-have список используется для совершенствования (развития) модели. По мере того как компьютер изучает новые атрибуты, он помещает их первоначально в may_have список и, после анализа примера-отклонения, он переносит их в must_have список, только если он узнает, что они должны быть частью описания класса.

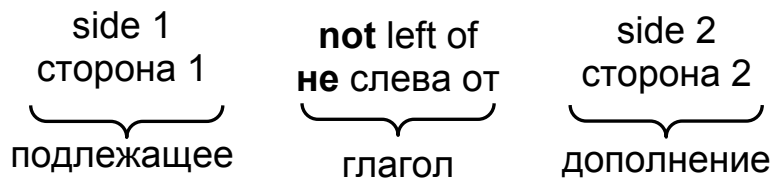
Для этой версии программы все примеры (правильные и отклонения) описываются предложениями, имеющими следующую структуру:

Подлежащее	глагол-сказуемое	дополнение
Subject	verb	object

В соответствие с этим форматом, пример арки может быть представлен:

side 1	left of	side 2
сторона 1	слева от	сторона 2
		
подлежащее	глагол	дополнение

Пример-отклонение формируется добавлением частицы "не" (not) к глаголу правильного примера.



Таким образом, каждый элемент `must_have` и `may_have` списков имеет три составляющих: `subject`, `verb`, `object`.

Каждый раз, как только вводится правильный пример, компьютер помещает его в `may_have` список. Вызывается процедура `generalize()`, которая сравнивает эти три составляющие примера с элементами как `may_have`, так и `must_have` списка. Если найдется элемент, который имеет такие же глагол и дополнение, как пример, но разные подлежащие, то `generalize()` комбинирует два отдельных подлежащих, используя слово "или" и модернизирует соответствующий элемент списка. Если же элемент списка и пример имеют одни и те же подлежащее и глагол, но различные дополнения, то `generalize()` соединяет два дополнения и модернизирует элемент списка. Таким образом, создается новая классификация (модернизируется).

Примеры-отклонения могут следовать не сразу же за правильными примерами, некоторые могут быть вообще пропущены. Если вводится пример-отклонение, то вызывается процедура `restrict()`, которая сравнивает каждый элемент в `may_have` списке с фразой примера-отклонения, исключив предварительно "не". Если обнаружено совпадение, то этот элемент переносится из `may_have` списка в `must_have` список. Этот шаг ограничивает классификацию.

Л/р N3. МАШИННОЕ САМООБУЧЕНИЕ.

ЦЕЛЬ. 1.Изучение основных принципов машинного самообучения.

2.Получение практических навыков работы с программой построения описаний понятий.

ВЫПОЛНЕНИЕ РАБОТЫ.

1.Суть метода (своими словами на 0,5 стр. максимум).

2.Описание правильных примеров и примеров-отклонений в порядке их предъявления.

3.Формат ввода примеров в ЭВМ.

РЕЗУЛЬТАТ: распечатка.

ВЫВОДЫ.

(L)earn, (D)isplay, or (Q)uit ? 1

Enter an exampl.

subject: block

verb: on top of

object: sides

Enter a near-miss (NL to skip).

subject: block

verb: not on top of

object: sides

Enter an exampl.

subject: side 1

verb: left of

object: side 2

Enter a near-miss (NL to skip).

subject: <NL>

Enter an exampl.

subject: cylinder
verb: on top of
object: sides

Enter a near-miss (NL to skip).

subject: <NL>

Enter an exampl.

subject: <NL>

(L)earn, (D)isplay, or (Q)uit ? d

Display of description:

May have:

side 1 left of side 2

Must have:

block or cylinder on top of sides

(L)earn, (D)isplay, or (Q)uit ? 1

Enter an exampl.

subject: sides
verb: made of
object: wood

Enter a near-miss (NL to scip).

subject: sides
verb: made of
object: wood

Enter an exampl.

subject: sides

verb: made of
object: metal

Enter a near-miss (NL to skip).

subject: <NL>

Enter an exampl.

subject: <NL>

(L)earn, (D)isplay, or (Q)uit ? d

Display of description:

May have:

side 1 left of side 2

Must have:

block or cylinder on top of sides

sides made of wood or metal

(L)earn, (D)isplay, or (Q)uit ? q