Сцилла и Харибда ИИ

необъясняемость и вычислительная неэффективность

Дмитрий Виноградов

ВЦ ФИЦ ИУ РАН

4 июля 2022 г.

Случай из жизни: проблема объясняемости

На конференции IJCAI-2017 (Melbourn) очень известный и уважаемый австрийский логик и специалист по ИИ Georg Gottlob, который работает в Technische Universität Wien и в University of Oxford, во время приглашенного доклада рассказал случай, который произошел с ним лично.

В Англии ему отказали в кредите по карте, потому что нейронная сеть обнаружила, что он зарегистрирован в доме, хозяин которого наделал долгов и скрылся.
Это было так, но этим человеком был не сам Готтлоб, а предыдущий владелец, у которого Готтлоб купил дом.
Но как это можно было обнаружить?

Символьная ИИ и Машинное обучение

Символьная Имитация Интеллекта

1. Индуктивное логическое программирование
2. Индуктивная логика (ДСМ-метод)
3. Байесовские сети доверия
4. Деревья решений

Машинное обучение

1. Нейросети: рекуррентные, сверточные, LTSM, трансформеры
2. Машина опорных векторов
3. Комитеты: bagging, boosting, случайный лес

	Символьная ИИ	Машинное обучение
Данные	мало	много
Сложность	высокая	низкая
Гипотезы	правила	параметры модели
Объясняемость	высокая	низкая
Передача знаний	есть	нет

Индуктивное логическое программирование

Расширенный обзор на основе пленарного доклада несостоявшейся IJCAI-2020: Andrew Cropper, Sebastijan Dumančić: Inductive logic programming at 30: a new introduction

Полезные статьи:

Plotkin, G. Automatic Methods of Inductive Inference. PhD Thesis, Edinburgh University (1971)

Shapiro, E.Y. Algorithmic Program Debugging. MIT Press, Cambridge (MA), 1983

Quinlan, J.R. Learning Logical Definitions from Relations // Machine Learning, 5, 239-266 (1990)

Muggleton, S.; Buntine, W.L. Machine Invention of First Order Predicate by Inverting Resolution // Proc. 5th International Conference on Machine Learning, Morgan Kaufmann, 339-352 (1988)

Muggleton, S. Inductive Logic Programming // New Generation Computing, 8(4), 295-318 (1991)

De Raedt, L.; Dehaspe, L. Clausal Discovery // Machine Learning, 26(2-3), 99-146 (1997)

Gottlob, G.; Leone, N.; Scarcello, F. On the Complexity of Some Inductive Logic Programming Problems // Proc. 7th International Workshop on Inductive Logic Programming, LNCS 1297, Springer 17-32 (1997)

Напоминание: язык логики предикатов

Переменная - строка, начинающая с заглавной буквы.
n-арная **функция** - строка, начинающая с прописной буквы, после которой в скобках идет список из *n* аргументов.
Константа - 0-арная функция.
Терм - переменная, константа, или функция, аргументы которой полностью заполнены термами. Для любого терма можно построить дерево разбора. Терм называется **осно'вным**, если не содержит переменных.
Подстановка - отображение переменных в термы. Значение терма при подстановке соответствует подклеиванию дерева разбора подставляемого терма вместо всякого вхождения соответствующей переменной.
n-арный **предикат** - строка, начинающая с прописной буквы, после которой в скобках идет список из *n* аргументов.
Атом - предикат, аргументы которого полностью заполнены термами. Он называется **основным**, если не содержит переменных.
not - унарная операция **отрицания**, понимаемая через неудачу вывода.
$\mathrm{not}\,atom$ - отрицание атома *atom*.
Положительный литерал - атом, отрицание атома - **отрицательный**.

Напоминание: язык логических программ

Программа - список правил. **Правило** - выражение вида $head \leftarrow body_1, \ldots, body_m$, понимаемое как

$$\forall X_1 \ldots \forall X_n \left[body_1 \& \ldots \& body_m \supset head \right],$$

где $head$ - некоторый атом, называемый **головой правила**, а $body_1, \ldots, body_m$ - список литералов, называемый **телом правила**.
Факт - правило с пустым телом. **Хорновское правило** - правило, в теле которого нет отрицательных литералов.
Программа называется **хорновской**, если все ее правила хорновские,
Запрос (к программе) - выражение вида $\leftarrow atom_1, \ldots, atom_k$.
Отрицание как неудача: для каждого правила $p(t_1, \ldots, t_n) \leftarrow atom_1, \ldots, atom_m, \mathrm{not} atom'_1, \ldots, \mathrm{not} atom'_k$ в **(LIFO) стек запросов** добавляются запросы $\leftarrow atom'_1, \ldots, \leftarrow atom'_k$.
Если вывод запроса $\leftarrow atom'_j$ $(1 \leq j \leq k)$ заканчивается успехом, то удаляем правило и запросы $\leftarrow atom'_{j+1}, \ldots, \leftarrow atom'_k$. Если все выводы запросов заканчиваются неудачами, то оставляют правило $p(t_1, \ldots, t_n) \leftarrow atom_1, \ldots, atom_m$.

Семантика логических программ

(Эрбранов) универсум имеет носителем множество основных термов H, а n-арная функция $f : H^n \to H$ отображает основные термы $t_1, \ldots, t_n$ в терм $f(t_1, \ldots, t_n)$.
Оценка - отображение $\theta : Vars \to H$ переменных $Vars$ в универсум (частный случай подстановки). **Значение** $t\theta \in H$ терма t при оценке θ - основный терм, получаемый как значение этого терма при соответствующей подстановке.
(Эрбранова) интерпретация I - подмножество основных атомов, которое задает значения каждого n-арного предиката p на наборе основных термов $t_1, \ldots, t_n$

$$p(t_1, \ldots, t_n) = \textbf{true} \Leftrightarrow p(t_1, \ldots, t_n) \in I$$

Интерпретация I называется **моделью** программы π, если для каждого правила $head \leftarrow body_1, \ldots, body_m \in \pi$ и каждой оценки θ истинна импликация $body_1\theta \& \ldots \& body_m\theta \supset head\theta$, где $p(t_1, \ldots, t_n)\theta = p(t_1\theta, \ldots, t_n\theta)$.

Задача

Доказать, что для хорновской программы π пересечение двух моделей π будет моделью π.

Наименьшая модель хорновской программы

Определим $I_\pi^0 = \{p(t_1\theta, \ldots, t_n\theta) : p(t_1, \ldots, t_n) \leftarrow \ \in \pi, \theta : Vars \to H\}$ - множество истинных фактов.
Для хорновской программы π добавим следствия на шаге $s+1$

$$I_\pi^{s+1} = \{p(t_1\theta, \ldots, t_n\theta) : p(t_1, \ldots, t_n) \leftarrow atom_1, \ldots, atom_m \in \pi, \\ \theta : Vars \to H, atom_1\theta \in I_\pi^s, \ldots, atom_m\theta \in I_\pi^s\}.$$

Минимальная интерпретация для хорновской программы π равна

$$I_\pi^\infty = \cup_{s=0}^\infty I_\pi^s.$$

Задача

Доказать, что для хорновской программы π минимальная интерпретация I_π^∞ будет (единственной) наименьшей моделью π.

Задача

Доказать, что у программы $\{p_1 \leftarrow \mathrm{not} p_2, p_2 \leftarrow \mathrm{not} p_1\}$ две минимальные модели $\{p_1\}$ и $\{p_2\}$, а у $\{p_1 \leftarrow \mathrm{not} p_2\}$ есть наименьшая модель $\{p_1\}$.

Устойчивые модели произвольной программы

Gelfond M.; Lifschitz V. The stable model semantics for logic programming // Proceedings of the 5th ICLP, 1070–1080 (1988)

Рассмотрим программу π и зафиксируем эбранову интерпретацию I.

Для правила $p(t_1, \ldots, t_n) \leftarrow atom_1, \ldots, atom_m, \mathrm{not}\, atom'_1, \ldots, \mathrm{not}\, atom'_k$ программы π и оценки $\theta : Vars \to H$

1. если найдется такое j, что $atom'_j\theta \in I$, то удалим это правило;
2. иначе заменим его на правило $p(t_1\theta, \ldots, t_n\theta) \leftarrow atom_1\theta, \ldots, atom_m\theta$.

Полученную хорновскую программу обозначим π^I.

Интерпретацию I назовем **устойчивой относительно** π, если I совпадает с наименьшей моделью программы π^I.

Теорема

Устойчивая относительно π интерпретация является минимальной моделью программы π^I, следовательно, и моделью π.

Задача

Доказать, что у программы $\{p_1 \leftarrow \mathrm{not}\, p_2, p_2 \leftarrow \mathrm{not}\, p_1\}$ обе минимальные модели $\{p_1\}$ и $\{p_2\}$ устойчивы, а у $\{p_1 \leftarrow \mathrm{not}\, p_1\}$ нет устойчивой модели.

Индуктивное обучение через вывод

Пусть B - программа общих знаний. E^+ - множество положительных фактов. E^- - множество отрицательных фактов. Множество правил H называется **гипотезой через вывод**, если

1. (полнота): для каждого $atom \in E^+$ выполнено $B \cup H \models atom$, т.е. запрос $\leftarrow atom$ успешно выводится из $B \cup H$.
2. (совместность): для каждого $atom \in E^-$ выполнено $B \cup H \not\models atom$, т.е. вывод запроса $\leftarrow atom$ из $B \cup H$ заканчивается неудачей.

Задача

Доказать, что $B \cup H \not\models atom$ эквивалентно выполнимости $B \cup H \cup \{\mathrm{not} atom\}$.

При наличии 2-арной функции:

Теорема (Тэрнлунд)

Существование модели у хорновской программы с функциями неразрешимо.

Индуктивное обучение для Datalog

Grädel, Erich; Kolaitis, Phokion G.; Libkin, Leonid; Maarten, Marx; Spencer, Joel; Vardi, Moshe Y.; Venema, Yde; Weinstein, Scott. Finite model theory and its applications, Berlin: Springer-Verlag, 2007

Datalog - декларативный фрагмент Prolog без функций, но с константами и с ограничениями на отрицание. Тогда унификация - замена констант на переменную, а выполнимость сводится к поиску моделей в логике высказываний.

Теорема (Даулинг, Гальер)

Проблема выполнимости хорновской программы в логике высказываний решается за линейное время от длины программы.

Массовая задача называется **NP-трудной**, если к ней полиномиально сводится (по Тьюрингу) любая задача из NP.

Дизъюнкция не более трех литералов называется **3-дизъюнктом**. **3-КНФ** - это конъюнкция 3-дизъюнктов (в любом числе).

Теорема (Кук)

Проблема 3CNF-SAT выполнимости 3-КНФ является NP-трудной.

Синтез познавательных процедур

ДСМ-метод автоматического порождения гипотез: Логические и эпистемологические основания (ред.: Аншаков О.М., Финн В.К.). М.: Эдиториал УРСС, 2009.

1. Индуктивное обобщение обучающих примеров в гипотезы о причинах целевого свойства.
2. Предсказание по аналогии.
3. Абдуктивное принятие гипотез на основании объяснения свойств обучающей выборки.
4. Пополнение обучающей выборки аналогами необъясненных обучающих примеров или добавление признаков.
5. Обнаружение эмпирических закономерностей.

Ganter, Bernhard; Wille, Rudolf. Formal Concept Analysis. Berlin: Springer, 1999.
Анализ формальных понятий - современный раздел теории решеток - обеспечил эффективные средства вычисления сходств и установил достаточность представления объектов битовыми строками фиксированной длины с побитовым умножением.

Описать окружность около выпуклого многоугольника

Многоугольники	**goal**	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
Равносторонний треугольник	1	1	0	0	1	1	0	1	1
Прямоугольный треугольник	1	1	0	1	0	0	0	0	0
Квадрат	1	0	1	1	1	1	1	1	1
Ромб	0	0	1	0	1	1	1	1	0
Прямоугольник	1	0	1	1	1	0	1	1	1
Параллелограмм	0	0	1	0	1	0	1	1	0
Равнобедренная трапеция	1	0	1	0	1	0	1	1	0
Прямоугольная трапеция	0	0	1	1	0	0	1	1	0
Равнобедренный треугольник	?	1	0	0	1	0	0	1	0

f_1: треугольник; f_2: четырехугольник; f_3: есть прямой угол; f_4: пара сторон равна; f_5: все стороны равны; f_6: есть пара параллельных сторон; f_7: пара углов равна; f_8: все углы равны.

- Гипотеза $\{f_1\}$ - вокруг любого треугольника можно
- Гипотеза $\{f_4, f_5, f_7, f_8\}$ - вокруг правильного многоугольника можно
- Гипотеза $\{f_2, f_3, f_4, f_6, f_7, f_8\}$ - вокруг прямоугольника можно
- Гипотеза $\{f_2, f_4, f_6, f_7\}$ имеет ромб как контр-пример!

Теорема об экспоненциальности числа сходств

Теорема (фольклор)

Существует такая обучающая выборка из n элементов, что число сходств будет равно 2^n.

Доказательство.

Пусть $O = \{o_1, o_2, \ldots, o_n\}$ - множество объектов, а $F = \{f_1, f_2, \ldots, f_n\}$ - множество признаков, и обучающая выборка такова:

$O \mid F$	f_1	f_2	$\ldots$	f_n
o_1	0	1	$\ldots$	1
o_2	1	0	$\ldots$	1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
o_n	1	1	$\ldots$	0

Ясно, что любая пара $\langle O \setminus \{o_{j_1}, \ldots, o_{j_k}\}, \{f_{j_1}, \ldots, f_{j_k}\}\rangle$ будет сходством, поэтому мы имеем Булеву алгебру всех 2^n подмножеств (=битовых строк). □

Вероятно приближенное-корректное обучение

Вапник В.Н., Червоненкис А.Я. Теория распознавания образов (статистические проблемы обучения). М.: Наука, 1974.
Valiant L.G. A Theory of Learnable // Communications of the ACM, 27(11), 1134–1142 (1984)
Процедура обучения должна

1. выбрать язык описания объектов (задать область X_n обучающих примеров);
2. научиться оценивать снизу число m случайных обучающих примеров $S = \{\langle \vec{x}_1, c(\vec{x}_1)\rangle, \ldots, \langle \vec{x}_m, c(\vec{x}_m)\rangle\}$, чтобы из них можно с заранее выбранной вероятностью $\geq 1 - \delta$ получать правило $h : X_n \to \{0, 1\}$, ошибающееся в не более чем заданной доле $\varepsilon > 0$ случаев (для любого $c : X_n \to \{0, 1\}$);
3. придумать алгоритм $A(S) = h_S$ нахождения такого правила за ограниченное (полиномом от n, $\frac{1}{\varepsilon}$ и $-\log \delta = \log\left(\frac{1}{\delta}\right)$) число шагов;
4. реализовать предсказание тестовых примеров $\vec{x} \in X_n$ с помощью найденного правила h_S.

Пример ВПК-обучения

Выбирают преемника стареющему вождю. Кандидат должен отправиться с шаманом в джунгли с целью обучиться правилу, какие растения лекарственные, а какие нет. Перед выходом ученик должен сказать, сколько растений m он хочет изучить. Если это число окажется неразумно большим, то его дисквалифицируют.

Для очередного растения $\vec{x}_t \in X_n$ шаман говорит, лекарственное ли это растение (положителен ли $c(\vec{x}_t) = 1$ этот пример) или нет (отрицательный пример $c(\vec{x}_t) = 0$). Все примеры $\vec{x}_t \in X_n$ появляются независимо и с одинаковой вероятностью. Когда они увидят объявленное учеником число m обучающих примеров $S = \{\langle \vec{x}_1, c(\vec{x}_1)\rangle, \ldots, \langle \vec{x}_m, c(\vec{x}_m)\rangle\}$, шаман с учеником возвращаются домой.

Затем ученик должен найти правило $h = h_S : X_n \to \{0, 1\}$, как распознавать лекарственные растения. Эффективность этого шага важна!

Наконец, ученик с шаманом отправляются в (те же самые) джунгли. Ученик обязан классифицировать первое встретившееся растение (тестовый пример) $\vec{x} \in X_n$, который появляется независимо от обучающей выборки и с распределением, совпадающим с обучающим. Если классификация ученика не совпадет с классификацией шамана (=неверна) $h_S(\vec{x}) \neq c(\vec{x})$, то ученика дисквалифицируют.

ВПК-обучение конъюнктивным понятиям

Определение

Пусть каждый объект x описывается множеством бинарных признаков $f_1, \ldots, f_n$, т.е. множество объектов $X \subseteq \{0,1\}^n$, где n - число признаков. Чтобы узнать значение признака на объекте $\vec{x}$, введем булевские переменные $p_1, \ldots, p_n : \{0,1\}^n \to \{0,1\}$, соответствующие проецированию на компоненты. Переменные вместе с их отрицаниями назовем ***литералами****.*

Понятие *– подмножество объектов $\{\vec{x} \in X : c(\vec{x}) = 1\}$, задаваемое булевой функцией $c : X \to \{0,1\}$, называемой* ***индикатором****. Если $c(\vec{x}) = 1$, то объект $\vec{x} \in X$ называется положительным примером понятия, в противном случае – отрицательным примером. Если индикатор представим конъюнкцией литералов $c = \{l'_{j_1}, \ldots, l'_{j_r}\}$, то такое понятие назовем* ***конъюнктивным****.*

Гипотеза *– список литералов $h = \{l_{i_1}, \ldots, l_{i_k}\}$, конъюнкция которых представляет кандидата на обучаемое понятие. Предсказание с помощью гипотезы осуществляется по булевой функции $h = l_{i_1} \& \ldots \& l_{i_k}$ в качестве индикатора.*

Алгоритм пересечения

Задача

Проверить, что, отождествляя объект $\vec{x} \in \{0,1\}^n$ с максимально непротиворечивым множеством литералов $\{l_1, \ldots, l_n\}$ ($l_j \in \{\neg f_j, f_j\}$), гипотеза h доопределяет пример $\vec{x}$ положительно, если и только если $h \subseteq \vec{x}$.

Алгоритм (пересечения)

1. *Перечислим элементы $S = \{\langle \vec{x}_1, c(\vec{x}_1)\rangle, \ldots, \langle \vec{x}_m, c(\vec{x}_m)\rangle\}$ в некотором порядке.*
2. *Положим $h_0 = L = \{\neg p_1, p_1, \ldots, \neg p_n, p_n\}$ (множество всех литералов).*
3. *По порядку проверяем $h_j(\vec{x}_{j+1}) \neq c(\vec{x}_{j+1})$. Если ошибка не происходит, то гипотеза не изменяется. Когда случается ошибка, из гипотезы удаляются те литералы, которых нет в неправильно предсказанном примере:*

$$h_j(\vec{x}_{j+1}) \neq c(\vec{x}_{j+1}) \Rightarrow h_{j+1} = h_j \setminus (L \setminus \vec{x}_{j+1}).$$

Легко проверить, что $h_j \setminus (L \setminus \vec{x}_{j+1}) = h_j \cap \vec{x}_{j+1}$ как множества литералов, что и дает название этому алгоритму.

Вспомогательные леммы

Лемма

Никакой литерал из обучаемого конъюнкта $c = \{l'_{i_1}, \ldots, l'_{i_k}\}$ *не удаляется алгоритмом пересечения, т.е.* $c \subseteq h_j$ *для всех* j.

Лемма

Алгоритм пересечения может ошибаться только на положительных примерах, т.е. возможен только случай $h_j(\vec{x}_j) = 0 \neq 1 = c(\vec{x}_j)$.

Определение

Литерал l *назовем* ***плохим***, *если* $\mathbb{P}\left[c(\vec{x}) = 1 \& l \notin \vec{x}\right] > \frac{\varepsilon}{2n}$.

Лемма

Если h *не содержит плохих литералов, то* $\mathbb{P}\left[h(\vec{x}) \neq c(\vec{x})\right] \leq \varepsilon$.

Выборки большого объема

Определение

Объем m выборки $S = \{\langle \vec{x}_1, c(\vec{x}_1)\rangle, \ldots, \langle \vec{x}_m, c(\vec{x}_m)\rangle\}$ назовем ***большим****, если* $m \geq \frac{2n\cdot[\ln(2n)-\ln(\delta)]}{\varepsilon}$.

Лемма

Если $m \geq \frac{2n\cdot[\ln(2n)-\ln(\delta)]}{\varepsilon}$, *то* $2n \cdot \left(1 - \frac{\varepsilon}{2n}\right)^m \leq \delta$.

Доказательство.

Из-за выпуклости функции $y = e^{-x}$ имеем, что касательная $y = 1 - x$ к $y = e^{-x}$ в $y_0 = 0$ лежит ниже самой функции.
Подставим $\frac{\varepsilon}{2n}$ в неравенство $1 - x \leq e^{-x}$ и возведем в степень m.
Получаем $\left(1 - \frac{\varepsilon}{2n}\right)^m \leq e^{-m\cdot\frac{\varepsilon}{2n}}$.
Остальное - простые вычисления. □

ВПК-обучаемость конъюнктивным понятиям

Теорема

Алгоритм пересечения доказывает эффективную ВПК-обучаемость конъюнктивным понятиям.

Доказательство.

Докажем, что при большом объеме выборки m вероятность того, что в гипотезе h останется хоть один плохой литерал, не превзойдет δ.
Пусть l – плохой литерал. Тогда $\mathbb{P}[l$ сохранится после одного примера$] =$
$= 1 - \mathbb{P}[l$ удалится первым примером $] \leq 1 - \frac{\varepsilon}{2n}$.
Поэтому $\mathbb{P}[l$ сохранится после m примеров$] \leq \left(1 - \frac{\varepsilon}{2n}\right)^m$.
Так как всего литералов ровно $2n$, то утверждение теоремы следует из неравенства Буля $\mathbb{P}\left[\cup_{j=1}^{k} A_j\right] \leqslant \sum_{j=1}^{k} \mathbb{P}[A_j]$ и предыдущей леммы.
Ясно, что $\frac{2n \cdot [\ln(2n) - \ln(\delta)]}{\varepsilon}$ мажорируется полиномом от n, $\frac{1}{\varepsilon}$ и $-\log \delta$, и число шагов алгоритма пересечения для обработки одного примера линейно зависит от n. □

Кандидаты в гипотезы о причинах

Обучающую выборку нужно понимать как бинарное отношение между элементами множества O, которые мы называем *именами объектов*, и элементами множества F, которые мы называем *признаками*.

$O \times F$	f_1	$\ldots$	f_{j_1}	f_{j_1+1}	$\ldots$	$f_{j_{m-1}}$	$\ldots$	f_{j_m}	$\ldots$	f_n
o_1	0	$\ldots$	0	0	$\ldots$	1	$\ldots$	0	$\ldots$	0
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
o_{i_1}	0	$\ldots$	**1**	1	$\ldots$	**1**	$\ldots$	**1**	$\ldots$	1
o_{i_1+1}	0	$\ldots$	0	0	$\ldots$	1	$\ldots$	1	$\ldots$	0
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
o_{i_l-1}	1	$\ldots$	**1**	0	$\ldots$	**1**	$\ldots$	**1**	$\ldots$	0
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
o_{i_l}	1	$\ldots$	**1**	1	$\ldots$	**1**	$\ldots$	**1**	$\ldots$	0
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
o_k	0	$\ldots$	1	0	$\ldots$	0	$\ldots$	0	$\ldots$	0

Формальное определение кандидатов

Для подмножества $A \subseteq O$ объектов его *общим фрагментом* называется подмножество $A' = \{f \in F : \forall o \in A\, [oIf]\} \subseteq F$. Полагаем $\emptyset' = F$.
На самом деле, это определение совпадает с последовательным вычислением побитового умножения строк, соответствующих отобранным во множество A объектов.
Для подмножества $B \subseteq F$ признаков его *сходством* называется подмножество $B' = \{o \in O : \forall f \in B\, [oIf]\} \subseteq O$. Полагаем $\emptyset' = O$.
Операции $' : 2^O \to 2^F$ и $' : 2^F \to 2^O$ называются *полярами* и задают *соответствие Галуа*.

Определение

Пару $\langle A, B \rangle$ *назовем* ***кандидатом***, *если* $A = B' \subseteq O$ *и* $B = A' \subseteq F$.
Подмножество $A \subseteq O$ *называем* ***списком родителей*** *кандидата, а* $B \subseteq F$ - ***(общим) фрагментом*** *кандидата.*
Равенство $A = B'$ *соответствует принципу* ***исчерпываемости***.

Операции «Замыкай-по-одному»

Идея: порождать только случайную подвыборку кандидатов с помощью случайных блужданий по решетке кандидатов.
Операция **замыкай-по-одному-вниз** на кандидате $\langle A, B\rangle$ и объекте $o \in O$ порождает кандидат

$$CbODown(\langle A, B\rangle, o) = \langle A, B\rangle \wedge \langle \{o\}'', \{o\}'\rangle = \langle (A \cup \{o\})'', B \cap \{o\}'\rangle.$$

Операция **замыкай-по-одному-вверх** на кандидате $\langle A, B\rangle$ и признаке $f \in F$ порождает кандидат

$$CbOUp(\langle A, B\rangle, f) = \langle A, B\rangle \vee \langle \{f\}', \{f\}''\rangle = \langle A \cap \{f\}', (B \cup \{f\})''\rangle.$$

Ускорение вычислений:
Если $o \in A$, то $CbODown(\langle A, B\rangle, o) = \langle A, B\rangle$. Аналогично, если $f \in B$, то $CbOUp(\langle A, B\rangle, f) = \langle A, B\rangle$.
В случае Булевой алгебры вычисления упрощаются
Если $o_j \notin A$, то $CbODown(\langle A, B\rangle, o_j) = \langle A \cup \{o_j\}, B \setminus \{f_j\}\rangle$, и если $f_j \notin B$, то $CbOUp(\langle A, B\rangle, f_j) = \langle A \setminus \{o_j\}, B \cup \{f_j\}\rangle$.

Алгоритм монотонной цепи Маркова

Алгоритм

Data: *множество обучающих (+)-примеров; внешние функции* $CbOUp(\ ,\ )$ *и* $CbODown(\ ,\ )$ *операций «замыкай-по-одному»*
Result: *кандидат* $\langle A, B\rangle$
$O :=$ *(+)-примеры,* $F :=$ *признаки;* $I \subseteq O \times F$ *- обучающая выборка для (+)-примеров* $A := O$; $B = O'$; $R := O \cup F$
for $(i := 0; i < T; i = i + 1)$ **do**
  Выбираем случайный элемент $r \in R$
  if $(r \in O)$ **then**
    $\langle A, B\rangle := CbODown(\langle A, B\rangle, r)$
  end
  else
    $\langle A, B\rangle := CbOUp(\langle A, B\rangle, r)$
  end
end

Случай Булевой алгебры

Определение

Кандидаты $\langle A_1, B_1 \rangle$ *и* $\langle A_2, B_2 \rangle$ *в случае Булевой алгебры называются* ***соседними****, если* $|(A_1 \setminus A_2) \cup (A_2 \setminus A_1)| = |(B_1 \setminus B_2) \cup (B_2 \setminus B_1)| = 1$.

Задача

Доказать, что для обучающей выборки для Булевой алгебры алгоритм монотонной цепи Маркова с вероятностью $\frac{1}{2}$ *остается на месте и с вероятностью* $\frac{1}{2n}$ *переходит из текущего кандидата на одного из n его соседей, то есть представляет собой ленивое случайное блуждание по соответствующему гиперкубу.*

Спаривающая цепь Маркова

Состоянием изменяемых переменных в цикле (= состоянием цепи Маркова) является упорядоченная пара кандидатов $\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle$.

Определение

Порядок *на кандидатах:* $\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle$, *если* $B_1 \subseteq B_2$.

Первоначально меньший кандидат совпадает с наименьшим кандидатом $Min := \langle O, O'\rangle$, а больший - с наибольшим $Max := \langle F', F\rangle$.
В цикле к обоим кандидатам применяется одна и та же операция *CbODown* с выбранным объектом, или *CbOUp* с выбранным признаком.

Лемма

Для всякой упорядоченной пары кандидатов $\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle$ *и любого* $o \in O$ *имеем* $CbODown(\langle A_1, B_1\rangle, o) \leq CbODown(\langle A_2, B_2\rangle, o)$.
Для всякой упорядоченной пары кандидатов $\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle$ *и любого* $f \in F$ *имеем* $CbOUp(\langle A_1, B_1\rangle, f) \leq CbOUp(\langle A_2, B_2\rangle, f)$.

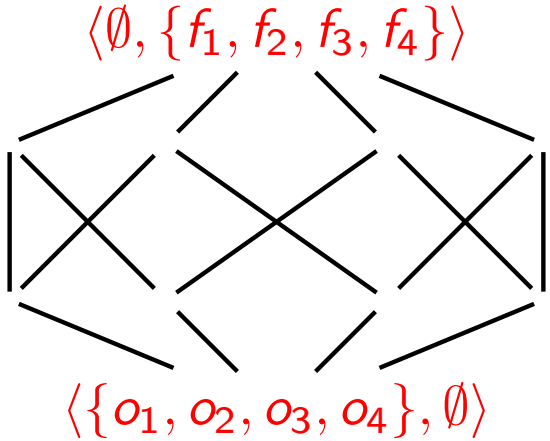
Процесс останавливается, когда меньший кандидат совпадет в большим. Тогда этот общий кандидат и выдается алгоритмом 3.

Алгоритм спаривающей цепи Маркова

Алгоритм

Data: *множество обучающих (+)-примеров; внешние функции* $CbOUp(\ ,\ )$ *и* $CbODown(\ ,\ )$ *операций «замыкай-по-одному»*
Result: *кандидат* $\langle A, B\rangle$
$O :=$ *(+)-примеры,* $F :=$ *признаки;* $I \subseteq O \times F$ *- обучающая выборка из (+)-примеров* $R := O \cup F$; $Min := \langle O, O'\rangle$; $Max := \langle F', F\rangle$
while $(Min \neq Max)$ **do**
  Выбираем случайный элемент $r \in R$
  if $(r \in O)$ **then**
    $Min := CbODown(Min, r)$; $Max := CbODown(Max, r)$
  end
  else
    $Min := CbOUp(Min, r)$; $Max := CbOUp(Max, r)$
  end
end
$\langle A, B\rangle := Min$

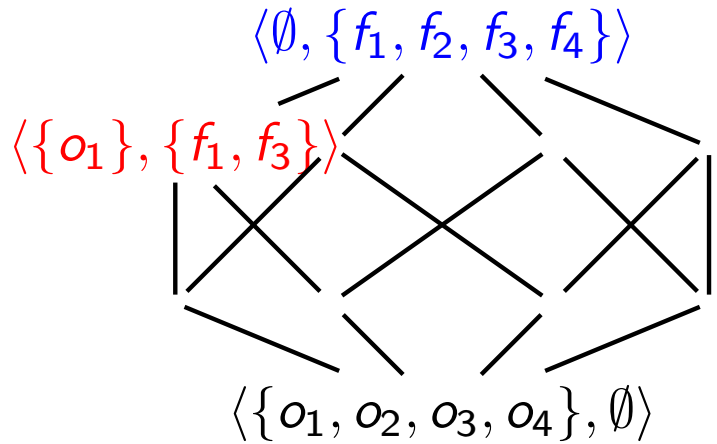
Как работает спаривающая цепь Маркова: шаг 0



Пример

верхний	f_1	f_2	f_3	f_4	нижний	f_1	f_2	f_3	f_4
o_1	1	0	1	0	o_1	***1***	***0***	***1***	***0***
o_2	1	0	0	1	o_2	***1***	***0***	***0***	***1***
o_3	0	1	1	0	o_3	***0***	***1***	***1***	***0***
o_4	0	1	0	1	o_4	***0***	***1***	***0***	***1***

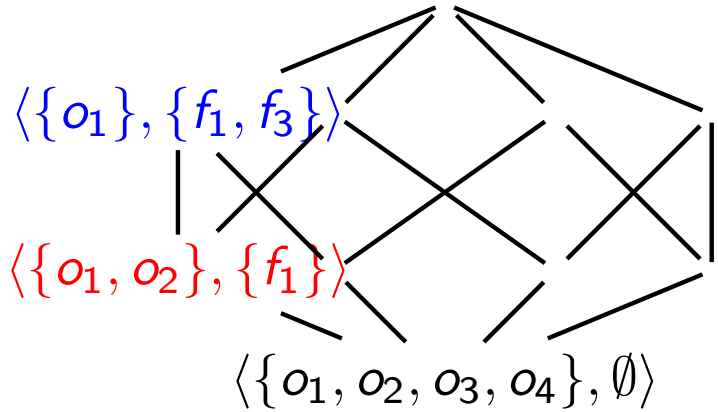
Как работает спаривающая цепь Маркова: выбор o_1



Пример

верхний	f_1	f_2	f_3	f_4	нижний	f_1	f_2	f_3	f_4
o_1	***1***	***0***	***1***	***0***	o_1	***1***	***0***	***1***	***0***
o_2	1	0	0	1	o_2	***1***	***0***	***0***	***1***
o_3	0	1	1	0	o_3	***0***	***1***	***1***	***0***
o_4	0	1	0	1	o_4	***0***	***1***	***0***	***1***

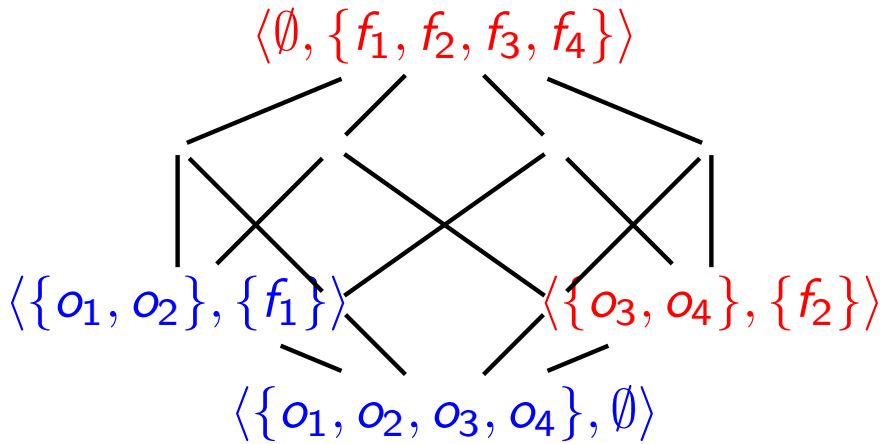
Как работает спаривающая цепь Маркова: выбор o_2



Пример

верхний	f_1	f_2	f_3	f_4	нижний	f_1	f_2	f_3	f_4
o_1	***1***	***0***	***1***	***0***	o_1	***1***	***0***	***1***	***0***
o_2	***1***	***0***	***0***	***1***	o_2	***1***	***0***	***0***	***1***
o_3	0	1	1	0	o_3	***0***	***1***	***1***	***0***
o_4	0	1	0	1	o_4	***0***	***1***	***0***	***1***

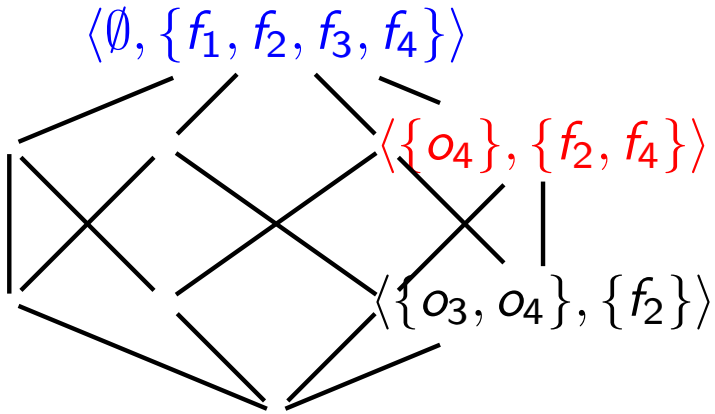
Как работает спаривающая цепь Маркова: выбор f_2



Пример

верхний	f_1	f_2	f_3	f_4	нижний	f_1	f_2	f_3	f_4
o_1	1	0	1	0	o_1	1	0	1	0
o_2	1	0	0	1	o_2	1	0	0	1
o_3	0	1	1	0	o_3	***0***	***1***	***1***	***0***
o_4	0	1	0	1	o_4	***0***	***1***	***0***	***1***

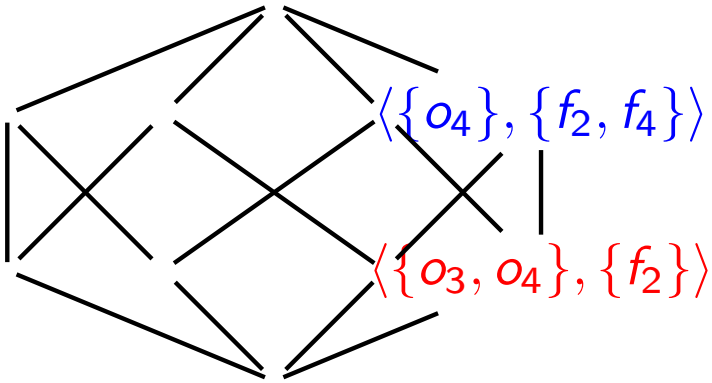
Как работает спаривающая цепь Маркова: выбор o_4



Пример

верхний	f_1	f_2	f_3	f_4	нижний	f_1	f_2	f_3	f_4
o_1	1	0	1	0	o_1	1	0	1	0
o_2	1	0	0	1	o_2	1	0	0	1
o_3	0	1	1	0	o_3	***0***	***1***	***1***	***0***
o_4	***0***	***1***	***0***	***1***	o_4	***0***	***1***	***0***	***1***

Как работает спаривающая цепь Маркова: выбор o_3



Пример

верхний	f_1	f_2	f_3	f_4	нижний	f_1	f_2	f_3	f_4
o_1	1	0	1	0	o_1	1	0	1	0
o_2	1	0	0	1	o_2	1	0	0	1
o_3	***0***	***1***	***1***	***0***	o_3	***0***	***1***	***1***	***0***
o_4	***0***	***1***	***0***	***1***	o_4	***0***	***1***	***0***	***1***

Алгоритмы как цепи Маркова

Теорема

Алгоритмы монотонной и спаривающей цепей соответствуют цепям Маркова.

Доказательство.

Операция *CbODown* (и *CbOUp*) относительно фиксированного объекта (признака) определяет детерминистскую функцию из множества кандидатов в себя. Поэтому каждая строка соответствующей матрицы перехода имеет единицу в одной из ячеек и нули в остальных местах. Очевидно, что такая матрица является стохастической.
Операции *CbODown* и *CbOUp* на кандидатах соответствуют (равномерно) взвешенной сумме таких матриц. Поэтому такая сумма является стохастической матрицей, что доказывает случай монотонной цепи Маркова.
Одновременное применение операций *CbODown* и *CbOUp* к упорядоченной паре кандидатов соответствует тензорному (Кронекеровскому) произведению матрицы преобразований саму на себя с ограничением на инвариантное подпространство таких пар (то, что это подпространство инвариантно, следует из предыдущей леммы). Поэтому матрица преобразования, соответствующая спаренному алгоритму, задает конечную цепь Маркова. □

Остановка спаривающей цепи Маркова

Определение

Состояние вида $\langle A_1, B_2 \rangle = \langle A_2, B_2 \rangle$ спаривающей цепи Маркова для совпадающей пары кандидатов называется ***эргодическим****. Состояние вида $\langle A_1, B_1 \rangle < \langle A_2, B_2 \rangle$ называется* ***невозвратным****.*

Теорема

Вероятность того, что состояние цепи Маркова в момент времени t окажется невозвратным, стремится к нулю, когда $t \to \infty$. □

Лемма (Бореля-Кантелли)

Если ряд $\sum_{j=1}^{\infty} \mathbb{P}(A_j) < \infty$ для вероятностей событий A_j сходится, то с вероятностью 1 происходит только конечное число событий A_j.

Задача

Доказать, что алгоритм спаривающей цепи Маркова останавливается с вероятностью 1.

Длина траекторий для Булевой алгебры

Теорема

Среднее время $\mathbb{E}[T]$ *склеивания для* n*-мерного гиперкуба равно*

$$\mathbb{E}[T] = \frac{n}{n} + \frac{n}{n-1} + \ldots + \frac{n}{1} = n \cdot (\sum_{j=1}^{n} \frac{1}{j}) \approx n \cdot \ln(n) + n \cdot \gamma + \frac{1}{2} \qquad (1)$$

При $n = 32$ средняя длина $\sum_{j=1}^{n} \frac{n}{j} \leq 130$.

Лемма (неравенство Чебышова)

$\mathbb{P}[\mid T - \mathbb{E}[T] \mid \geq \varepsilon] \leq \mathbb{D}[T]/\varepsilon^2$. □

Теорема

$\mathbb{P}[T \geq (1+\varepsilon) \cdot n \cdot \ln(n)] \to 0$ *при* $n \to \infty$ *для любого* $\varepsilon > 0$.

При $n = 32$ Булев гиперкуб содержит $4,294,967,296$ вершин. 1000 траекторий спаривающей цепи Маркова породят около $260,000$ элементов гиперкуба.

Алгоритм индуктивного обобщения

Алгоритм

Data: *множество обучающих (+)- и (-)-примеров; число N ВКФ-гипотез*
Result: *выборка S ВКФ-гипотез*
$O :=$ *(+)-примеры,* $F :=$ *признаки;* $I \subseteq O \times F$ *- обучающая выборка для (+)-примеров;* $C :=$ *(-)-примеры;* $S := \emptyset$; $i := 0$
while $(i < N)$ **do**
  породить кандидата $\langle A, B \rangle$ *с помощью цепи Маркова*
  $hasObstacle :=$ ***false***
  for $(c \in C)$ **do**
    if $(B \subseteq \{c\}')$ **then**
      $hasObstacle :=$ ***true***
    end
  end
  if $(hasObstacle =$ ***false***$)$ **then**
    $S := S \cup \{\langle A, B \rangle\}$
    $i := i + 1$
  end
end

Алгоритм предсказания по аналогии

Алгоритм

Data: *расширенная выборка* S^+ *ВКФ-гипотез, файл тестовых* (?)*-примеров*
Result: *предсказанное целевое свойство у каждого из* (?)*-примеров*
$X :=$ (?)*-примеры*
for $(o \in X)$ **do**
  $PredictPositively(o) :=$ ***false***
  for $(\langle A, B\rangle \in S^+)$ **do**
    if $(B \subseteq o')$ **then**
      $PredictPositively(o) :=$ ***true***
    end
  end
end

Определение

Объект o*, описываемый фрагментом* $o' \subseteq F$ *(множеством признаков),* ***предсказывается положительным*** *с помощью ВКФ-гипотезы* $\langle A, B\rangle$*, если* $B \subseteq o'$*.*

Нижние полупространства для предсказания

Если число признаков равно $n = |F|$, то можно рассматривать вершины n-мерного гиперкуба $\{0,1\}^n \subseteq \boldsymbol{R}^n$.
Каждый объект o, предъявляемый для предсказания, задает семейство нижних полупространств в $\boldsymbol{R}^n$:

Определение

Нижнее полупространство $H^{\downarrow}(o)$, *определяемое объектом* o *с фрагментом* $o' \subseteq F$, *задается линейным неравенством* $x_{j_1} + \ldots + x_{j_k} < \frac{1}{2}$, *где* $F \setminus o' = \{f_{j_1}, \ldots, f_{j_k}\}$. *Допускается также вырожденное нижнее полупространство* $0 < \frac{1}{2}$, *соответствующее* $o' = F$, *и совпадающее со всем* $\boldsymbol{R}^n$.

Связь с доопределением по аналогии

Лемма

Объект o предсказывается положительным тогда и только тогда, когда в любом нижнем полупространстве $H^{\downarrow}(o)$ содержится фрагмент B хотя одной ВКФ-гипотезы $\langle A, B\rangle$.

Определение

*Объект o назовем ε-**важным**, если суммарная вероятность появления таких ВКФ-гипотез $\langle A, B\rangle$, что $B \in H^{\downarrow}(o)$ будет больше ε.*

Теперь нас будет интересовать только вероятность ошибки «первого рода» (отказ от положительного предсказания): требуется найти такое число N, зависящее от ε и δ, чтобы с вероятностью, большей $1 - \delta$, каждый ε-важный объект будет предсказываться положительно каким-то элементом случайной выборки объема N.

Необходимое число ВКФ-гипотез

Теорема

Для n признаков и любых $\varepsilon > 0$ и $1 > \delta > 0$ достаточно породить

$$N \geq \frac{n \cdot \ln 2 - \ln \delta}{\varepsilon}$$

ВКФ-гипотез, чтобы вероятностью $> 1 - \delta$ все ε-важные объекты могли быть предсказаны положительно.

Цепь Маркова порождает гипотезы, при этом независимые траектории порождают независимые элементы решетки кандидатов. Тестовые примеры задают подмножества точек (отсекаемые гиперплоскостями) на гиперкубе, где должны оказаться ВКФ-гипотезы. Это дуально парадигме Вапника-Червоненкиса (там гипотезы определяют подмножества, куда должны попасть обучающие точки).