МУЛЬТИПАРАДИГМЕННЕ ПРОГРАМУВАННЯ

Лекція 9

Парадигма ймовірнісного програмування

У 1814 році П’єр-Симон Лаплас писав: **здебільшого найважливіші життєві питання є насправді лише завданнями теорії ймовірностей**.

Через сто років після цих слів на такі питання можна було відповісти тільки одним способом (зберігаючи вірність думку Лапласа): проаналізувати кожну задачу на папері, висловити результат у вигляді формули і обчислити значення формули, вручну підставивши в неї числа. Наступ ери комп’ютер мало що змінило. Просто стало можливо обчислювати складніші формули, а аналіз за допомогою пера і паперу тепер може займати сотні сторінок.

Для аналізу ймовірнісної завдань і необхідно побудувати вірогідну модель, в якій описується простір можливих результатів і кожному з них якимось чином зіставляється числова ймовірність. Раніше імовірнісні моделі формулювалися на суміші природної мови і напівформальному математичної нотації. На основі моделі за допомогою деяких математичних маніпуляцій виводилася формула або алгоритм для обчислення відповідей. Обидві стадії були трудомісткими, чреватими помилками і залежали від конкретного завдання, тому застосування теорії ймовірностей на практиці стикалися з серйозними обмеженнями.
Всупереч Лапласа, найважливіші життєві питання залишалися невирішеними.

Першим великим просуванням стала розробка формальних мов, зокрема байесівських і марковських мереж, для вираження імовірнісних моделей. У формального мови є точний синтаксис, який визначає, які вислови допустимі, і точна семантика, яка визначає, що означає кожне допустиме вираз (тобто яка саме імовірнісна модель представлена даним виразом). Тому з’явилася можливість описувати імовірнісні моделі в машиночитабельному вигляді і розробити єдиний алгоритм обчислення наслідків будь-якої виразності ймовірнісної моделі.

Парадигма імовірнісного програмування - нова захоплююча область досліджень, яка приваблює все більший інтерес. Поступово вона прокладає собі шлях з академічних кіл в світ програмістів. По суті справи, розподіл усіх програмування - це новий спосіб створення імовірнісних моделей, що дозволяють прогнозувати або виводити нові факти, яких немає в результатах спостережень. Імовірнісні міркування давно вважаються одним з основних підходів до машинного навчання, де модель описує те, що відомо з досвіду.

Але раніше такі системи були обмежені простими фіксованими структурами типу Байєсових мереж. Розподіл усіх програмування звільняє системи імовірнісних міркувань від цих кайданів, надаючи всю міць мов програмування для представлення моделей. Можна вважати, що це аналог переходу від логічних схем до мов високого рівня.

Що таке ймовірнісне програмування?

Розподіл усіх программирование - це спосіб створення систем, що допомагають приймати рішення в умовах невизначеності. Багато рішень, що приймаються нами щодня, мають на увазі облік релевантних факторів, які ми не спостерігаємо безпосередньо. Історично одним із способів прийняття рішень в таких умовах невизначеності стали системи імовірнісного міркування. Розподіл усіх міркування дозволяє об'єднати наші знання про ситуацію з ймовірними законами і таким чином взяти до уваги не спостерігаються чинники, важливі для прийняття рішення.

До недавнього часу область застосування систем імовірнісного міркування була обмежена, і до багатьох виникають на практиці завдань їх вдавалося застосувати з великими труднощами. Розподіл усіх програмування - це новий підхід, що дозволив спростити побудову систем імовірнісного міркування і розширити їх придатність.

Щоб зрозуміти, в чому сенс імовірнісного програмування, почнемо з розгляду того, як приймається рішення в умовах невизначеності і яку роль в цьому відіграють суб'єктивні судження. Потім подивимося, як може допомогти система імовірнісного міркування.

Ми познайомимося з трьома видами міркувань, властивих таким системам. Після цього стане зрозуміло, що таке ймовірнісне програмування і як воно дозволяє використовувати всю міць мов програмування для побудови систем імовірнісних міркувань.

1. Як ми висловлюємо суб'єктивне судження?

У реальному світі цікаві для нас питання рідко допускають недвозначну відповідь: так чи ні. Наприклад, при виведенні н а ри нок нового продукту хотілося б знати, добре л і він буде продаватися. Ви вважаєте, що його чекає успіх, тому що продукт добре спроектований і вивчення ринку показало, що в ньому є потреба, але повної впевненості немає. Бути може, ваш конкурент представить кращий продукт, а, можливо, у вашому продукті виявиться фатальний недолік, через який ринок від нього відвернеться. Або економічна ситуація повернеться до гіршого. Якщо ви хочете стовідсоткової впевненості, то ніколи не зможете прийняти рішення про початок продажів.


Показатели прошлого года
Показатели будущего года
?

2. Системи імовірнісних міркувань допомагають приймати рішення

Імовірнісні міркування - це підхід, в якому модель предметної області використовується для прийняття решени й в умовах невизначеності. Візьмемо приклад з області футболу. Припустимо, що за статистикою 9% кутових завершуються голом. Потрібно передбачити результат конкретного кутового удару. Зростання центрального нападаючи ющего атакуючої команди дорівнює 193 см, і відомо, що він відмінно грає головою. Основного воротаря команди, що захищається тільки що забрали на носилках, і на заміну йому вийшов воротар, який проводить свій перший матч.

Крім того, дме сильний вітер, який ускладнює контроль над польотом м'яча. І як за таких умов вирахувати ймовірність?
Як система імовірнісних міркувань пророкує НАСЛІДКИ кутового удару побачимо на наступному малюнку:


1. Вы кодируете свои знания об угловых ударах и релевантных факторах
Система вероятностных рассуждений
Модель углового удара
Алгоритм вывода
2. Вы предоставляете знания о предусловиях конкретного углового удара
Факт
• Высокий центральный нападающий
• Неопытный вратарь
• Сильный ветер
Запрос
• Будет ли забит гол?
3. Вы говорите системе, что хотите узнать
4. Система применяет алгоритм вывода для предсказания исхода
Ответ
• Да: 20 %; Нет: 80 %
5. Система возвращает ответ в виде набора вероятностей

На малюнку показано, як отримувати відповідь за допомогою системи імовірнісних міркувань. Всі свої знання про кутових ударах і релевантних чинниках ми кодуємо у вигляді моделі кутового. Потім ми надаємо факти про конкретний кутовому ударі: що центральний нападаючий високий, що воротар недосвідчений і що дме сильний вітер. Ми говоримо системі, що хотіли б дізнатися, чи буде забитий гол. Алгоритм виведення повертає відповідь: гол буде забитий з ймовірністю 20%.

Співвідношення між моделлю, що надається вами інформацією та відповідями на за тання математично строго визначено законами теорії ймовірностей.

Процес використання моделі для отримання відповідей н а запити при відомих фактах називається **імовірнісним висновком**, або просто **висновком**. На щастя, розроблені алгоритми, які виконують необхідні обчислення, приховуючи від вас всю математику. Вони називаються алгоритмами виведення.

Основні компоненти системи імовірнісних міркувань:


Система вероятностных рассуждений
Вероятностная модель
Алгоритм вывода
Ответ
Вероятностная модель выражает общие знания о ситуации
Алгоритм вывода используется моделью для нахождения ответов на запросы при известных фактах
Ответы на запросы имеют вид вероятностей различных исходов
Факты содержат конкретную информацию о ситуации
Факты
Запросы
Запросы выражают то, что поможет принять решение

Система імовірнісних міркувань може міркувати трьома способами.

• Передбачення майбутніх подій. Ми вже бачили це на малюнку раніше, де система передбачає, чи буде в даній ситуації забитий гол. Факти зазвичай включають інформацію про поточну ситуацію, наприклад зростання центрального нападника, досвідченість воротаря і силу вітру.

• Висновок причини подій. Прокрутимо плівку вперед на 10 секунд. Високий центральний нападаючий тільки що забив гол головою, проштовхнувши м'яч під корпусом воротаря. Що можна сказати про це воротаря-новобранця при таких фактах? Чи можна зробити висновок, що йому не вистачає вміння? На малюнку далі показано, як використовувати систему імовірнісних міркувань для відповіді на це питання. Береться та ж модель кутового удару, що і для прогнозу голи. (Це корисна властивість імовірнісних міркувань: одну і ту ж модель можна використовувати як для передбачення майбутнього результату, так і для пояснення причин, що призвели до відомого результату.) Факти ті ж, що і раніше, плюс той факт, що гол забитий. Запит стосується кваліфікації воротаря, а відповідь дає можливості різноманітних рівнів кваліфікації.


Факты включают знание предусловий и исходов
Система вероятностных рассуждений
Модель углового удара
Алгоритм вывода
Факты
• Предусловия
› Высокий центральный нападающий
› Неопытный вратарь
› Сильный ветер
• Исход
› Забит гол
Запрос
• Какова квалификация вратаря?
Алгоритм вывода рассуждает о запросе, зная все известные предусловия и исходы
Ответ
• Плохая: 50 %; Средняя: 30 %; Хорошая: 20 %
Задается запрос о возможной причине данного исхода

• Навчання на минулі події, щоб краще прогнозувати майбутні. Прокрутимо плівку ще на 10 хвилин вперед. Та ж команда заробила ще один кутовий. Все так же, як і раніше - високий центральний нападаючий, недосвідчений воротар - тільки вітер тепер стих. Розподіл усіх міркування дозволяє вико ьзовать інформацію про те, що трапилося при
подачі попереднього кутового, щоб краще передбачити результат наступного. На малюнку далі показано, як це робиться. До складу фактів входить все те, що і минулого разу (з позначкою, що це було минулого разу), а також нова інформація про поточну ситуацію.

Відповідаючи на питання, чи буде забитий гол на цей раз, алгоритм виведення початку визначає властивості ситуації, які привели до голу в перший раз, наприклад, кваліфікацію нападника і воротаря. А потім ці властивості використовуються для передбачення результату в новій ситуації.

Факты включают знание предусловий и исходов прошлых ситуаций, а также предусловия текущей ситуации

Алгоритм вывода рассуждает о предыдущих ситуациях, пытаясь определить факторы (например, квалификацию вратаря), существенные и в новой ситуации

Алгоритм использует эти факторы, чтобы улучшить предсказание исхода в новой ситуации

Задается запрос об исходе в новой ситуации

Беручи до уваги факти, які стосуються результату минулого кутового, система імовірнісного міркування може краще передбачити результат наступного.
Запити перерахованих видів можуть допомогти в прийнятті різних рішень.

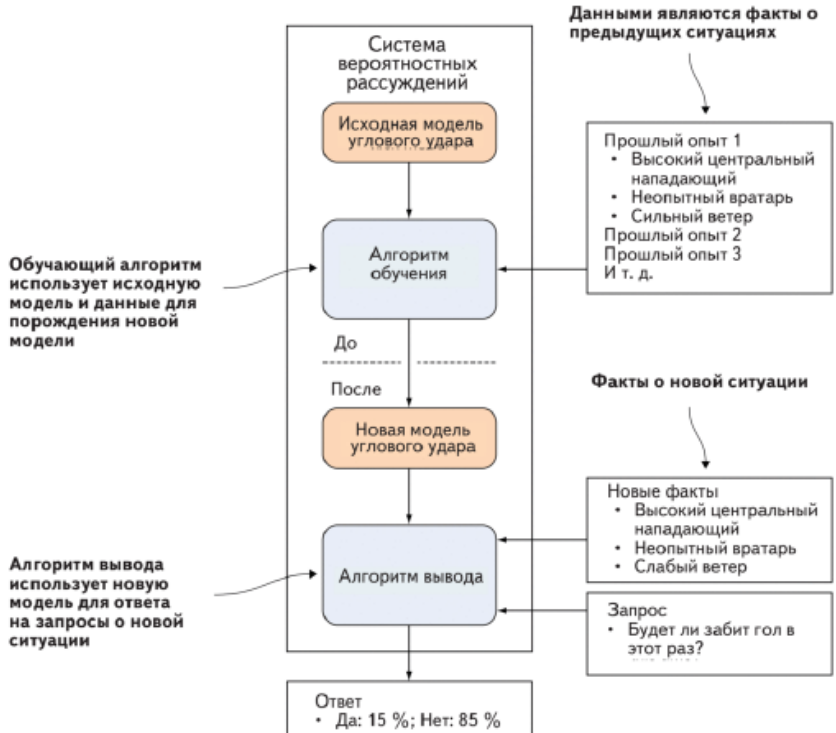
- Знаючи ймовірність голи при наявності або відсутності додаткового захисника, можна вирішити, чи не варто замінити атакуючого захисником.
- На основі оцінки кваліфікації воротаря можна вирішити, яку зарплату запропонувати йому в наступному контракті.
- Отримавши інформацію про воротаря, можна вирішити, чи варто ставити його на наступну гру.

Навчання поліпшеної моделі

Вище були описані способи міркування про конкретні ситуації при відомих фактах. Але система імовірнісних міркувань дозволяє зробити ще одну річ: навчаючись на минулому досвіді, поліпшити загальні знання. третій спосіб міркувань - це використання минулих результатів для кращого передбачення результату в конкретній нової ситуації. А зараз ми говоримо про поліпшення самої моделі. Якщо є великий минулий досвід,тобто інформація про багатьох кутових ударах, то чому б не навчити нову модель, що представляє загальні знання про те, що зазвичай відбувається при подачі кутового? На малюнку нижче показано, що це досягається за допомогою алгоритму навчання.

На відміну від алгоритму виведення, його завдання полягає не в тому, щоб відповідати на питання, а в тому, щоб породити нову модель. На вхід алгоритму навчання подається вихідна модель, а він оновлює її на основі досвіду. Потім нову модель можна використовувати для відповіді на майбутні запити. Імовірно відповіді нової моделі будуть більш обгрунтовані, ніж відповіді вихідної.

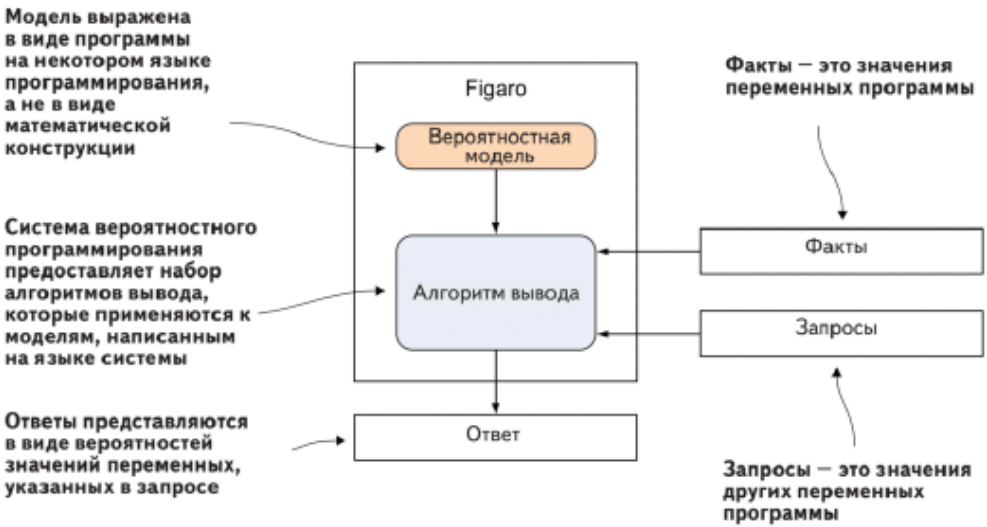
За допомогою алгоритму навчання можна навчити нову модель на основі минулого досвіду, а потім використовувати її для майбутніх висновків.


Данными являются факты о предыдущих ситуациях
Система вероятностных рассуждений
Исходная модель углового удара
Прошлый опыт 1
• Высокий центральный нападающий
• Неопытный вратарь
• Сильный ветер
Прошлый опыт 2
Прошлый опыт 3
И т. д.
Обучающий алгоритм использует исходную модель и данные для порождения новой модели
Алгоритм обучения
До
После
Факты о новой ситуации
Новая модель углового удара
Новые факты
• Высокий центральный нападающий
• Неопытный вратарь
• Слабый ветер
Алгоритм вывода использует новую модель для ответа на запросы о новой ситуации
Алгоритм вывода
Запрос
• Будет ли забит гол в этот раз?
Ответ
• Да: 15 %; Нет: 85 %

У будь-якій системі імовірнісних міркувань використовується мова представлення, на якому виражаються імовірнісні моделі. Таких мов багато. Можливо, про деякі ви чули, наприклад, про байесовских мережах (їх ще називають мережами довіри) або прихованих марковських моделях. Від мови уявлення залежить, які моделі зможе обробити система і як вони виглядають. Безліч моделей, які представлені мовою, називається виразною силою мови. Для розробки додатків потрібно, щоб виразна сила була якомога більше.
Система імовірнісного програмування - це просто система імовірнісних міркувань, для якої мовою подання є мова програмування. Говорячи мовою програмувати, ми маємо на увазі, що він має всі можливості, які прийнято очікувати від типового мови програмування: змінними, розвиненою системою типів даних, засобами управління потоком виконання, функціями і т.д.

Як ви незабаром побачите, мови імовірнісного програмування здатні висловити широкий спектр імовірнісних моделей, що далеко виходить за рамки більшості традиційних систем імовірнісних міркувань. Виразна сила мов імовірнісного програмування досить висока.

На малюнку нижче ілюструється співвідношення між системами імовірнісного програмування і імовірнісних міркувань в загальному випадку. Порівняйте цей малюнок з малюнком представленим раніше, щоб стали ясніше відмінності між двома системами. Основна зміна полягає в тому, що моделі виражаються у вигляді програм на деякій мові програмування, а не у вигляді математичних конструкцій типу байєсівської мережі. Тому факти, запити та відповіді стають змінними програми. Факти можна уявити конкретними зіаченіямі змінних, запит - це отримання значення змінної, а відповідь - ймовірності того, що змінні приймають ті чи інші значення. Крім того, система імовірнісного програмування зазвичай включає набір алгоритмів ви вода. Ці алгоритми застосовуються до моделей, написаним на мові системи.


Модель выражена в виде программы на некотором языке программирования, а не в виде математической конструкции
Figaro
Вероятностная модель
Факты — это значения переменных программы
Система вероятностного программирования предоставляет набор алгоритмов вывода, которые применяются к моделям, написанным на языке системы
Алгоритм вывода
Факты
Запросы
Ответы представляются в виде вероятностей значений переменных, указанных в запросе
Ответ
Запросы — это значения других переменных программы

Система імовірнісного програмування - це система імовірнісних міркувань, в якій для подання імовірнісних моделей застосовується мова програмування.

Подання імовірнісних моделей у вигляді програм

Головна ідея будь-якої мови програмування - виконання. Ми виконуємо програму, яка породжує певний вихід. Імовірнісна програма в цьому сенсі аналогічна, тільки вона може мати не один, а кілька шляхів виконання, кожен з яких породжує свій вихід. Яким шляхом слідувати, визначається випадковим вибором, що здійснюються всередині програми. Кожен випадковий вибір має кілька можливих результатів, і в програмі закодована ймовірність кожного результату. Тому можна вважати, що імовірнісна програма - це програма, при виконанні якої випадковим чином генеруються вихідні дані.

Ця ідея ілюструється на малюнку нижче.

Тут система імовірнісного програмування містить програму кутового удару, яка описує випадковий процес генерації результату кутового. Програма приймає ряд вхідних даних; в нашому прикладі це зростання центрального нападника, досвідченість воротаря і сила вітру. На основі цих даних програма випадковим чином виконується і генерує вихідні дані. Кожне випадкове виконання дає на виході конкретний результат. Оскільки кожен випадковий вибір може мати кілька випадків, то існує багато можливих шляхів виконання, що дають різні результати. Будь-який заданий результат, наприклад взяття воріт, може бути згенерований на різних шляхах виконання.


1. Вероятностная программа описывает процесс случайной генерации выходных данных по известным входным
Вероятностная система программирования
Программа углового удара
Высокий центральный нападающий
Неопытный вратарь
Сильный ветер
Случайное выполнение
Нет гола
Гол!!!
Нет гола
Чуть не забили
2. При заданных входных данных программа может случайно выполняться многократно и каждый раз порождать новые результаты
3. Вероятность некоторого результата — это вероятность выбора пути выполнения, порождающего такой результат. В данном случае вероятность гола равна ¼

Розглянемо, як ця програма визначає імовірнісну модель. Будь-яке конкретне виконання програми є результатом послідовності випадкових виборів. Кожен випадковий вибір відбувається з певною ймовірністю. Якщо перемножити всі ці ймовірності, то вийде ймовірність шляху виконання. Таким чином, програма визначає ймовірність кожного шляху виконання. Якщо уявити собі, що програма проганяється багаторазово, то частка прогонів, на яких генерується заданий шлях виконання, дорівнює його ймовірності. Імовірність деякого результату дорівнює частці прогонів, на яких було отримано цей результат. На малюнку раніше гол виходить в 1/4 всіх прогонів, тому ймовірність голи дорівнює 1/4.

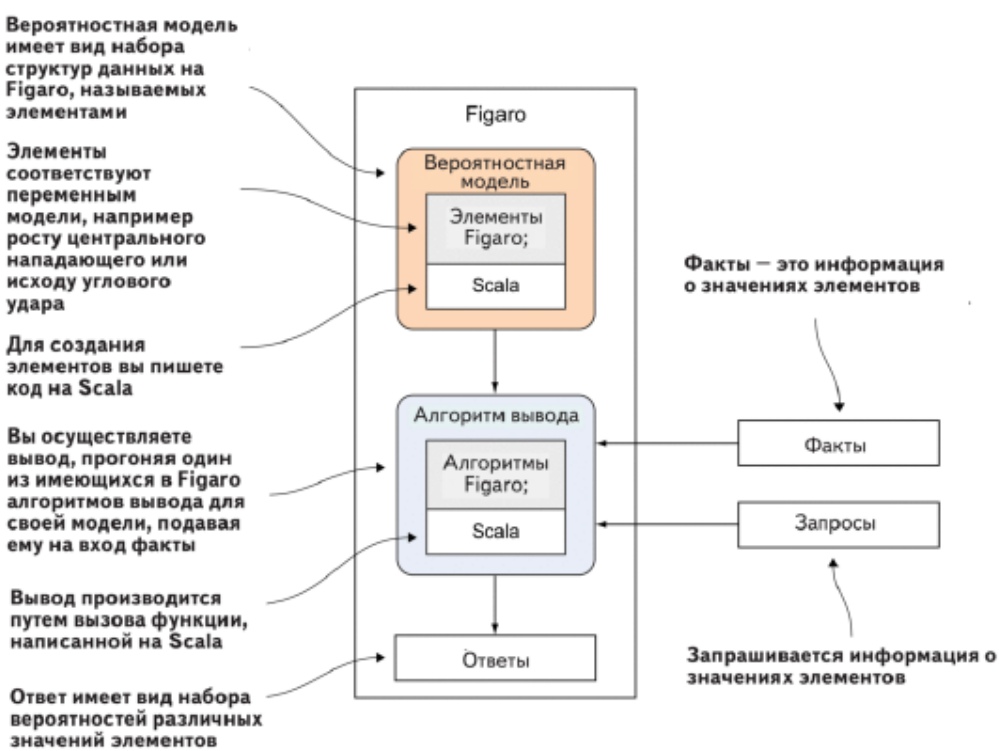
ВВЕДЕННЯ ДО МОВИ ЙМОВІРНІСНОГО ПРОГРАМУВАННЯ

Вихідний код Фігаро відкритий, підтримка здійснюється за допомогою GitHub, а сама система розробляється з 2009 року. Вона реалізована у вигляді бібліотеки, написаної на Скала.

На малюнку нижче показано, як в Figaro використовується Scala для реалізації системи імовірнісного програмування. Почнемо з ймовірнісної моделі. У Figaro модель складається з структур даних, які називаються елементами. Кожен елемент представляє змінну, яка може приймати довільне число значень.

Структури даних реалізовані на Scala, і для створення моделі, в якій вони використовуються, ви пишете програму на Scala. Програмі можна подати на вхід факти, що містять інформацію про значеннях елементів, а в запиті вказати, які вихідні елементи вас цікавлять. Ви вибираєте один з вбудованих в Figaro алгоритмів вибору і застосовуєте його до своєї моделі, щоб отримати відповідь на запит при заданих фактах. Алгоритми виведення реалізовані на Скала, а застосування алгоритм зводиться до виклику функції Scala.

Результатом виведення є ймовірності різних значень елементів, зазначених у запиті.


Вероятностная модель имеет вид набора структур данных на Figaro, называемых элементами
Элементы соответствуют переменным модели, например росту центрального нападающего или исходу углового удара
Для создания элементов вы пишете код на Scala
Вы осуществляете вывод, прогоняя один из имеющихся в Figaro алгоритмов вывода для своей модели, подавая ему на вход факты
Вывод производится путем вызова функции, написанной на Scala
Ответ имеет вид набора вероятностей различных значений элементов
Figaro
Вероятностная модель
Элементы Figaro;
Scala
Алгоритм вывода
Алгоритмы Figaro;
Scala
Ответы
Факты – это информация о значениях элементов
Факты
Запросы
Запрашивается информация о значениях элементов

Той факт, що Figaro написана на Scala, дає ряд переваг. Деякі з них пов'язані з використанням мови загального призначення замість спеціалізованого, інші - зі специфікою саме Scala. Спочатку зупинимося на перевагах вкладення в мову загального призначення.

Факти можна представляти, використовуючи програму на осяжний мовою. Наприклад, програма може прочитати дані з файлу, якимось чином бработать їх і представити у вигляді фактів для моделі Figaro. На спеціалізованому незалежному мовою зробити це було б набагато важче.

• Аналогічно відповіді, повернуті Figaro, можна використовувати в програмі. Наприклад, якщо менеджер футбольної команди працює з деякою програмою, то ця програма може запросити ймовірність голи з тим, щоб порекомендувати менеджеру варіанти дій.

• Код на мові загального призначення можна включити в вірогідну програму. Розглянемо, наприклад, фізичну модель траєкторії м'яча після удару головою. Цю модель можна було б зробити частиною елемента Figaro.
• Для побудови моделі Фігаро можна використовувати стандартні прийоми програмування. Наприклад, можна завести словник, який містить елементи Figaro, які відповідають усім гравцям команди, і вибирати з нього елементи, виходячи з ситуації, що складається на поле.

Тепер перерахуємо причини, за якими Скала особливо добре підходить на роль осяжний мови для системи імовірнісного програмування.

1. Оскільки Scala - *функціональна мова програмування*, Figaro також отримує всі переваги функціонального програмування, що дозволяє природно за приписувати багато моделей.

2. Scala - *об'єктно-орієнтована мова*, і те, що він поєднує в собі функціональні і об'єктно-орієнтовані риси, є додатковою перевагою. Figaro також є об'єктно-орієнтованим.

Розглянемо ряд переваг Фігаро не пов'язаний з вкладенням в Scala.

1. На Фігаро можна уявити надзвичайно широкий спектр імовірнісних моделей. Елементи Фігаро можуть набувати значень будь-якого типу, в тому числі: булеві, цілі, з подвійною точністю, масиви, дерева, графи і т. Д. Зв'язки між елементами можна визначити за допомогою довільної функції.
2. Figaro надає розвинені засоби завдання фактів за допомогою умов і обмежень.
3. У Figaro є великий набір алгоритмів виводу.
4. Figaro дозволяє представляти динамічні моделі ситуацій, зраджую трудящих в часі, і міркувати про них.

5. Figaro дає можливість включати в модель явні рішення і підтримує висновок оптимальних рішень.

Є кілька причин, за якими Figaro - зручна мова для вивчення ймовірного програмування.

1. Будучи реалізований у вигляді бібліотеки на Scala, Figaro може бути використаний в програмах, написаних на Java і Скала, що спрощує його інтеграцію з додатками.
2. З тієї ж причини Фігаро дозволяє використовувати для побудови моделей все багатство осяжної мови.
3. Scala - сучасна передова мова програмування, що володіє безліччю корисних засобів для організації програм, і все це ви автоматично отримуєте в своє розпорядження, працюючи з Figaro.
4. Figaro своєму розпорядженні повний набір алгоритмів.

Побудова простої системи ймовірнісного програмування

Основна ідея полягає в тому, що Figaro містить такі засоби для подання імовірнісних моделей і виконання виведення з них, які недоступні без імовірнісного програмування.

Уявіть, що людина прокидається вранці, дивиться, ясно чи на вулиці, і вимовляє привітання, залежне від погоди. Так відбувається два дні поспіль. Крім того, погода в другий день залежить від першого: якщо в перший день було ясно, то ймовірність, що і на завтра буде сонячно, підвищується. Ці пропозиції на природній мові можна записати кількісно, як показано далі.

Погода сегодня		
Ясно	0.2	
Пасмурно	0.8	
Приветствие сегодня		
Если сегодня на улице ясно	«Здравствуй, мир!»	0.6
	«Здравствуй, вселенная!»	0.4
Если сегодня на улице пасмурно	«Здравствуй, мир!»	0.2
	«О нет, только не это»	0.8
Погода завтра		
Если сегодня на улице ясно	Ясно	0.8
	Пасмурно	0.2
Если сегодня на улице пасмурно	Ясно	0.05
	Пасмурно	0.95

Приветствие завтра		
Если завтра на улице ясно	«Здравствуй, мир!»	0.6
	«Здравствуй, вселенная!»	0.4
Если завтра на улице пасмурно	«Здравствуй, мир!»	0.2
	«О нет, только не это»	0.8

Намети три завдання, які повинна вирішувати ця модель. Раніше ми бачили три типи міркувань, доступних ймовірнісній моделі: передбачення майбутнього, висновок минулих подій, що призвели до нинішнього результату, і навчання на минулому досвіді для кращого передбачення майбутнього. Все це ми зробимо за допомогою нашої простої моделі. Точніше, завдання формулюються в такий спосіб.

1. Передбачити сьогоднішнє вітання.
2. Знаючи, що сьогодні було вимовлено вітання "Здрастуй, світ!", зробити висновок про те, ясно чи на вулиці.
3. Знаючи, що сьогодні було вимовлено вітання "Здрастуй, світ!", навчитися передбачати завтрашнє вітання.

А тепер познайомимося з кодом, написаним на Scala / Figaro.

```
import com.cra.figaro.language.{Flip, Select}
import com.cra.figaro.library.compound.If
import com.cra.figaro.algorithm.factored.VariableElimination

object HelloWorld {
   val sunnyToday = Flip(0.2)
   val greetingToday = If(sunnyToday,
        Select(0.6 -> "Здравствуй, мир!", 0.4 -> "Здравствуй, вселенная!"),
        Select(0.2 -> "Здравствуй, мир!", 0.8 -> "О нет, только не это"))
  val sunnyTomorrow = If(sunnyToday, Flip(0.8), Flip(0.05))
  val greetingTomorrow = If(sunnyTomorrow,
        Select(0.6 -> "Hello, world!", 0.4 -> "Howdy, universe!"),
        Select(0.2 -> "Hello, world!", 0.8 -> "Oh no, not again"))
```


```
def predict() {
  val result = VariableElimination.probability(greetingToday,
                "Здравствуй, мир!")
  println("Сегодня будет приветствие \"Здравствуй, мир!\" " +
          "с вероятностью " + result + ".")
}
```

❸

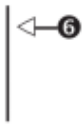
```
def infer() {
  greetingToday.observe("Здравствуй, мир!")
  val result = VariableElimination.probability(sunnyToday, true)
  println("Если сегодня произнесено приветствие  \"Здравствуй, мир\", " +
          "то будет солнечно с вероятностью " + result + ".")
}
```

❹

```
def learnAndPredict() {
  greetingToday.observe("Здравствуй, мир!")
  val result = VariableElimination.probability(greetingTomorrow,
                "Hello, world!")
  println("Если сегодня произнесено приветствие  \"Здравствуй, мир!\", " +
          "то завтра будет сказано \"Здравствуй, мир!\" " +
          "с вероятностью " + result + ".")
}
```

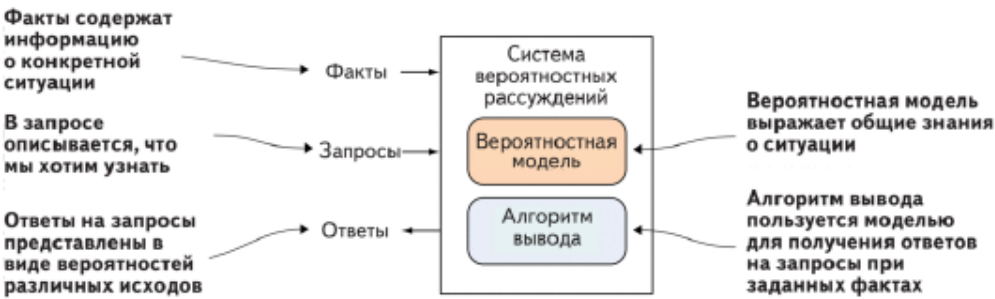
❺

```
  def main(args: Array[String]) {
    predict()
    infer()
    learnAndPredict()
  }
}
```



❶ — Импортируем конструкции Figaro
❷ — Определяем модель
❸ — Предсказываем сегодняшнее приветствие, применяя алгоритм вывода
❹ — Применяем алгоритм вывода, чтобы вывести сегодняшнюю погоду из того, что сегодня было произнесено приветствие «Здравствуй, мир!»
❺ — Применяя алгоритм вывода, обучаемся предсказывать завтрашнее приветствие, зная, что сегодня было произнесено «Здравствуй, мир!»
❻ — Метод main, который выполняет все задачи

ДОВІДНИК ПО ФІГАРО



Огляд основних частин системи імовірнісних міркувань

Загальні знання виражаються у вигляді моделі Figaro.
Знання про конкретну ситуацію представлені у вигляді фактів.
Запит говорить системі, що ми хочемо дізнатися.
Алгоритм виведення Figaro приймає запит і використовує модель для отримання відповідей на запити.
На малюнку нижче поданий взаємозв’язок основних концепцій Figaro.

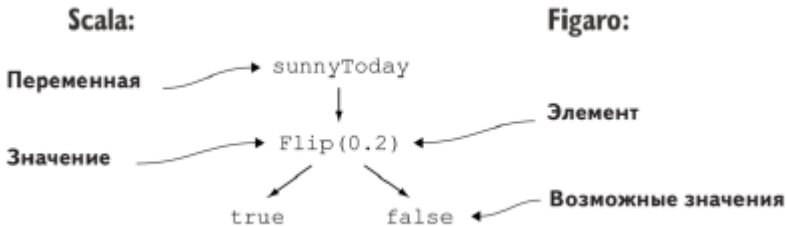

Модель Figaro состоит из набора структур данных, называемых элементами
Факты включают наблюдения, условия и ограничения на значения элементов
Модель Figaro
Атомарные элементы
Составные элементы
Apply
Chain
Атомарные элементы являются базовыми строительными блоками
Составные элементы связывают более простые элементы
Apply и Chain – важные примеры составных элементов
Факты
Замечания
Условия
Ограничения
Алгоритмы вывода Figaro
Instantiate
Run
Clean up
Алгоритмы Figaro вычисляют информацию о запрошенных элементах
Обычно экземпляр алгоритма создается, исполняется и в конце уничтожается
Запросы
Целевые элементы
Ответы
Вероятности значений
Наиболее вероятные значения
В Figaro имеются методы для выполнения всех перечисленных шагов
В запросах указываются целевые элементы
В ответах содержится информация о целевых элементах

Приклад побудови ймовірнісної моделі.
Почнемо з побудови найпростішої моделі Figaro, в якій буде всього один атомарний елемент. Але спочатку необхідно імпортувати необхідні конструкції Figaro:

```
import com.cra.figaro.language._
```

В результаті імпортуються всі класи з пакета **com.cra.figaro.language**, який містить базові конструкції Figaro. Один з цих класів називається **Flip**. З його допомогою можна побудувати просту модель:

```
val sunnyToday = Flip(0 . 2 )
```



Зв'язок між змінними і значеннями Scala і елементами Фігаро і їх можливими значеннями.

Важливо чітко розуміти, що відносяться до Scala, а що - до Figaro. У цьому рядку ми створили змінну Scala **`sunnyToday`** і привласнили їй значення **`Flip(0.2)`**. Значення **`Flip(0. 2)`** - це елемент Figaro, що представляє випадковий процес, який породжує значення **`true`** з ймовірністю 0.2 і **`false`** - з ймовірністю 0.8.

Елемент - це структура даних, що представляє процес, який випадковим чином породжує деяке значення. Число випадків випадкового процесу може бути довільним. Кожен можливий результат називається значенням процесу. Таким чином, **`Flip(0.2)`** - елемент з двома можливими булевими значеннями: **`true`** і **`false`**.

Отже, ми маємо змінну Scala, приймаючу значення в сенсі Scala. Це значення є елементом Figaro і представляє різні можливі результати процесу.

Отже, ми побудували просту модель. Тепер запустимо алгоритм виведення і запитаємо ймовірність того, що змінна **sunnyToday** дорівнює **true**. Спочатку потрібно імпортувати використовується алгоритм виведення:

```
import
com.cra.figaro.algorithm.factored.VariableElimination
```

Тут імпортується алгоритм виключення змінних, що відноситься до класу точних алгоритмів, тобто він точно обчислює ймовірності, виходячи з моделі і фактів. Імовірнісний висновок - складне завдання, тому іноді точні алгоритми працюють дуже довго або їм не вистачає пам'яті.

Figaro пропонує також наближені алгоритми, які зазвичай обчислюють відповіді, близькі до точних. Оскільки всі моделі тут прості, то алгоритм виключення змінних
буде працювати без проблем.

У Figaro є команда, що дозволяє за одну дію задати запит, виконати алгоритм і отримати відповідь:

```
println(      VariableElimination.probability
( sunnyToday,true ))
```

Ця команда надрукує 0.2. Наша модель складається тільки з елемента **`Flip(0.2)`**, який приймає значення true з ймовірністю 0.2. Алгоритм виключення змінних правильно обчислює ймовірність, що **`sunnyToday`** дорівнює **`true`**, - а вона дорівнює 0.2.

В наступній лекції ми розглянемо принципи побудову моделей і завдання спостережень в парадигмі ймовірнісного програмування.