

Деревья принятия решений


Сергей Николенко

Машинное обучение — ИТМО, осень 2006

Outline

- 1 Введение в курс
 - Кто я такой
 - О чём этот курс: машинное обучение
 - Краткий обзор курса
 - Программирование: Python
- 2 Введение в лекцию: мотивация и пример
 - Зачем всё это надо
 - Структура дерева принятия решений
 - Пример

Кто я такой

- Сергей Николенко, аспирант ПОМИ РАН.
- Научные интересы: теория сложности, криптография, высшая алгебра, алгебраическая геометрия, искусственный интеллект.
- Email: `sergey@logic.pdmi.ras.ru`, `smartnik@inbox.ru`
- Homepage:
`http://logic.pdmi.ras.ru/~sergey/`
- LiveJournal:  [smartnik](#)

Машинное обучение

Что такое «машинное обучение»?

Определение

Компьютерная программа обучается по мере накопления опыта относительно некоторого класса задач T и целевой функции P , если качество решения этих задач (относительно P) улучшается с получением нового опыта.

Машинное обучение

Что такое «машинное обучение»?

Определение

Компьютерная программа обучается по мере накопления опыта относительно некоторого класса задач T и целевой функции P , если качество решения этих задач (относительно P) улучшается с получением нового опыта.

Что мы будем изучать

- Деревья принятия решений
- Нейронные сети
- Генетические алгоритмы
- Байесовское обучение: оценки максимального правдоподобия, скрытые марковские модели
- Байесовские сети

О программировании

Примеры в курсе будут на языке Python.

Достоинства:

- Простой и интуитивный
- Трудно сделать ошибку
- Программы компактные и легко читаются

Другие языки

Альтернативы:

- Ruby
- Perl
- Lisp

Программы и иллюстрации на любых языках будут с благодарностью приняты.

Outline

- 1 Введение в курс
 - Кто я такой
 - О чём этот курс: машинное обучение
 - Краткий обзор курса
 - Программирование: Python
- 2 Введение в лекцию: мотивация и пример
 - Зачем всё это надо
 - Структура дерева принятия решений
 - Пример

Пример

Задача: выиграет ли «Зенит» свой следующий матч?

Параметры:

- выше ли находится соперник по турнирной таблице;
- дома ли играется матч;
- пропускает ли матч кто-либо из лидеров команды;
- идёт ли дождь.

Мы знаем об исходах нескольких матчей и хотим предсказать исход следующего матча, параметры которого нам ещё не встречались.

Постановка задачи

Главная задача:

- Классификация данных
- Аппроксимация заданной булевой функции

То есть имеется *частично* заданная функция f , и мы хотим понять, как она работает на ещё не известных примерах.

Постановка задачи

Дано:

- Атрибуты (параметры функции)
- Тестовые примеры ($f(0, 0, 1)$, $f(0, 1, 1)$, $f(1, 1, 0)$, $f(1, 1, 1)$)

Нужно:

- Продолжить функцию на другие значения атрибутов (найти $f(0, 0, 0)$)
- Сделать это красиво и экономично

Дерево принятия решений

Дерево принятия решений — это дерево. На нём есть метки:

- В узлах, не являющиеся листьями: атрибуты, по которым различаются случаи
- В листьях: значения целевой функции
- На рёбрах: значения атрибута, из которого исходит ребро

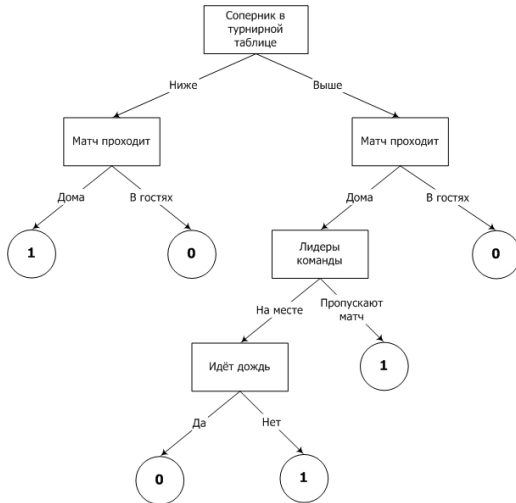
Чтобы классифицировать новый случай, нужно спуститься по дереву до листа и выдать соответствующее значение.

Начальные данные

Таблица: Как играет «Зенит».

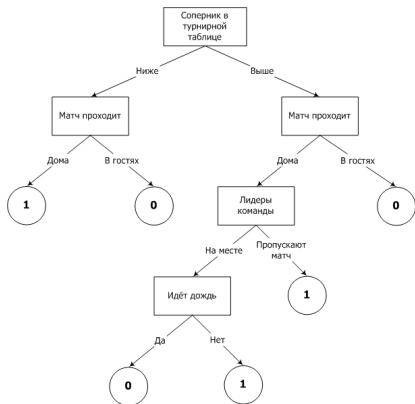
Соперник	Играем	Лидеры	Дождь	Победа
Выше	Дома	На месте	Да	Нет
Выше	Дома	На месте	Нет	Да
Выше	Дома	Пропускают	Нет	Да
Ниже	Дома	Пропускают	Нет	Да
Ниже	В гостях	Пропускают	Нет	Нет
Ниже	Дома	Пропускают	Да	Да
Выше	В гостях	На месте	Да	Нет
Ниже	В гостях	На месте	Нет	???

Само дерево



Его использование

Как использовать:

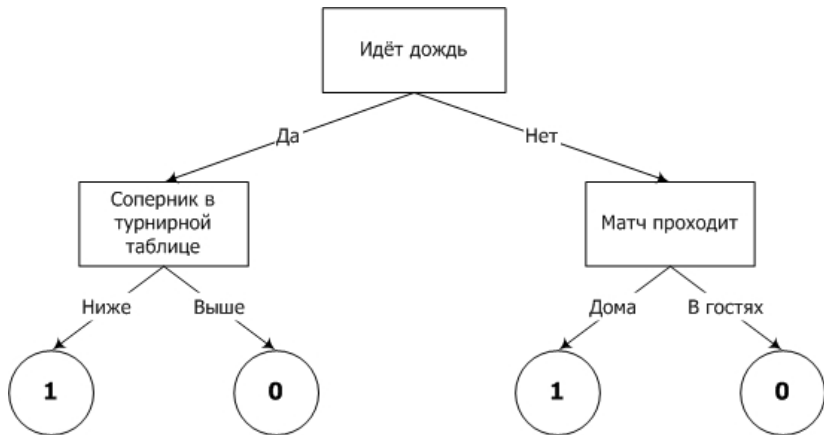


Соперник = Ниже
Играем = В гостях
Лидеры = На месте
Дождь = Нет
Победа = ???

Спускаемся по дереву, выбирая нужные атрибуты, и получаем ответ: судя по нашему дереву, «Зенит» этот матч должен проиграть.

Оптимальное дерево

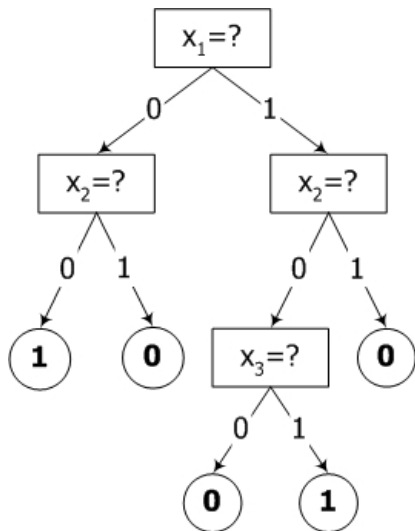
Это большое дерево. А вот дерево для тех же самых данных, но куда меньше:



Outline

- 3 Основные понятия
 - Деревья принятия решений и булевские функции
 - Энтропия и прирост информации
- 4 Алгоритм ID3
 - Сам алгоритм
 - Реализация ID3 на языке Python

Деревья и булевские функции



Из дерева принятия решений легко добыть булевскую функцию в ДНФ.

Например, дерево на рисунке соответствует функции:

$$f(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 \vee x_1 \bar{x}_2 x_3.$$

Упражнения

Упражнение

Нарисовать деревья принятия решений, соответствующие функциям:

- 1 $x \vee (y \wedge \bar{z})$;
- 2 $(x \wedge \bar{y}) \vee (y \wedge \bar{z} \wedge t)$;
- 3 $(x \vee y) \wedge (\bar{y} \vee z)$.

Алгоритм построения

Как строить дерево:

- Выбираем очередной атрибут Q , помещаем его в корень
- Для всех его значений i :
 - Оставляем из тестовых примеров только те, у которых значение атрибута Q равно i
 - Рекурсивно строим дерево в этом потомке
- Выдаём полученное дерево

Главная проблема:

- Как выбирать новый атрибут?

Алгоритм построения

Как строить дерево:

- Выбираем очередной атрибут Q , помещаем его в корень
- Для всех его значений i :
 - Оставляем из тестовых примеров только те, у которых значение атрибута Q равно i
 - Рекурсивно строим дерево в этом потомке
- Выдаём полученное дерево

Главная проблема:

- Как выбирать новый атрибут?

Энтропия

Определение

Предположим, что имеется множество A из n элементов, m из которых обладают некоторым свойством S . Тогда энтропия множества A по отношению к свойству S — это

$$H(A, S) = -\frac{m}{n} \log_2 \frac{m}{n} - \frac{n-m}{n} \log_2 \frac{n-m}{n}.$$

Энтропия зависит от пропорции, в которой разделяется множество. Чем «ровнее» поделили, тем больше энтропия.

Энтропия

Если свойство S не бинарное, а может принимать s различных значений, каждое из которых реализуется в m_i случаях, то

$$H(A, S) = - \sum_{i=1}^s \frac{m_i}{n} \log \frac{m_i}{n}.$$

Энтропия — это среднее количество битов, которые требуются, чтобы закодировать атрибут S у элемента множества A . Если вероятность появления S равна $1/2$, то энтропия равна 1, и нужен полноценный бит; а если S появляется не равновероятно, то можно закодировать последовательность элементов A более эффективно.

Энтропия: пример

В нашем примере из 7 матчей «Зенит» три проиграл и четыре выиграл. Поэтому исходная энтропия

$$H(A, \text{Победа}) = -\frac{4}{7} \log_2 \frac{4}{7} - \frac{3}{7} \log_2 \frac{3}{7} \approx 0.9852.$$

Прирост информации

Атрибут для классификации нужно выбирать так, чтобы после классификации энтропия (относительно целевой функции) стала как можно меньше.

Определение

Предположим, что множество A элементов, характеризующихся свойством S , классифицировано посредством атрибута Q , имеющего q возможных значений. Тогда прирост информации (information gain) определяется как

$$\text{Gain}(A, Q) = H(A, S) - \sum_{i=1}^q \frac{|A_i|}{|A|} H(A_i, S),$$

где A_i — множество элементов A , на которых атрибут Q имеет значение i .

Прирост информации: пример

Теперь вычислим приросты информации для различных атрибутов:

$$\begin{aligned} \text{Gain}(A, \text{Соперник}) &= H(A, \text{Победа}) - \frac{4}{7}H(A_{\text{выше}}, \text{Победа}) - \\ &\quad - \frac{3}{7}H(A_{\text{ниже}}, \text{Победа}) \approx \\ &\approx 0.9852 - \frac{4}{7} \left(-\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} \right) - \\ &\quad - \frac{3}{7} \left(-\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} \right) \approx 0.0202. \end{aligned}$$

Мы явно выбрали не слишком удачный атрибут для корня дерева...

Прирост информации: пример

$$\text{Gain}(A, \text{Играем}) \approx 0.4696.$$

$$\text{Gain}(A, \text{Лидеры}) \approx 0.1281.$$

$$\text{Gain}(A, \text{Дождь}) \approx 0.1281.$$

Прирост информации советует сначала классифицировать по тому, домашний ли матч или гостевой.

Упражнение

Дерево (проверьте) получится глубины 3. Как нужно модифицировать выбор атрибутов, чтобы получить дерево глубины 2, причём с меньшим количеством узлов, чем в приведённом выше?

Outline

- 3 Основные понятия
 - Деревья принятия решений и булевы функции
 - Энтропия и прирост информации
- 4 Алгоритм ID3
 - Сам алгоритм
 - Реализация ID3 на языке Python

Алгоритм ID3

$ID3(A, S, Q)$

- Создать корень дерева.
- Если S выполняется на всех элементах A , поставить в корень метку 1 и выйти.
- Если S не выполняется ни на одном элементе A , поставить в корень метку 0 и выйти.
- Если $Q = \emptyset$, то:
 - если S выполняется на половине или большей части A , поставить в корень метку 1 и выйти;
 - если S не выполняется на большей части A , поставить в корень метку 0 и выйти.

- Выбрать $Q \in \mathcal{Q}$, для которого $\text{Gain}(A, Q)$ максимален.
- Поставить в корень метку Q .
- Для каждого значения q атрибута Q :
 - добавить нового потомка корня и пометить соответствующее исходящее ребро меткой q ;
 - если в A нет случаев, для которых Q принимает значение q (т.е. $|A_q| = 0$), то пометить этого потомка в зависимости от того, на какой части A выполняется S (аналогично пункту 1);
 - иначе запустить $ID3(A_q, S, \mathcal{Q} \setminus \{Q\})$ и добавить его результат как поддерево с корнем в этом потомке.

Реализация ID3 на языке Python

- Будет в lecture notes
- Будет на моей страничке:
<http://logic.pdmi.ras.ru/~sergey/index.php?page=teaching>

Упражнение
Разобраться!

Outline

- 5 Проблемы критерия прироста информации
 - Когда критерий прироста информации не работает
 - Gain Ratio
 - Индекс Гини

- 6 Оверфиттинг
 - Что это такое
 - Как с ним бороться

Проблема критерия прироста информации

Проблема: прирост информации выбирает атрибуты, у которых больше всего значений. Например, пусть в таблице игр были записаны ещё и даты матчей. Прирост информации:

$$\begin{aligned} \text{Gain}(A, \text{Дата}) &= H(A, \text{Победа}) - \\ &- \sum_{i=1}^n \frac{1}{n} H(A_{\text{Дата}=i}, \text{Победа}) = H(A, \text{Победа}), \end{aligned}$$

потому что в каждой из веток только один случай, и энтропия в каждой ветке равна нулю.

Прирост информации — максимальный из возможных, но полученное дерево абсолютно бесполезно.

Gain Ratio

Gain Ratio учитывает не только количество информации, требуемое для записи результата, но и количество информации, требуемое для разделения по текущему атрибуту.
Поправка:

$$\text{SplitInfo}(A, Q) = - \sum_{i=1}^q \frac{|A_q|}{|A|} \log_2 \frac{|A_q|}{|A|},$$

Сам критерий — максимизация величины

$$\text{GainRatio}(A, Q) = \frac{\text{Gain}(A, Q)}{\text{SplitInfo}(A, Q)}.$$

Gain Ratio: пример

У атрибута «Дата»

$$\text{SplitInfo}(A, \text{Дата}) = - \sum_{i=1}^7 \frac{1}{7} \log_2 \frac{1}{7} \approx 2.80735 \dots,$$

и Gain Ratio получается равным

$$\text{GainRatio}(A, \text{Дата}) = \frac{\text{Gain}(A, \text{Дата})}{\text{SplitInfo}(A, \text{Дата})} \approx 0.350935 \dots$$

А для атрибута, показывающего, где проходит матч,

$$\text{SplitInfo}(A, \text{Играем}) = -\frac{5}{7} \log_2 \frac{5}{7} - \frac{2}{7} \log_2 \frac{2}{7} \approx 0.86312 \dots,$$

и итоговый Gain Ratio получается

$$\text{GainRatio}(A, \text{Играем}) = \frac{\text{Gain}(A, \text{Играем})}{\text{SplitInfo}(A, \text{Играем})} \approx 0.5452 \dots$$

Индекс Гини

Для набора тестов A и свойства S , имеющего s значений, этот индекс вычисляется как

$$\text{Gini}(A, S) = 1 - \sum_{i=1}^s \frac{|A_i|}{|A|}.$$

Соответственно, для набора тестов A , атрибута Q , имеющего q значений, и целевого свойства S , имеющего s значений, индекс вычисляется следующим образом:

$$\text{Gini}(A, Q, S) = \text{Gini}(A, S) - \sum_{j=1}^q \frac{|A_j|}{|A|} \text{Gini}(A_j, S).$$

Упражнения

Упражнение

Добавить в предыдущую реализацию возможность выбирать атрибуты по Gain Ratio.

Упражнение

Добавить в предыдущую реализацию возможность выбирать атрибуты по индексу Гини.

Outline

- 5 Проблемы критерия прироста информации
 - Когда критерий прироста информации не работает
 - Gain Ratio
 - Индекс Гини

- 6 Оверфиттинг
 - Что это такое
 - Как с ним бороться

Оверфиттинг

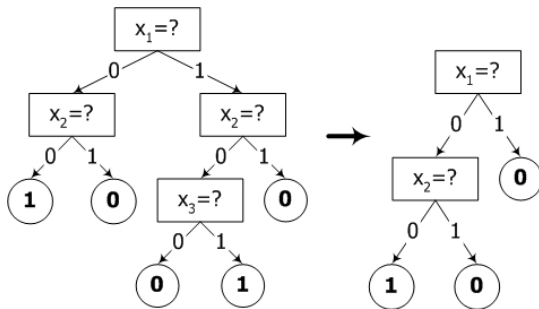
- ID3 удовлетворяет *всем* данным
- Но часть данных могут быть «шумом» или содержать ошибки
- Из-за этого дерево сильно растёт и хуже работает

Оверфиттинг: пример

- Пусть «Зенит» дома выигрывает в 90% случаев, и ни от чего это больше не зависит.
- И среди исходных данных имеется одно домашнее поражение
- ID3 учтёт все «причины» и будет в дальнейшем предсказывать, что «Зенит» проиграет в аналогичных ситуациях
- Но на самом деле он будет выигрывать с вероятностью 90%

Обрезание

Надо научиться обрезать лишние ветки. Обычно это делают так: ветку заменяют на значение, которое принимает большинство тестовых примеров в этой ветке.



Как выяснить, какие ветки обрезать?

Обрезание: общий алгоритм

- Построим дерево по части исходных данных
- Тестировать будем на оставшейся части
- Для каждой вершины:
 - Обрежем ветку с корнем в этой вершине
 - Если обрезанное дерево будет лучше справляться с тестами, так и оставим обрезанную ветку, иначе вернём как было

Упражнение

Добавить в предыдущую реализацию обрезание лишних веток.

Спасибо за внимание!

- Lecture notes, слайды и коды программ появятся на моей homepage:
`http://logic.pdmi.ras.ru/~sergey/index.php?page=teaching`
- Присылайте любые замечания, коды программ на других языках, решения упражнений, новые численные примеры и прочее по адресам:
`sergey@logic.pdmi.ras.ru, smartnik@inbox.ru`