Probably approximately correct learning (PAC learning)

Арсения Шихова, БПМИ142

18 апреля 2017г.

- Что такое обучение с учителем:
  - Дана обучающая выборка $(x_1, y_1), \ldots, (x_n, y_n)$
  - Дан класс функций A, обычно функции какого-то конкретного вида, зависящие от вектора параметров
  - Для фиксированного алгоритма $a \in A$ дана функция потерь вида $\sum\limits_{i=1}^{n} L(a(x_i), y_i)$
  - Надо найти $\min\limits_{a \in A} \sum\limits_{i=1}^{n} L(a(x_i), y_i)$

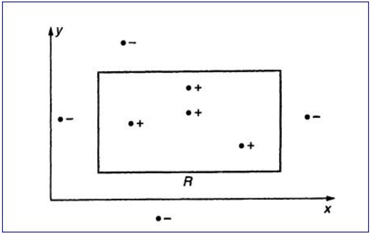
- Что такое обучение с учителем:
  - Дана обучающая выборка $(x_1, y_1), \dots, (x_n, y_n)$
  - Дан класс функций A, обычно функции какого-то конкретного вида, зависящие от вектора параметров
  - Для фиксированного алгоритма $a \in A$ дана функция потерь вида $\sum\limits_{i=1}^{n} L(a(x_i), y_i)$
  - Надо найти $\min\limits_{a \in A} \sum\limits_{i=1}^{n} L(a(x_i), y_i)$
- Можно ли найти решение за полиномиальное от n время?

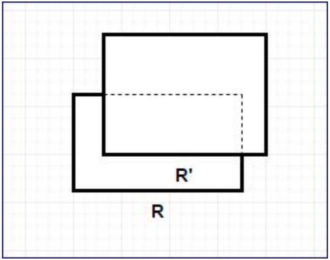
- Что такое обучение с учителем:
  - Дана обучающая выборка $(x_1, y_1), \dots, (x_n, y_n)$
  - Дан класс функций A, обычно функции какого-то конкретного вида, зависящие от вектора параметров
  - Для фиксированного алгоритма $a \in A$ дана функция потерь вида $\sum\limits_{i=1}^{n} L(a(x_i), y_i)$
  - Надо найти $\min\limits_{a \in A} \sum\limits_{i=1}^{n} L(a(x_i), y_i)$
- Можно ли найти решение за полиномиальное от n время?
- Теоретические обоснования — делать предположения о:
  - Свойствах семейства алгоритмов A
  - Свойствах генератора данных

- Что такое обучение с учителем:
  - Дана обучающая выборка $(x_1, y_1), \ldots, (x_n, y_n)$
  - Дан класс функций A, обычно функции какого-то конкретного вида, зависящие от вектора параметров
  - Для фиксированного алгоритма $a \in A$ дана функция потерь вида $\sum\limits_{i=1}^{n} L(a(x_i), y_i)$
  - Надо найти $\min\limits_{a \in A} \sum\limits_{i=1}^{n} L(a(x_i), y_i)$
- Можно ли найти решение за полиномиальное от n время?
- Теоретические обоснования — делать предположения о:
  - Свойствах семейства алгоритмов A
  - Свойствах генератора данных
- PAC-обучение — это:
  - Есть какое-то неизвестное, но фиксированное распределение D на всём признаковом пространстве $\mathbb{X}$, из которого берутся объекты
  - Независимо и случайно выбираем m точек из D и по ним строим алгоритм такой, что с вероятностью не больше δ даёт ошибку, превосходящую ε для некоторых малых ε, δ

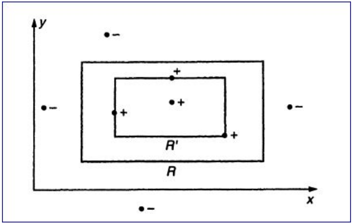
Пример задачи: Rectangle learning game

Играет один человек. Ему нужно найти координаты вершин фиксированного прямоугольника R со сторонами, параллельными осям координат. Один ход: игрок выбирает случайную точку $p \in \mathbb{R}^2$ и получает 1, если $p \in R$, и 0 иначе. Точно решить эту задачу за конечное число шагов нельзя. Но можно найти какой-то прямоугольник R', который будет близок к R.

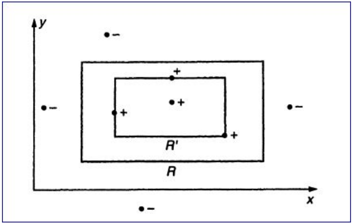




Идея: если игрок нашёл довольно много ($\geqslant 2$) точек, принадлежащих R, то можно построить самый маленький $R' \subseteq R$



Идея: если игрок нашёл довольно много ($\geqslant 2$) точек, принадлежащих R, то можно построить самый маленький $R' \subseteq R$



Но насколько хорошо R' приближает R?

- Предположим, что все точки берутся независимо из фиксированного распределения D, не обязательно равномерного

- Предположим, что все точки берутся независимо из фиксированного распределения D, не обязательно равномерного
- $\sum\limits_{p\in\mathbb{R}^2} Pr\left[p \in R\triangle R'\right]$ — ошибка на R'; так как согласно выбранной стратегии $R' \subseteq R$, то $R\triangle R' = R\backslash R'$

- Предположим, что все точки берутся независимо из фиксированного распределения D, не обязательно равномерного
- $\sum\limits_{p\in\mathbb{R}^2} Pr\left[p \in R\triangle R'\right]$ — ошибка на R'; так как согласно выбранной стратегии $R' \subseteq R$, то $R\triangle R' = R\backslash R'$
- Зафиксируем малые ε и δ $(0 < \varepsilon, \delta < 1)$ и найдём такое m, что игрок, сделавший m ходов, получит R', ошибка на котором будет не больше ε с вероятностью $1 - \delta$

- $R \backslash R'$ — объединение четырёх пересекающихся прямоугольных полос; достаточно потребовать для каждой из них, чтобы вероятность попадания случайной точки из распределения D была не более чем $\frac{\varepsilon}{4}$

- $R \backslash R'$ — объединение четырёх пересекающихся прямоугольных полос; достаточно потребовать для каждой из них, чтобы вероятность попадания случайной точки из распределения D была не более чем $\frac{\varepsilon}{4}$
- Если это не так для верхней полосы T' — вероятность того, что m точек из D не попали в T' меньше или равна $(1 - \frac{\varepsilon}{4})^m$

- $R \backslash R'$ — объединение четырёх пересекающихся прямоугольных полос; достаточно потребовать для каждой из них, чтобы вероятность попадания случайной точки из распределения D была не более чем $\frac{\varepsilon}{4}$
- Если это не так для верхней полосы T' — вероятность того, что m точек из D не попали в T' меньше или равна $(1 - \frac{\varepsilon}{4})^m$
- Так как для любых событий A, B: $Pr[A \cup B] \leqslant Pr[A] + Pr[B]$, то вероятность того, что m точек, взятых независимо из D, не попадут в $R \backslash R'$, не больше $4(1 - \frac{\varepsilon}{4})^m$

Итак, мы доказали теорему:

$\forall$ *распределения* D *на* $\mathbb{R}^2$, $\forall\ 0 < \varepsilon, \delta < 1$ *достаточно взять* m *такое, что* $4(1 - \frac{\varepsilon}{4})^m \leqslant \delta$, *и получить алгоритм, который с вероятностью* $1 - \delta$ *решает задачу поиска прямоугольника с ошибкой не более чем* ε.

- X — **признаковое пространство**; например $\mathbb{R}^2$

- X — **признаковое пространство**; например $\mathbb{R}^2$
- **Представление** $c : X \to \{0, 1\}$ — можно понимать как подмножество объектов X, удовлетворяющих некоторому свойству, то есть $\{x \in X : c(x) = 1\}$; например фиксированный прямоугольник R

- X — **признаковое пространство**; например $\mathbb{R}^2$
- **Представление** $c : X \to \{0, 1\}$ — можно понимать как подмножество объектов X, удовлетворяющих некоторому свойству, то есть $\{x \in X : c(x) = 1\}$; например фиксированный прямоугольник R
- **Класс представлений** C — объединение нескольких представлений; например множество всевозможных прямоугольников со сторонами, параллельными осям координат

- X — **признаковое пространство**; например $\mathbb{R}^2$
- **Представление** $c : X \to \{0, 1\}$ — можно понимать как подмножество объектов X, удовлетворяющих некоторому свойству, то есть $\{x \in X : c(x) = 1\}$; например фиксированный прямоугольник R
- **Класс представлений** C — объединение нескольких представлений; например множество всевозможных прямоугольников со сторонами, параллельными осям координат
- D — фиксированное распределение на X

- X — **признаковое пространство**; например $\mathbb{R}^2$
- **Представление** $c : X \to \{0, 1\}$ — можно понимать как подмножество объектов X, удовлетворяющих некоторому свойству, то есть $\{x \in X : c(x) = 1\}$; например фиксированный прямоугольник R
- **Класс представлений** C — объединение нескольких представлений; например множество всевозможных прямоугольников со сторонами, параллельными осям координат
- D — фиксированное распределение на X
- Если h — представление в X, то для другого представления c **ошибка на** h: $Error(h) = \sum\limits_{x \in X} Pr[x] \cdot [c(x) \neq h(x)]$; например вероятность попасть в $R \triangle R'$

- X — **признаковое пространство**; например $\mathbb{R}^2$
- **Представление** $c : X \to \{0, 1\}$ — можно понимать как подмножество объектов X, удовлетворяющих некоторому свойству, то есть $\{x \in X : c(x) = 1\}$; например фиксированный прямоугольник R
- **Класс представлений** C — объединение нескольких представлений; например множество всевозможных прямоугольников со сторонами, параллельными осям координат
- D — фиксированное распределение на X
- Если h — представление в X, то для другого представления c **ошибка на** h: $Error(h) = \sum_{x \in X} Pr[x] \cdot [c(x) \neq h(x)]$; например вероятность попасть в $R \triangle R'$
- **Оракул** $EX(c, D)$ — процедура, возвращающая пару $(x, c(x))$, где x взято случайным образом из распределения D; предполагаем, что оракул работает за фиксированое небольшое количество времени

Мы хотим строить „хорошие“ алгоритмы, то есть такие, что:

- Число вызовов оракула не очень большое
- Алгоритм возвращает представление h такое, что $Error(h)$ не очень большое

А если говорить формально, то:

Определение

*Пусть заданы признаковое пространство X и класс представлений C. C называется PAC-**обучаемым**, если существует алгоритм L такой, что для любого распределения D, любого представления $c \in C$ и для любых $0 < \varepsilon, \delta < 1$:*

- *L принимает на вход ε, δ и $EX(c, D)$*
- *L после m вызовов оракула возвращает представление h*
- *С вероятностью не менее $1 - \delta$ $Error(h) \leqslant \varepsilon$*

*Если при этом m зависит полиномиально от $\frac{1}{\varepsilon}$ и $\frac{1}{\delta}$, то C **эффективно** PAC-**обучаемый**.*

Рассмотрим более общую модель:

- X_n — какое-то множество, определяемое константой n; обычно $X_n = \{0, 1\}^n$ или $X_n = \mathbb{R}^n$

Рассмотрим более общую модель:

- X_n — какое-то множество, определяемое константой n; обычно $X_n = \{0, 1\}^n$ или $X_n = \mathbb{R}^n$
- C_n — класс представлений в X_n

Рассмотрим более общую модель:

- X_n — какое-то множество, определяемое константой n; обычно $X_n = \{0,1\}^n$ или $X_n = \mathbb{R}^n$
- C_n — класс представлений в X_n
- $X = \bigcup\limits_{n=1}^{\infty} X_n$; $C = \bigcup\limits_{n=1}^{\infty} C_n$

Рассмотрим более общую модель:

- X_n — какое-то множество, определяемое константой n; обычно $X_n = \{0,1\}^n$ или $X_n = \mathbb{R}^n$
- C_n — класс представлений в X_n
- $X = \bigcup\limits_{n=1}^{\infty} X_n$; $C = \bigcup\limits_{n=1}^{\infty} C_n$
- Для фиксированного алфавита Σ существует отображение $R : \Sigma^* \to X$; то есть каждое представление $c \subseteq X$ как-то кодируется, возможно неоднозначно; часто $\Sigma = \{0,1\}$

Рассмотрим более общую модель:

- X_n — какое-то множество, определяемое константой n; обычно $X_n = \{0,1\}^n$ или $X_n = \mathbb{R}^n$
- C_n — класс представлений в X_n
- $X = \bigcup\limits_{n=1}^{\infty} X_n$; $C = \bigcup\limits_{n=1}^{\infty} C_n$
- Для фиксированного алфавита Σ существует отображение $R : \Sigma^* \to X$; то есть каждое представление $c \subseteq X$ как-то кодируется, возможно неоднозначно; часто $\Sigma = \{0,1\}$
- **Размер представления** $size(c) = \min\limits_{\sigma \in \Sigma^* : R(\sigma) = c} size(\sigma)$, где $size(\sigma)$ — просто длина строки

Тогда можно дать определение эффективной *PAC*-обучаемости аналогично имеющемуся, только требовать от алгоритма, чтобы время его работы было полиномом от $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$, n и $size(c)$

- Для любых ε и δ, $c \in C$, D...

- Для любых ε и δ, $c \in C$, D...
- А что если заменить в определении PAC-обучаемости любые ε и δ на какие-то фиксированные ε_0 и δ_0?

Теорема

Множество всех возможных конъюнкций литералов эффективно PAC-обучаемо.

Постановка задачи:

- Вход: число n; оракул $EX(c, D)$, где c какая-то конъюнкция литералов, D — распределение на $\{0, 1\}^n$; ε; δ
- Выход: конъюнкция литералов h такая, что с вероятностью $1 - \delta$ ошибка h на C не превосходит ε; то есть $\sum\limits_{a \in \{0,1\}^n} Pr[a] \cdot [h(a) \neq c(a)] \leqslant \varepsilon$

Процесс построения алгоритма:

1. Обозначим $h = x_1 \wedge \neg x_1 \wedge x_2 \wedge \neg x_2 \wedge \ldots \wedge x_n \wedge \neg x_n$

Процесс построения алгоритма:

1. Обозначим $h = x_1 \wedge \neg x_1 \wedge x_2 \wedge \neg x_2 \wedge \ldots \wedge x_n \wedge \neg x_n$
2. После каждого вызова оракула, получая пару $(a, c(a))$:
   - Если $c(a) = 0$, не делать ничего
   - Если $c(a) = 1$, то $\forall\ i : 1 \leqslant i \leqslant n$:
     если $a_i = 0$, то удаляем из h литерал x_i; если $a_i = 1$, удаляем $\neg x_i$

- Заметим, что если литерал z встречается в c, то он не будет удалён из h

- Заметим, что если литерал z встречается в c, то он не будет удалён из h
- Для литерала z, не встречающегося в c, обозначим вероятность быть удалённым после одного вызова оракула
  $p(z) = \sum_{a \in \{0,1\}^n} Pr[a] \cdot [c(a) = 1 \wedge z = 0 \text{ в } a]$

- Заметим, что если литерал z встречается в c, то он не будет удалён из h
- Для литерала z, не встречающегося в c, обозначим вероятность быть удалённым после одного вызова оракула $p(z) = \sum\limits_{a \in \{0,1\}^n} Pr[a] \cdot [c(a) = 1 \wedge z = 0 \text{ в } a]$
- $Error(h) \leqslant \sum\limits_{z \in h} p(z)$

- Заметим, что если литерал z встречается в c, то он не будет удалён из h
- Для литерала z, не встречающегося в c, обозначим вероятность быть удалённым после одного вызова оракула $p(z) = \sum\limits_{a \in \{0,1\}^n} Pr[a] \cdot [c(a) = 1 \wedge z = 0 \text{ в } a]$
- $Error(h) \leqslant \sum\limits_{z \in h} p(z)$
- Назовём **плохим** литерал z, если $p(z) > \frac{\varepsilon}{2n}$

- Заметим, что если литерал z встречается в c, то он не будет удалён из h
- Для литерала z, не встречающегося в c, обозначим вероятность быть удалённым после одного вызова оракула $p(z) = \sum\limits_{a \in \{0,1\}^n} Pr[a] \cdot [c(a) = 1 \wedge z = 0 \text{ в } a]$
- $Error(h) \leqslant \sum\limits_{z \in h} p(z)$
- Назовём **плохим** литерал z, если $p(z) > \frac{\varepsilon}{2n}$
- Вероятность того, что после m вызовов оракула в h останется хотя бы один плохой литерал, не превосходит $2n\left(1 - \frac{\varepsilon}{2n}\right)^m \leqslant \delta$

- Обобщение задачи о прямоугольнике на n-мерное пространство эффективно PAC-обучаемо
- Множество всех 3-ДНФ* не является эффективно PAC-обучаемым, если предположить, что $NP \neq RP$**

*Дизъюнкция конъюнкций не более чем трёх переменных

**Прочитать про сложностные классы NP и RP можно здесь и здесь в Википедии, а доказательство — в книге Вазирани (см. список литературы), глава 1.4, стр. 18-22

Список литературы:

1. https://jeremykun.com/2014/01/02/probably-approximately-correct-a-formal-theory-of-learning/
2. M. Kearns, U. Vazirani. An Introduction to Computational Learning Theory. Глава 1.

Спасибо за внимание! Вопросы?