Байесовское обучение II. Обучение концептам.

Сергей Николенко

Машинное обучение — ИТМО, осень 2006

Outline

Concept learning

- Обучение концептам, иногда ещё называют «формированием понятий».
- Метод обучения, принципиально похожий на деревья принятия решений, но с другими результатами
- Здесь немного не на месте, но хорошая иллюстрация.
- Это ещё один метод, результаты которого будут соответствовать MAP для тех же условий.

Два алгоритма

- Find-S — попроще и похуже.
- Candidate Elimination — посложнее и получше.

Основные принципы

- Обычная задача классификации — игры «Зенита».
- Будем рассматривать гипотезы в виде набора причин, из которых следует, что целевая функция равна 1.
- Гипотеза — набор значений атрибутов. Может содержать ? (любое значение) или $\emptyset$ (ни одного значения, пустая гипотеза).

Примеры гипотез

Для игр «Зенита» атрибуты:

$$\langle \text{Соперник}, \text{Играем}, \text{Лидеры}, \text{Дождь} \rangle$$

Примеры гипотез:

$$\langle \text{Выше}, \text{Дома}, ?, ? \rangle$$

$$\langle ?, \text{В гостях}, \text{Играют}, ? \rangle$$

$$\langle ?, ?, ?, ? \rangle$$

$$\langle \emptyset, \emptyset, \emptyset, \emptyset \rangle$$

Более общие и более частные гипотезы

- На гипотезах есть естественный порядок.
- Гипотеза h_1 называется *более общей*, чем гипотеза h_2 (обозначаем $h_1 \succ h_2$), если для всякого тестового примера d, если $h_2(d) = 1$, то и $h_1(d) = 1$.
- Гипотеза $\langle \emptyset, \ldots, \emptyset \rangle$ является самой частной из всех гипотез, гипотеза $\langle ?, \ldots, ? \rangle$ — самой общей.

Find-S: идея

- Алгоритм Find-S очень прост: нужно начать с самой частной гипотезы, а затем обобщать её так, чтобы она включала в себя все тестовые примеры.
- На каждом шаге, если текущая гипотеза h неправильно классифицирует пример $(h(d) = 0)$, нужно искать минимальную h' из тех, для которых $h' \succ h$ и $h' = 0$; т.е., просто заменять неподходящие значения на ?.
- В результате получится максимально частная гипотеза, отвечающая всем тестовым данным.
- Негативные тестовые примеры вообще игнорируются.

Алгоритм Find-S

$\texttt{FindS}(D)$

- $h := \langle \emptyset, \ldots, \emptyset \rangle$.
- Для всех $d \in D^+$:
  - Если $h(d) = 0$:
    - $h' := h$.
    - Для каждого атрибута a, если $h[a] \neq d[a]$, то $h'[a] := ?$.
    - $h := h'$.
- Выдать h.

Find-S: пример

- Возьмём игры «Зенита» и тестовые примеры (только положительные)

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

Find-S: пример

- $h = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle$.
- Первый пример: Выше Дома На месте Нет
- h выдаёт 0, значит, надо обобщать. Максимально частная гипотеза просто совпадёт с этим тестовым примером.

Find-S: пример

- $h = \langle$Выше, Дома, На месте, Нет$\rangle$.
- Второй пример: Выше Дома Пропускают Нет
- h выдаёт 0. Максимально частная гипотеза, объединяющая h и этот пример, обобщит значение атрибута Лидеры.

Find-S: пример

- $h = \langle$Выше, Дома, ?, Нет$\rangle$.
- Третий пример: Ниже Дома Пропускают Нет
- h выдаёт 0. Максимально частная гипотеза, объединяющая h и этот пример, обобщит значение атрибута Соперник.

Find-S: пример

- Итого получаем гипотезу, объясняющую все тестовые примеры:

$$h = \langle ?, \text{Дома}, ?, \text{Нет} \rangle.$$

За и против

- Очень простой и быстрый алгоритм.
- Совсем не выразительный — нужно, чтобы целевая функция выражалась такой гипотезой, т.е. фактически одной веткой дерева; дизъюнкции не разрешаются.
- Но зато, если всё же выражается, то полученный результат будет правильно классифицировать и негативные примеры тоже (т.к. результат максимально частный).

Общие принципы

- Find-S позволяет найти наиболее частную гипотезу, совместную с положительными примерами.
- Что напрашивается?

Общие принципы

- Find-S позволяет найти наиболее частную гипотезу, совместную с положительными примерами.
- Что напрашивается?
- Хочется найти наиболее общую гипотезу, совместную с негативными примерами.

Общие принципы

- Find-S позволяет найти одну из возможных гипотез, а их на самом деле миллион.
- Что напрашивается?

Общие принципы

- Find-S позволяет найти одну из возможных гипотез, а их на самом деле миллион.
- Что напрашивается?
- Хочется искать два множества: множество минимальных относительно $\succ$ (наиболее частных) гипотез, совместных с данными, и множество максимальных (наиболее общих).

Общие принципы

- Find-S позволяет найти одну из возможных гипотез, а их на самом деле миллион.
- Что напрашивается?
- Хочется искать два множества: множество минимальных относительно $\succ$ (наиболее частных) гипотез, совместных с данными, и множество максимальных (наиболее общих). В этом случае то, что получится, будет границами всего множества допустимых гипотез, и любая другая гипотеза, совместная с тестовыми данными, будет находиться между этими двумя.

Сеттинг

- Итак, есть два множества: G — множество максимальных (общих) гипотез, и S — множество минимальных (частных) гипотез.
- Когда приходит новый пример d, есть несколько возможностей:

Сеттинг

- Итак, есть два множества: G — множество максимальных (общих) гипотез, и S — множество минимальных (частных) гипотез.
- Когда приходит новый пример d, есть несколько возможностей:
- $t_d = 1$
- В этом случае, если оказывается, что гипотеза из G неверно классифицирует d, её нужно удалить.
- А если неверно классифицируют гипотезы из S, их нужно соответственно обобщить. Но обобщить так, чтобы подходила хоть какая-то из гипотез из G.

Сеттинг

- Итак, есть два множества: G — множество максимальных (общих) гипотез, и S — множество минимальных (частных) гипотез.
- Когда приходит новый пример d, есть несколько возможностей:
- $t_d = 0$
- В этом случае, если оказывается, что гипотеза из S неверно классифицирует d, её нужно удалить.
- А если неверно классифицируют гипотезы из G, их нужно соответственно специализировать (добавить все минимальные специализации, совместные с d и такие, чтобы была соответствующая гипотеза в h).

Алгоритм Candidate Elimination

$\texttt{CandidateElimination}(D)$

- $H := \{\langle \emptyset, \ldots, \emptyset \rangle\}$.
- $G := \{\langle ?, \ldots, ? \rangle\}$.
- Для всех $d \in D$:
  - Если $t_d = 1$:
    - Для каждой $h \in G$, если $h(d) = 0$, $G := G \setminus \{h\}$.
    - Для каждой $h \in S$, если $h(d) = 0$, $S := S \setminus \{h\} \cup \mathrm{Gen}(h)$, где $\mathrm{Gen}(h) = \min\{h' \mid h' \succ h, h'(d) = 1, \exists h_g \in G : \ h_g \succ h'\}$.
  - Если $t_d = 0$:
    - Для каждой $h \in S$, если $h(d) = 1$, $S := S \setminus \{h\}$.
    - Для каждой $h \in G$, если $h(d) = 1$, $G := G \setminus \{h\} \cup \mathrm{Spe}(h)$, где $\mathrm{Spe}(h) = \max\{h' \mid h \succ h', h'(d) = 0, \exists h_s \in S : \ h' \succ h_s\}$.
- Выдать G, H.

Candidate elimination: пример

- Возьмём игры «Зенита» и тестовые примеры (в том числе негативные)

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

Candidate elimination: пример

- $H = \{\langle \emptyset, \emptyset, \emptyset, \emptyset \rangle\}$, $G := \{\langle ?, ?, ?, ? \rangle\}$.
- Первый пример: Выше Дома На месте Нет Да
- $H[0]$ выдаёт 0, значит, надо обобщать. Максимально частная гипотеза просто совпадёт с этим тестовым примером.

Candidate elimination: пример

- $H = \{\langle \text{Выше}, \text{Дома}, \text{На месте}, \text{Нет}\rangle\}$, $G := \{\langle ?, ?, ?, ?\rangle\}$.
- Второй пример:

  Ниже В гостях Пропускают Нет Нет
- $G[0]$ выдаёт 0 — надо специализировать. Максимально общие гипотезы, которые обобщают $H[0]$ и выдают 0:

$$\langle \text{Выше}, ?, ?, ?\rangle, \langle ?, \text{Дома}, ?, ?\rangle, \langle ?, ?, \text{На месте}, ?\rangle.$$

Candidate elimination: пример

- $H = \{\langle \text{Выше}, \text{Дома}, \text{На месте}, \text{Нет} \rangle\}$,
  $G = \{\langle \text{Выше}, ?, ?, ? \rangle, \langle ?, \text{Дома}, ?, ? \rangle, \langle ?, ?, \text{На месте}, ? \rangle\}$.
- Третий пример: Ниже Дома Пропускают Да Нет
- $G[1]$ выдаёт 1; $\text{Spe}(G[1]) =$
  $\{\langle \text{Выше}, \text{Дома}, ?, ? \rangle, \langle ?, \text{Дома}, \text{На месте}, ? \rangle, \langle ?, \text{Дома}, ?, \text{Нет} \rangle\}$.

Candidate elimination: пример

- $H = \{\langle \text{Выше}, \text{Дома}, \text{На месте}, \text{Нет} \rangle\}$,

  $$G = \{\langle \text{Выше}, ?, ?, ? \rangle, \langle ?, ?, \text{На месте}, ? \rangle, \langle ?, \text{Дома}, ?, \text{Нет} \rangle\}.$$

- Четвёртый пример:

  Выше Дома Пропускают Нет Да

- $G[1]$ выдаёт 0, поэтому её удаляем.
- $H[0]$ надо обобщить, получим $\langle \text{Выше}, \text{Дома}, ?, \text{Нет} \rangle$.

Candidate elimination: пример

- Итого получаем максимально частную гипотезу, объясняющую все тестовые примеры:

$$h = \langle ?, \text{Дома}, ?, \text{Нет} \rangle$$

и набор максимально общих гипотез, совместных со всеми данными:

$$G = \{\langle \text{Выше}, ?, ?, ? \rangle, \langle ?, \text{Дома}, ?, \text{Нет} \rangle\}.$$

За и против

- Хорошо работает, если целевая функция содержится во множестве возможных гипотез.
- В противном случае может сойтись к пустому множеству.
- Это фактически единственный недостаток (но очень серьёзный, потому что множество гипотез довольно невыразительное).

Упражнения

Упражнение

Реализовать алгоритм Find-S для классификации с теми же форматами входных и выходных данных, что ID3.

Упражнение

Реализовать алгоритм Candidate Elimination для классификации с теми же форматами входных и выходных данных, что ID3.

Outline

Вспоминаем прошлую лекцию

- Для поиска MAP нужно научиться искать $p(h)$ и $p(D|h)$.
- Пусть выполняются следующие условия:
  - В D нет шума (т.е. все тестовые примеры с правильными ответами).
  - Целевая функция c лежит в H.
  - Нет априорных причин верить, что одна из гипотез более вероятна, чем другая.
- Они выполняются, например, в задачах классификации.

Вспоминаем прошлую лекцию

- Из третьего условия следует:

$$p(h) = \frac{1}{|H|} \text{ для всех } h \in H.$$

- $p(D|h)$ — вероятность наблюдать значения целевых функций $D = \langle d_1, \ldots, d_m \rangle$ для фиксированного набора входных данных $\langle x_1, \ldots, x_m \rangle$ при условии гипотезы h. Поскольку шума нет, $p(d_i|h) = 1$, если $d_i = h(x_i)$, и 0 в противном случае. Итого:

$$p(D|h) = \begin{cases} 1, & \text{если } d_i = h(x_i) \text{ для всех } d_i \in D, \\ 0, & \text{в противном случае.} \end{cases}$$

Вспоминаем прошлую лекцию

- Давайте подсчитаем вероятность $p(D)$. $\mathrm{Cons}(D)$ — множество гипотез $h \in H$, совместимых с D. Тогда:

$$p(D) = \sum_{h \in H} p(D|h)p(h) = \sum_{h \in \mathrm{Cons}(D)} \frac{1}{|H|} = \frac{|\mathrm{Cons}(d)}{|H|}.$$

- Итого получается:

$$p(h|D) = \begin{cases} \frac{1}{|\mathrm{Cons}(d)|}, & \text{если } d_i = h(x_i) \text{ для всех } d_i \in D, \\ 0, & \text{в противном случае.} \end{cases}$$

- То есть каждая гипотеза, совместимая со всеми данными — максимальная апостериорная гипотеза. Пока вроде бы ничего нового.

Зачем нужен MAP для анализа алгоритмов

- Казалось бы, мы ничего нового не узнали: алгоритм выдаёт MAP, ну и что?
- Важно другое — важны *предположения*, в которых мы смогли это доказать.

Зачем нужен MAP для анализа алгоритмов

- Казалось бы, мы ничего нового не узнали: алгоритм выдаёт MAP, ну и что?
- Важно другое — важны *предположения*, в которых мы смогли это доказать.
- Пусть выполняются следующие условия:
  - В D нет шума (т.е. все тестовые примеры с правильными ответами).
  - Целевая функция c лежит в H.
  - Нет априорных причин верить, что одна из гипотез более вероятна, чем другая.

Зачем нужен MAP для анализа алгоритмов

- Казалось бы, мы ничего нового не узнали: алгоритм выдаёт MAP, ну и что?
- Важно другое — важны *предположения*, в которых мы смогли это доказать.
- Иначе говоря, мы поняли, что алгоритм обучения концептам Find-S работает оптимальным образом, если гипотезы априори равновероятны, и среди тестовых примеров нет шума. То же верно для ID3, например. А если гипотезы неравновероятны, можно сделать лучше.

Зачем нужен MAP для анализа алгоритмов

- Казалось бы, мы ничего нового не узнали: алгоритм выдаёт MAP, ну и что?
- Важно другое — важны *предположения*, в которых мы смогли это доказать.
- Байесовский метод позволил установить *границы применимости* алгоритмов. Теперь мы знаем, когда их можно применять смело, а когда можно искать более хорошие алгоритмы. Это *очень* важно для AI.

Нейронные сети

- В нейронных сетях мы минимизировали среднеквадратичную ошибку. Почему?
- Как мы сейчас увидим, именно её нужно минимизировать для того, чтобы получить MAP — в определённых предположениях, разумеется.

Предположения

- Мы хотим выучить функцию $f : \mathcal{X} \longrightarrow \mathbb{R}$.
- Есть набор тестовых примеров

$$D = \{\langle x_1, t_1 \rangle, \ldots, \langle x_m, t_m \rangle\}.$$

- Мы предполагаем, что $t_i = f(x_i) + e_i$, где e_i — *равномерно распределённый* шум с нулевым средним. Это и есть основное предположение.
- Кроме того, предполагаем независимость тестовых примеров.

Вывод

- Мы ищем $h_{MAP} = \text{argmax}_H p(D|h) = \text{argmax}_H \prod_{i=1}^{m} p(t_i|h)$.
- $p(t_i|h)$ — нормальное распределение с вариацией σ и с центром в $\mu = f(x_i) = h(x_i)$ (т.к. при условии h).

Вывод

- Мы ищем $h_{MAP} = \text{argmax}_H p(D|h) = \text{argmax}_H \prod_{i=1}^{m} p(t_i|h)$.
-

$$h_{MAP} = \text{argmax}_H \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - h(x_i))^2} =$$

$$= \text{argmax}_H \sum_{i=1}^{m} \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(d_i - h(x_i))^2 \right) =$$

$$= \text{argmin}_H \sum_{i=1}^{m} (d_i - h(x_i))^2.$$

Вывод

- Мы ищем $h_{MAP} = \text{argmax}_H p(D|h) = \text{argmax}_H \prod_{i=1}^{m} p(t_i|h)$.
- Вот и доказали, что минимизация среднеквадратичной ошибки ведёт к MAP. Это фактически оправдывает применение нейронных сетей.

Бритва Оккама

- Обычно пишут так: «Entia non sunt multiplicanda praeter necessitatem» («Не следует умножать сущности без необходимости»).
- Сам Оккам так не писал, самое близкое — «Numquam ponenda est pluralitas sine necessitate» («Не следует утверждать многое без необходимости»)
- Выдвигалась и Джоном Дунсом Скотом, и Фомой Аквинским, и ещё Аристотелем; Оккам просто активно применял её.
- Базовый философский принцип — неужели его можно доказать математически?

MAP и бритва Оккама

- $$h_{MAP} = \text{argmax}_{h \in H} p(D|h)p(h) =$$
$$= \text{argmax}_{h \in H} \log_2 p(D|h) + \log_2 p(h) =$$
$$= \text{argmin}_{h \in H} - \log_2 p(D|h) - \log_2 p(h).$$

MAP и бритва Оккама

- $h_{MAP} = \operatorname{argmin}_{h \in H} - \log_2 p(D|h) - \log_2 p(h)$.
- Но $-\log_2 p(D|h)$ — это длина описания D при условии использования гипотезы h в оптимальном кодировании (по Шеннону), а $-\log_2 p(h)$ — длина описания самой гипотезы h.

MAP и бритва Оккама

- $h_{MAP} = \operatorname{argmin}_{h \in H} - \log_2 p(D|h) - \log_2 p(h)$.
- Иначе говоря, поиск MAP рекомендует не умножать сущности — использовать кратчайшую из возможных записей описываемой ситуации! Это ещё называется MDL — Minimum Description Length principle.

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`