

Построение метрик и отбор признаков

К. В. Воронцов, А. В. Зухба
vokov@forecsys.ru
a_l@mail.ru

февраль 2014

Содержание

1 Задача построения метрики

- Постановка задачи
- Методы решения
- Качество метрики
- Проклятие размерности

2 Задача отбора признаков

- Жадные алгоритмы и стратегии перебора
- Стохастический поиск

Задача построения метрики

X — объекты, Y — ответы (идентификаторы классов);
 $X^\ell = (x_i, y_i)_{i=1}^\ell$ — обучающая выборка;
 $f_j: X \rightarrow D_j$, $j = 1, \dots, n$ — признаки объектов (features).
Матрица «объекты–признаки» (features data)

$$F = \|f_j(x_i)\|_{\ell \times n} = \begin{pmatrix} f_1(x_1) & \dots & f_n(x_1) \\ \dots & \dots & \dots \\ f_1(x_\ell) & \dots & f_n(x_\ell) \end{pmatrix}$$

Задача построения метрики:

- Как построить функцию расстояния $\rho: X \times X \rightarrow [0, \infty)$?
- Как оценить качество функции расстояния?

Гипотеза компактности:

Схожие объекты, как правило, лежат в одном классе.

Стандартные метрики

Взвешенное расстояние Минковского

$$\rho_p(x, z) = \left(\sum_{i=1}^n w_i |f_i(x) - f_i(z)|^p \right)^{\frac{1}{p}}; \quad w_i \geq 0.$$

Евклидова метрика ($p = 2$)

$$\rho_2(x, z) = \left(\sum_{i=1}^n |f_i(x) - f_i(z)|^2 \right)^{\frac{1}{2}}.$$

Манхэттенское расстояние ($p = 1$) Минимальная длина пути из x в z при условии, что можно двигаться только параллельно осям координат.

Стандартные метрики

Метрика Чебышева ($p = \infty$) выбирает наибольшее из расстояний между векторами по каждой координате:

$$\rho_{\infty}(x, z) = \max_{i=1, \dots, n} |f_i(x) - f_i(z)|.$$

«Считающее» расстояние ($p = 0$) равно числу координат, по которым векторы x и z различаются:

$$\rho_0(x, z) = \sum_{i=1}^n [f_i(x) \neq f_i(z)].$$

Стандартные метрики

Косинусная мера

$$\rho_{\cos}(x, z) = \arccos\left(\frac{\langle x, z \rangle}{\|x\| \|z\|}\right) = \arccos\left(\frac{\sum_{i=1}^n f_i(x) f_i(z)}{(\sum_{i=1}^n f_i^2(x))^{0,5} (\sum_{i=1}^n f_i^2(z))^{0,5}}\right)$$

Расстояние Джаккарда используется, если объектами являются множества.

$$\rho_J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}.$$

Редакторское расстояние равно минимальному числу вставок и удалений символов, с помощью которых можно преобразовать первую строку ко второй.

Стандартные метрики

Расстояние Махалобиса

$$\rho_m(x, z) = \sqrt{(x - z)^T S^{-1} (x - z)}.$$

$$\hat{S} = \frac{1}{n-1} X^T X$$

Теорема

Выборочная ковариационная матрица является неотрицательно определенной.

LOO и профиль компактности

Контроль по отдельным объектам (leave-one-out CV): $k = 1$,

$$\text{LOO}(\mu, X^L) = \frac{1}{L} \sum_{i=1}^L Q(\mu(X^L \setminus \{x_i\}), \{x_i\}).$$

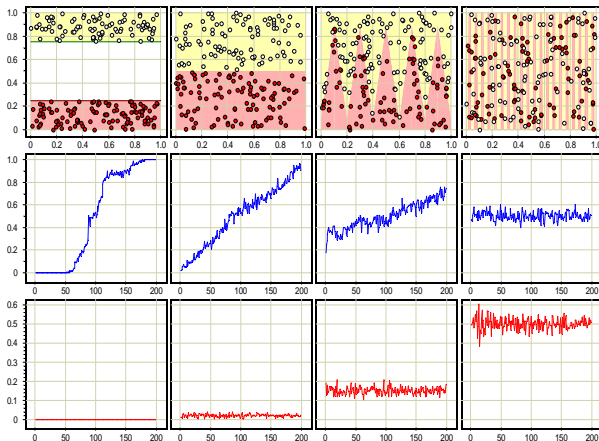
Профиль компактности выборки X^L — это функция доли объектов x_i , у которых m -й сосед $x_i^{(m)}$ лежит в другом классе:

$$K(m, X^L) = \frac{1}{L} \sum_{i=1}^L [y_i \neq y_i^{(m)}]; \quad m = 1, \dots, L-1,$$

где $x_i^{(m)}$ — m -й сосед объекта x_i среди X^L ;

$y_i^{(m)}$ — ответ на m -м соседе объекта x_i .

Профили компактности



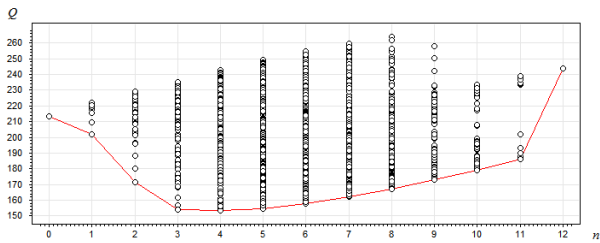
средний ряд: профили компактности,
нижний ряд: зависимость CCV от длины контроля k .

Проклятие размерности

Проклятие размерности — проблема, связанная с экспоненциальным возрастанием количества данных из-за увеличения размерности пространства.

- Трудоемкость вычислений.
- Необходимость хранения огромного количества данных.
- Увеличение доли шумов.
- Расстояния во всех парах объектов стремятся к одному и тому же значению, а значит, становятся неинформативными.

Алгоритм полного перебора (Full Search)

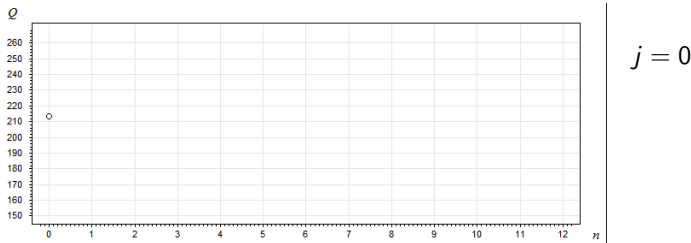


Вход: множество \mathcal{F} , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, где j — сложность наборов:
- 3: найти лучший набор сложности j :

$$\mathcal{G}_j := \arg \min_{\mathcal{G}: |\mathcal{G}|=j} Q(\mathcal{G});$$
- 4: **если** $Q(\mathcal{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathcal{G}_j)$;
- 5: **если** $j - j^* \geq d$ **то вернуть** \mathcal{G}_{j^*} ;

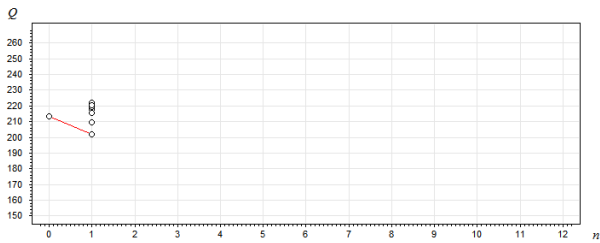
Алгоритм полного перебора (Full Search)



Вход: множество \mathcal{F} , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, где j — сложность наборов:
- 3: найти лучший набор сложности j :
 $\mathcal{G}_j := \arg \min_{\mathcal{G}: |\mathcal{G}|=j} Q(\mathcal{G})$;
- 4: **если** $Q(\mathcal{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathcal{G}_j)$;
- 5: **если** $j - j^* \geq d$ **то вернуть** \mathcal{G}_{j^*} ;

Алгоритм полного перебора (Full Search)

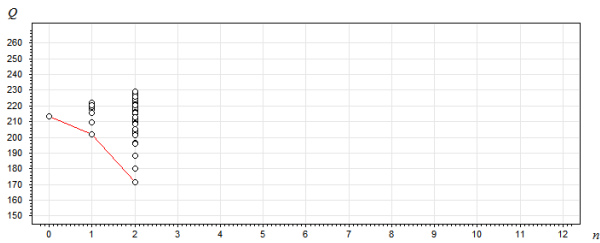


$$j = 1$$
$$j^* = 1$$

Вход: множество \mathcal{F} , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, где j — сложность наборов:
- 3: найти лучший набор сложности j :
 $\mathcal{G}_j := \arg \min_{\mathcal{G}: |\mathcal{G}|=j} Q(\mathcal{G})$;
- 4: **если** $Q(\mathcal{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathcal{G}_j)$;
- 5: **если** $j - j^* \geq d$ **то вернуть** \mathcal{G}_{j^*} ;

Алгоритм полного перебора (Full Search)



$$j = 2$$

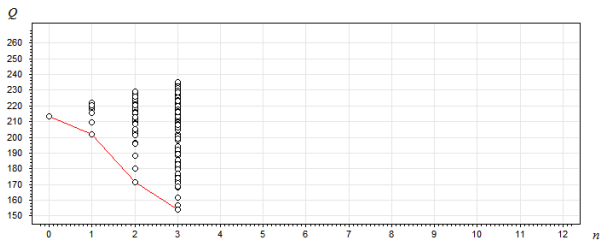
$$j^* = 2$$

Вход: множество \mathcal{F} , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, где j — сложность наборов:
- 3: найти лучший набор сложности j :

$$\mathcal{G}_j := \arg \min_{\mathcal{G}: |\mathcal{G}|=j} Q(\mathcal{G});$$
- 4: **если** $Q(\mathcal{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathcal{G}_j)$;
- 5: **если** $j - j^* \geq d$ **то вернуть** \mathcal{G}_{j^*} ;

Алгоритм полного перебора (Full Search)



$$j = 3$$

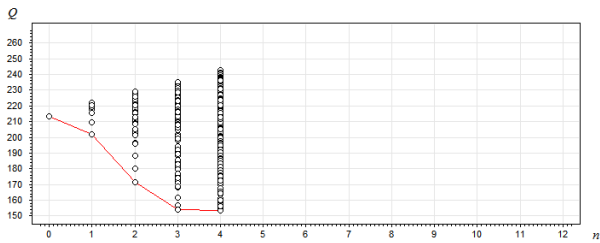
$$j^* = 3$$

Вход: множество \mathcal{F} , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, где j — сложность наборов:
- 3: найти лучший набор сложности j :

$$\mathcal{G}_j := \arg \min_{\mathcal{G}: |\mathcal{G}|=j} Q(\mathcal{G});$$
- 4: **если** $Q(\mathcal{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathcal{G}_j)$;
- 5: **если** $j - j^* \geq d$ **то вернуть** \mathcal{G}_{j^*} ;

Алгоритм полного перебора (Full Search)



$$j = 4$$

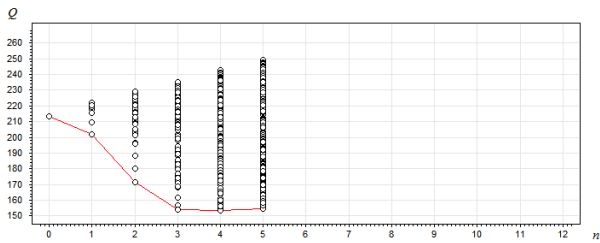
$$j^* = 4$$

Вход: множество \mathcal{F} , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, где j — сложность наборов:
- 3: найти лучший набор сложности j :

$$\mathcal{G}_j := \arg \min_{\mathcal{G}: |\mathcal{G}|=j} Q(\mathcal{G});$$
- 4: **если** $Q(\mathcal{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathcal{G}_j)$;
- 5: **если** $j - j^* \geq d$ **то вернуть** \mathcal{G}_{j^*} ;

Алгоритм полного перебора (Full Search)



$$j = 5$$

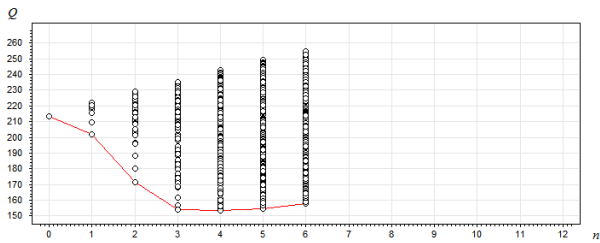
$$j^* = 4$$

Вход: множество \mathcal{F} , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, где j — сложность наборов:
- 3: найти лучший набор сложности j :

$$\mathcal{G}_j := \arg \min_{\mathcal{G}: |\mathcal{G}|=j} Q(\mathcal{G});$$
- 4: **если** $Q(\mathcal{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathcal{G}_j)$;
- 5: **если** $j - j^* \geq d$ **то вернуть** \mathcal{G}_{j^*} ;

Алгоритм полного перебора (Full Search)



$$j = 6$$

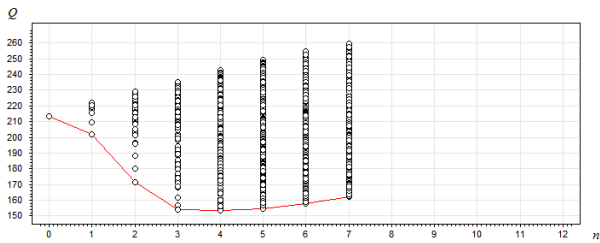
$$j^* = 4$$

Вход: множество \mathcal{F} , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, где j — сложность наборов:
- 3: найти лучший набор сложности j :

$$\mathcal{G}_j := \arg \min_{\mathcal{G}: |\mathcal{G}|=j} Q(\mathcal{G});$$
- 4: **если** $Q(\mathcal{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathcal{G}_j)$;
- 5: **если** $j - j^* \geq d$ **то вернуть** \mathcal{G}_{j^*} ;

Алгоритм полного перебора (Full Search)



$$j = 7$$

$$j^* = 4$$

Вход: множество \mathcal{F} , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, где j — сложность наборов:
- 3: найти лучший набор сложности j :

$$\mathcal{G}_j := \arg \min_{\mathcal{G}: |\mathcal{G}|=j} Q(\mathcal{G});$$
- 4: **если** $Q(\mathcal{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathcal{G}_j)$;
- 5: **если** $j - j^* \geq d$ **то вернуть** \mathcal{G}_{j^*} ;

Алгоритм полного перебора (Full Search)

Преимущества:

- простота реализации;
- гарантированный результат;
- полный перебор эффективен, когда
 - информативных признаков не много, $j^* \lesssim 5$;
 - всего признаков не много, $n \lesssim 20..100$.

Недостатки:

- в остальных случаях ооооооочень долго — $O(2^n)$;
- чем больше перебирается вариантов, тем больше переобучение (особенно, если лучшие из вариантов существенно различны и одинаково плохи).

Способы устранения:

- эвристические методы сокращённого перебора.

Алгоритм жадного добавления (Add)

Вход: множество \mathcal{F} , критерий Q , параметр d ;

- 1: $\mathcal{G}_0 := \emptyset$; $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, где j — сложность наборов:
- 3: найти признак, наиболее выгодный для добавления:
$$f^* := \arg \min_{f \in \mathcal{F} \setminus \mathcal{G}_{j-1}} Q(\mathcal{G}_{j-1} \cup \{f\});$$
- 4: добавить этот признак в набор:
$$\mathcal{G}_j := \mathcal{G}_{j-1} \cup \{f^*\};$$
- 5: **если** $Q(\mathcal{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathcal{G}_j)$;
- 6: **если** $j - j^* \geq d$ **то вернуть** \mathcal{G}_{j^*} ;

Алгоритм жадного добавления (Add)

Преимущества:

- работает быстро — $O(n^2)$, точнее $O(n(j^* + d))$;
- возможны быстрые инкрементные алгоритмы, пример — *шаговая регрессия* (step-wise regression).

Недостатки:

- Add склонен включать в набор лишние признаки.

Способы устранения:

- Add-Del — чередование добавлений и удалений (см. далее);
- поиск в ширину (см. ещё далее).

Алгоритм поочерёдного добавления и удаления (Add-Del)

1: $\mathcal{G}_0 := \emptyset$; $Q^* := Q(\emptyset)$; $t := 0$; — инициализация;

2: **повторять**

3: **пока** $|\mathcal{G}_t| < n$ добавлять признаки (Add):

4: $t := t + 1$; — началась следующая итерация;

5: $f^* := \arg \min_{f \in \mathcal{F} \setminus \mathcal{G}_{t-1}} Q(\mathcal{G}_{t-1} \cup \{f\})$; $\mathcal{G}_t := \mathcal{G}_{t-1} \cup \{f^*\}$;

6: **если** $Q(\mathcal{G}_t) < Q^*$ **то** $t^* := t$; $Q^* := Q(\mathcal{G}_t)$;

7: **если** $t - t^* \geq d$ **то прервать цикл**;

8: **пока** $|\mathcal{G}_t| > 0$ удалять признаки (Del):

9: $t := t + 1$; — началась следующая итерация;

10: $f^* := \arg \min_{f \in \mathcal{G}_{t-1}} Q(\mathcal{G}_{t-1} \setminus \{f\})$; $\mathcal{G}_t := \mathcal{G}_{t-1} \setminus \{f^*\}$;

11: **если** $Q(\mathcal{G}_t) < Q^*$ **то** $t^* := t$; $Q^* := Q(\mathcal{G}_t)$;

12: **если** $t - t^* \geq d$ **то прервать цикл**;

13: **пока** значения критерия $Q(\mathcal{G}_{t^*})$ уменьшаются;

14: **вернуть** \mathcal{G}_{t^*} ;

Алгоритм поочерёдного добавления и удаления (Add-Del)

Преимущества:

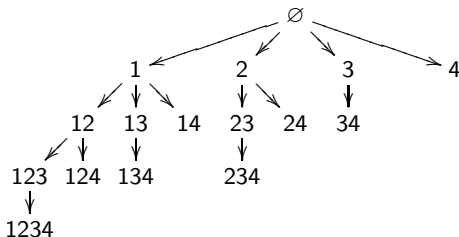
- как правило, лучше, чем Add и Del по отдельности;
- возможны быстрые инкрементные алгоритмы,

Недостатки:

- работает дольше, оптимальность не гарантирует.

Поиск в глубину (DFS, метод ветвей и границ)

Пример: дерево наборов признаков, $n = 4$



Основные идеи:

- нумерация признаков по возрастанию номеров — чтобы избежать повторов при переборе подмножеств;
- если набор \mathcal{S} бесперспективен, то больше не пытаться его наращивать.

Поиск в глубину (DFS, метод ветвей и границ)

Обозначим Q_j^* — значение критерия на самом лучшем наборе мощности j из всех до сих пор просмотренных.

Оценка бесперспективности набора признаков \mathcal{G} :
набор \mathcal{G} не наращивается, если

$$\exists j: \quad Q(\mathcal{G}) \geq \kappa Q_j^* \quad \text{и} \quad |\mathcal{G}| \geq j + d,$$

$d \geq 0$ — целочисленный параметр,
 $\kappa \geq 1$ — вещественный параметр.

Чем меньше d и κ , тем сильнее сокращается перебор.

Поиск в глубину (DFS, метод ветвей и границ)

Вход: множество \mathcal{F} , критерий Q , параметры d и κ ;

- 1: **ПРОЦЕДУРА** Нарастить (\mathcal{G}) ;
 - 2: **если** найдётся $j \leq |\mathcal{G}| - d$ такое, что $Q(\mathcal{G}) \geq \kappa Q_j^*$, **то**
 - 3: **выход**;
 - 4: $Q_{|\mathcal{G}|}^* := \min\{Q_{|\mathcal{G}|}^*, Q(\mathcal{G})\}$;
 - 5: **для всех** $f_s \in \mathcal{F}$ таких, что $s > \max\{t \mid f_t \in \mathcal{G}\}$
 Нарастить $(\mathcal{G} \cup \{f_s\})$;
-

- 6: Инициализация массива лучших значений критерия:
 $Q_j^* := Q(\emptyset)$ для всех $j = 1, \dots, n$;
- 7: Упорядочить признаки по убыванию информативности;
- 8: Нарастить (\emptyset) ;
- 9: **вернуть** \mathcal{G} , для которого $Q(\mathcal{G}) = \min_{j=1, \dots, n} Q_j^*$;

Поиск в ширину (BFS)

Он же *многорядный итерационный алгоритм МГУА*
(МГУА — метод группового учёта аргументов).

Философский принцип *неокончателных решений* Габора:
принимая решения, следует оставлять максимальную свободу
выбора для принятия последующих решений.

Усовершенствуем алгоритм Add:
на каждой j -й итерации будем строить не один набор,
а множество из B_j наборов, называемое j -м *рядом*:

$$R_j = \{\mathcal{G}_j^1, \dots, \mathcal{G}_j^{B_j}\}, \quad \mathcal{G}_j^b \subseteq \mathcal{F}, \quad |\mathcal{G}_j^b| = j, \quad b = 1, \dots, B_j.$$

где $B_j \leq B$ — параметр *ширины поиска*.

Поиск в ширину (BFS)

Вход: множество \mathcal{F} , критерий Q , параметры d, B ;

- 1: первый ряд состоит из всех наборов длины 1:
 $R_1 := \{\{f_1\}, \dots, \{f_n\}\}; \quad Q^* = Q(\emptyset);$
- 2: **для всех** $j = 1, \dots, n$, где j — сложность наборов:
- 3: отсортировать ряд $R_j = \{\mathcal{G}_j^1, \dots, \mathcal{G}_j^{B_j}\}$
по возрастанию критерия: $Q(\mathcal{G}_j^1) \leq \dots \leq Q(\mathcal{G}_j^{B_j});$
- 4: **если** $B_j > B$ **то**
- 5: $R_j := \{\mathcal{G}_j^1, \dots, \mathcal{G}_j^B\};$ — B лучших наборов ряда;
- 6: **если** $Q(\mathcal{G}_j^1) < Q^*$ **то** $j^* := j; \quad Q^* := Q(\mathcal{G}_j^1);$
- 7: **если** $j - j^* \geq d$ **то вернуть** $\mathcal{G}_{j^*}^1;$
- 8: породить следующий ряд:
 $R_{j+1} := \{\mathcal{G} \cup \{f\} \mid \mathcal{G} \in R_j, f \in \mathcal{F} \setminus \mathcal{G}\};$

Поиск в ширину (BFS)

- **Трудоёмкость:**
 $O(Bn^2)$, точнее $O(Bn(j^* + d))$.
- **Проблема дубликатов:**
после сортировки (шаг 3) проверить на совпадение только соседние наборы с равными значениями внутреннего и внешнего критерия.
- **Адаптивный отбор признаков:**
на шаге 8 добавлять к j -му ряду только признаки f с наибольшей информативностью $l_j(f)$:

$$l_j(f) = \sum_{b=1}^{B_j} [f \in \mathcal{G}_j^b].$$

Генетический алгоритм поиска (идея и терминология)

$\mathcal{G} \subseteq \mathcal{F}$ — индивид (в МГУА «модель»);

$R_t := \{\mathcal{G}_t^1, \dots, \mathcal{G}_t^{B_t}\}$ — поколение (в МГУА — «ряд»);

$\beta = (\beta_j)_{j=1}^n$, $\beta_j = [f_j \in \mathcal{G}]$ — хромосома, кодирующая \mathcal{G} ;

Бинарная операция скрещивания $\beta = \beta' \times \beta''$:

$$\beta_j = \begin{cases} \beta'_j, & \text{с вероятностью } 1/2; \\ \beta''_j, & \text{с вероятностью } 1/2; \end{cases}$$

Унарная операция мутации $\beta = \sim \beta'$

$$\beta_j = \begin{cases} 1 - \beta'_j, & \text{с вероятностью } p_m; \\ \beta'_j, & \text{с вероятностью } 1 - p_m; \end{cases}$$

где параметр p_m — вероятность мутации.

Генетический (эволюционный) алгоритм

Вход: множество \mathcal{F} , критерий Q , параметры: d, p_m ,
 B — размер популяции, T — число поколений;

-
- 1: инициализировать случайную популяцию из B наборов:
 $B_1 := B$; $R_1 := \{\mathcal{G}_1^1, \dots, \mathcal{G}_1^{B_1}\}$; $Q^* := Q(\emptyset)$;
 - 2: **для всех** $t = 1, \dots, T$, где t — номер поколения:
 - 3: ранжирование индивидов: $Q(\mathcal{G}_t^1) \leq \dots \leq Q(\mathcal{G}_t^{B_t})$;
 - 4: **если** $B_t > B$ **то**
 - 5: селекция: $R_t := \{\mathcal{G}_t^1, \dots, \mathcal{G}_t^B\}$;
 - 6: **если** $Q(\mathcal{G}_t^1) < Q^*$ **то** $t^* := t$; $Q^* := Q(\mathcal{G}_t^1)$;
 - 7: **если** $t - t^* \geq d$ **то вернуть** $\mathcal{G}_{t^*}^1$;
 - 8: породить $t+1$ -е поколение путём скрещиваний и мутаций:
 $R_{t+1} := \{\sim(\mathcal{G}' \times \mathcal{G}'') \mid \mathcal{G}', \mathcal{G}'' \in R_t\} \cup R_t$;

Эвристики для управления процессом эволюции

- Увеличивать вероятности перехода признаков от более успешного родителя к потомку.
- Накапливать оценки информативности признаков.
Чем более информативен признак, тем выше вероятность его включения в набор во время мутации.
- Применение совокупности критериев качества.
- Скрещивать только лучшие индивиды (элитаризм).
- Переносить лучшие индивиды в следующее поколение.
- В случае стагнации увеличивать вероятность мутаций.
- Параллельно выращивается несколько изолированных популяций (островная модель эволюции).

Генетический (эволюционный) алгоритм

Преимущества:

- it is fun!
- возможность введения различных эвристик;
- решает задачи даже с очень большим числом признаков.

Недостатки:

- относительно медленная сходимость;
- отсутствие теории;
- подбор параметров — непростое искусство;

Случайный поиск — упрощенный генетический алгоритм

Модификация: шаг 8

- породить $t+1$ -е поколение путём многократных *мутаций*:

$$R_{t+1} := \{\sim \mathcal{G}, \dots, \sim \mathcal{G} \mid \mathcal{G} \in R_t\} \cup R_t;$$

Недостатки:

- ничем не лучше ГА;
- очень медленная сходимость.

Способ устранения:

- СПА — случайный поиск с адаптацией.

Основная идея адаптации:

- увеличивать вероятность появления тех признаков, которые часто входят в наилучшие наборы,
- одновременно уменьшать вероятность появления признаков, которые часто входят в наихудшие наборы.

Случайный поиск с адаптацией (СПА)

Вход: множество \mathcal{F} , критерий Q , параметры d, j_0, T, r, h ;

- 1: $p_1 = \dots = p_n := 1/n$; — равные вероятности признаков;
- 2: **для всех** $j = j_0, \dots, n$, где j — сложность наборов:
- 3: **для всех** $t = 1, \dots, T$, где t — номер итерации:
- 4: r случайных наборов признаков из распределения $\{p_1, \dots, p_n\}$:
 $R_{jt} := \{\mathcal{G}_{jt}^1, \dots, \mathcal{G}_{jt}^r\}, \quad |\mathcal{G}_{jt}^1| = \dots = |\mathcal{G}_{jt}^r| = j$;
- 5: $\mathcal{G}_{jt}^{\min} := \arg \min_{\mathcal{G} \in R_{jt}} Q(\mathcal{G})$; — лучший из r наборов;
- 6: $\mathcal{G}_{jt}^{\max} := \arg \max_{\mathcal{G} \in R_{jt}} Q(\mathcal{G})$; — худший из r наборов;
- 7: $H := 0$; наказание для всех $f_s \in \mathcal{G}_{jt}^{\max}$:
 $\Delta p_s := \min\{p_s, h\}; \quad p_s := p_s - \Delta p_s; \quad H := H + \Delta p_s$;
- 8: поощрение для всех $f_s \in \mathcal{G}_{jt}^{\min}$: $p_s := p_s + H/j$;
- 9: $\mathcal{G}_j := \arg \min_{\mathcal{G} \in R_{j1}, \dots, R_{jT}} Q(\mathcal{G})$; — лучший набор сложности j ;
- 10: **если** $Q(\mathcal{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathcal{G}_j)$;
- 11: **если** $j - j^* \geq d$ **то вернуть** \mathcal{G}_{j^*} ;

Случайный поиск с адаптацией (СПА)

Рекомендации по выбору параметров r, T, h :

$T \approx 10..50$ — число итераций;

$r \approx 20..100$ — число наборов, создаваемых на каждой итерации;

$h \approx \frac{1}{rn}$ — скорость адаптации;

Преимущества:

- трудоёмкость порядка $O(Tr(j^* + d))$ операций;
- меньшее число параметров, по сравнению с генетикой;
- довольно быстрая сходимость.

Недостатки:

- при большом числе признаков СПА малоэффективен.

Резюме в конце лекции

- Отбор признаков надо вести по внешним критериям.
- Для отбора признаков могут использоваться любые эвристические методы дискретной оптимизации

$$Q(\mathcal{G}) \rightarrow \min_{\mathcal{G} \subseteq \mathcal{F}} .$$

- Большинство эвристик эксплуатируют две основные идеи:
 - не все признаки информативны;
 - $Q(\mathcal{G})$ изменяется не сильно при небольшом изменении \mathcal{G} .
- МГУА, ГА, СПА очень похожи — на их основе легко создавать различные «симбиотические» алгоритмы.
- Отбор признаков в беспереборных методах происходит благодаря негладкости регуляризатора.