Обобщающая способность
Методы отбора признаков

Воронцов Константин Вячеславович
vokov@forecsys.ru
http://www.MachineLearning.ru/wiki?title=User:Vokov

Этот курс доступен на странице вики-ресурса
http://www.MachineLearning.ru/wiki
«Машинное обучение (курс лекций, К.В.Воронцов)»

Видеолекции: http://shad.yandex.ru/lectures

5 марта 2015

Содержание

Задачи выбора метода обучения

Дано: X — пространство объектов; Y — множество ответов;
$X^\ell = (x_i, y_i)_{i=1}^\ell$ — обучающая выборка, $y_i = y^*(x_i)$;
$A_t = \{a \colon X \to Y\}$ — модели алгоритмов, $t \in T$;
$\mu_t \colon (X \times Y)^\ell \to A_t$ — методы обучения, $t \in T$.

Найти: метод μ_t с наилучшей *обобщающей способностью*.

Частные случаи:

- выбор лучшей модели A_t (model selection);
- выбор метода обучения μ_t для заданной модели A (в частности, оптимизация *гиперпараметров*);
- отбор признаков (features selection):
  $F = \{f_j \colon X \to D_j \colon j = 1, \ldots, n\}$ — множество признаков;
  метод обучения μ_J использует только признаки $J \subseteq F$.

Как оценить качество обучения по прецедентам?

$\mathscr{L}(a, x)$ — функция потерь алгоритма a на объекте x;

$Q(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathscr{L}(a, x_i)$ — *функционал качества* a на X^ℓ.

Внутренний критерий оценивает качество на обучении X^ℓ:

$$Q_\mu(X^\ell) = Q\big(\mu(X^\ell), X^\ell\big).$$

Недостаток: эта оценка смещена, т.к. μ минимизирует её же.

Внешний критерий оценивает качество «вне обучения», например, по отложенной (hold-out) контрольной выборке X^k:

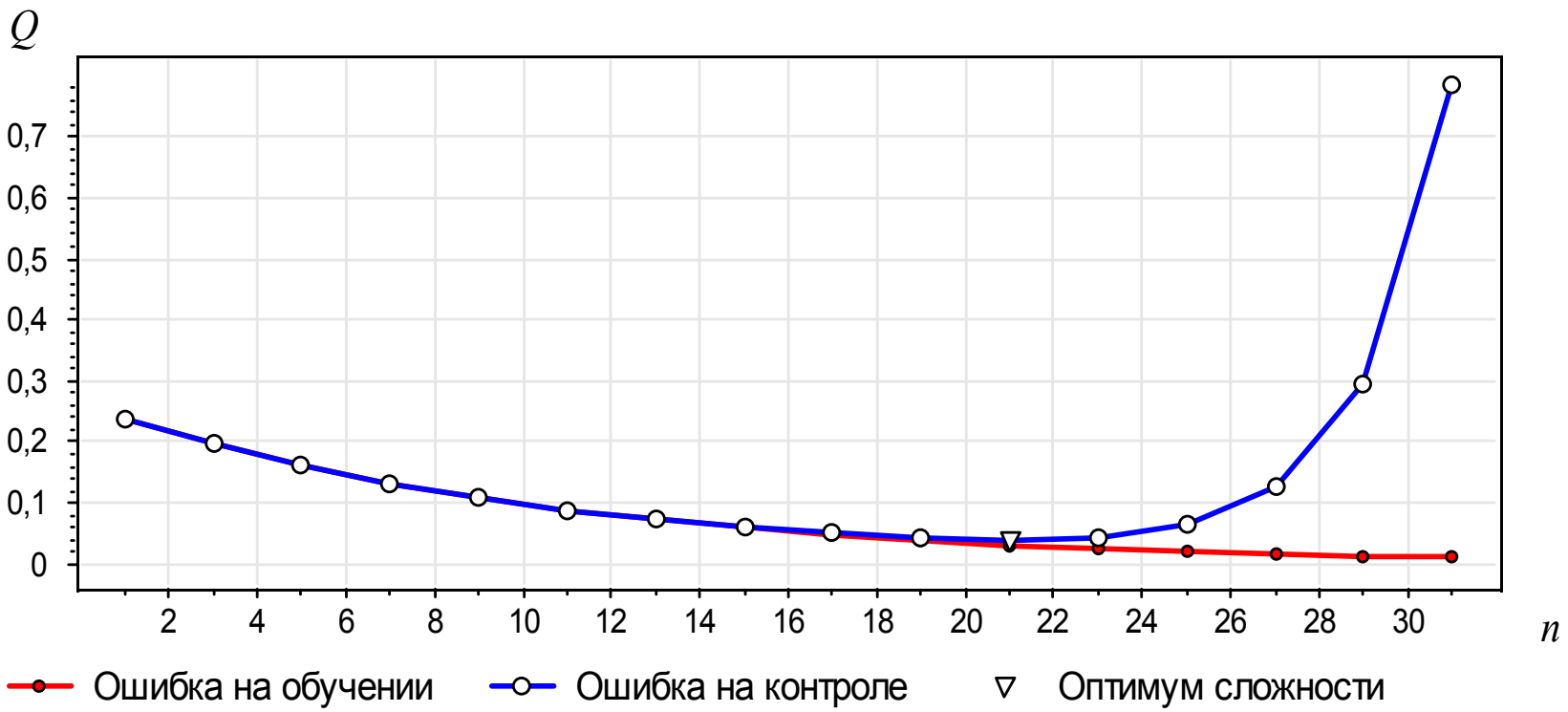
$$Q_\mu(X^\ell, X^k) = Q\big(\mu(X^\ell), X^k\big).$$

Недостаток: эта оценка зависит от разбиения $X^L = X^\ell \sqcup X^k$.

Основное отличие внешних критериев от внутренних

Внутренний критерий монотонно убывает с ростом сложности модели (например, числа признаков).

Внешний критерий имеет характерный минимум, соответствующий оптимальной сложности модели.



Кросс-проверка (cross-validation, CV)

Усреднение оценок hold-out по заданному N — множеству разбиений $X^L = X_n^\ell \sqcup X_n^k$, $n = 1, \ldots, N$:

$$\mathrm{CV}(\mu, X^L) = \frac{1}{|N|} \sum_{n \in N} Q_\mu(X_n^\ell, X_n^k).$$

Частные случаи — разные способы задания N.

1. Случайное множество разбиений.

2. *Полная кросс-проверка* (complete cross-validation, CCV):
N — множество всех $C_{\ell+k}^k$ разбиений.

Недостаток: оценка CCV вычислительно слишком сложна.
Используются либо малые k, либо комбинаторные оценки CCV.

Эмпирические оценки кросс-проверки

3. Скользящий контроль (leave one out CV): $k = 1$,

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

Недостатки LOO: ресурсоёмкость, высокая дисперсия.

4. Кросс-проверка по q блокам (q-fold CV): случайное разбиение $X^L = X_1^{\ell_1} \sqcup \cdots \sqcup X_q^{\ell_q}$ на q блоков (почти) равной длины,

$$\mathrm{CV}_q(\mu, X^L) = \frac{1}{q} \sum_{n=1}^{q} Q_\mu\left(X^L \setminus X_n^{\ell_n}, X_n^{\ell_n}\right).$$

Недостатки q-fold CV:

- оценка существенно зависит от разбиения на блоки;
- каждый объект лишь один раз участвует в контроле.

Эмпирические оценки скользящего контроля

5. Контроль t раз по q блокам ($t \times q$-fold CV)
— стандарт «де факто» для тестирования методов обучения.

Выборка X^L разбивается t раз случайным образом на q блоков

$$X^L = X_{s1}^{\ell_1} \sqcup \cdots \sqcup X_{sq}^{\ell_q}, \quad s = 1, \ldots, t, \quad \ell_1 + \cdots + \ell_q = L;$$

$$\mathrm{CV}_{t \times q}(\mu, X^L) = \frac{1}{t} \sum_{s=1}^{t} \frac{1}{q} \sum_{n=1}^{q} Q_\mu\left(X^L \setminus X_{sn}^{\ell_n}, X_{sn}^{\ell_n}\right).$$

Преимущества $t \times q$-fold CV:

- увеличением t можно улучшать точность оценки (компромисс между точностью и временем вычислений);
- каждый объект участвует в контроле ровно t раз;
- оценивание доверительных интервалов (95% при $t = 40$).

Критерии непротиворечивости моделей

Идея: Если модель верна, то алгоритмы, настроенные по разным частям данных, не должны противоречить друг другу.

1. По одному случайному разбиению $X^\ell \sqcup X^k = X^L$, $\ell = k$:

$$D_1(\mu, X^L) = \frac{1}{L} \sum_{i=1}^{L} \left|\mu(X^\ell)(x_i) - \mu(X^k)(x_i)\right|.$$

2. Аналог $\mathrm{CV}_{t\times 2}$: по t разбиениям $X^L = X_s^\ell \sqcup X_s^k$, $s = 1, \ldots, t$:

$$D_t(\mu, X^L) = \frac{1}{t} \sum_{s=1}^{t} \frac{1}{L} \sum_{i=1}^{L} \left|\mu(X_s^\ell)(x_i) - \mu(X_s^k)(x_i)\right|.$$

Недостатки:

- длина обучения сокращается в 2 раза;
- трудоёмкость возрастает в 2 раза.

Аналитические оценки и их обращение

Основная идея аналитического подхода:

1. Получить верхнюю оценку вероятности переобучения R_ε, справедливую для любой выборки X^L, широкого класса моделей A и методов обучения μ:

$$R_\varepsilon(\mu, X^L) = \mathsf{P}\Big[Q_\mu(X^\ell, X^k) - Q_\mu(X^\ell) \geqslant \varepsilon\Big] \leqslant \eta(\varepsilon, A).$$

2. Тогда для любой X^L, любых A и μ и любого $\eta \in (0,1)$ с вероятностью не менее $(1-\eta)$ справедлива оценка

$$Q_\mu(X^\ell, X^k) \leqslant Q_\mu(X^\ell) + \varepsilon(\eta, A),$$

где $\varepsilon(\eta, A)$ — функция штрафа на A, обратная к $\eta(\varepsilon, A)$, не зависящая от скрытой контрольной выборки X^k.

3. Оптимизировать метод обучения: $Q_\mu(X^\ell) + \varepsilon(\eta, A) \to \min\limits_{\mu \in M}$.

Критерии регуляризации

Регуляризатор — аддитивная добавка к внутреннему критерию, обычно штраф за сложность (complexity penalty) модели A:

$$Q_{\text{рег}}(\mu, X^\ell) = Q_\mu(X^\ell) + \text{штраф}(A),$$

Линейные модели: $A = \{a(x) = \text{sign}\langle w, x\rangle\}$ — классификация,

$A = \{a(x) = \langle w, x\rangle\}$ — регрессия.

L_2-регуляризация (ридж-регрессия, weight decay):

$$\text{штраф}(w) = \tau\|w\|_2^2 = \tau\sum_{j=1}^{n} w_j^2.$$

L_1-регуляризация (LASSO):

$$\text{штраф}(w) = \tau\|w\|_1 = \tau\sum_{j=1}^{n} |w_j|.$$

L_0-регуляризация (AIC, BIC):

$$\text{штраф}(w) = \tau\|w\|_0 = \tau\sum_{j=1}^{n} \big[w_j \neq 0\big].$$

Разновидности L_0-регуляризации

Информационный критерий Акаике (Akaike Information Criterion):

$$\mathrm{AIC}(\mu, x) = Q_\mu(X^\ell) + \frac{2\hat{\sigma}^2}{\ell}|J|,$$

где $\hat{\sigma}^2$ — оценка дисперсии ошибки $\mathsf{D}\big(y_i - a(x_i)\big)$,
J — подмножество используемых признаков.

Байесовский информационный критерий (Bayes Inform. Criterion):

$$\mathrm{BIC}(\mu, X^\ell) = \frac{\ell}{\hat{\sigma}^2}\left(Q_\mu(X^\ell) + \frac{\hat{\sigma}^2 \ln \ell}{\ell}|J|\right).$$

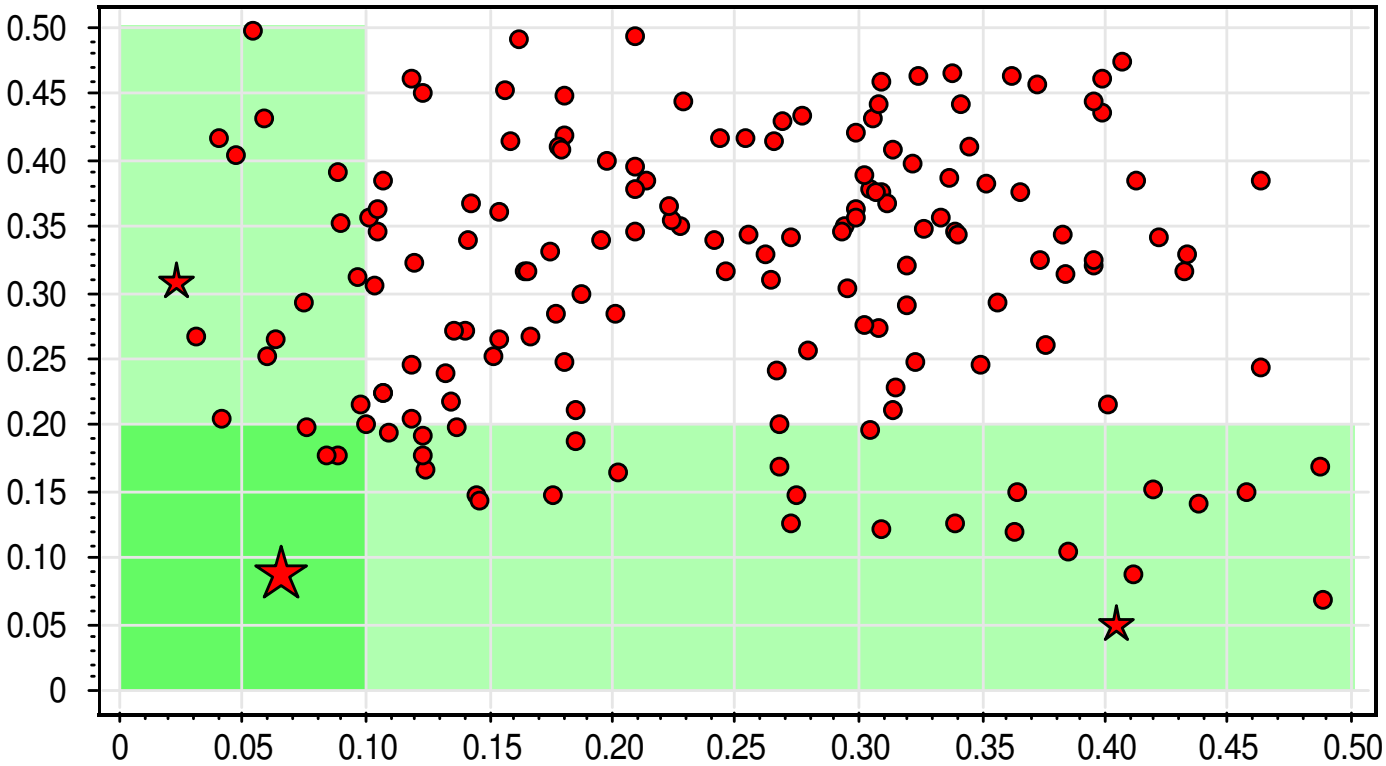
Оценка Вапника-Червоненкиса (VC-bound):

$$\mathrm{VC}(\mu, X^\ell) = Q_\mu(X^\ell) + \sqrt{\frac{h}{\ell}\ln\frac{2e\ell}{h} + \frac{1}{\ell}\ln\frac{9}{4\eta}},$$

h — VC-размерность; для линейных, опять-таки, $h = |J|$;
η — уровень значимости; обычно $\eta = 0.05$.

Выбор модели по совокупности внешних критериев

Модель, немного неоптимальная по обоим критериям, скорее всего, лучше, чем модель, оптимальная по одному критерию, но не оптимальная по другому.



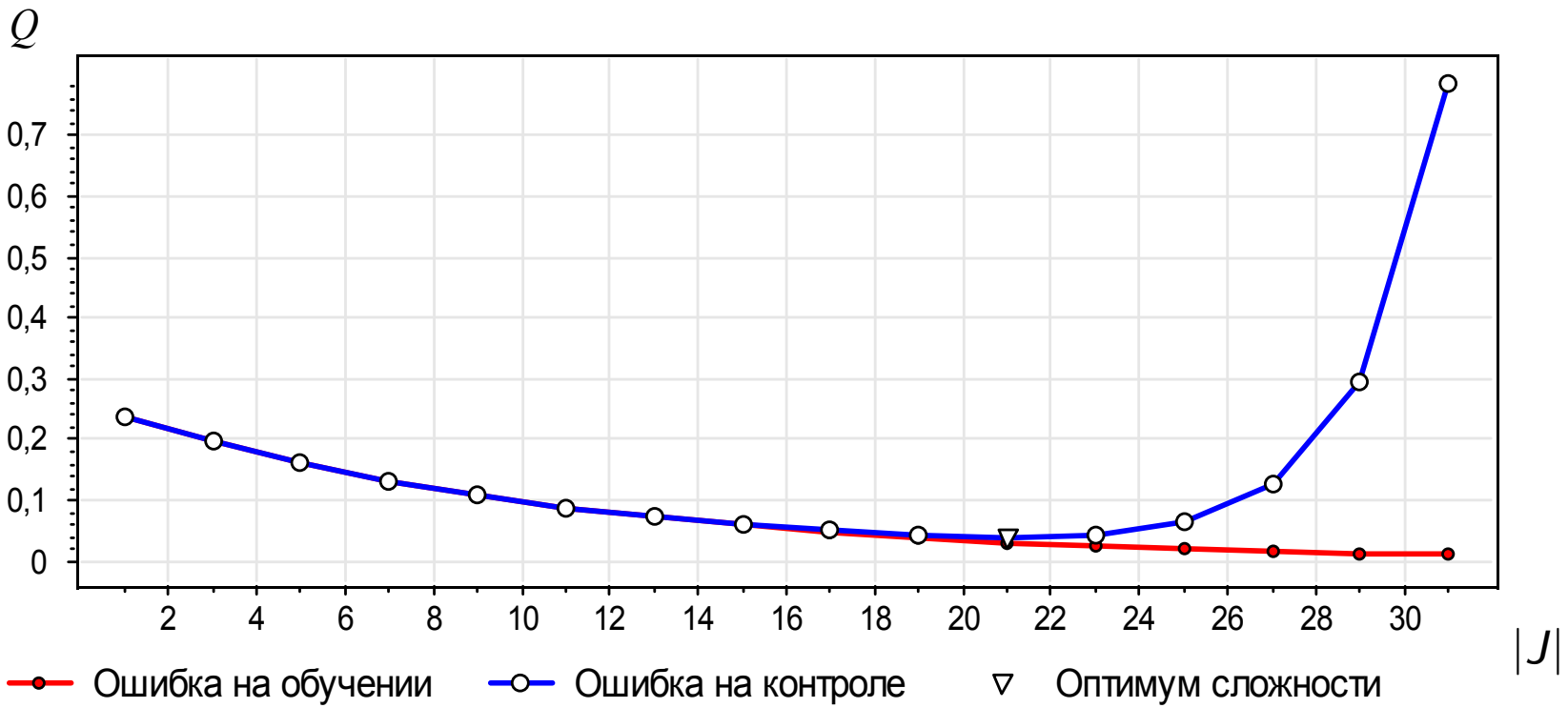
Задача отбора признаков по внешнему критерию

$F = \{f_j : X \to D_j : j = 1, \ldots, n\}$ — множество признаков;

μ_J — метод обучения, использующий только признаки $J \subseteq F$;

$Q(J) = Q(\mu_J, X^\ell)$ — выбранный внешний критерий.

$Q(J) \to \min$ — задача дискретной оптимизации.



Задача отбора признаков в логических закономерностях

Закономерность R — *конъюнкция* пороговых условий:

$$R(x) = \bigwedge_{j \in J} \big[f_j(x) \geqslant a_j \big].$$

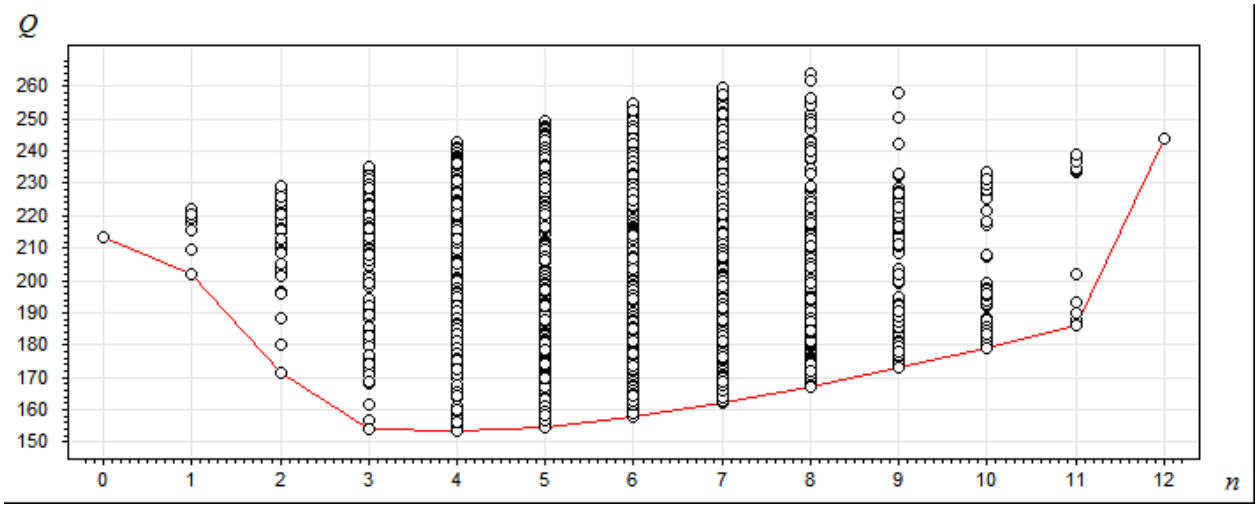
Критерий информативности относительно класса c:

$$I(p,n) \to \max_{J, \{a_j\}}; \quad \begin{cases} p(R) = \#\big\{x_i \colon R(x_i)=1 \text{ и } y_i=c\big\} \to \max \\ n(R) = \#\big\{x_i \colon R(x_i)=1 \text{ и } y_i \neq c\big\} \to \min \end{cases}$$

Аналогично внешним критериям,
информативность $I(p,n)$ имеет оптимум по сложности $|J|$:

- слишком мало признаков $\Rightarrow$ большие n, низкая $I(p,n)$
- оптимально признаков $\Rightarrow$ малые n, большие p, высокая $I(p,n)$
- слишком много признаков $\Rightarrow$ малые $p+n$, низкая $I(p,n)$

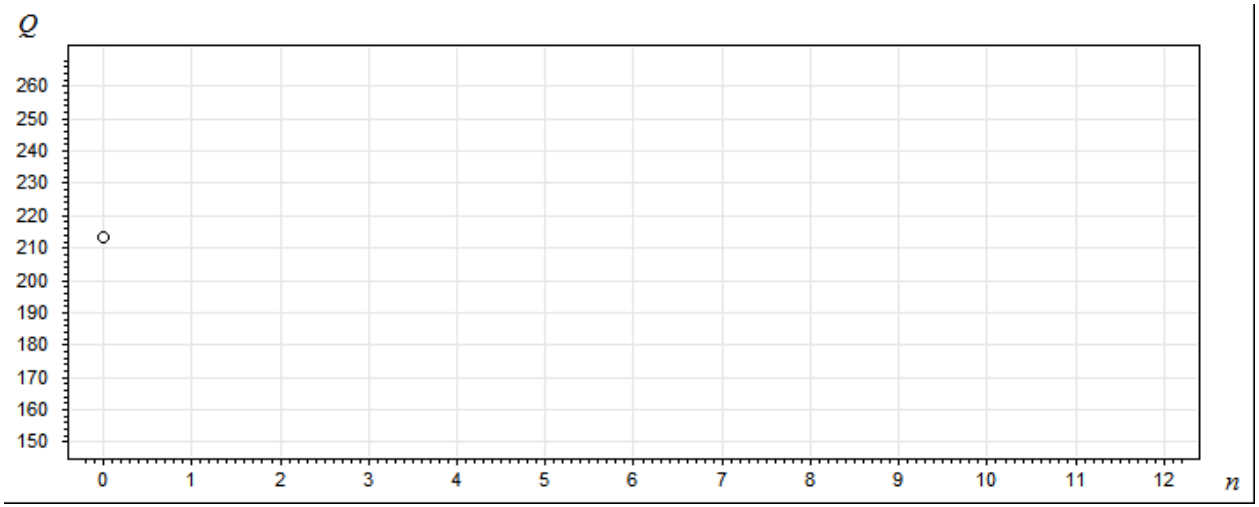
Алгоритм полного перебора (Full Search)



Вход: множество F, критерий Q, параметр d;

1: $Q^* := Q(\varnothing)$; — инициализация;
2: **для всех** $j = 1, \ldots, n$, где j — сложность наборов:
3: найти лучший набор сложности j:
$J_j := \arg\min_{J:\,|J|=j} Q(J)$;
4: **если** $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;
5: **если** $j - j^* \geqslant d$ **то вернуть** J_{j^*};

Алгоритм полного перебора (Full Search)

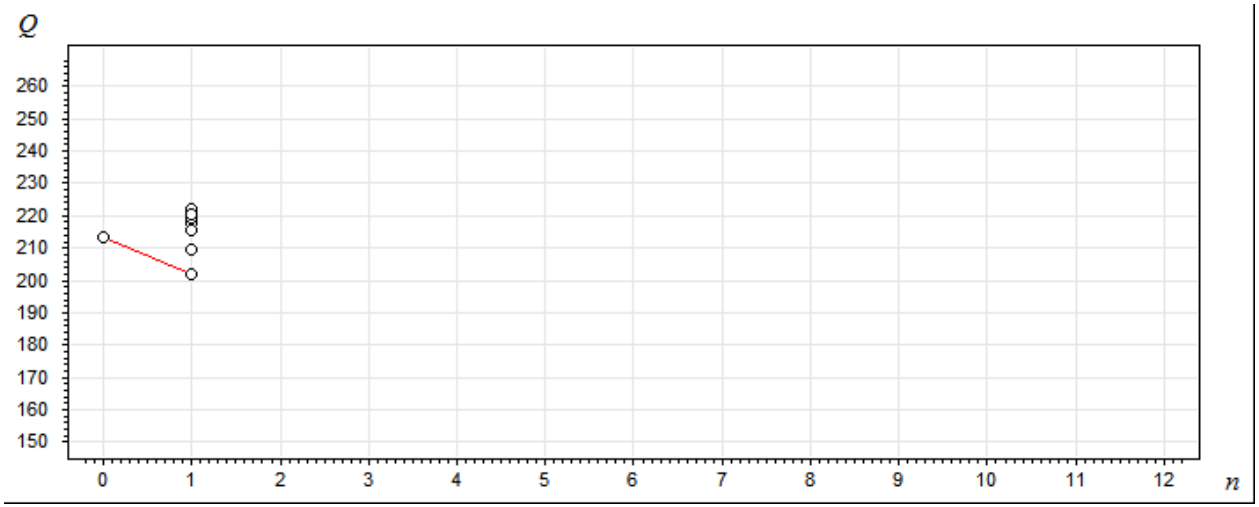


$d = 3$
$j = 0$

Вход: множество F, критерий Q, параметр d;

1: $Q^* := Q(\varnothing)$; — инициализация;
2: **для всех** $j = 1, \ldots, n$, где j — сложность наборов:
3: найти лучший набор сложности j:
$J_j := \arg\min_{J:\,|J|=j} Q(J)$;
4: **если** $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;
5: **если** $j - j^* \geqslant d$ **то вернуть** J_{j^*};

Алгоритм полного перебора (Full Search)

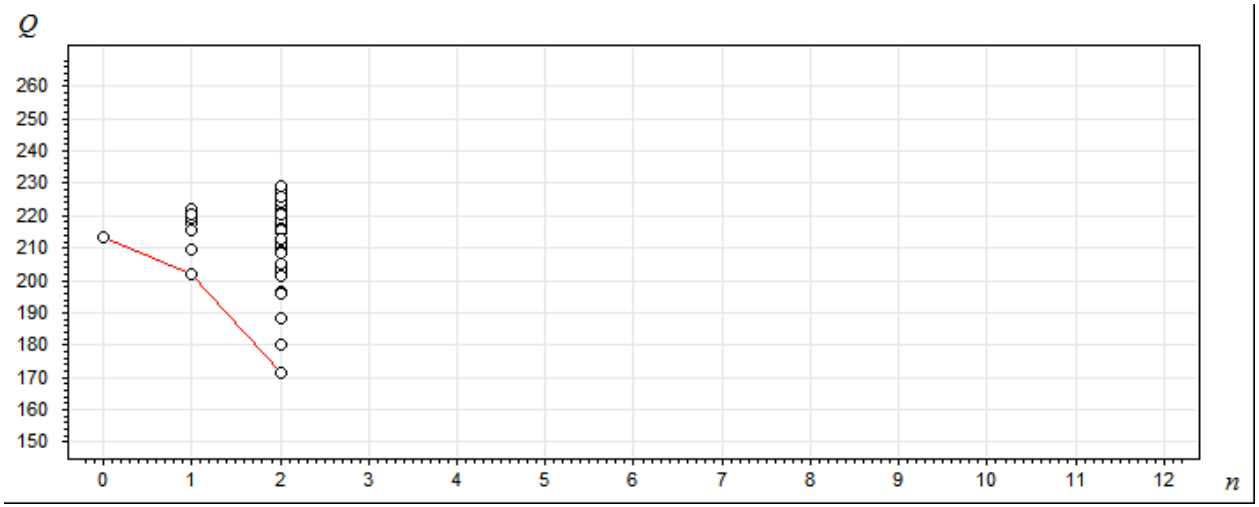


$d = 3$
$j = 1$
$j^* = 1$

Вход: множество F, критерий Q, параметр d;

1: $Q^* := Q(\varnothing)$; — инициализация;
2: **для всех** $j = 1, \ldots, n$, где j — сложность наборов:
3: найти лучший набор сложности j:
$J_j := \arg\min_{J:\,|J|=j} Q(J)$;
4: **если** $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;
5: **если** $j - j^* \geqslant d$ **то вернуть** J_{j^*};

Алгоритм полного перебора (Full Search)

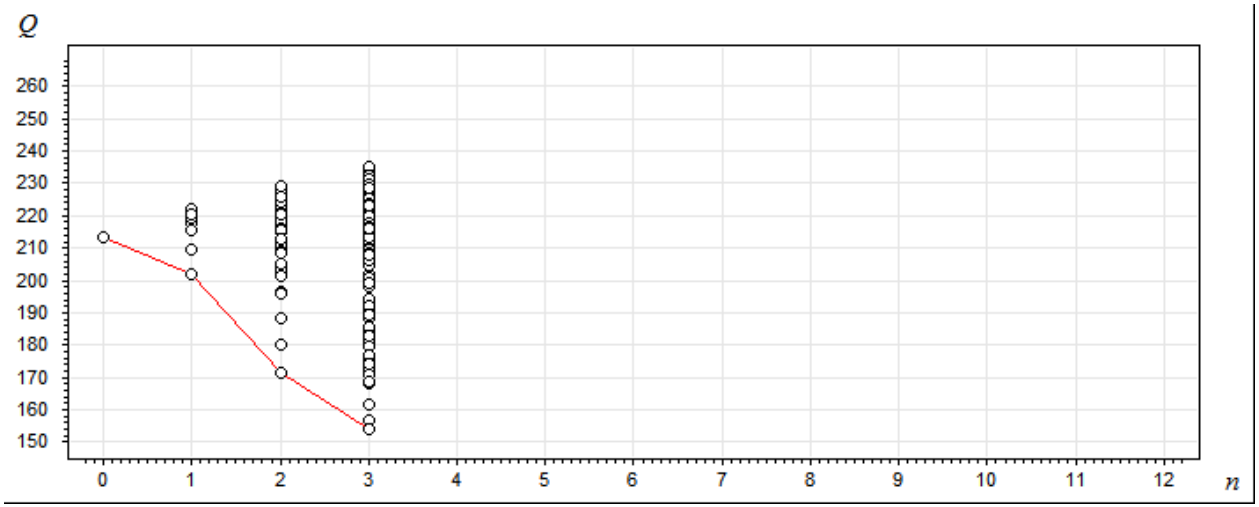


$d = 3$
$j = 2$
$j^* = 2$

Вход: множество F, критерий Q, параметр d;

1: $Q^* := Q(\varnothing)$; — инициализация;
2: **для всех** $j = 1, \ldots, n$, где j — сложность наборов:
3: найти лучший набор сложности j:
$J_j := \arg\min_{J:\,|J|=j} Q(J)$;
4: **если** $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;
5: **если** $j - j^* \geqslant d$ **то вернуть** J_{j^*};

Алгоритм полного перебора (Full Search)

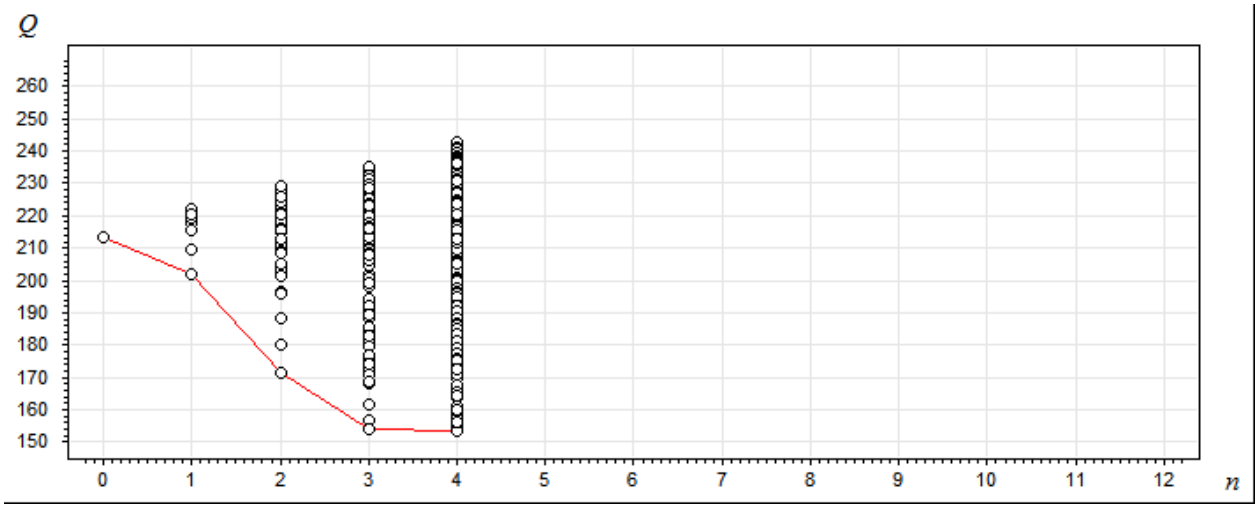


$d = 3$
$j = 3$
$j^* = 3$

Вход: множество F, критерий Q, параметр d;

1: $Q^* := Q(\varnothing)$; — инициализация;
2: **для всех** $j = 1, \ldots, n$, где j — сложность наборов:
3: найти лучший набор сложности j:
$J_j := \arg\min_{J:\,|J|=j} Q(J)$;
4: **если** $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;
5: **если** $j - j^* \geqslant d$ **то вернуть** J_{j^*};

Алгоритм полного перебора (Full Search)

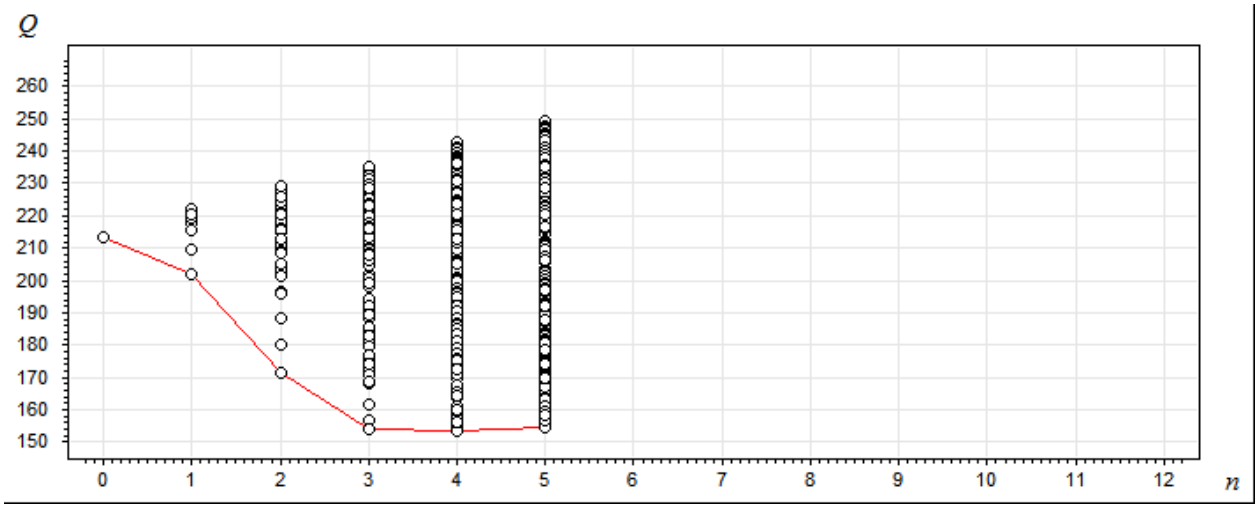


$d = 3$
$j = 4$
$j^* = 4$

Вход: множество F, критерий Q, параметр d;

1: $Q^* := Q(\varnothing)$; — инициализация;
2: **для всех** $j = 1, \ldots, n$, где j — сложность наборов:
3: найти лучший набор сложности j:
$J_j := \arg\min_{J:\,|J|=j} Q(J)$;
4: **если** $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;
5: **если** $j - j^* \geqslant d$ **то вернуть** J_{j^*};

Алгоритм полного перебора (Full Search)

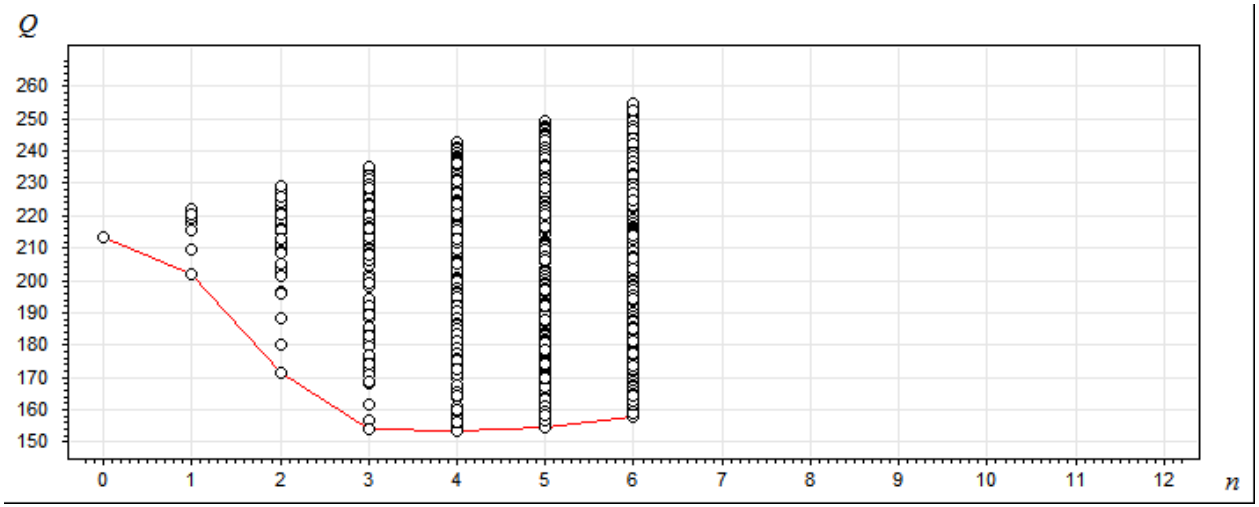


$d = 3$
$j = 5$
$j^* = 4$

Вход: множество F, критерий Q, параметр d;

1: $Q^* := Q(\varnothing)$; — инициализация;
2: **для всех** $j = 1, \ldots, n$, где j — сложность наборов:
3: найти лучший набор сложности j:
$J_j := \arg\min_{J:\,|J|=j} Q(J)$;
4: **если** $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;
5: **если** $j - j^* \geqslant d$ **то вернуть** J_{j^*};

Алгоритм полного перебора (Full Search)

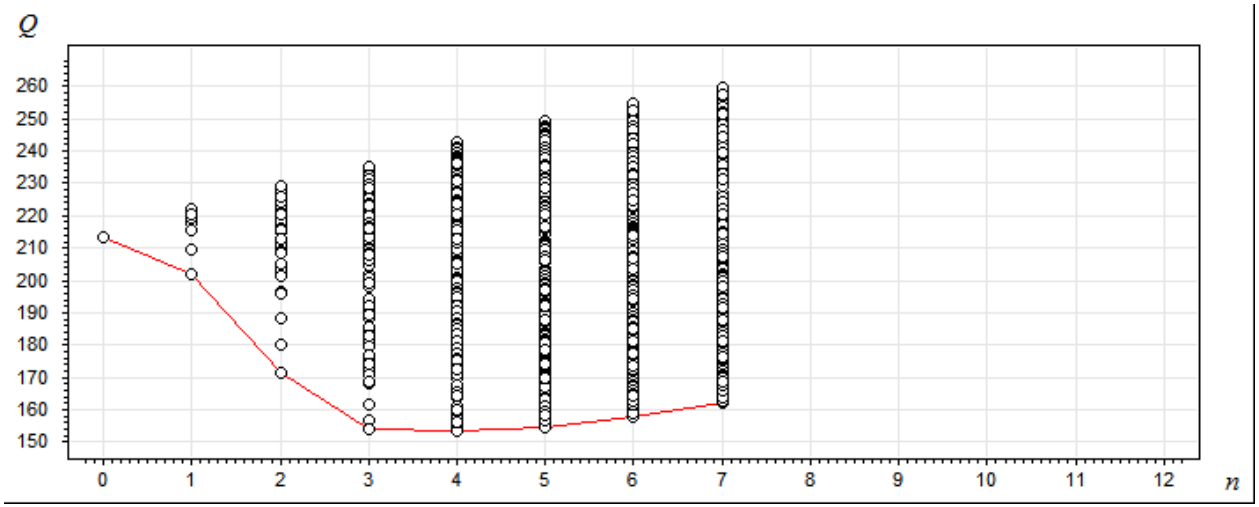


$d = 3$
$j = 6$
$j^* = 4$

Вход: множество F, критерий Q, параметр d;

1: $Q^* := Q(\varnothing)$; — инициализация;
2: **для всех** $j = 1, \ldots, n$, где j — сложность наборов:
3: найти лучший набор сложности j:
$J_j := \arg\min_{J:\,|J|=j} Q(J)$;
4: **если** $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;
5: **если** $j - j^* \geqslant d$ **то вернуть** J_{j^*};

Алгоритм полного перебора (Full Search)



$d = 3$
$j = 7$
$j^* = 4$

Вход: множество F, критерий Q, параметр d;

1: $Q^* := Q(\varnothing)$; — инициализация;
2: **для всех** $j = 1, \ldots, n$, где j — сложность наборов:
3: найти лучший набор сложности j:
$J_j := \arg\min_{J:\,|J|=j} Q(J)$;
4: **если** $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;
5: **если** $j - j^* \geqslant d$ **то вернуть** J_{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 F, критерий Q, параметр d;

1: $J_0 := \varnothing$; $Q^* := Q(\varnothing)$; — инициализация;
2: **для всех** $j = 1, \ldots, n$, где j — сложность наборов:
3: найти признак, наиболее выгодный для добавления:
$f^* := \arg\min_{f \in F \setminus J_{j-1}} Q\left(J_{j-1} \cup \{f\}\right)$;
4: добавить этот признак в набор:
$J_j := J_{j-1} \cup \{f^*\}$;
5: **если** $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;
6: **если** $j - j^* \geqslant d$ **то вернуть** J_{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)

$J_0 := \varnothing$; $Q^* := Q(\varnothing)$; $t := 0$; — инициализация;
повторять

пока $|J_t| < n$ добавлять признаки (Add):
  $t := t + 1$; — началась следующая итерация;
  $f^* := \arg\min\limits_{f \in F \setminus J_{t-1}} Q\big(J_{t-1} \cup \{f\}\big)$; $J_t := J_{t-1} \cup \{f^*\}$;
  если $Q(J_t) < Q^*$ **то** $t^* := t$; $Q^* := Q(J_t)$;
  если $t - t^* \geqslant d$ **то прервать цикл**;

пока $|J_t| > 0$ удалять признаки (Del):
  $t := t + 1$; — началась следующая итерация;
  $f^* := \arg\min\limits_{f \in J_{t-1}} Q\big(J_{t-1} \setminus \{f\}\big)$; $J_t := J_{t-1} \setminus \{f^*\}$;
  если $Q(J_t) < Q^*$ **то** $t^* := t$; $Q^* := Q(J_t)$;
  если $t - t^* \geqslant d$ **то прервать цикл**;

пока значения критерия $Q(J_{t^*})$ уменьшаются;
вернуть J_{t^*};

Алгоритм поочерёдного добавления и удаления (Add-Del)

Преимущества:

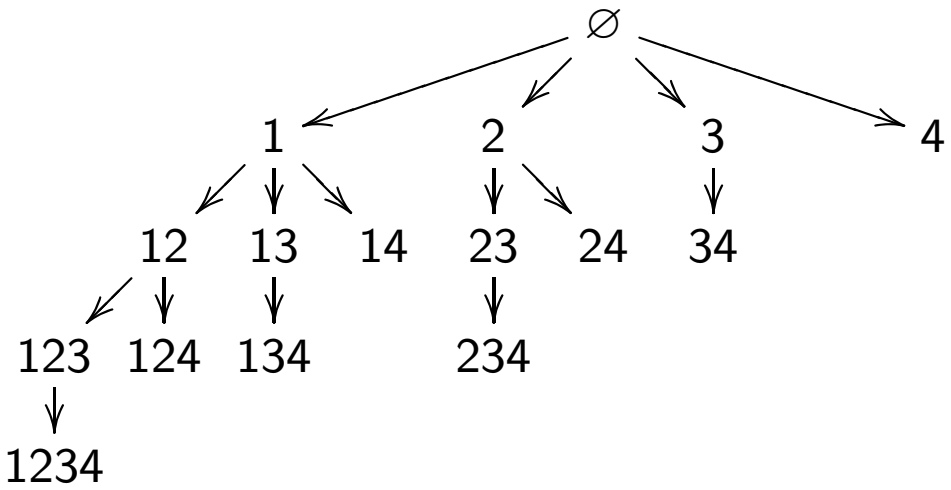
- как правило, лучше, чем Add и Del по отдельности;
- возможны быстрые инкрементные алгоритмы, пример — *шаговая регрессия* (step-wise regression).

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 J 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 J:
набор J не наращивается, если

$$\exists j: \quad Q(J) \geqslant \varkappa Q_j^* \quad \text{и} \quad |J| \geqslant j + d,$$

$d \geqslant 0$ — целочисленный параметр,
$\varkappa \geqslant 1$ — вещественный параметр.

Чем меньше d и $\varkappa$, тем сильнее сокращается перебор.

Поиск в глубину (DFS, метод ветвей и границ)

Вход: множество F, критерий Q, параметры d и $\varkappa$;

ПРОЦЕДУРА Нарастить (J);
если найдётся $j \leqslant |J| - d$ такое, что $Q(J) \geqslant \varkappa Q^*_j$, **то**
    выход;
$Q^*_{|J|} := \min\{Q^*_{|J|}, Q(J)\}$;
для всех $f_s \in F$ таких, что $s > \max\{t \mid f_t \in J\}$
    Нарастить $(J \cup \{f_s\})$;

Инициализация массива лучших значений критерия:
$Q^*_j := Q(\varnothing)$ для всех $j = 1, \ldots, n$;
Упорядочить признаки по убыванию информативности;
Нарастить $(\varnothing)$;
вернуть J, для которого $Q(J) = \min\limits_{j=1,\ldots,n} Q^*_j$;

Поиск в ширину (BFS)

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

Философия — принцип *неокончательных 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*:

$$R_j = \{J_j^1, \ldots, J_j^{B_j}\}, \quad J_j^b \subseteq F, \quad |J_j^b| = j, \quad b = 1, \ldots, B_j.$$

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

Поиск в ширину (BFS)

Вход: множество F, критерий Q, параметры d, B;

1: первый ряд состоит из всех наборов длины 1:
$R_1 := \{\{f_1\}, \ldots, \{f_n\}\}$; $Q^* = Q(\varnothing)$;
2: **для всех** $j = 1, \ldots, n$, где j — сложность наборов:
3: отсортировать ряд $R_j = \{J_j^1, \ldots, J_j^{B_j}\}$
по возрастанию критерия: $Q(J_j^1) \leqslant \ldots \leqslant Q(J_j^{B_j})$;
4: **если** $B_j > B$ **то**
5: $R_j := \{J_j^1, \ldots, J_j^B\}$; — B лучших наборов ряда;
6: **если** $Q(J_j^1) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j^1)$;
7: **если** $j - j^* \geqslant d$ **то вернуть** $J_{j^*}^1$;
8: породить следующий ряд:
$R_{j+1} := \{J \cup \{f\} \mid J \in R_j,\ f \in F \setminus J\}$;

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 $I_j(f)$:

$$I_j(f) = \sum_{b=1}^{B_j} [f \in J_j^b].$$

Генетический алгоритм поиска (идея и терминология)

$J \subseteq F$ — *индивид* (в МГУА «модель»);

$R_t := \left\{ J_t^1, \ldots, J_t^{B_t} \right\}$ — *поколение* (в МГУА — «ряд»);

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

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 F, критерий Q, параметры: d, p_m,
B — размер популяции, T — число поколений;

1: инициализировать случайную популяцию из B наборов:
$B_1 := B$; $R_1 := \left\{J_1^1, \ldots, J_1^{B_1}\right\}$; $Q^* := Q(\varnothing)$;
2: **для всех** $t = 1, \ldots, T$, где t — номер поколения:
3: ранжирование индивидов: $Q(J_t^1) \leqslant \ldots \leqslant Q(J_t^{B_t})$;
4: **если** $B_t > B$ **то**
5: селекция: $R_t := \left\{J_t^1, \ldots, J_t^B\right\}$;
6: **если** $Q(J_t^1) < Q^*$ **то** $t^* := t$; $Q^* := Q(J_t^1)$;
7: **если** $t - t^* \geqslant d$ **то вернуть** $J_{t^*}^1$;
8: породить t+1-е поколение путём скрещиваний и мутаций:
$R_{t+1} := \left\{\sim(J' \times J'') \;\middle|\; J', J'' \in R_t\right\} \cup R_t$;

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 теоретических гарантий;
- подбор параметров — непростое искусство;

Случайный поиск — упрощенный генетический алгоритм

Модификация: шаг 8

- породить $t+1$-е поколение путём многократных *мутаций*:
  $R_{t+1} := \left\{\sim J, \ldots, \sim J \;\middle|\; J \in R_t\right\} \cup R_t$;

Недостатки:

- ничем не лучше ГА;
- сходимость ещё медленнее.

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 F, критерий Q, параметры d, j_0, T, r, h;

$p_1 = \dots = p_n := 1/n$; — равные вероятности признаков;
для всех $j = j_0, \dots, n$, где j — сложность наборов:
  для всех $t = 1, \dots, T$, где t — номер итерации:
    r случайных наборов признаков из распределения $\{p_1, \dots, p_n\}$:
    $R_{jt} := \{J_{jt}^1, \dots, J_{jt}^r\}, \quad |J_{jt}^1| = \dots = |J_{jt}^r| = j$;
    $J_{jt}^{\min} := \arg\min\limits_{J \in R_{jt}} Q(J)$; — лучший из r наборов;
    $J_{jt}^{\max} := \arg\max\limits_{J \in R_{jt}} Q(J)$; — худший из r наборов;
    $H := 0$; наказание для всех $f_s \in J_{jt}^{\max}$:
    $\Delta p_s := \min\{p_s, h\}$; $p_s := p_s - \Delta p_s$; $H := H + \Delta p_s$;
    поощрение для всех $f_s \in J_{jt}^{\min}$: $p_s := p_s + H/j$;
  $J_j := \arg\min\limits_{J \in R_{j1}, \dots, R_{jT}} Q(J)$; — лучший набор сложности j;
  если $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;
  если $j - j^* \geqslant d$ **то вернуть** J_{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;

$h \approx \frac{1}{r\,n}$ — 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.

Лбов Г. С. Выбор эффективной системы зависимых признаков // Вычислительные системы, 1965, Т. 19, С. 21–34.

Загоруйко Н. Г., Ёлкина В. Н., Лбов Г. С. Алгоритмы обнаружения эмпирических закономерностей. Новосибирск: Наука, 1985.

Резюме в конце лекции

- Критерий $Q(J)$ должен иметь оптимум по сложности:
  — внешний критерий — оценка обобщающей способности,
  — информативность — для логических правил.
- Для отбора признаков могут использоваться любые эвристические методы дискретной оптимизации

$$Q(J) \to \min_{J \subseteq F}.$$

- Большинство эвристик эксплуатируют две основные идеи:
  — не все признаки информативны;
  — $Q(J)$ изменяется не сильно при небольшом изменении J.
- МГУА, ГА, СПА очень похожи — на их основе легко создавать различные «симбиотические» алгоритмы.