Мат.модели машинного обучения: оценивание качества обучения, методы отбора признаков

Воронцов Константин Вячеславович
k.v.vorontsov@phystech.edu
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)»

МФТИ • 25 октября 2024

Содержание

Анализ ошибок классификации

Задача классификации на два класса: y_i, $a(x_i) \in \{-1, +1\}$.

	модель классификации	учитель
TP, True Positive	$a(x_i) = +1$	$y_i = +1$
TN, True Negative	$a(x_i) = -1$	$y_i = -1$
FP, False Positive	$a(x_i) = +1$	$y_i = -1$
FN, False Negative	$a(x_i) = -1$	$y_i = +1$

FP: ложноположительно, ошибка I рода, «ложная тревога»

FN: ложноотрицательно, ошибка II рода, «пропуск цели»

Правильность классификации (чем больше, тем лучше):

$$\text{Accuracy} = \frac{1}{\ell}\sum_{i=1}^{\ell}[a(x_i) = y_i] = \frac{\text{TP} + \text{TN}}{\text{FP} + \text{FN} + \text{TP} + \text{TN}}$$

Недостаток: не учитывается дисбаланс численности классов, различие цены ошибки I и II рода.

Функции потерь, зависящие от штрафов за ошибку

Задача классификации на два класса, $y_i \in \{-1, +1\}$.
Модель классификации: $a(x; w, w_0) = \text{sign}\big(g(x, w) - w_0\big)$.
Чем больше w_0, тем больше x_i таких, что $a(x_i) = -1$.

Пусть λ_y — штраф за ошибку на объекте класса y.
Функция потерь теперь зависит от штрафов:

$$\mathscr{L}(w, x_i) = \lambda_{y_i}\big[a(x_i; w, w_0) \neq y_i\big] = \lambda_{y_i}\big[(g(x_i, w) - w_0)y_i < 0\big].$$

Проблема

На практике штрафы $\{\lambda_y\}$ могут пересматриваться

- Нужен удобный способ выбора w_0 в зависимости от $\{\lambda_y\}$, не требующий построения w заново.
- Нужна характеристика качества модели $g(x, w)$, не зависящая от штрафов $\{\lambda_y\}$ и численности классов.

Определение ROC-кривой

Кривая ошибок ROC (receiver operating characteristic).
Каждая точка кривой соответствует некоторому $a(x; w, w_0)$.

- по оси X: доля *ошибочных положительных классификаций* (FPR — false positive rate):

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} = \frac{\sum_{i=1}^{\ell} [y_i = -1][a(x_i; w, w_0) = +1]}{\sum_{i=1}^{\ell} [y_i = -1]};$$

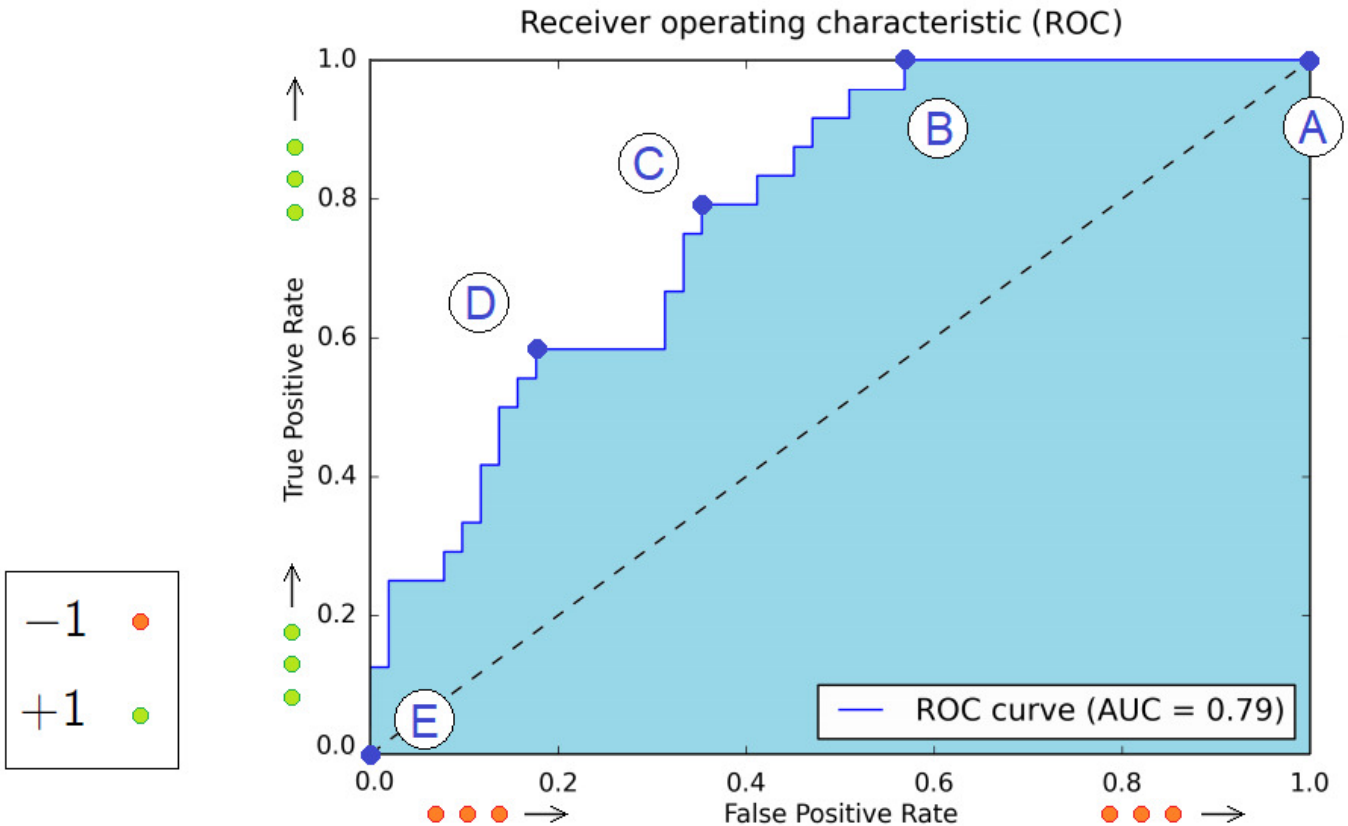
  $1 - \text{FPR}$ называется *специфичностью* алгоритма a.

- по оси Y: доля *правильных положительных классификаций* (TPR — true positive rate):

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\sum_{i=1}^{\ell} [y_i = +1][a(x_i; w, w_0) = +1]}{\sum_{i=1}^{\ell} [y_i = +1]};$$

  TPR называется также *чувствительностью* алгоритма a.

ROC-кривая и площадь под кривой AUC (Area Under Curve)



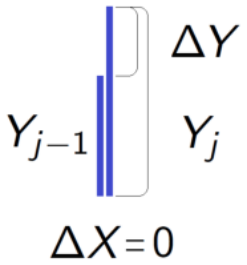
ABCDE — положения порога w_0 на оси значений функции g

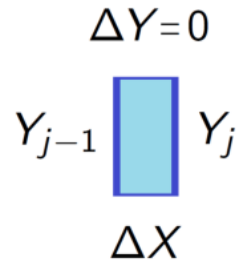
Алгоритм эффективного построения ROC-кривой

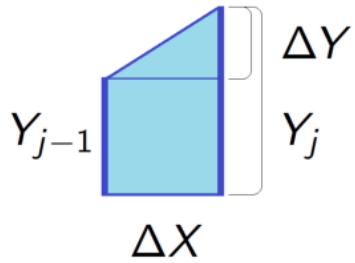
Вход: выборка $\{x_i\}_{i=1}^{\ell}$; дискриминантная функция $g(x, w)$;
Выход: ROC-кривая $(X_j, Y_j)_{j=0}^{k}$, $k \leqslant \ell$ и площадь AUC

$\ell_y := \sum_{i=1}^{\ell} [y_i = y]$, для всех $y \in Y$;
упорядочить $\{x_i\}$ по убыванию $g_i = g(x_i, w)$: $g_1 \geqslant \ldots \geqslant g_\ell$;
$(X_0, Y_0) := (0, 0)$; AUC $:= 0$; $\Delta X := 0$; $\Delta Y := 0$; $j := 1$;

для $i := 1, \ldots, \ell$
- $\Delta X := \Delta X + \frac{1}{\ell_-}[y_i = -1]$;
- $\Delta Y := \Delta Y + \frac{1}{\ell_+}[y_i = +1]$;
- **если** $(g_i \neq g_{i-1})$ **то**
  - $X_j := X_{j-1} + \Delta X$;
  - $Y_j := Y_{j-1} + \Delta Y$;
  - AUC $:=$ AUC $+ \frac{1}{2}(Y_{j-1} + Y_j)\Delta X$;
  - $j := j + 1$; $\Delta X := 0$; $\Delta Y := 0$;







Градиентная максимизация AUC

Модель классификации: $a(x_i, w, w_0) = \text{sign}\big(g(x_i, w) - w_0\big)$.

AUC — это доля правильно упорядоченных пар (x_i, x_j):

$$\text{AUC}(w) = \frac{1}{\ell_-}\sum_{i=1}^{\ell}\big[y_i = -1\big]\text{TPR}_i =$$

$$= \frac{1}{\ell_-\ell_+}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\big[y_i < y_j\big]\big[g(x_i, w) < g(x_j, w)\big] \to \max_w$$

Явная максимизация аппроксимированного AUC:

$$1 - \text{AUC}(w) \leqslant Q(w) = \sum_{i,j:\, y_i<y_j} L\underbrace{\big(g(x_j, w) - g(x_i, w)\big)}_{M_{ij}(w)} \to \min_w$$

$L(M)$ — убывающая функция отступа,

$M_{ij}(w)$ — новое понятие отступа для пар объектов.

Алгоритм SG для максимизации AUC

Возьмём для простоты линейный классификатор:

$$g(x,w) = \langle x, w\rangle, \qquad M_{ij}(w) = \langle x_j - x_i, w\rangle, \qquad y_i < y_j.$$

Вход: выборка X^ℓ, темп обучения h, темп забывания λ;
Выход: вектор весов w;

инициализировать веса w_j, $j = 0, \ldots, n$;
инициализировать оценку: $\bar{Q} := \frac{1}{\ell_+\ell_-} \sum_{i,j} [y_i < y_j]\, L(M_{ij}(w))$;

повторять
  выбрать пару объектов (i,j): $y_i < y_j$, случайным образом;
  вычислить потерю: $\varepsilon_{ij} := L(M_{ij}(w))$;
  сделать градиентный шаг: $w := w - h\, L'(M_{ij}(w))(x_j - x_i)$;
  оценить функционал: $\bar{Q} := (1-\lambda)\bar{Q} + \lambda\varepsilon_{ij}$;
пока значение $\bar{Q}$ и/или веса w не сойдутся;

Логарифм правдоподобия, log-loss

Вероятностная модель бинарной классификации, $y_i \in \{-1, +1\}$:

$$a(x, w) = \text{sign}\big(g(x, w) - w_0\big), \qquad g(x, w) = \text{P}(y = +1 | x, w).$$

Проблема: ROC и AUC инвариантны относительно монотонных преобразований дискриминантной функции $g(x, w)$.

Критерий логарифма правдоподобия (log-loss):

$$Q(w) = \sum_{i=1}^{\ell} [y_i = +1] \ln g(x, w) + [y_i = -1] \ln\big(1 - g(x, w)\big) \to \max_w$$

Вероятностная модель многоклассовой классификации:

$$a(x) = \arg\max_{y \in Y} \text{P}(y | x, w);$$

$$Q(w) = \sum_{i=1}^{\ell} \ln \text{P}(y_i | x_i, w) \to \max_w$$

Точность и полнота бинарной классификации

В информационном поиске не важен TN:

$$\text{Точность, Precision} = \frac{TP}{TP+FP}$$

$$\text{Полнота, Recall} = \frac{TP}{TP+FN}$$

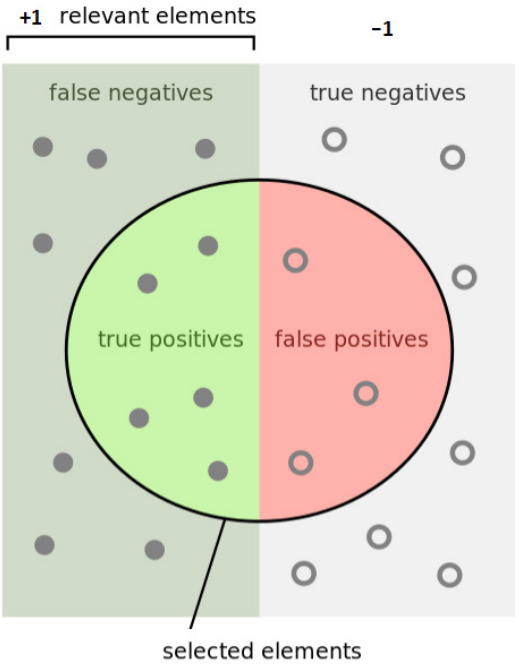
Precision — доля релевантных среди найденных
Recall — доля найденных среди релевантных

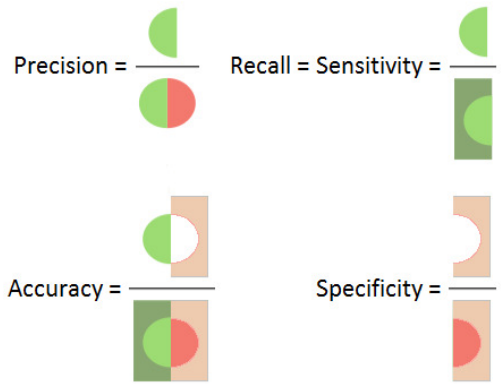
В медицинской диагностике:

$$\text{Чувствительность, Sensitivity} = \frac{TP}{TP+FN}$$

$$\text{Специфичность, Specificity} = \frac{TN}{TN+FP}$$

Sensitivity — доля верных положительных диагнозов
Specificity — доля верных отрицательных диагнозов





Точность и полнота многоклассовой классификации

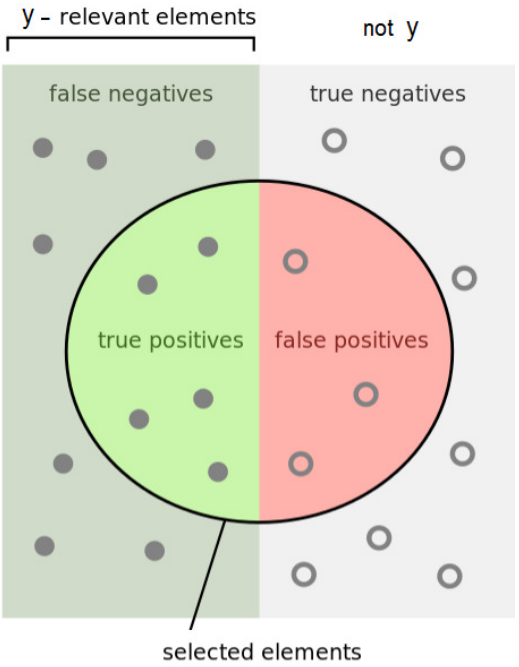
Для каждого класса $y \in Y$:
TP_y — верные положительные
FP_y — ложные положительные
FN_y — ложные отрицательные

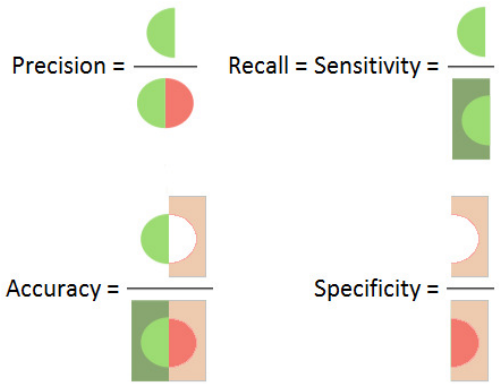
Точность и полнота с микроусреднением:

$$\text{Precision: } P = \frac{\sum_y \mathrm{TP}_y}{\sum_y (\mathrm{TP}_y + \mathrm{FP}_y)};$$

$$\text{Recall: } R = \frac{\sum_y \mathrm{TP}_y}{\sum_y (\mathrm{TP}_y + \mathrm{FN}_y)};$$

Микроусреднение не чувствительно
к ошибкам на малочисленных классах





Точность и полнота многоклассовой классификации

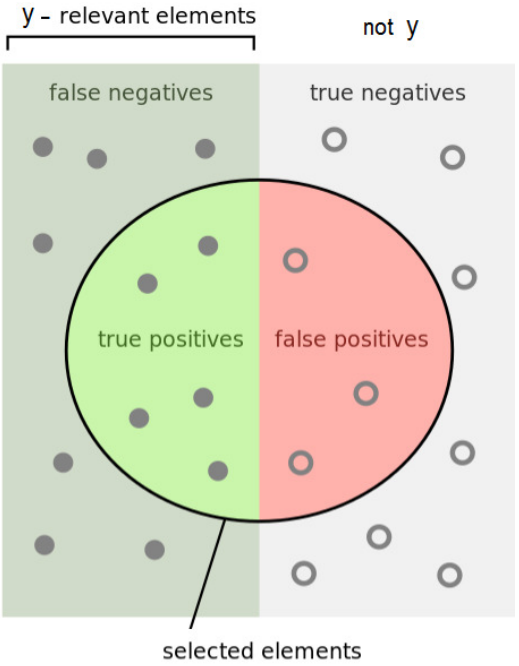
Для каждого класса $y \in Y$:
TP_y — верные положительные
FP_y — ложные положительные
FN_y — ложные отрицательные

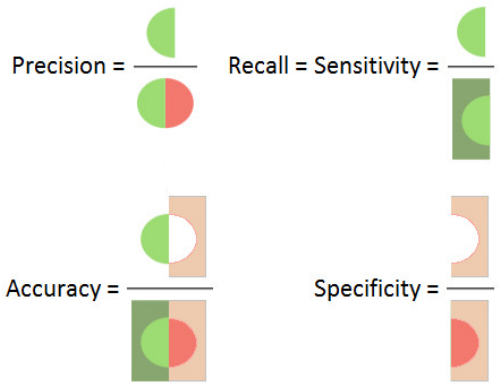
Точность и полнота с макроусреднением:

$$\text{Precision: } P = \frac{1}{|Y|} \sum_y \frac{\mathrm{TP}_y}{\mathrm{TP}_y + \mathrm{FP}_y};$$

$$\text{Recall: } R = \frac{1}{|Y|} \sum_y \frac{\mathrm{TP}_y}{\mathrm{TP}_y + \mathrm{FN}_y};$$

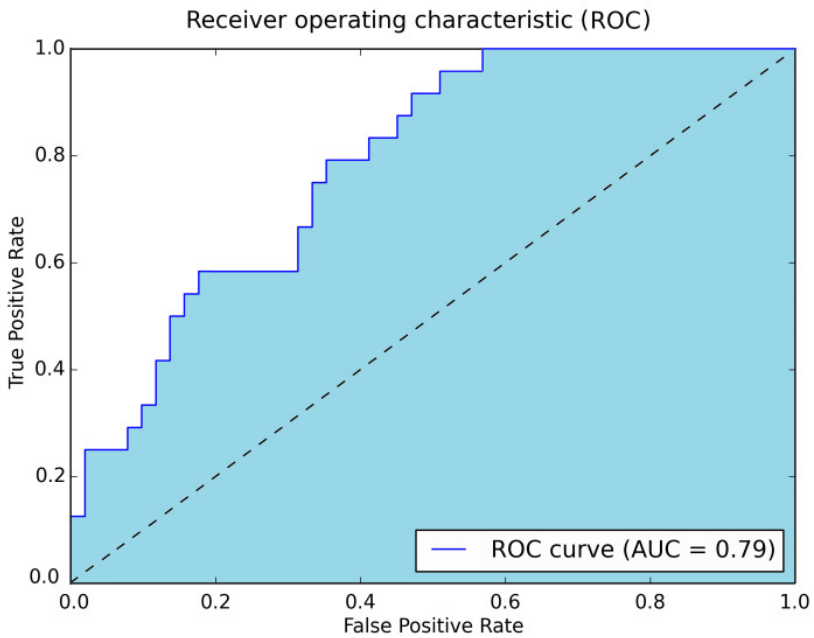
Макроусреднение чувствительно
к ошибкам на малочисленных классах

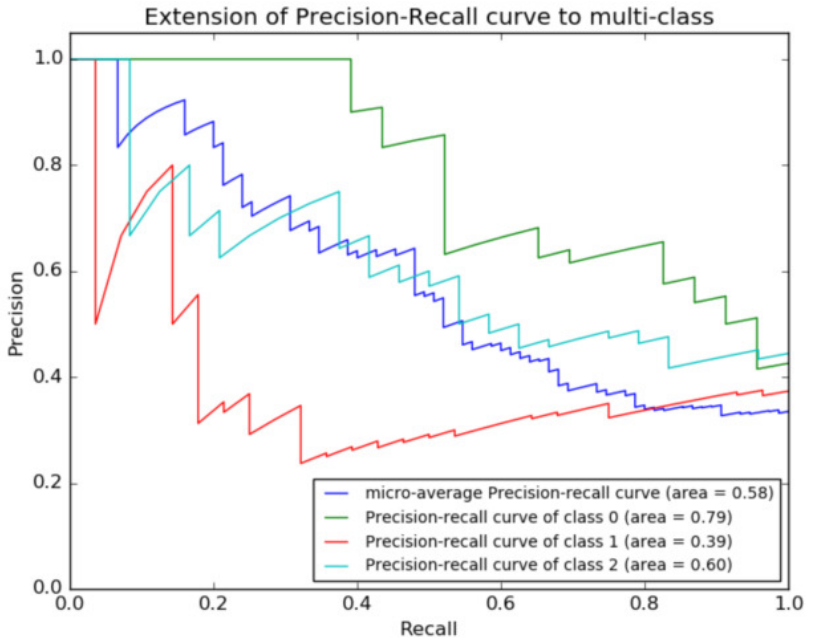




Кривые ROC и Precision-Recall

Модель классификации: $a(x) = \text{sign}\big(\langle x, w\rangle - w_0\big)$
Каждая точка кривой соответствует значению порога w_0





AUROC — площадь под ROC-кривой
AUPRC — площадь под кривой Precision-Recall

Примеры из Python scikit learn: `http://scikit-learn.org/dev`

Резюме. Оценки качества классификации

- Чувствительность и специфичность лучше подходят для задач с несбалансированными классами
- Логарифм правдоподобия (log-loss) лучше подходит для оценки качества вероятностной модели классификации.
- Точность и полнота лучше подходят для задач поиска, когда доля объектов релевантного класса очень мала.

Агрегированные оценки:

- AUC лучше подходит для оценивания качества, когда соотношение цены ошибок не фиксировано.
- AUPRC — площадь под кривой точность–полнота.
- $F_1 = \frac{2PR}{P+R}$ — F-мера, другой способ агрегирования P и R.
- $F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P+R}$ — F_β-мера: чем больше β, тем важнее R.

Задачи выбора модели и метода обучения

Дано: X — пространство объектов; Y — множество ответов;
$X^\ell = (x_i, y_i)_{i=1}^\ell$ — обучающая выборка, $y_i = y(x_i)$;
$A_t = \{a \colon X \times W_t \to Y\}$ — параметрические модели, $t \in T$
W_t — пространство параметров модели A_t
$\mu_t \colon (X \times Y)^\ell \to W_t$ — 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 (feature selection):
  $F = \{f_j \colon X \to D_j \colon j = 1, \ldots, n\}$ — множество признаков;
  метод обучения μ_J использует только признаки $J \subseteq F$.

Обобщающая (предсказательная) способность метода

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

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

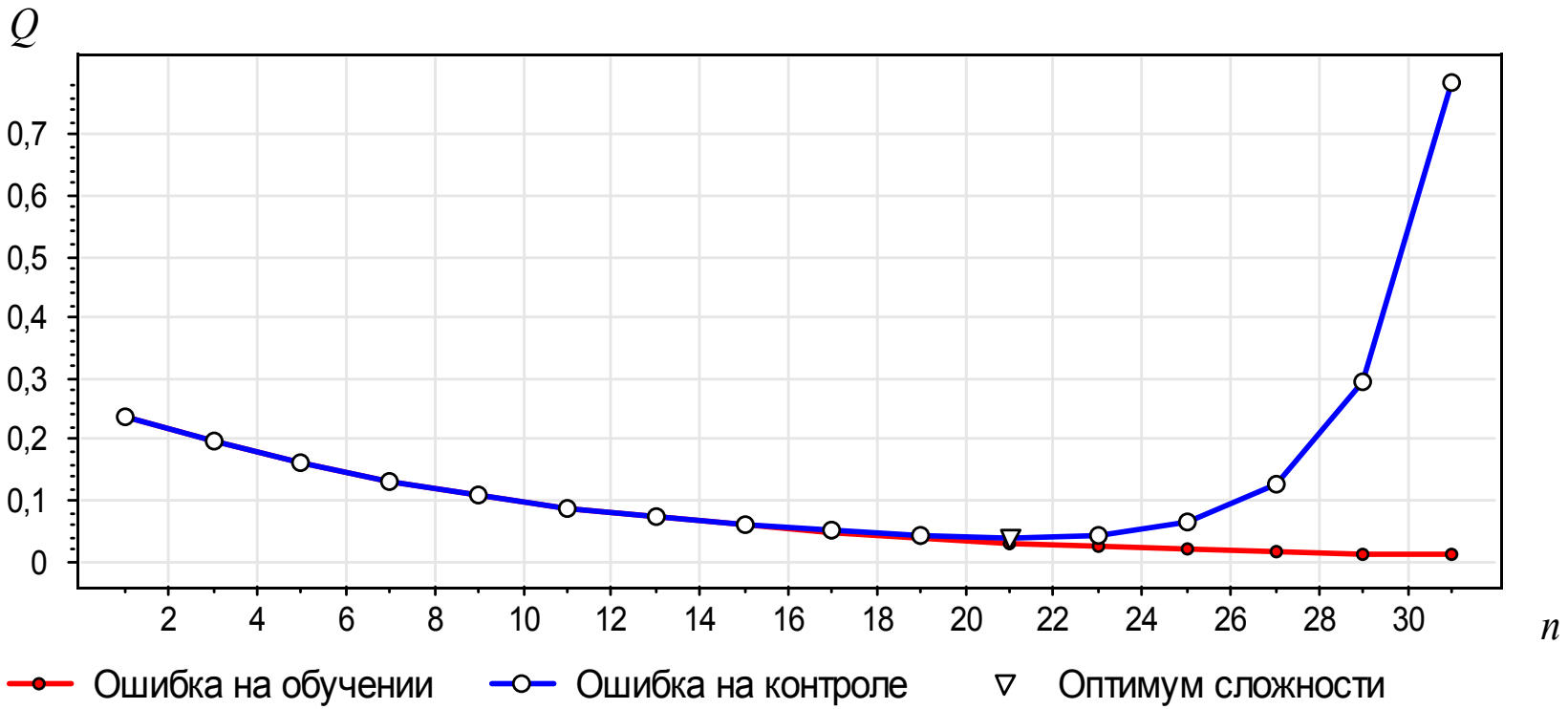
$$Q_\mu(X^\ell, X^k) = Q\big(\mu(X^\ell), X^k\big).$$

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 множества N.

1. $|N| = 1$ — единственное разбиение: hold-out.

2. N — случайное множество разбиений: метод Монте-Карло.

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

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

Скользящий контроль и поблочная кросс-проверка

4. 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.

5. 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.

Многократная поблочная кросс-проверка

6. 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 $X^L = X_s^\ell \sqcup X_s^k$, $s = 1, \ldots, t$:

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

Преимущество: походит также для обучения без учителя.

Недостатки:

— объём обучения сокращается в 2 раза;

— трудоёмкость возрастает в $2t$ раз.

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):

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

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):

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

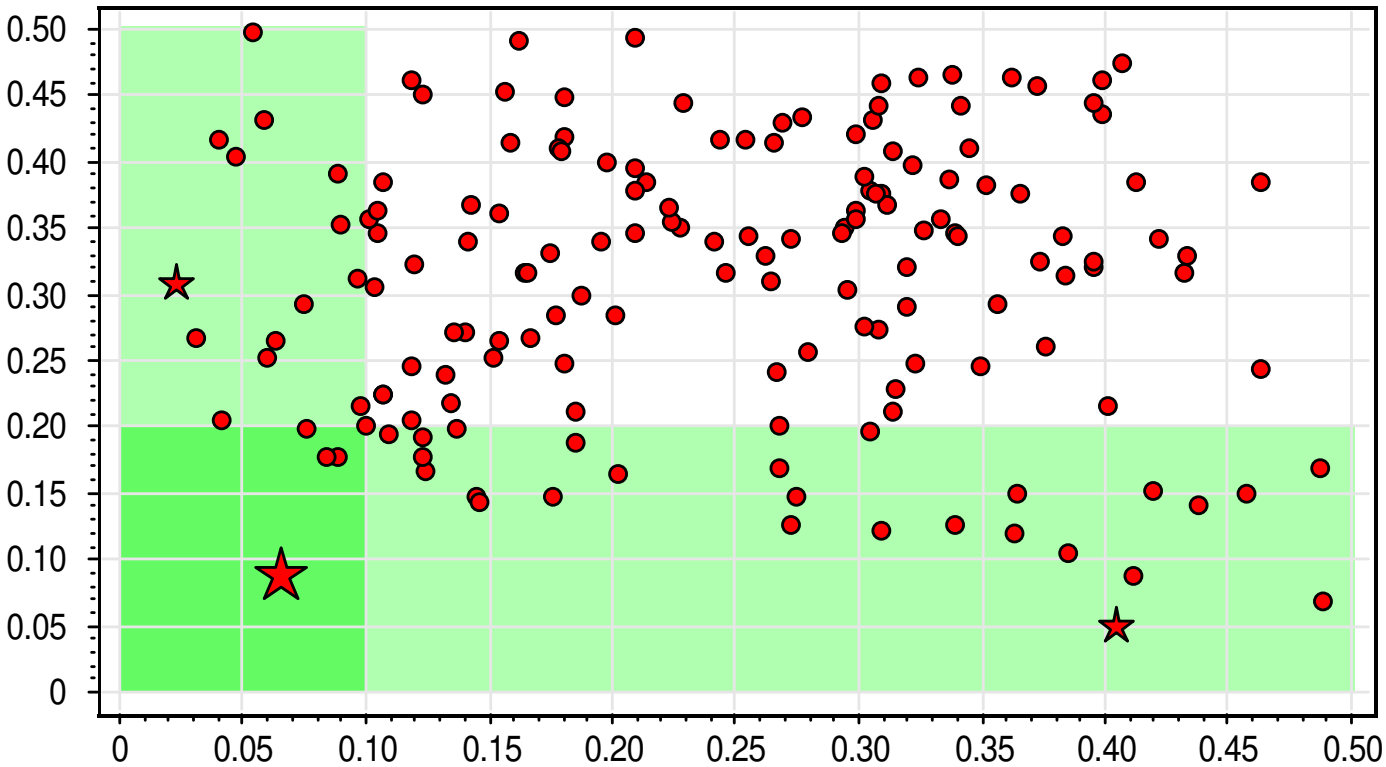
Оценка Вапника-Червоненкиса (VC-bound):

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

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

Многокритериальный выбор модели

Модель, немного неоптимальная по обоим критериям, может оказаться лучше, чем модель, оптимальная по одному критерию, но сильно не оптимальная по другому.



Задача отбора признаков по внешнему критерию

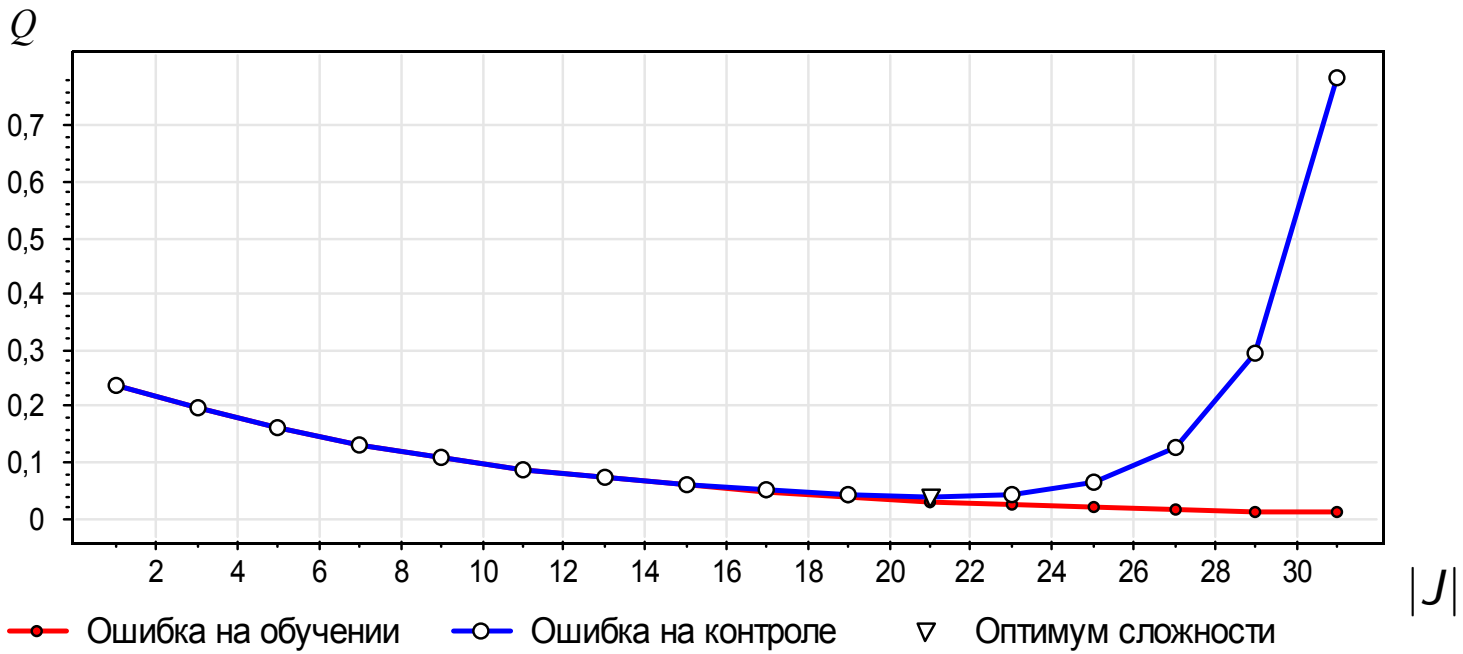
$F = \{f_j : X \to D_j : j = 1, \ldots, n\}$ — множество признаков;

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

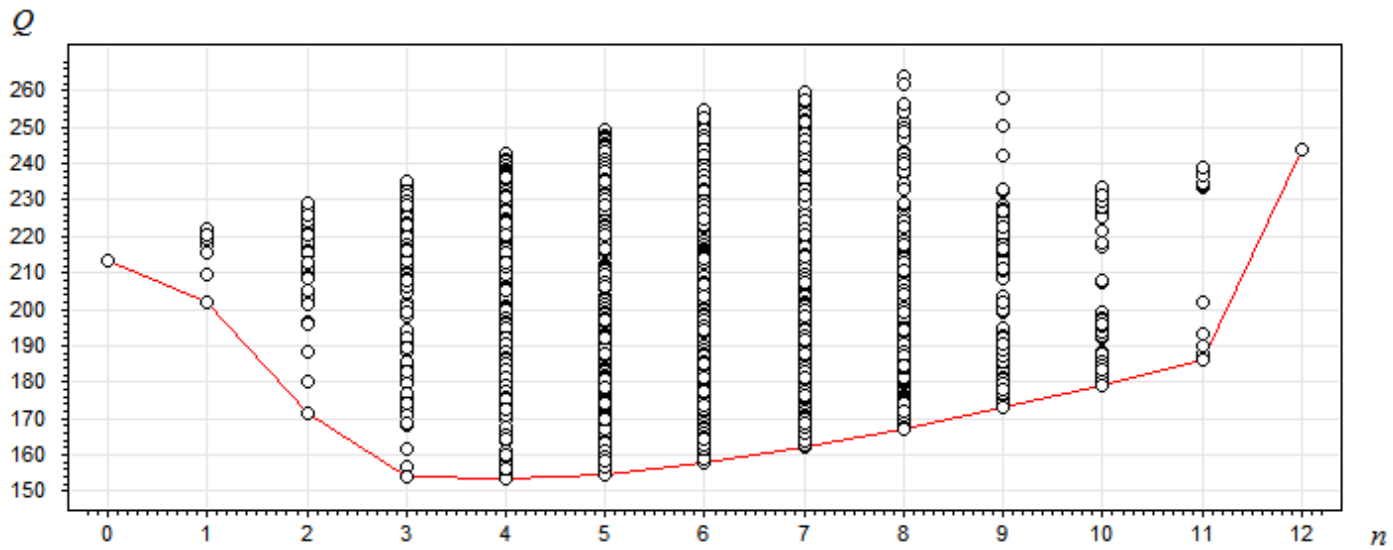
$Q(J) = Q(\mu_J, X^\ell)$ — выбранный внешний критерий.

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

Внутренний критерий и внешний критерий:



Алгоритм полного перебора (Full Search)



Вход: множество F, внешний критерий Q, параметр d;

инициализация: $Q^* := Q(\varnothing)$;

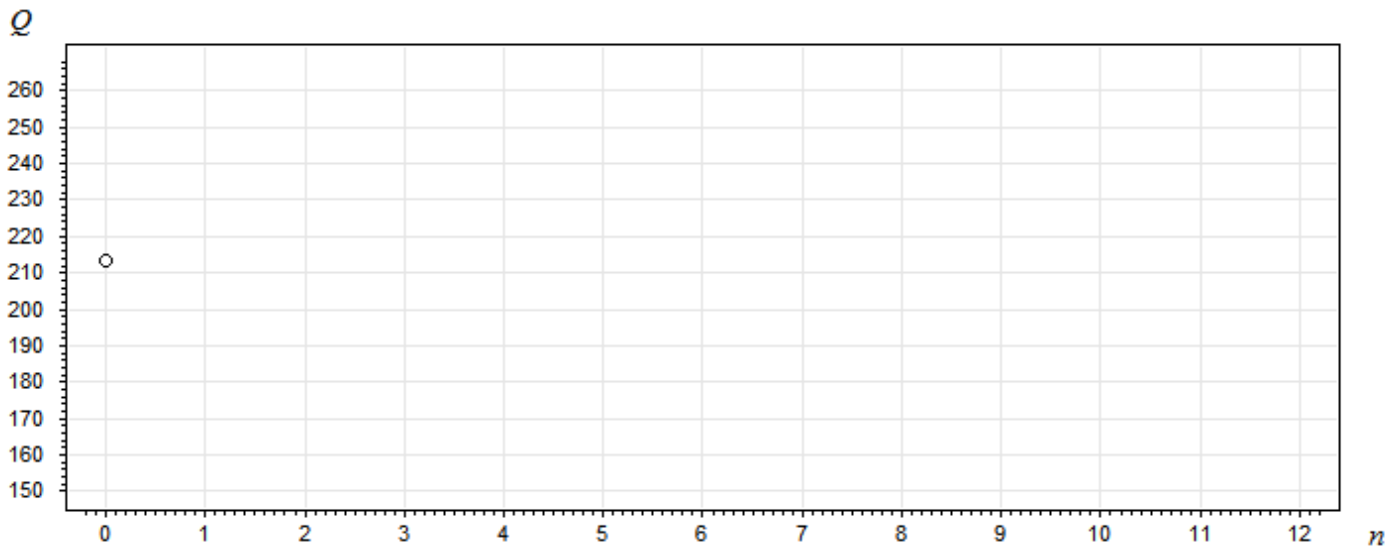
для $j = 1, \ldots, n$, где j — сложность наборов

$J_j := \arg\min_{J:\,|J|=j} Q(J)$ — найти 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^*};

Алгоритм полного перебора (Full Search)



$d = 3$
$j = 0$

Вход: множество F, внешний критерий Q, параметр d;

инициализация: $Q^* := Q(\varnothing)$;

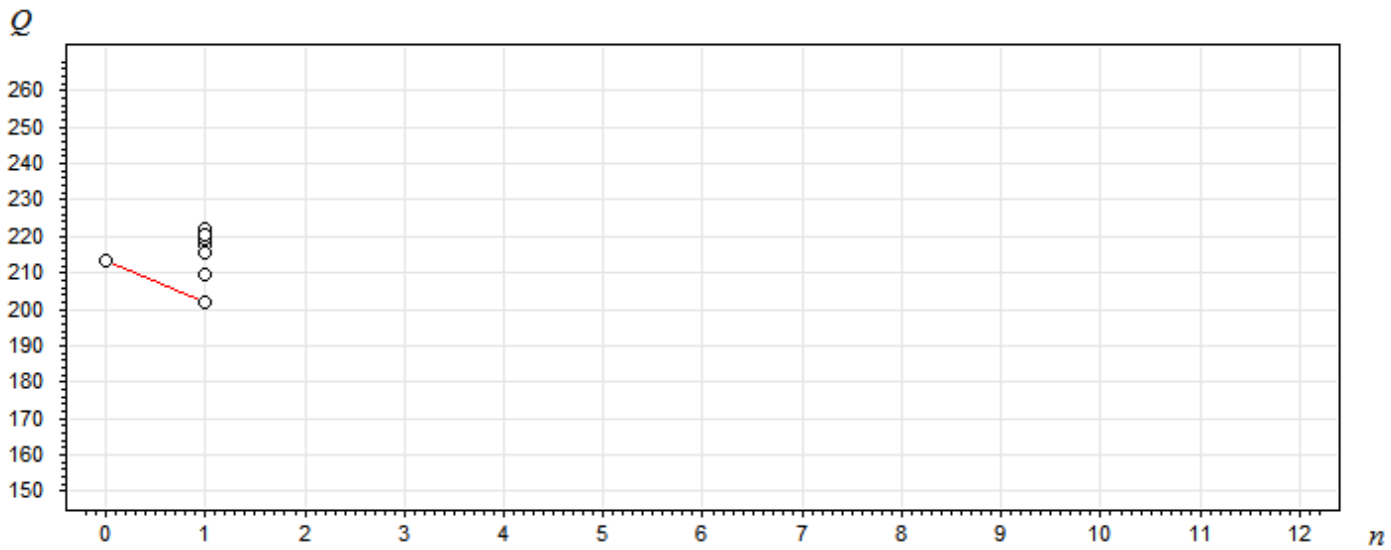
для $j = 1, \ldots, n$, где j — сложность наборов

  $J_j := \arg\min_{J:\,|J|=j} Q(J)$ — найти 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^*};

Алгоритм полного перебора (Full Search)



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

Вход: множество F, внешний критерий Q, параметр d;

инициализация: $Q^* := Q(\varnothing)$;

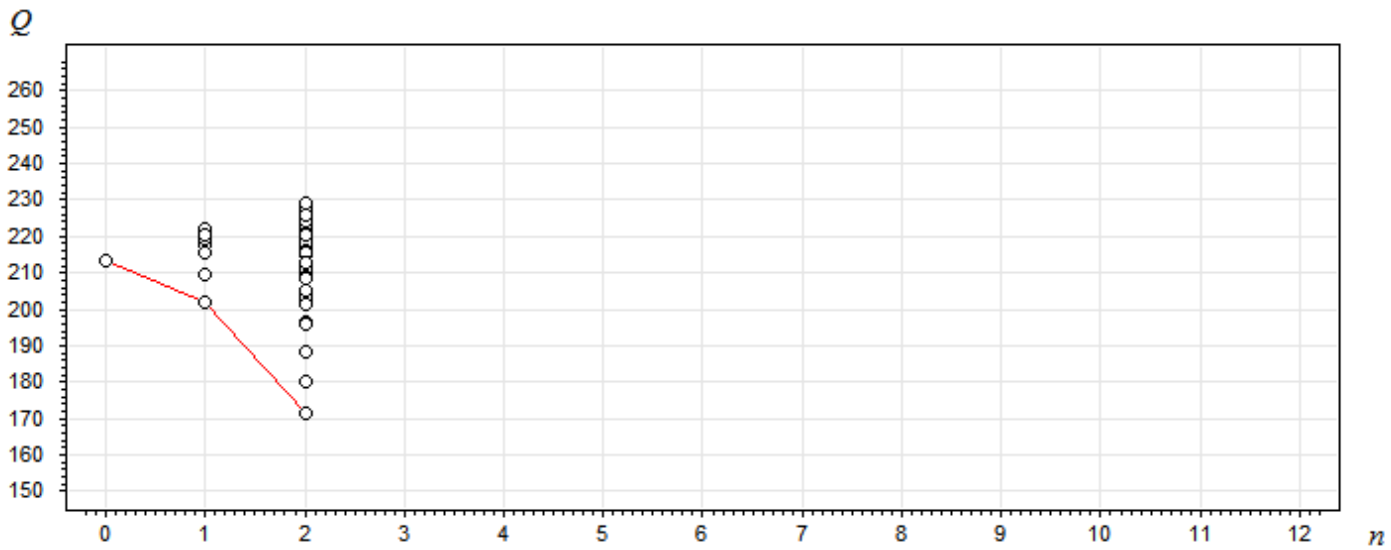
для $j = 1, \ldots, n$, где j — сложность наборов

  $J_j := \arg\min_{J:\,|J|=j} Q(J)$ — найти 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^*};

Алгоритм полного перебора (Full Search)



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

Вход: множество F, внешний критерий Q, параметр d;

инициализация: $Q^* := Q(\varnothing)$;

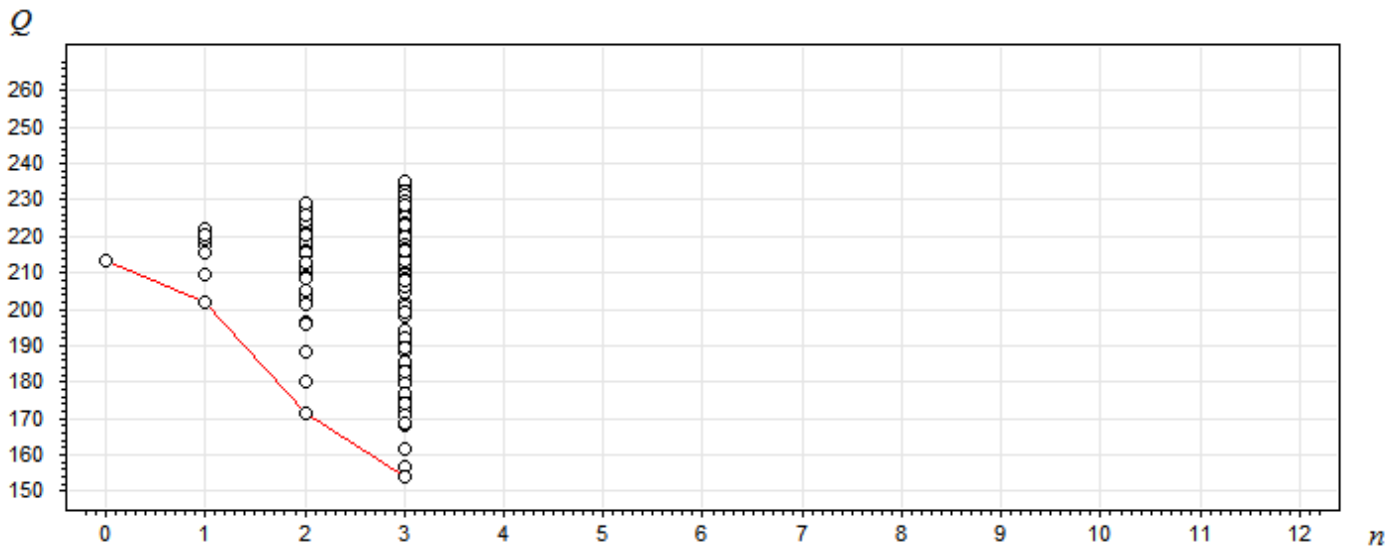
для $j = 1, \ldots, n$, где j — сложность наборов

  $J_j := \arg\min_{J:\,|J|=j} Q(J)$ — найти 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^*};

Алгоритм полного перебора (Full Search)



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

Вход: множество F, внешний критерий Q, параметр d;

инициализация: $Q^* := Q(\varnothing)$;

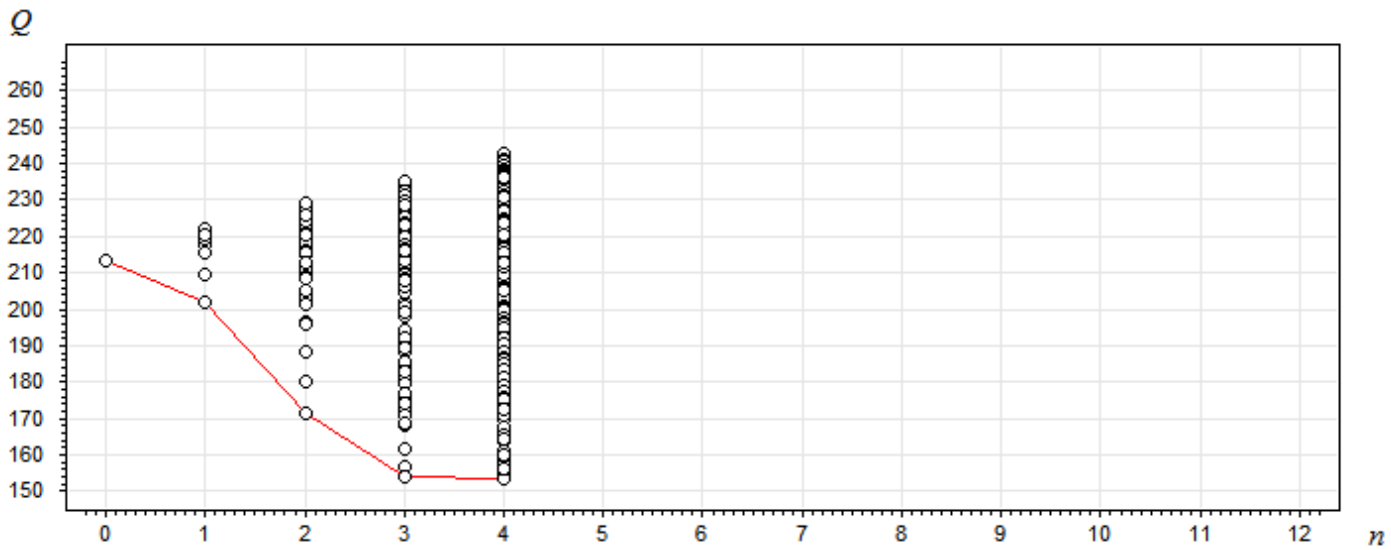
для $j = 1, \ldots, n$, где j — сложность наборов

  $J_j := \arg\min_{J:\,|J|=j} Q(J)$ — найти 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^*};

Алгоритм полного перебора (Full Search)



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

Вход: множество F, внешний критерий Q, параметр d;

инициализация: $Q^* := Q(\varnothing)$;

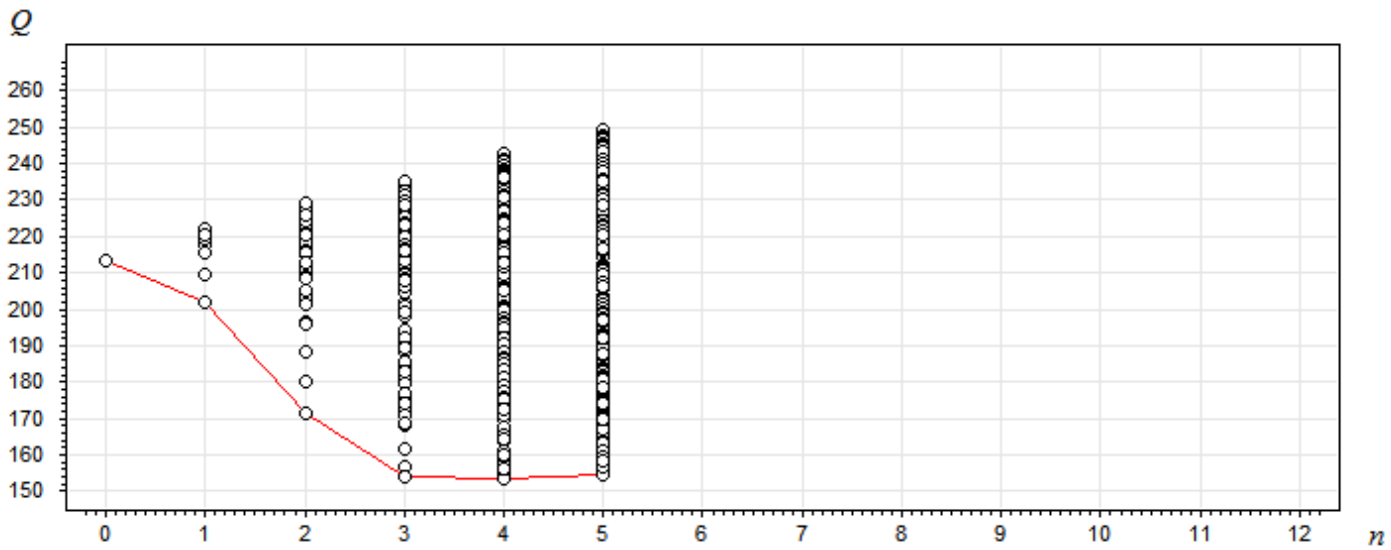
для $j = 1, \ldots, n$, где j — сложность наборов

  $J_j := \arg\min_{J:\,|J|=j} Q(J)$ — найти 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^*};

Алгоритм полного перебора (Full Search)



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

Вход: множество F, внешний критерий Q, параметр d;

инициализация: $Q^* := Q(\varnothing)$;

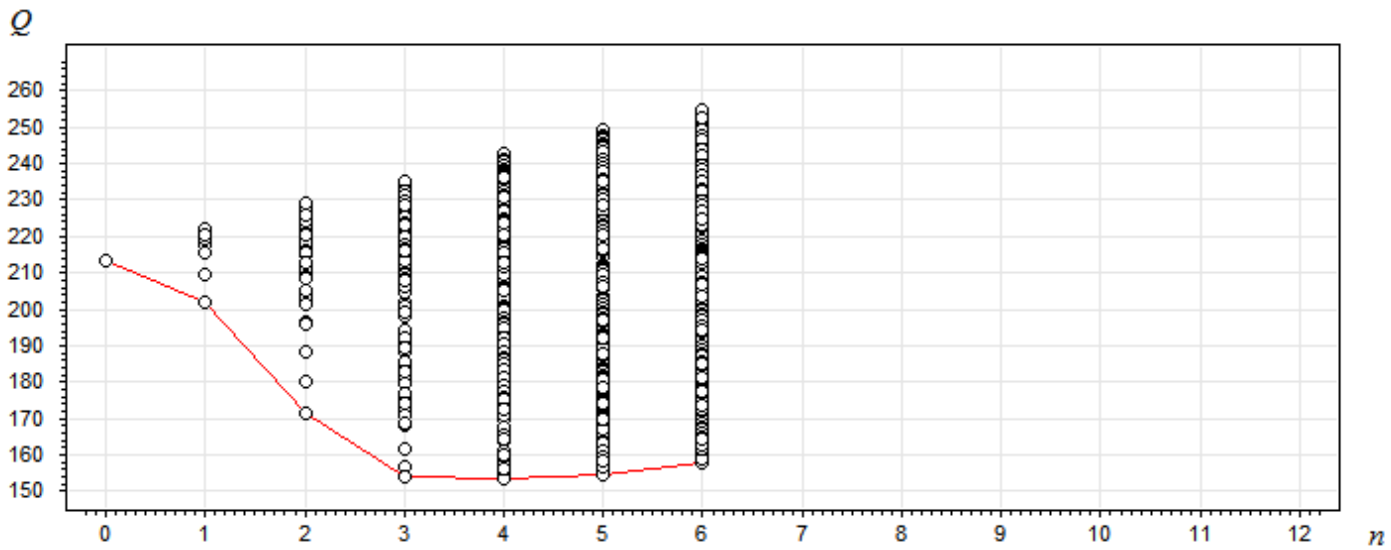
для $j = 1, \ldots, n$, где j — сложность наборов

  $J_j := \arg\min_{J:\,|J|=j} Q(J)$ — найти 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^*};

Алгоритм полного перебора (Full Search)



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

Вход: множество F, внешний критерий Q, параметр d;

инициализация: $Q^* := Q(\varnothing)$;

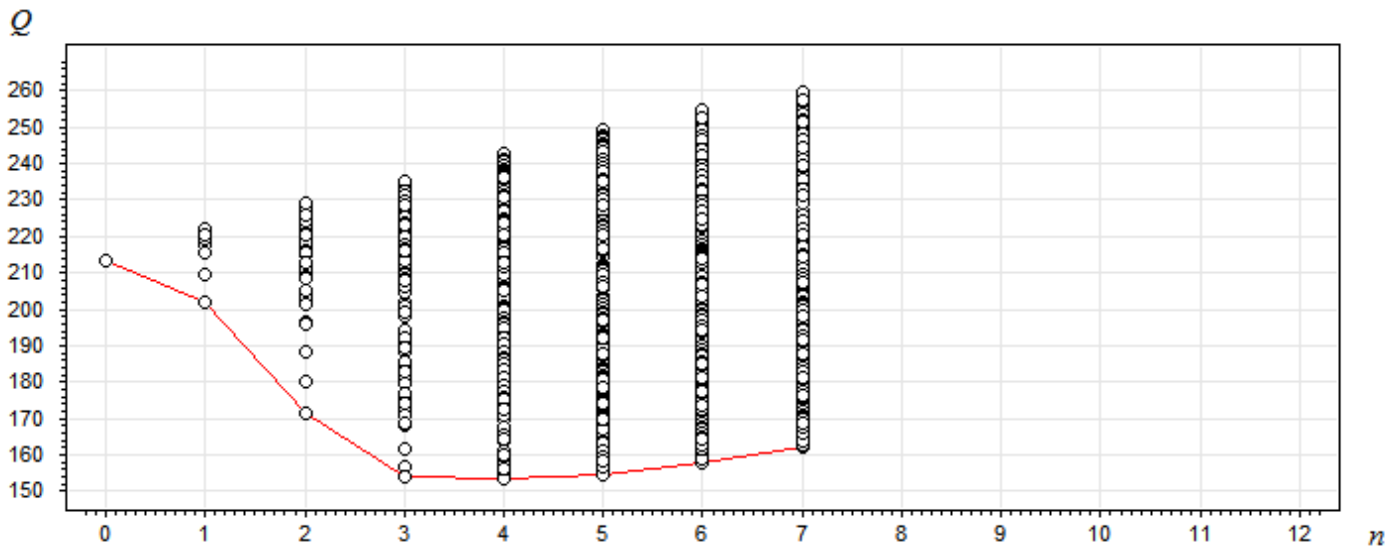
для $j = 1, \ldots, n$, где j — сложность наборов

  $J_j := \arg\min_{J:\,|J|=j} Q(J)$ — найти 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^*};

Алгоритм полного перебора (Full Search)



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

Вход: множество F, внешний критерий Q, параметр d;

инициализация: $Q^* := Q(\varnothing)$;

для $j = 1, \ldots, n$, где j — сложность наборов

  $J_j := \arg\min_{J:\,|J|=j} Q(J)$ — найти 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^*};

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;

инициализация: $J_0 := \varnothing$; $Q^* := Q(\varnothing)$;

для $j = 1, \ldots, n$, где j — сложность наборов:

найти признак, наиболее выгодный для добавления:

$f^* := \arg\min_{f \in F \setminus J_{j-1}} Q\left(J_{j-1} \cup \{f\}\right)$;

добавить этот признак в набор:

$J_j := J_{j-1} \cup \{f^*\}$;

если $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;

если $j - j^* \geqslant d$ **то вернуть** J_{j^*};

Преимущество: скорость $O(n^2)$, точнее $O(nj^*)$, вместо $O(2^n)$

Недостаток: склонность включать в набор лишние признаки

Способы устранения: Del, Add-Del, Beam Search

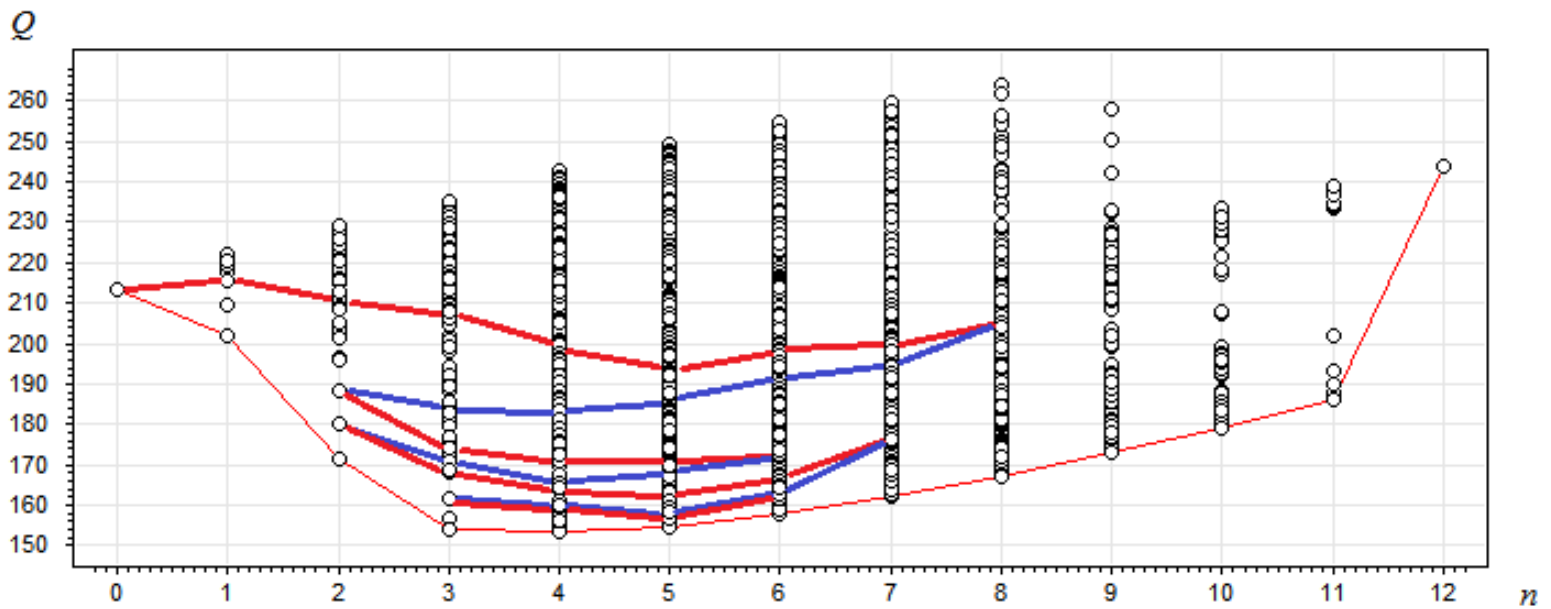
Алгоритм поочерёдного добавления и удаления (Add-Del)

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.

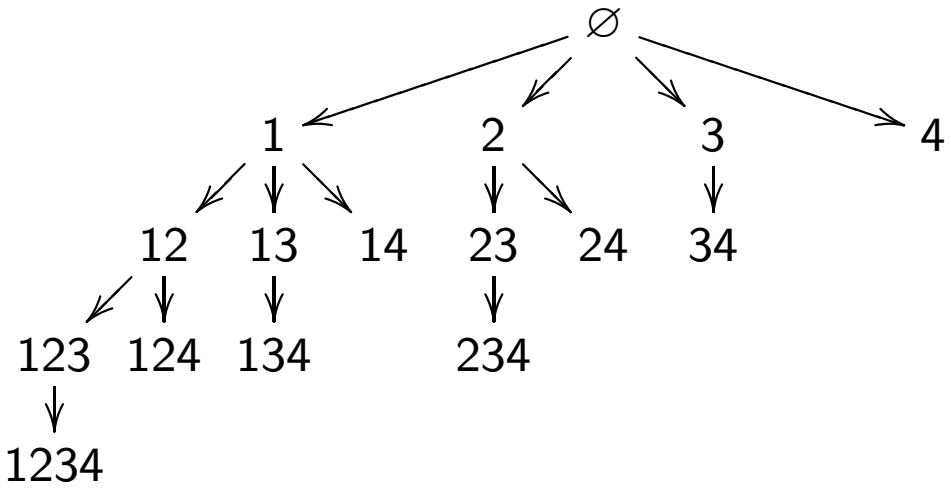


Алгоритм поочерёдного добавления и удаления (Add-Del)

инициализация: $J_0 := \varnothing$; $Q^* := Q(\varnothing)$; $t := 0$;
повторять
  пока $|J_t| < n$ добавлять признаки (итерации Add):
    $t := t + 1$ — началась следующая итерация;
    $f^* := \arg\min_{f \in F \setminus J_{t-1}} Q\big(J_{t-1} \cup \{f\}\big)$; $J_t := J_{t-1} \cup \{f^*\}$;
    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 (итерации Del):
    $t := t + 1$ — началась следующая итерация;
    $f^* := \arg\min_{f \in J_{t-1}} Q\big(J_{t-1} \setminus \{f\}\big)$; $J_t := J_{t-1} \setminus \{f^*\}$;
    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^*};

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.

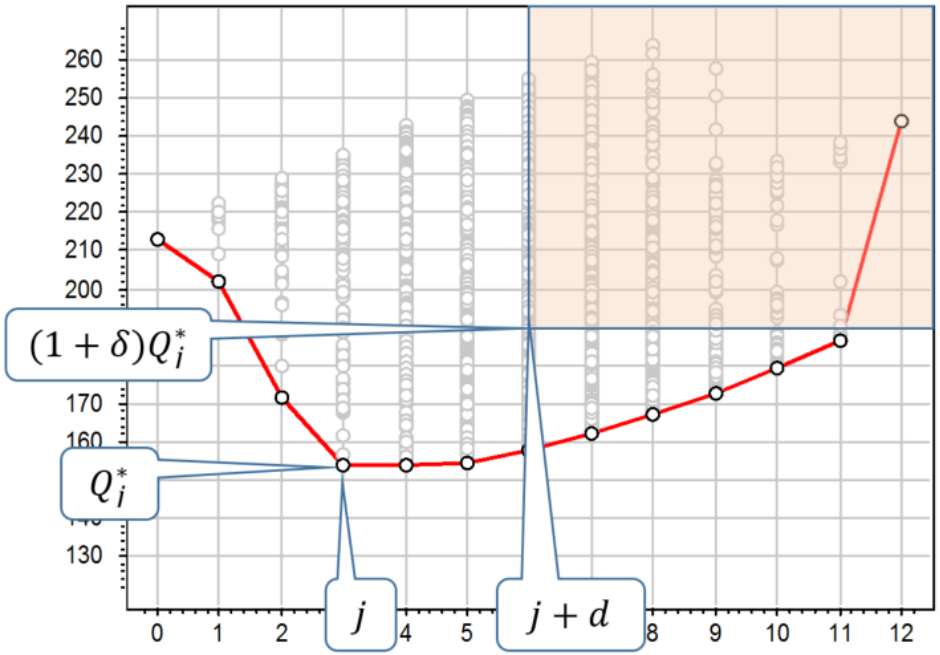
Поиск в глубину (DFS, метод ветвей и границ)

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

Оценка перспективности:
набор J не наращивается,
если найдётся j такой, что

$$\begin{cases} Q(J) \geqslant (1+\delta)Q_j^*; \\ |J| \geqslant j + d; \end{cases}$$

$d \geqslant 0$ и $\delta \geqslant 0$ — параметры.



Чем меньше d и δ, тем сильнее сокращается перебор.

Поиск в глубину (DFS, метод ветвей и границ)

Вход: множество F, критерий Q, параметры d и δ;

процедура *нарастить* $(J \subseteq F)$

  если найдётся j: $j \leqslant |J| - d$ и $Q(J) \geqslant (1+\delta)Q_j^*$, **то**

    набор J бесперспективный; **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 $(J \cup \{f_s\})$;

инициализировать массив лучших значений критерия:

$Q_j^* := Q(\varnothing)$ для всех $j = 1, \ldots, n$;

упорядочить признаки по убыванию информативности;

нарастить $(\varnothing)$;

вернуть J, для которого $Q(J) = \min\limits_{j=1,\ldots,n} Q_j^*$;

Усечённый поиск в ширину (Beam Search)

Поиск пучком (не пучка, не луча, не лучом, не «лучевой поиск»)
Он же поиск в ширину — breadth-first search (BFS)
Он же *многорядный итерационный алгоритм МГУА*
(Метод Группового Учёта Аргументов А.Г.Ивахненко)

Принцип неокончательных решений Габора: оставлять больше свободы 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 пучка поиска*.

Ивахненко А. Г., Юрачковский Ю. П. Моделирование сложных систем по экспериментальным данным, 1987.

Усечённый поиск в ширину (Beam Search)

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

первый ряд состоит из всех наборов длины 1:
$R_1 := \{\{f_1\}, \ldots, \{f_n\}\}$; $Q^* = Q(\varnothing)$;

для $j = 1, \ldots, n$, где j — сложность наборов:

- отсортировать ряд $R_j = \{J_j^1, \ldots, J_j^{B_j}\}$
  по возрастанию критерия: $Q(J_j^1) \leqslant \ldots \leqslant Q(J_j^{B_j})$;
- **если** $B_j > B$ **то**
  - $R_j := \{J_j^1, \ldots, J_j^B\}$ — оставить B лучших наборов ряда;
- **если** $Q(J_j^1) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j^1)$;
- **если** $j - j^* \geqslant d$ **то вернуть** $J_{j^*}^1$;
- породить следующий ряд:
  $R_{j+1} := \{J \cup \{f\} \mid J \in R_j,\ f \in F \setminus J\}$;

Усечённый поиск в ширину: дополнительные эвристики

- **Трудоёмкость:**
  $O(Bn^2)$, точнее $O(Bn(j^* + d))$.
- **Проблема дубликатов:**
  после сортировки $Q\left(J_j^1\right) \leqslant \ldots \leqslant Q\left(J_j^{B_j}\right)$ проверить на совпадение только соседние наборы с равными значениями внутреннего и внешнего критерия.
- **Адаптивный отбор признаков:**
  на последнем шаге добавлять к j-му ряду только признаки f с наибольшей информативностью $I_j(f)$:

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

Эволюционный 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* (crossover) $\beta = \beta' \times \beta''$:

- вариант 1: $\beta_j = \rho_j \beta'_j + (1 - \rho_j)\beta''_j$, $\rho_j \sim \text{uni}(0,1)$
- вариант 2: $\beta = (\beta'_1, \ldots, \beta'_s, \beta''_{s+1}, \ldots, \beta''_n)$, $s \sim \text{uni}(1, \ldots, n)$, надо задавать «естественное» ранжирование признаков

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

- $\beta_j = \rho_j(1 - \beta'_j) + (1 - \rho_j)\beta'_j$, $\rho_j \sim \text{bin}(p_m)$, где p_m — параметр *вероятности мутации*.

Эволюционный (генетический) алгоритм

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

инициализировать случайную популяцию из B наборов:
$B_1 := B$; $R_1 := \left\{J_1^1, \ldots, J_1^{B_1}\right\}$; $Q^* := Q(\varnothing)$;

для $t = 1, \ldots, T$, где t — номер поколения:

- ранжирование индивидов: $Q(J_t^1) \leqslant \ldots \leqslant Q(J_t^{B_t})$;
- **если** $B_t > B$ **то** селекция: $R_t := \left\{J_t^1, \ldots, J_t^B\right\}$;
- **если** $Q(J_t^1) < Q^*$ **то** $t^* := t$; $Q^* := Q(J_t^1)$;
- **если** $t - t^* \geqslant d$ **то вернуть** $J_{t^*}^1$;
- 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).

Обобщение: случайный поиск с адаптацией (СПА)

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

равные вероятности признаков: $p_1 = \cdots = p_n := 1/n$;
инициализировать случайную популяцию из B_1 наборов:
$R_1 := \left\{ J_1^1, \ldots, J_1^{B_1} \sim \{p_1, \ldots, p_n\} \right\}$; $Q^* := Q(\varnothing)$;
для $t = 1, \ldots, T$, где t — номер поколения:
- ранжирование индивидов: $Q(J_t^1) \leqslant \ldots \leqslant Q(J_t^{B_t})$;
- **если** $B_t > B$ **то** селекция: $R_t := \left\{ J_t^1, \ldots, J_t^B \right\}$;
- **если** $Q(J_t^1) < Q^*$ **то** $t^* := t$; $Q^* := Q(J_t^1)$;
- **если** $t - t^* \geqslant d$ **то вернуть** $J_{t^*}^1$;
- увеличить p_j для признаков из лучших наборов;
- уменьшить p_j для признаков из худших наборов;
- породить $t{+}1$-е поколение из B_t наборов:
$R_{t+1} := \left\{ J_{t+1}^1, \ldots, J_{t+1}^{B_t} \sim \{p_1, \ldots, p_n\} \right\} \cup R_t$;

Попытка обоснования. Теорема схемы

Схема — вектор $H = (h_1, \ldots, h_n)$, где $h_j \in \{0, 1, *\}$

$o(H)$ — *порядок схемы*, число не$*$ в схеме

$d(H)$ — *длина схемы*, расстояние между первым и последним не$*$, число мест, в которых кроссовер может нарушить схему

$f(H, t)$ — *степень приспособленности* схемы, среднее Q по всем векторам, подходящим под схему в поколении t

$\bar{f}(t) = f(*^n, t)$ — *средняя приспособленность* популяции

p_c — вероятность кроссовера (только второй вариант)

p_m — вероятность мутации

Теорема схемы [Холланд, 1975]

Число индивидов схемы H в популяции поколения t:

$$\mathrm{E}m(H, t+1) \geqslant m(H, t)\frac{f(H, t)}{\bar{f}(t)}\left(1 - p_c\frac{d(H)}{n-1} - p_m o(H)\right)$$

Интерпретация теоремы схемы

Число индивидов схемы H в популяции поколения t:

$$\mathsf{E}m(H,t+1) \geqslant m(H,t)\frac{f(H,t)}{\bar{f}(t)}\left(1 - p_c\frac{d(H)}{n-1} - p_m o(H)\right)$$

- *Строительный блок* — схема H с низким порядком $o(H)$, короткой длиной $d(H)$, высокой приспособленностью $f(H,t)$
- Если приспособленность строительного блока выше средней в популяции, то число его индивидов будет расти экспоненциально в последующих популяциях
- **Гипотеза** (building block hypothesis): «строительные блоки объединяются, чтобы сформировать ещё лучшие блоки»

John Henry Holland. Adaptation in natural and artificial systems. 1992
David White. An overview of schema theory. 2014

Резюме. Методы отбора признаков

- Для отбора признаков могут использоваться любые эвристические методы дискретной оптимизации

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

- $Q(J)$ должен быть внешним критерием, с характерным минимумом по сложности модели
- Большинство эвристик эксплуатируют две основные идеи:
  — признаки ранжируются по их полезности;
  — $Q(J)$ изменяется не сильно при малом изменении J.
- МГУА, ЭА и СПА очень похожи — на их основе можно изобретать новые «симбиотические» мета-эвристики.