

Оценивание качества классификации

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>

- 1 Оценки качества классификации**
 - Чувствительность, специфичность, ROC, AUC
 - Правдоподобие вероятностной модели классификации
 - Точность, полнота, AUC-PR
- 2 Внешние критерии обобщающей способности**
 - Внутренние и внешние критерии
 - Эмпирические внешние критерии
 - Аналитические внешние критерии
- 3 Методы отбора признаков**
 - Полный перебор
 - Жадные алгоритмы
 - Поиск в ширину и генетический алгоритм

Анализ ошибок классификации

Задача классификации на два класса, $y_i \in \{-1, +1\}$.

Алгоритм классификации $a(x_i) \in \{-1, +1\}$

	ответ классификатора	правильный ответ
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	$a(x_i) = -1$	$y_i = +1$

Доля правильных классификаций (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 ни численность (дисбаланс) классов, ни цена ошибки на объектах разных классов.

Функции потерь, зависящие от штрафов за ошибку

Задача классификации на два класса, $y_i \in \{-1, +1\}$.

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

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:

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

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):

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

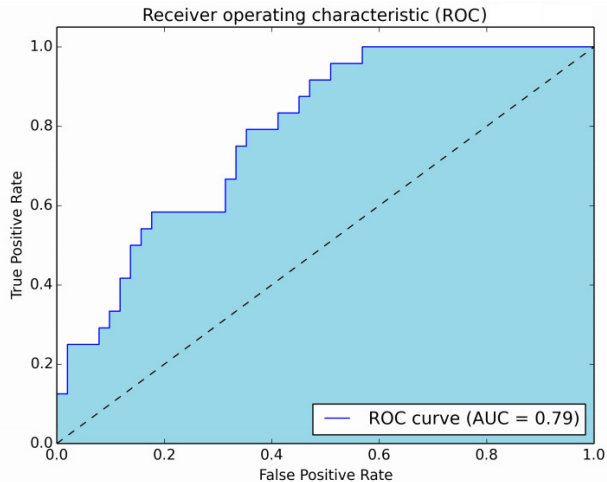
$1 - \text{FPR}(a)$ называется *специфичностью* алгоритма a .

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

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

$\text{TPR}(a)$ называется также *чувствительностью* алгоритма a .

Пример ROC-кривой



Пример из Python scikits learn: <http://scikit-learn.org/dev>

Алгоритм эффективного построения ROC-кривой

Вход: выборка X^ℓ ; дискриминантная функция $g(x, w)$;

Выход: $\{(FPR_i, TPR_i)\}_{i=0}^\ell$, AUC — площадь под ROC-кривой

$\ell_y := \sum_{i=1}^\ell [y_i = y]$, для всех $y \in Y$;

упорядочить выборку X^ℓ по убыванию значений $g(x_i, w)$;

поставить первую точку в начало координат:

$(FPR_0, TPR_0) := (0, 0)$; AUC := 0;

для $i := 1, \dots, \ell$

если $y_i = -1$ **то**

$FPR_i := FPR_{i-1} + \frac{1}{\ell_-}$; $TPR_i := TPR_{i-1}$;

$AUC := AUC + \frac{1}{\ell_-} TPR_i$;

иначе

$FPR_i := FPR_{i-1}$; $TPR_i := TPR_{i-1} + \frac{1}{\ell_+}$;

Градиентная максимизация AUC

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

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

$$\begin{aligned} \text{AUC}(w) &= \frac{1}{\ell_-} \sum_{i=1}^{\ell} [y_i = -1] \text{TPR}_i = \\ &= \frac{1}{\ell_- \ell_+} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} [y_i < y_j] [g(x_i, w) < g(x_j, w)] \rightarrow \max_w. \end{aligned}$$

Явная максимизация аппроксимированного AUC:

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

где $\mathcal{L}(M)$ — убывающая функция отступа,

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

Алгоритм SG для максимизации AUC

Возьмём для простоты линейный классификатор:

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

Вход: выборка X^ℓ , темп обучения h , темп забывания λ ;

Выход: вектор весов w ;

инициализировать веса w_j , $j = 0, \dots, n$;

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

повторять

выбрать **пару объектов** (i, j) : $y_i < y_j$, случайным образом;

вычислить потерю: $\varepsilon_{ij} := \mathcal{L}(M_{ij}(w))$;

сделать градиентный шаг: $w := w - h \mathcal{L}'(M_{ij}(w))(x_j - x_i)$;

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 классификации, $y_i \in \{-1, +1\}$:

$$g(x, w) = P(y = +1|x, w).$$

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

Критерий логарифма правдоподобия (log-loss):

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

Вероятностная модель многоклассовой классификации:

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

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

Оценки качества двухклассовой классификации

В информационном поиске:

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

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

Precision — доля релевантных среди найденных

Recall — доля найденных среди релевантных

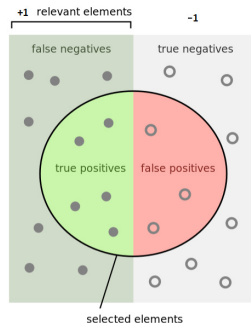
В медицинской диагностике:

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

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

Sensitivity — доля верных положительных диагнозов

Specificity — доля верных отрицательных диагнозов



$$\text{Precision} = \frac{\text{Green}}{\text{Green} + \text{Red}}$$

$$\text{Recall} = \text{Sensitivity} = \frac{\text{Green}}{\text{Green} + \text{Grey}}$$

$$\text{Accuracy} = \frac{\text{Green} + \text{Grey}}{\text{Green} + \text{Red} + \text{Grey}}$$

$$\text{Specificity} = \frac{\text{Grey}}{\text{Grey} + \text{Red}}$$

Точность и полнота многоклассовой классификации

Для каждого класса $y \in Y$:

TP_y — верные положительные

FP_y — ложные положительные

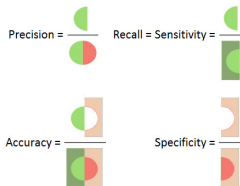
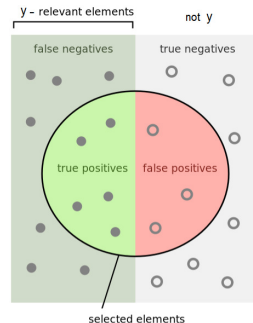
FN_y — ложные отрицательные

Точность и полнота **с микроусреднением**:

$$\text{Precision: } P = \frac{\sum_y TP_y}{\sum_y (TP_y + FP_y)}$$

$$\text{Recall: } R = \frac{\sum_y TP_y}{\sum_y (TP_y + FN_y)}$$

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

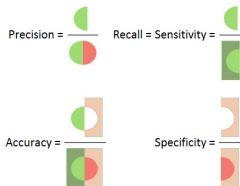
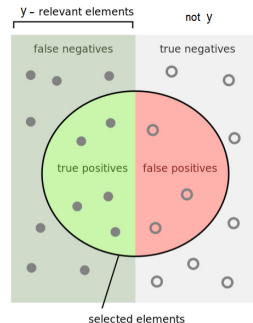
FN_y — ложные отрицательные

Точность и полнота **с макроусреднением**:

$$\text{Precision: } P = \frac{1}{|Y|} \sum_y \frac{TP_y}{TP_y + FP_y};$$

$$\text{Recall: } R = \frac{1}{|Y|} \sum_y \frac{TP_y}{TP_y + FN_y};$$

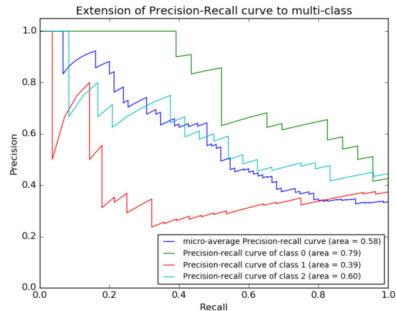
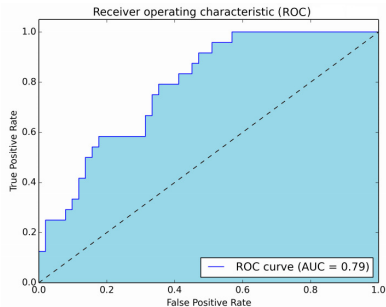
Макроусреднение чувствительно
к ошибкам на малочисленных классах



Кривые ROC и Precision-Recall

Модель классификации: $a(x) = \text{sign}(\langle x, w \rangle - w_0)$

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 .
- $F_\beta = \frac{(1+\beta^2)PR}{\beta^2P+R}$ — F_β -мера: чем больше β , тем важнее R .

Задачи выбора метода обучения

Дано: X — пространство объектов; Y — множество ответов;

$X^\ell = (x_i, y_i)_{i=1}^\ell$ — обучающая выборка, $y_i = y^*(x_i)$;

$A_t = \{a: X \rightarrow Y\}$ — модели алгоритмов, $t \in T$;

$\mu_t: (X \times Y)^\ell \rightarrow A_t$ — 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):
 $F = \{f_j: X \rightarrow D_j: j = 1, \dots, n\}$ — множество признаков;
метод обучения μ_J использует только признаки $J \subseteq F$.

Как оценить качество обучения по прецедентам?

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

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

Внутренний критерий оценивает качество на обучении X^ℓ :

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

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 :

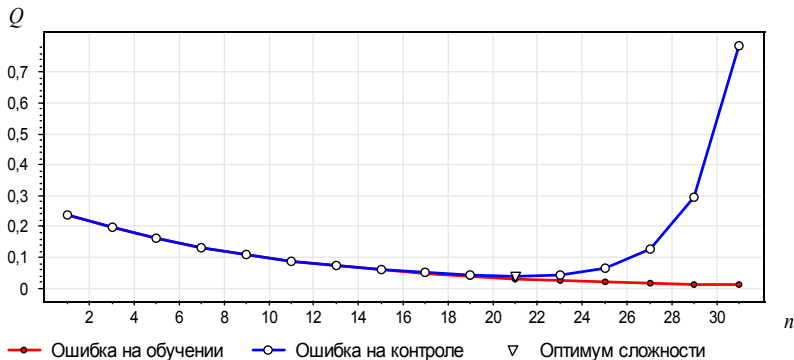
$$Q_\mu(X^\ell, X^k) = Q(\mu(X^\ell), X^k).$$

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 $X^L = X_n^\ell \sqcup X_n^k$, $n = 1, \dots, N$:

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

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.

Скольльзящий контроль и поблочная кросс-проверка

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

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

Недостатки LOO: ресурсоёмкость, высокая дисперсия.

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

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

Недостатки q -fold CV:

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

Множественная поблочная кросс-проверка

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

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

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

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 $X^L \sqcup X^k = X^L$, $\ell = k$:

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

2. Аналог $CV_{t \times 2}$: по t разбиениям $X^L = X_s^\ell \sqcup X_s^k$, $s = 1, \dots, t$:

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

Недостатки:

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

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):

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

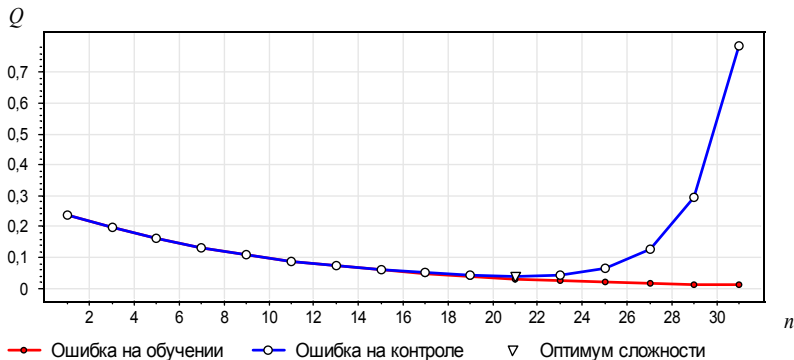
Задача отбора признаков по внешнему критерию

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

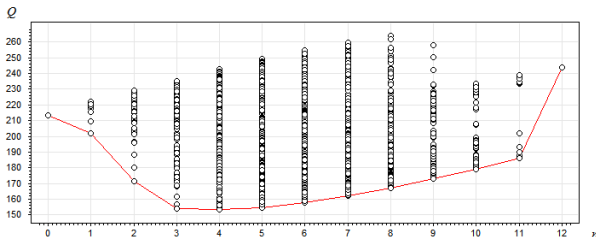
μ_J — метод обучения, использующий только признаки $J \subseteq F$;

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

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



Алгоритм полного перебора (Full Search)

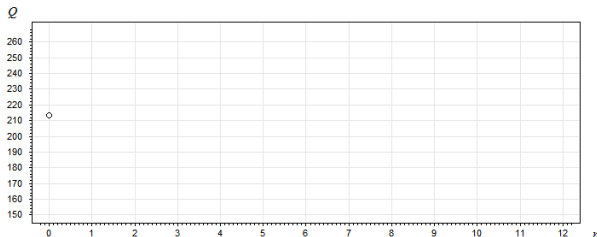


Вход: множество F , критерий Q , параметр d ;

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

$$J_j := \arg \min_{J: |J|=j} Q(J);$$
- 4: **если** $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;
- 5: **если** $j - j^* \geq d$ **то вернуть** J_{j^*} ;

Алгоритм полного перебора (Full Search)



$$d = 3$$

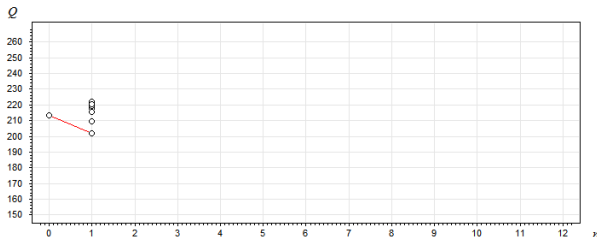
$$j = 0$$

Вход: множество 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 :

$$J_j := \arg \min_{J: |J|=j} Q(J)$$
;
- 4: **если** $Q(J_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j)$;
- 5: **если** $j - j^* \geq d$ **то вернуть** J_{j^*} ;

Алгоритм полного перебора (Full Search)



$$d = 3$$

$$j = 1$$

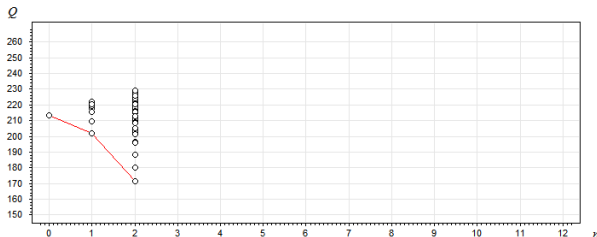
$$j^* = 1$$

Вход: множество F , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, 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** $j - j^* \geq d$ **то вернуть** J_{j^*} ;

Алгоритм полного перебора (Full Search)



$$d = 3$$

$$j = 2$$

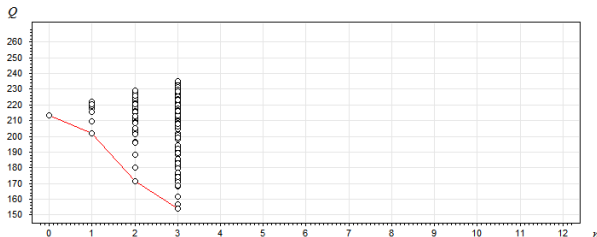
$$j^* = 2$$

Вход: множество F , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, 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** $j - j^* \geq d$ **то вернуть** J_{j^*} ;

Алгоритм полного перебора (Full Search)



$$d = 3$$

$$j = 3$$

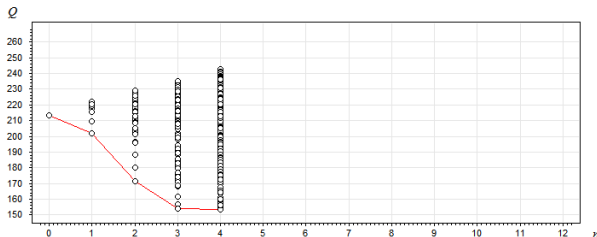
$$j^* = 3$$

Вход: множество F , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, 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** $j - j^* \geq d$ **то вернуть** J_{j^*} ;

Алгоритм полного перебора (Full Search)



$$d = 3$$

$$j = 4$$

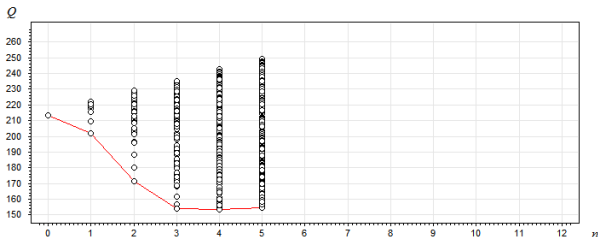
$$j^* = 4$$

Вход: множество F , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: для всех $j = 1, \dots, n$, 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 $j - j^* \geq d$ то вернуть J_{j^*} ;

Алгоритм полного перебора (Full Search)



$$d = 3$$

$$j = 5$$

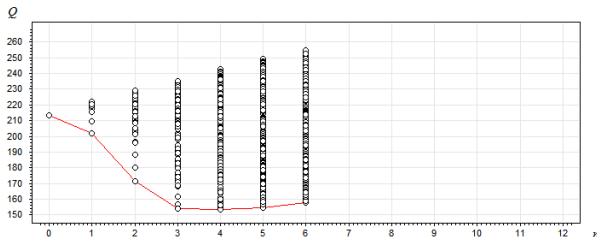
$$j^* = 4$$

Вход: множество F , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, 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** $j - j^* \geq d$ **то вернуть** J_{j^*} ;

Алгоритм полного перебора (Full Search)



$$d = 3$$

$$j = 6$$

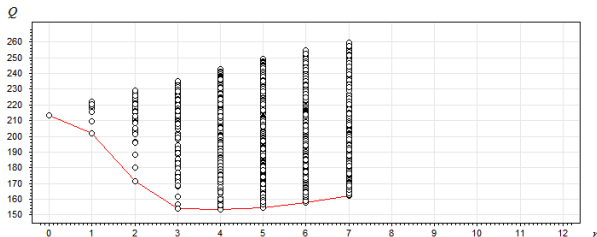
$$j^* = 4$$

Вход: множество F , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, 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** $j - j^* \geq d$ **то вернуть** J_{j^*} ;

Алгоритм полного перебора (Full Search)



$$d = 3$$

$$j = 7$$

$$j^* = 4$$

Вход: множество F , критерий Q , параметр d ;

- 1: $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, 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** $j - j^* \geq d$ **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 оооооочень 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: $J_0 := \emptyset$; $Q^* := Q(\emptyset)$; — инициализация;
- 2: **для всех** $j = 1, \dots, n$, где j — сложность наборов:
- 3: найти признак, наиболее выгодный для добавления:
$$f^* := \arg \min_{f \in F \setminus J_{j-1}} Q(J_{j-1} \cup \{f\});$$
- 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** $j - j^* \geq d$ **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:

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

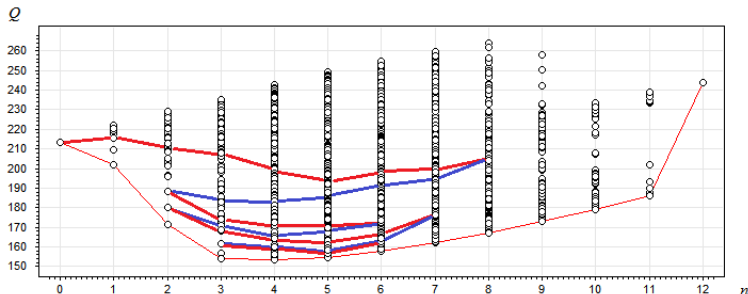
Алгоритм поочередного добавления и удаления (Add-Del)

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)

1: $J_0 := \emptyset$; $Q^* := Q(\emptyset)$; $t := 0$; — инициализация;

2: **повторять**

3: **пока** $|J_t| < n$ добавлять признаки (Add):

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

5: $f^* := \arg \min_{f \in F \setminus J_{t-1}} Q(J_{t-1} \cup \{f\})$; $J_t := J_{t-1} \cup \{f^*\}$;

6: **если** $Q(J_t) < Q^*$ **то** $t^* := t$; $Q^* := Q(J_t)$;

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

8: **пока** $|J_t| > 0$ удалять признаки (Del):

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

10: $f^* := \arg \min_{f \in J_{t-1}} Q(J_{t-1} \setminus \{f\})$; $J_t := J_{t-1} \setminus \{f^*\}$;

11: **если** $Q(J_t) < Q^*$ **то** $t^* := t$; $Q^* := Q(J_t)$;

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

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

14: **вернуть** J_{t^*} ;

Поиск в ширину (breadth-first search, BFS)

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

Философия — 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*:

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

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

Поиск в ширину (breadth-first search, BFS)

Вход: множество 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 $R_j = \{J_j^1, \dots, J_j^{B_j}\}$

по возрастанию критерия: $Q(J_j^1) \leq \dots \leq Q(J_j^{B_j})$;

4: **если** $B_j > B$ **то**

5: $R_j := \{J_j^1, \dots, J_j^B\}$; — B лучших наборов ряда;

6: **если** $Q(J_j^1) < Q^*$ **то** $j^* := j$; $Q^* := Q(J_j^1)$;

7: **если** $j - j^* \geq d$ **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: дополнительные эвристики

- **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 $I_j(f)$:

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

Эволюционный алгоритм поиска (идея и терминология)

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

$R_t := \{J_t^1, \dots, J_t^{B_t}\}$ — 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.

Эволюционный (генетический) алгоритм

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

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

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).

Обобщение: случайный поиск с адаптацией (СПА)

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

- 1: **равные вероятности признаков:** $p_1 = \dots = p_n := 1/n$;
- 2: инициализировать случайную популяцию из B_1 наборов:
 $R_1 := \{J_1^1, \dots, J_1^{B_1} \sim \{p_1, \dots, p_n\}\}$; $Q^* := Q(\emptyset)$;
- 3: **для всех** $t = 1, \dots, T$, где t — номер поколения:
- 4: ранжирование индивидов: $Q(J_t^1) \leq \dots \leq Q(J_t^{B_t})$;
- 5: **если** $B_t > B$ **то** селекция: $R_t := \{J_t^1, \dots, J_t^B\}$;
- 6: **если** $Q(J_t^1) < Q^*$ **то** $t^* := t$; $Q^* := Q(J_t^1)$;
- 7: **если** $t - t^* \geq d$ **то вернуть** $J_{t^*}^1$;
- 8: **увеличить** p_j **для признаков из лучших наборов;**
- 9: **уменьшить** p_j **для признаков из худших наборов;**
- 10: породить $t+1$ -е поколение из B_t наборов:
 $R_{t+1} := \{J_{t+1}^1, \dots, J_{t+1}^{B_t} \sim \{p_1, \dots, p_n\}\} \cup R_t$;

- Для отбора признаков могут использоваться любые эвристические методы дискретной оптимизации

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

- $Q(J)$ 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» алгоритмы.