Обобщающая способность
Методы отбора признаков

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

20 февраля 2018 г.

Содержание

Задача статистического (машинного) обучения с учителем

Задача восстановления зависимости $y\colon X \to Y$
по точкам *обучающей выборки* (x_i, y_i), $i = 1, \ldots, \ell$.

Дано: $x_i \mapsto \big(f_1(x_i), \ldots, f_n(x_i)\big)$ — объекты обучающей выборки,
$y_i = y(x_i)$ — правильные ответы, $i = 1, \ldots, \ell$:

$$\begin{pmatrix} f_1(x_1) & \ldots & f_n(x_1) \\ \ldots & \ldots & \ldots \\ f_1(x_\ell) & \ldots & f_n(x_\ell) \end{pmatrix} \xrightarrow{y} \begin{pmatrix} y_1 \\ \ldots \\ y_\ell \end{pmatrix}$$

Найти: функцию $a(x)$, способную давать правильные ответы
на *тестовых объектах* $\tilde{x}_i \mapsto \big(f_1(\tilde{x}_i), \ldots, f_n(\tilde{x}_i)\big)$, $i = 1, \ldots, k$:

$$\begin{pmatrix} f_1(\tilde{x}_1) & \ldots & f_n(\tilde{x}_1) \\ \ldots & \ldots & \ldots \\ f_1(\tilde{x}_k) & \ldots & f_n(\tilde{x}_k) \end{pmatrix} \xrightarrow{a?} \begin{pmatrix} a(\tilde{x}_1) \\ \ldots \\ a(\tilde{x}_k) \end{pmatrix}$$

Типы признаков и типы задач

Типы признаков, $f_j : X \to D_j$, в зависимости от D_j:

- $D_j = \{0, 1\}$ — *бинарный* признак;
- $|D_j| < \infty$ — *номинальный* признак;
- D_j упорядочено — *порядковый* признак;
- $D_j = \mathbb{R}$ — *количественный* признак.

Типы задач, в зависимости от Y:

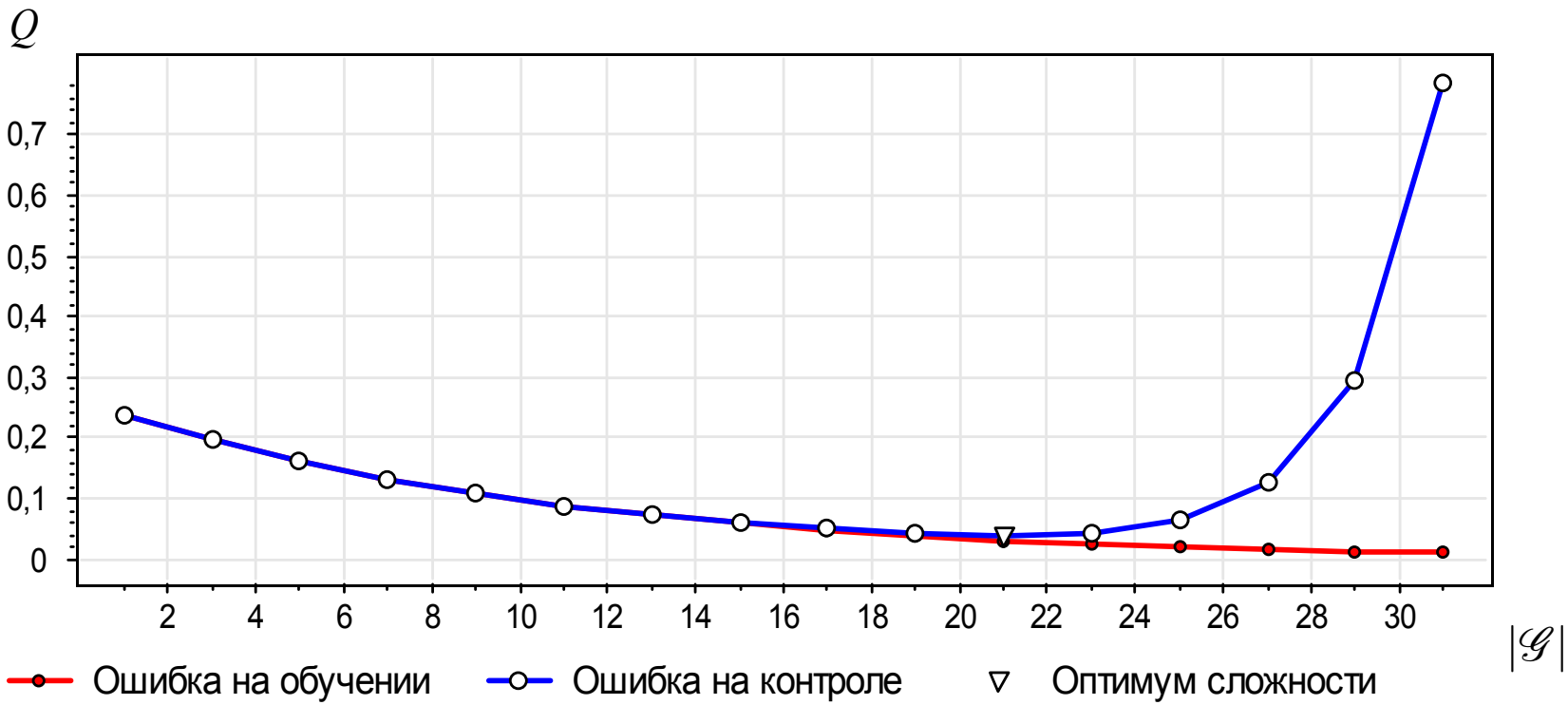
- $Y = \{0, 1\}$ или $Y = \{-1, +1\}$ — *классификация* на 2 класса;
- $Y = \{1, \ldots, M\}$ — на M непересекающихся классов;
- $Y = \{0, 1\}^M$ — на M классов, которые могут пересекаться;
- $Y = \mathbb{R}$ — задача восстановления регрессии;
- Y упорядочено — задача ранжирования (learning to rank).

Задача отбора признаков

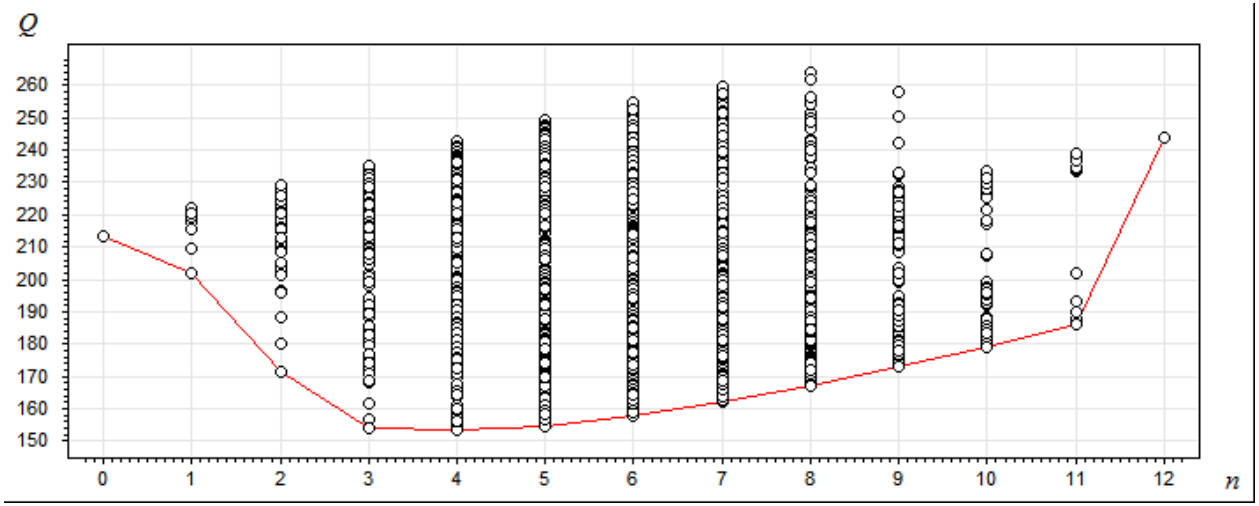
$\mathscr{F} = \{f_j : X \to D_j : j = 1, \ldots, n\}$ — множество признаков;

$\mu_{\mathscr{G}}$ — метод обучения, использующий только признаки $\mathscr{G} \subseteq \mathscr{F}$;

$Q(\mathscr{G}) = Q(\mu_{\mathscr{G}}, X^{\ell})$ — какой-либо критерий.



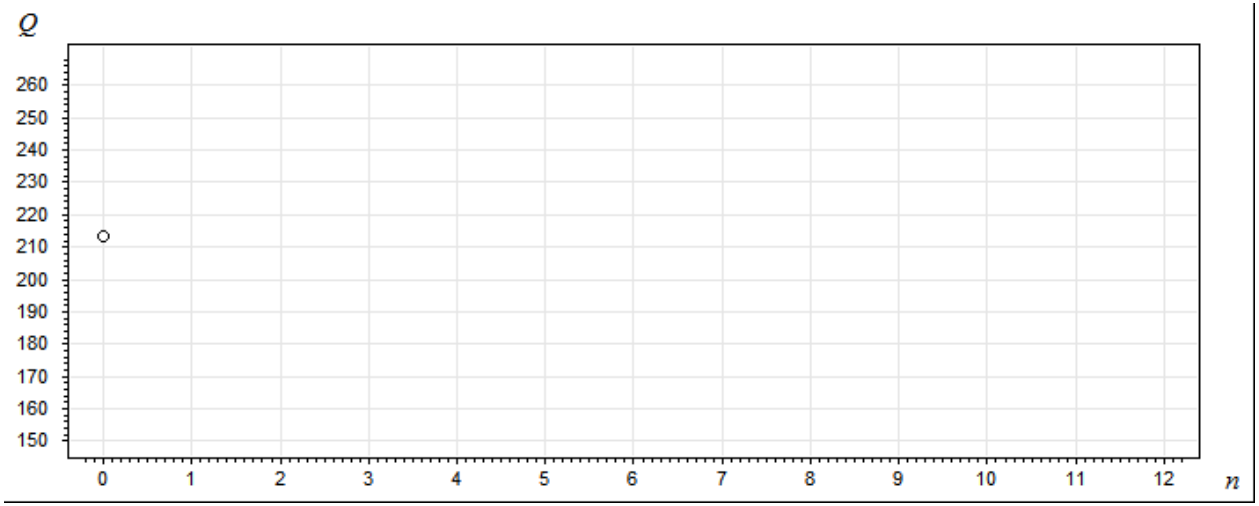
Алгоритм полного перебора (Full Search)



Вход: множество $\mathscr{F}$, 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:
$\mathscr{G}_j := \arg\min_{\mathscr{G}:\,|\mathscr{G}|=j} Q(\mathscr{G})$;
4: **если** $Q(\mathscr{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathscr{G}_j)$;
5: **если** $j - j^* \geqslant d$ **то вернуть** $\mathscr{G}_{j^*}$;

Алгоритм полного перебора (Full Search)

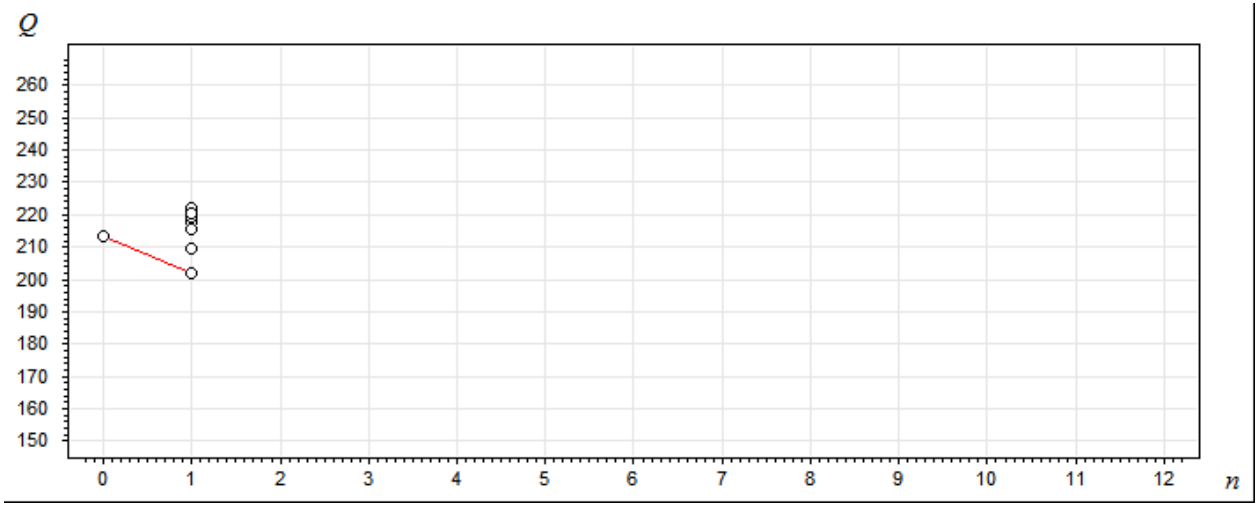


$j = 0$

Вход: множество $\mathscr{F}$, 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:
$\mathscr{G}_j := \arg\min_{\mathscr{G}:\,|\mathscr{G}|=j} Q(\mathscr{G})$;
4: **если** $Q(\mathscr{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathscr{G}_j)$;
5: **если** $j - j^* \geqslant d$ **то вернуть** $\mathscr{G}_{j^*}$;

Алгоритм полного перебора (Full Search)

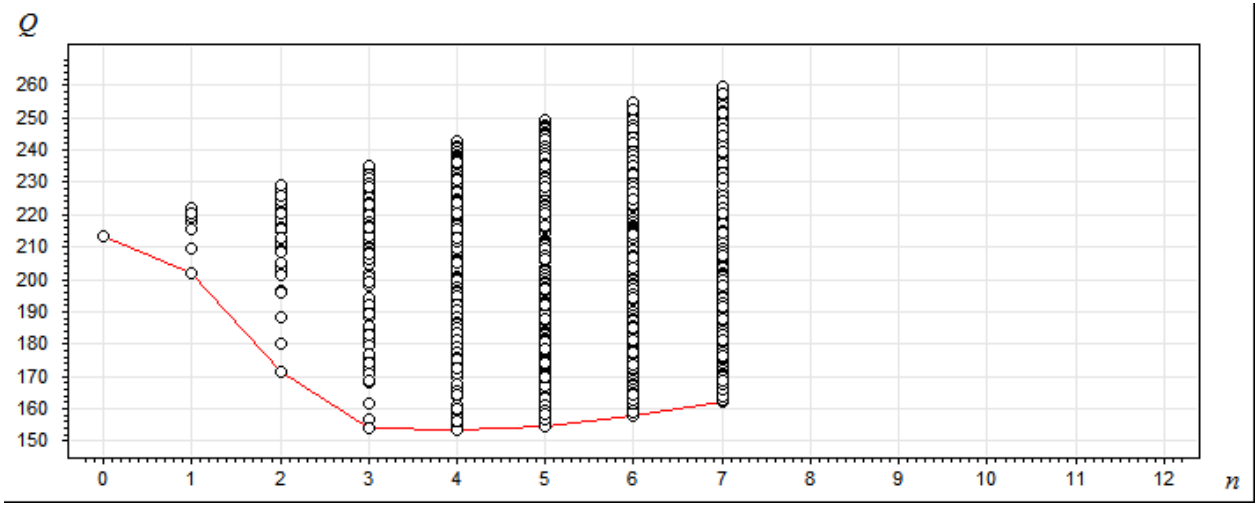


$j = 1$
$j^* = 1$

Вход: множество $\mathscr{F}$, 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:
$\mathscr{G}_j := \arg\min_{\mathscr{G}:\,|\mathscr{G}|=j} Q(\mathscr{G})$;
4: **если** $Q(\mathscr{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathscr{G}_j)$;
5: **если** $j - j^* \geqslant d$ **то вернуть** $\mathscr{G}_{j^*}$;

Алгоритм полного перебора (Full Search)



$j = 7$
$j^* = 4$

Вход: множество $\mathscr{F}$, 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:
$\mathscr{G}_j := \arg\min_{\mathscr{G}:\,|\mathscr{G}|=j} Q(\mathscr{G})$;
4: **если** $Q(\mathscr{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathscr{G}_j)$;
5: **если** $j - j^* \geqslant d$ **то вернуть** $\mathscr{G}_{j^*}$;

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 $\mathscr{F}$, критерий Q, параметр d;

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

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: $\mathscr{G}_0 := \varnothing$; $Q^* := Q(\varnothing)$; $t := 0$; — инициализация;
2: **повторять**

3: **пока** $|\mathscr{G}_t| < n$ добавлять признаки (Add):
4: $t := t + 1$; — началась следующая итерация;
5: $f^* := \arg\min\limits_{f \in \mathscr{F} \setminus \mathscr{G}_{t-1}} Q\big(\mathscr{G}_{t-1} \cup \{f\}\big)$; $\mathscr{G}_t := \mathscr{G}_{t-1} \cup \{f^*\}$;
6: **если** $Q(\mathscr{G}_t) < Q^*$ **то** $t^* := t$; $Q^* := Q(\mathscr{G}_t)$;
7: **если** $t - t^* \geqslant d$ **то прервать цикл**;

8: **пока** $|\mathscr{G}_t| > 0$ удалять признаки (Del):
9: $t := t + 1$; — началась следующая итерация;
10: $f^* := \arg\min\limits_{f \in \mathscr{G}_{t-1}} Q\big(\mathscr{G}_{t-1} \setminus \{f\}\big)$; $\mathscr{G}_t := \mathscr{G}_{t-1} \setminus \{f^*\}$;
11: **если** $Q(\mathscr{G}_t) < Q^*$ **то** $t^* := t$; $Q^* := Q(\mathscr{G}_t)$;
12: **если** $t - t^* \geqslant d$ **то прервать цикл**;

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

Алгоритм поочерёдного добавления и удаления (Add-Del)

Преимущества:

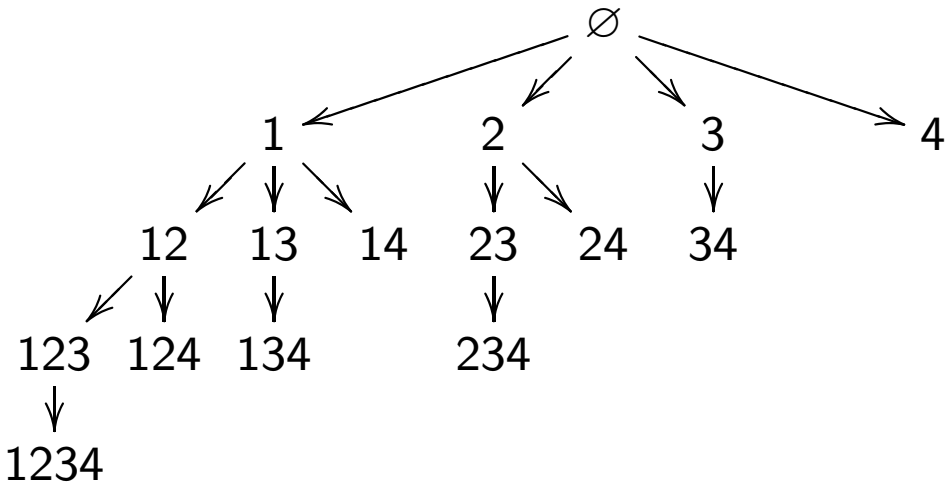
- как правило, лучше, чем Add и Del по отдельности;
- возможны быстрые инкрементные алгоритмы, пример — *шаговая регрессия* (step-wise regression).

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 $\mathscr{G}$ 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 $\mathscr{G}$:
набор $\mathscr{G}$ не наращивается, если

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

$d \geqslant 0$ — 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 $\mathscr{F}$, критерий Q, параметры d и $\varkappa$;

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

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

Поиск в ширину (BFS)

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

Философский 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*:

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

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

Поиск в ширину (BFS)

Вход: множество $\mathscr{F}$, 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 $R_j = \{\mathscr{G}_j^1, \ldots, \mathscr{G}_j^{B_j}\}$
по возрастанию критерия: $Q(\mathscr{G}_j^1) \leqslant \ldots \leqslant Q(\mathscr{G}_j^{B_j})$;
4: **если** $B_j > B$ **то**
5: $R_j := \{\mathscr{G}_j^1, \ldots, \mathscr{G}_j^B\}$; — B лучших наборов ряда;
6: **если** $Q(\mathscr{G}_j^1) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathscr{G}_j^1)$;
7: **если** $j - j^* \geqslant d$ **то вернуть** $\mathscr{G}_{j^*}^1$;
8: породить следующий ряд:
$R_{j+1} := \{\mathscr{G} \cup \{f\} \mid \mathscr{G} \in R_j,\ f \in \mathscr{F} \setminus \mathscr{G}\}$;

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 $I_j(f)$:

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

Генетический алгоритм поиска (идея и терминология)

$\mathscr{G} \subseteq \mathscr{F}$ — *индивид* (в МГУА «модель»);
$R_t := \{\mathscr{G}_t^1, \ldots, \mathscr{G}_t^{B_t}\}$ — *поколение* (в МГУА — «ряд»);

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

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 $\mathscr{F}$, критерий Q, параметры: d, p_m,
B — размер популяции, T — число поколений;

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

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 теории;
- подбор параметров — непростое искусство;

Случайный поиск — упрощенный генетический алгоритм

Модификация: шаг 8

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

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 $\mathscr{F}$, 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 $\{p_1, \dots, p_n\}$:
    $R_{jt} := \{\mathscr{G}_{jt}^1, \dots, \mathscr{G}_{jt}^r\}$, $|\mathscr{G}_{jt}^1| = \dots = |\mathscr{G}_{jt}^r| = j$;
    $\mathscr{G}_{jt}^{\min} := \arg\min\limits_{\mathscr{G} \in R_{jt}} Q(\mathscr{G})$; — лучший из r наборов;
    $\mathscr{G}_{jt}^{\max} := \arg\max\limits_{\mathscr{G} \in R_{jt}} Q(\mathscr{G})$; — худший из r наборов;
    $H := 0$; наказание для всех $f_s \in \mathscr{G}_{jt}^{\max}$:
    $\Delta p_s := \min\{p_s, h\}$; $p_s := p_s - \Delta p_s$; $H := H + \Delta p_s$;
    поощрение для всех $f_s \in \mathscr{G}_{jt}^{\min}$: $p_s := p_s + H/j$;
  $\mathscr{G}_j := \arg\min\limits_{\mathscr{G} \in R_{j1}, \dots, R_{jT}} Q(\mathscr{G})$; — лучший набор сложности j;
  если $Q(\mathscr{G}_j) < Q^*$ **то** $j^* := j$; $Q^* := Q(\mathscr{G}_j)$;
  если $j - j^* \geqslant d$ **то вернуть** $\mathscr{G}_{j^*}$;

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;

$h \approx \frac{1}{r\,n}$ — 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.

Обучение регрессии — это оптимизация

Задача регрессии, $Y = \mathbb{R}$

1. Выбираем *модель регрессии*, например, линейную:
$$a(x, w) = \langle x, w \rangle = \sum_{j=1}^{n} f_j(x) w_j, \qquad x, w \in \mathbb{R}^n$$
2. Выбираем функцию потерь (например, квадратичную):
$$\mathscr{L}(a, y) = (a - y)^2$$
3. Минимизируем потери *методом наименьших квадратов*:
$$Q(w) = \frac{1}{\ell} \sum_{i=1}^{\ell} \big(a(x_i, w) - y_i\big)^2 \to \min_w$$
4. Проверяем прогностическую (обобщающую) способность:
$$\tilde{Q}(w) = \frac{1}{k} \sum_{i=1}^{k} \big(a(\tilde{x}_i, w) - \tilde{y}_i\big)^2$$

Loss Functions of dotted forecasts

Mean squared error (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 .$$

Mean absolute error (MAE):

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| .$$

Mean absolute percentage error(MAPE):

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right| .$$

Symmetric mean absolute percentage error (SMAPE):

$$SMAPE = \frac{200}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{\hat{y}_i + y_i}\right| .$$

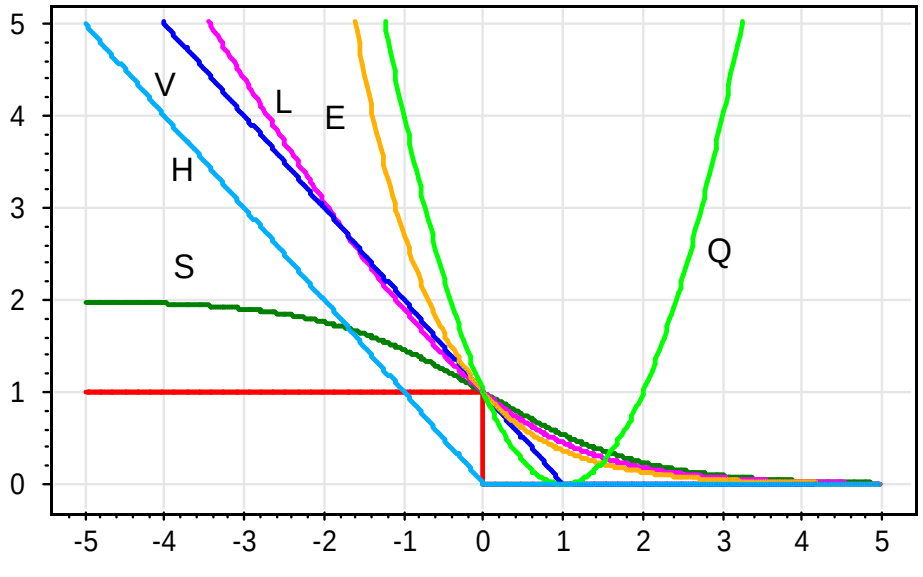
Обучение классификации — тоже оптимизация

Задача классификации, $Y = \{-1, +1\}$

1. Выбираем *модель классификации*, например, линейную:
$$a(x, w) = \text{sign}\langle x, w\rangle$$
2. Выбираем функцию потерь (бинарную или её аппроксимацию):
$$\mathscr{L}(a, y) = \big[\langle x_i, w\rangle y_i < 0\big] \leqslant \mathscr{L}\big(\langle x_i, w\rangle y_i\big)$$
3. Минимизируем *частоту ошибок* на обучающей выборке:
$$Q(w) = \frac{1}{\ell}\sum_{i=1}^{\ell}\big[a(x_i, w)y_i < 0\big] \leqslant \frac{1}{\ell}\sum_{i=1}^{\ell}\mathscr{L}\big(\langle x_i, w\rangle y_i\big) \to \min_w$$
4. Проверяем прогностическую (обобщающую) способность:
$$\tilde{Q}(w) = \frac{1}{k}\sum_{i=1}^{k}\big[\langle \tilde{x}_i, w\rangle \tilde{y}_i < 0\big]$$

Непрерывные аппроксимации пороговой функции потерь

Часто используемые непрерывные функции потерь $\mathscr{L}(M)$:



$V(M) = (1 - M)_+$ — кусочно-линейная (SVM);
$H(M) = (-M)_+$ — кусочно-линейная (Hebb's rule);
$L(M) = \log_2(1 + e^{-M})$ — логарифмическая (LR);
$Q(M) = (1 - M)^2$ — квадратичная (FLD);
$S(M) = 2(1 + e^{M})^{-1}$ — сигмоидная (ANN);
$E(M) = e^{-M}$ — экспоненциальная (AdaBoost);
$[M < 0]$ — пороговая функция потерь.

Балансировка ошибок I и II рода на примере линейного алгоритма

Задача классификации на два класса, $Y = \{-1, +1\}$;
λ_y — штраф за ошибку на объекте класса y;
модель классификации: $a(x, w, w_0) = \text{sign}\big(f(x, w) - w_0\big)$.

- $f(x, w) = \langle x, w \rangle$ — не зависит от $\{\lambda_y\}$;
- $w_0 = \ln \frac{\lambda_-}{\lambda_+}$ — зависит только от $\{\lambda_y\}$.

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

Постановка задачи

- Нужен удобный способ выбора w_0 в зависимости от $\{\lambda_y\}$, не требующий построения w заново.
- Нужна характеристика качества классификатора, инвариантная относительно выбора $\{\lambda_y\}$.

Определение ROC-кривой

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):

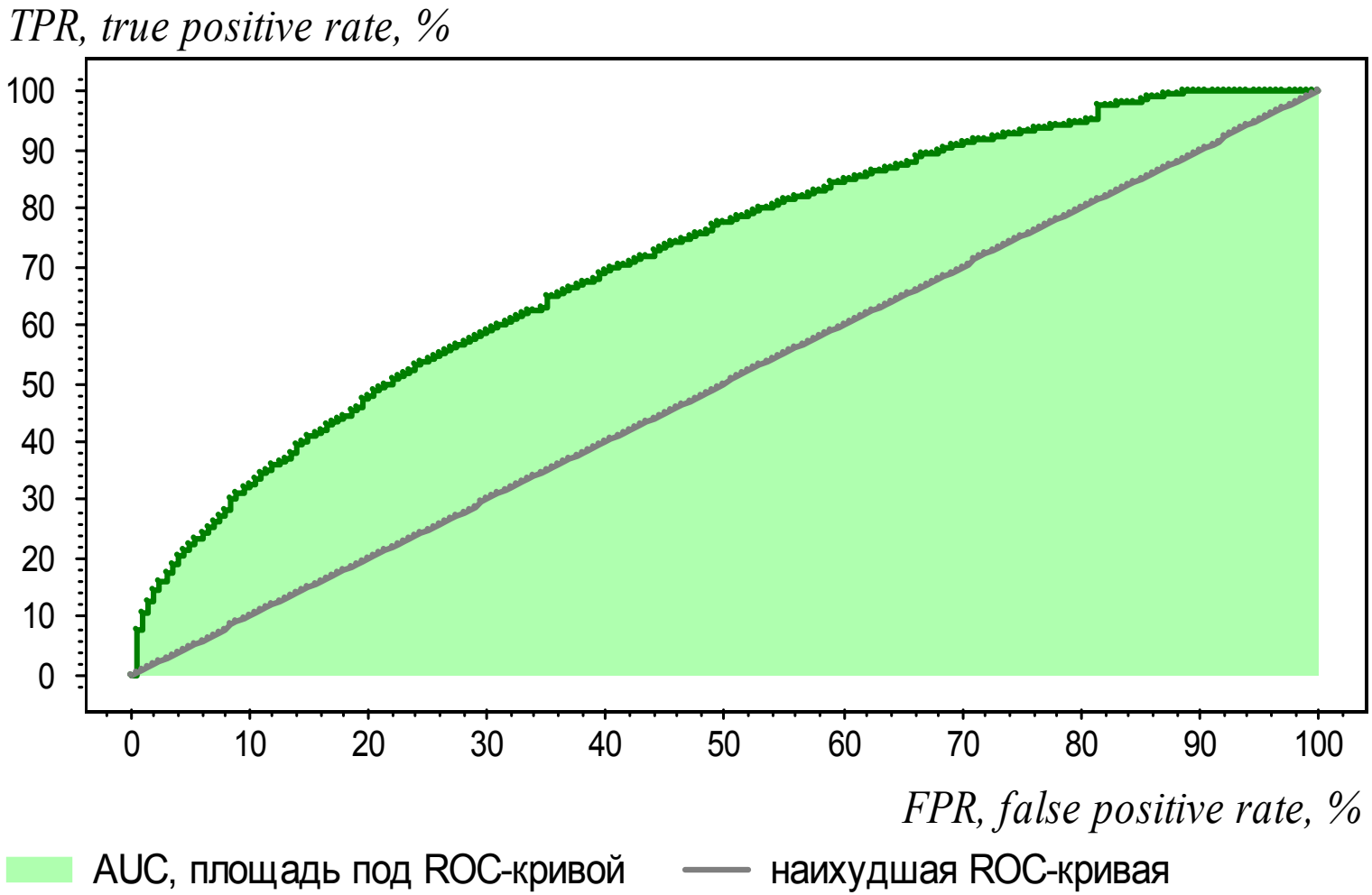
$$\text{FPR}(a, X^\ell) = \frac{\sum_{i=1}^{\ell} [y_i = -1][a(x_i; w, w_0) = +1]}{\sum_{i=1}^{\ell} [y_i = -1]};$$

  $1 - \text{FPR}(a)$ 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



Алгоритм эффективного построения ROC-кривой

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

Выход: $\{(\mathrm{FPR}_i, \mathrm{TPR}_i)\}_{i=0}^{\ell}$, 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 $f(x_i, w)$;

поставить первую точку в начало координат:
$(\mathrm{FPR}_0, \mathrm{TPR}_0) := (0, 0)$; $\mathrm{AUC} := 0$;

для $i := 1, \ldots, \ell$

  если $y_i = -1$ **то** — сместиться на один шаг вправо:

    $\mathrm{FPR}_i := \mathrm{FPR}_{i-1} + \frac{1}{\ell_-}$; $\mathrm{TPR}_i := \mathrm{TPR}_{i-1}$;

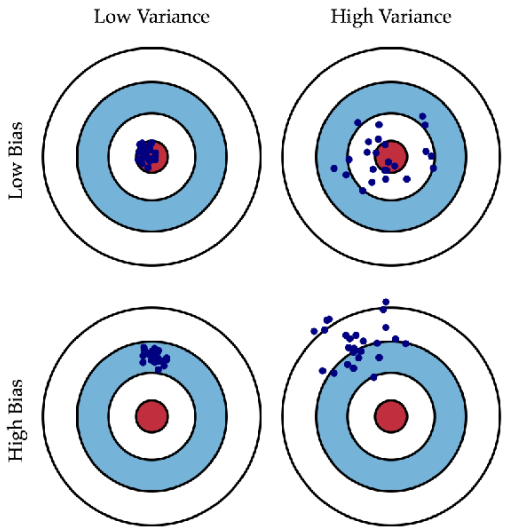
    $\mathrm{AUC} := \mathrm{AUC} + \frac{1}{\ell_-}\mathrm{TPR}_i$;

  иначе — сместиться на один шаг вверх:

    $\mathrm{FPR}_i := \mathrm{FPR}_{i-1}$; $\mathrm{TPR}_i := \mathrm{TPR}_{i-1} + \frac{1}{\ell_+}$;

Разложение ошибки на смещение и разброс

- **шум** — ошибка идеального алгоритма
- **смещение (bias)** — отклонение среднего ответа обученного алгоритма от ответа идеального алгоритма
- **разброс (variance)** — разброс ответов обученных алгоритмов относительно среднего ответа



Квадратичная функция потерь

Задача регрессии: $Y = \mathbb{R}$
Квадратичная функция потерь: $L(y, a) = \big(a(x) - y\big)^2$
Вероятностная постановка: $X^\ell = (x_i, y_i)_{i=1}^\ell \sim p(x, y)$
Метод обучения: $\mu\colon 2^X \to A$, т.е. выборка $\to$ алгоритм

Среднеквадратичный риск:

$$R(a) = \mathsf{E}_{x,y}(a(x) - y)^2 = \int_X \int_Y \big(a(x) - y\big)^2 p(x, y)\, dx\, dy$$

Минимум среднеквадратичного риска, «недостижимый идеал»:

$$a^*(x) = \mathsf{E}(y|x) = \int_Y y\, p(y|x)\, dx$$

Основная мера качества метода обучения μ:

$$Q(\mu) = \mathsf{E}_{X^\ell} \mathsf{E}_{x,y} \big(\mu(X^\ell)(x) - y\big)^2$$

Разложение ошибки на шум, вариацию и смещение

Теорема

В случае квадратичной функции потерь для любого μ

$$Q(\mu) = \underbrace{\mathsf{E}_{x,y}\big(a^*(x) - y\big)^2}_{\text{шум (noise)}} + \underbrace{\mathsf{E}_{x,y}\big(\bar{a}(x) - a^*(x)\big)^2}_{\text{смещение (bias)}} + \underbrace{\mathsf{E}_{x,y}\mathsf{E}_{X^\ell}\big(\mu(X^\ell)(x) - \bar{a}(x)\big)^2}_{\text{разброс (variance)}},$$

$\bar{a}(x) = \mathsf{E}_{X^\ell}\big(\mu(X^\ell)(x)\big)$ — средний ответ обученного алгоритма

Метод k ближайших соседей

Вероятностная модель данных: $p(y|x) = f(x) + \mathcal{N}(0, \sigma^2)$
Метод k ближайших соседей:

$$a(x) = \frac{1}{k} \sum_{j=1}^{k} y(x^{(j)}),$$

где $x^{(j)}$ — j-й сосед объекта x
$a^*(x) = f(x)$ — истинная зависимость
$\bar{a}(x) = \frac{1}{k} \sum_{j=1}^{k} f(x^{(j)})$ — средний ответ

Разложение bias-variance:

$$Q(\mu) = \underbrace{\sigma^2}_{\text{шум}} + \underbrace{\mathrm{E}_{x,y}\Big(\bar{a}(x) - f(x)\Big)^2}_{\text{смещение}} + \underbrace{\tfrac{1}{k}\sigma^2}_{\text{разброс}}$$