Методы понижения размерности

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

23 апреля 2018

Содержание

Задачи выбора признаков, модели, 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:

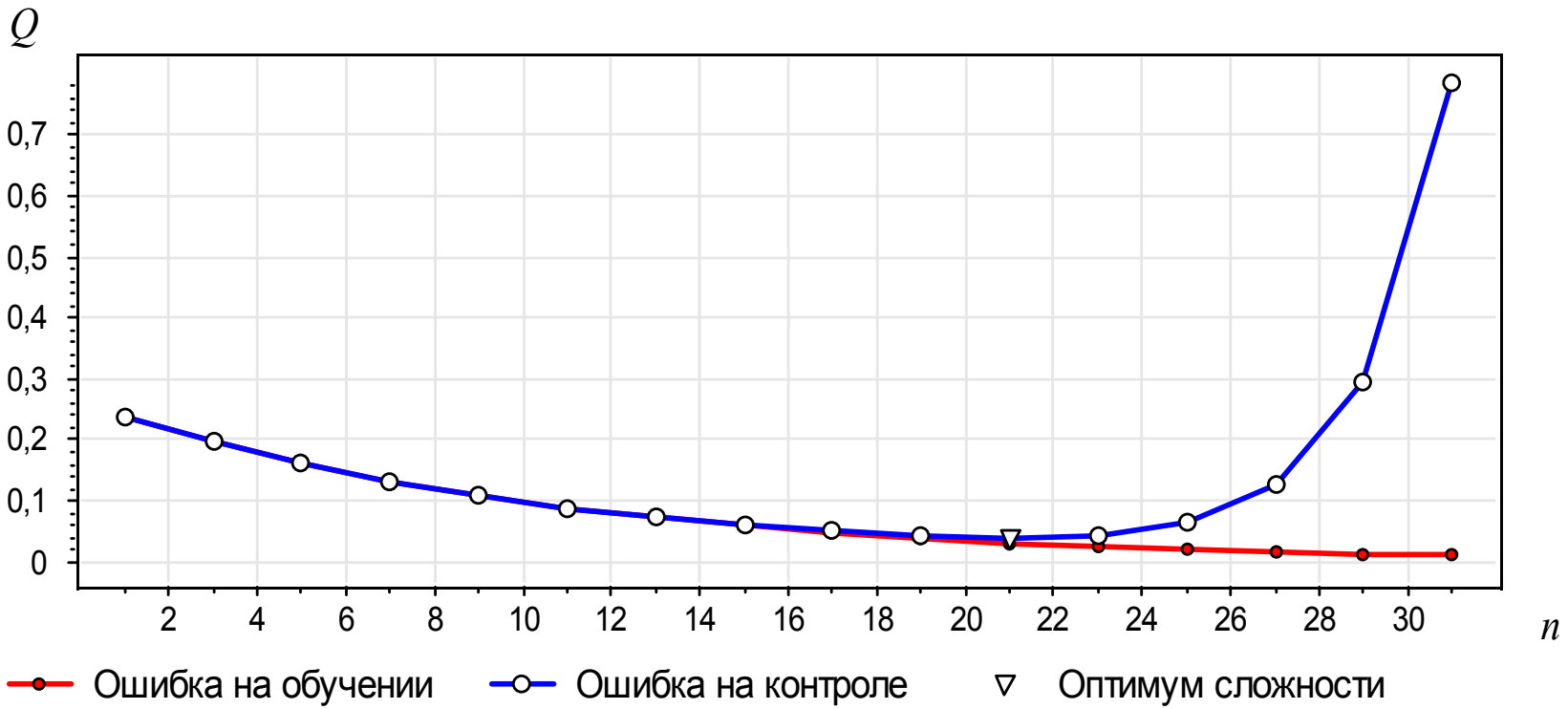
$$Q_\mu(X^\ell, X^k) = Q\big(\mu(X^\ell), X^k\big).$$

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.

Скользящий 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 $X^L = X_1^{\ell_1} \sqcup \cdots \sqcup X_q^{\ell_q}$ на q блоков (почти) равной длины,

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

Недостатки q-fold CV:

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

Многократная поблочная кросс-проверка

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.

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 $\eta = 0.05$.

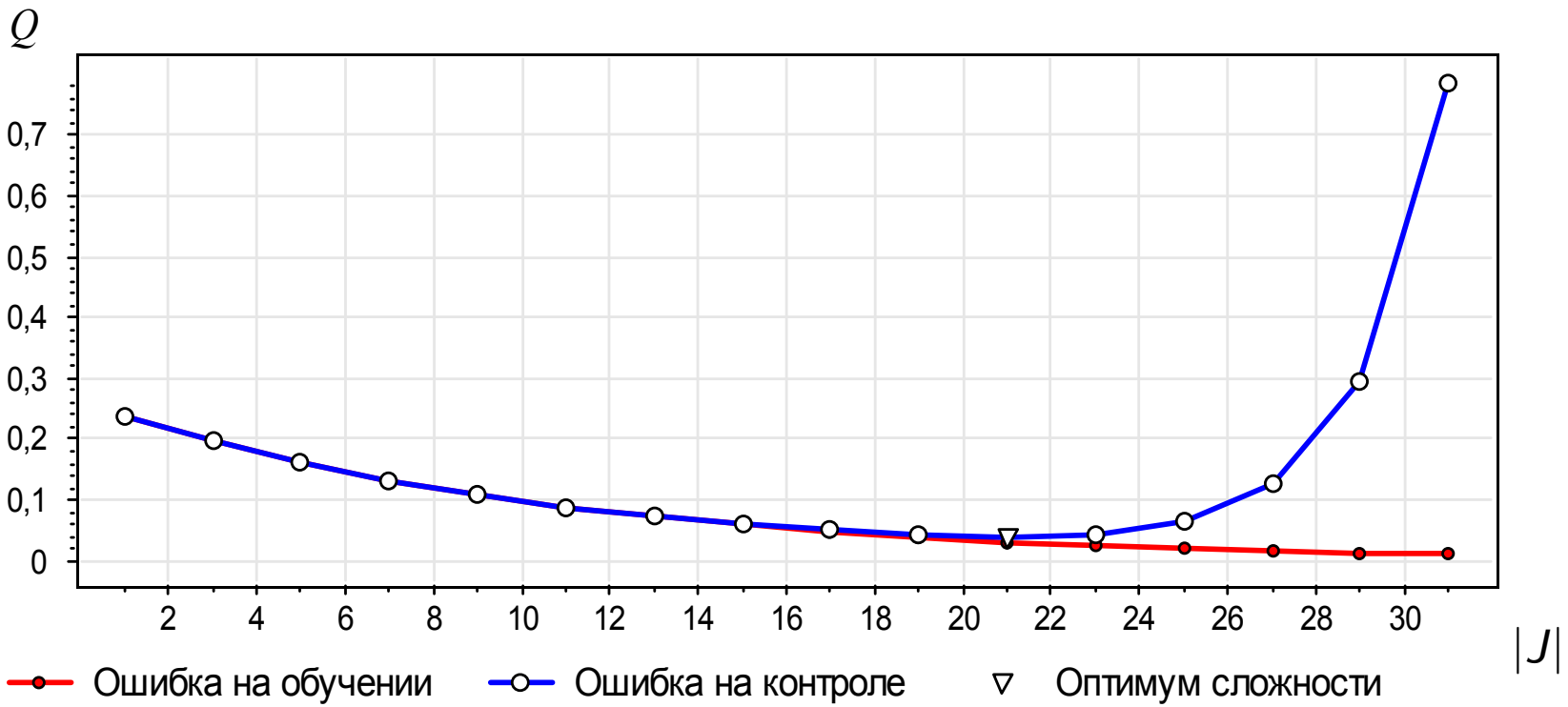
Задача отбора признаков по внешнему критерию

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

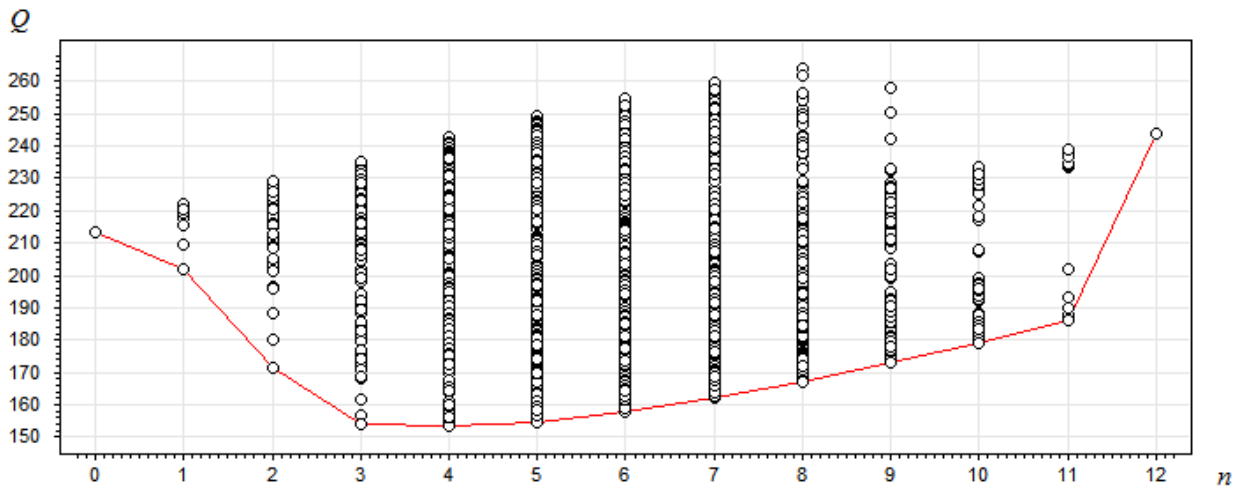
μ_J — метод обучения, использующий только признаки $J \subseteq F$;

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

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



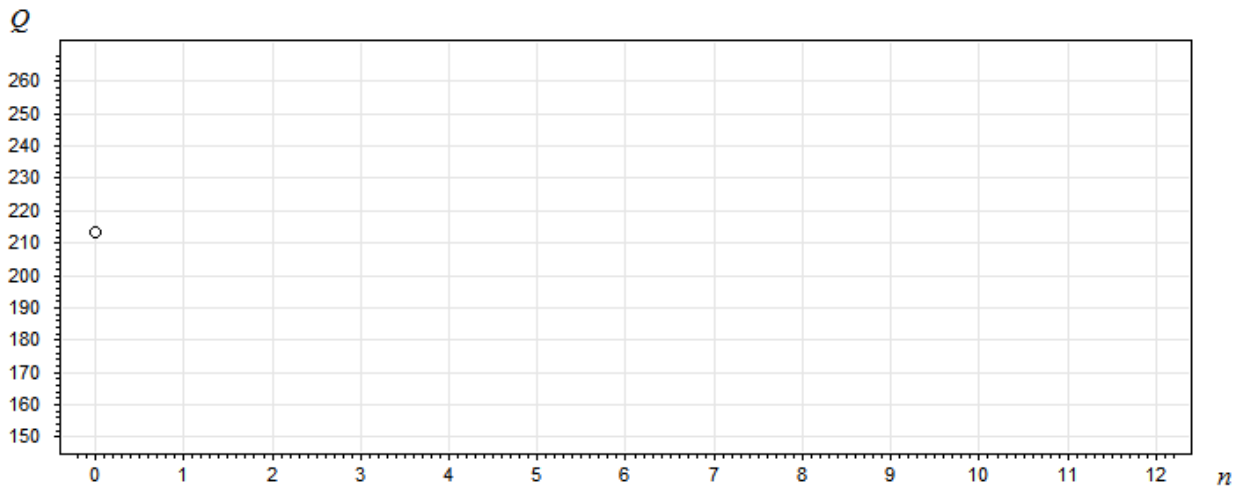
Алгоритм полного перебора (Full Search)



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)

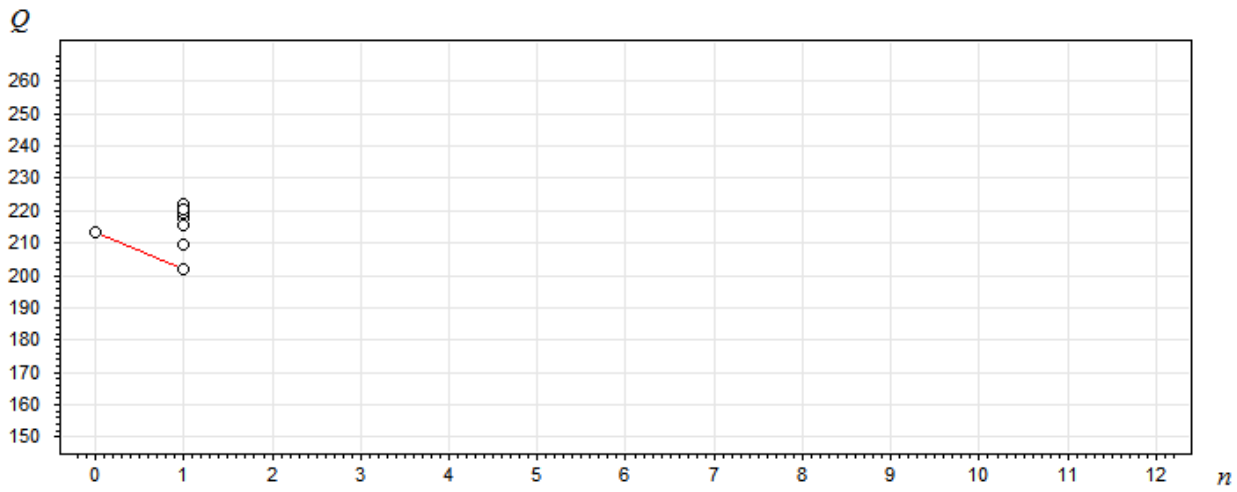


$d = 3$
$j = 0$

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)

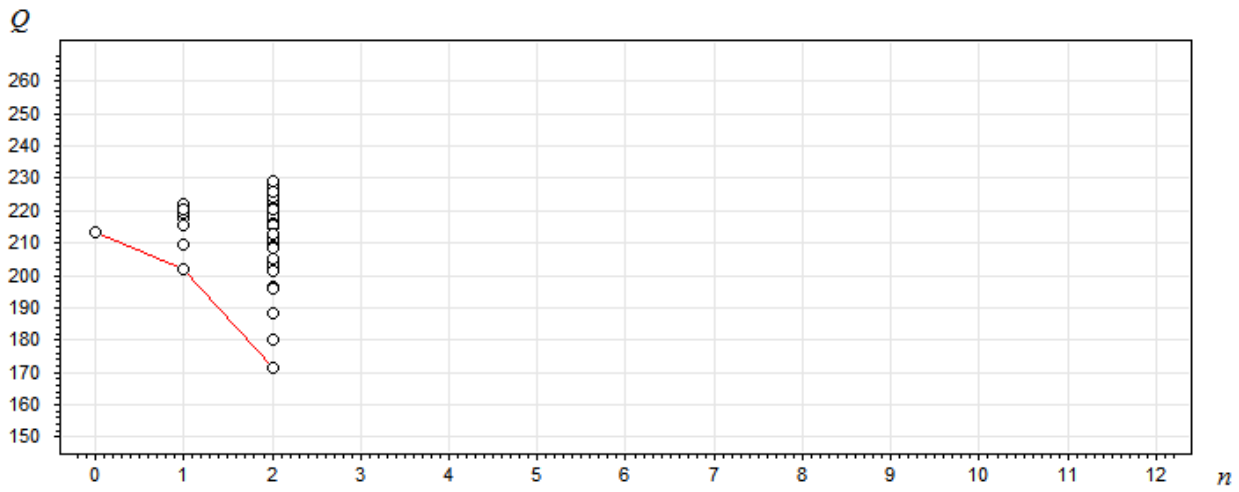


$d = 3$
$j = 1$
$j^* = 1$

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)

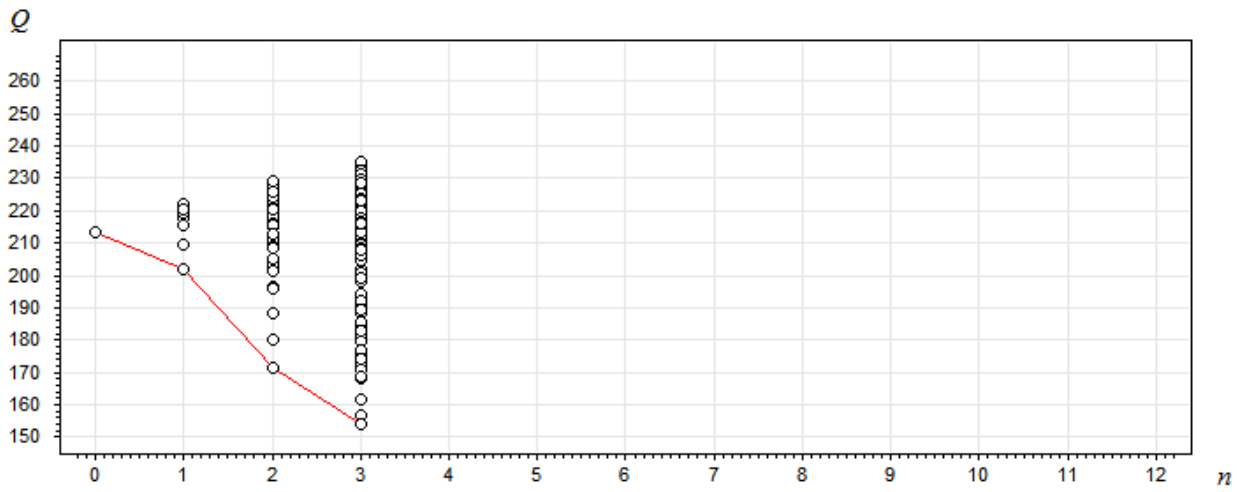


$d = 3$
$j = 2$
$j^* = 2$

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)

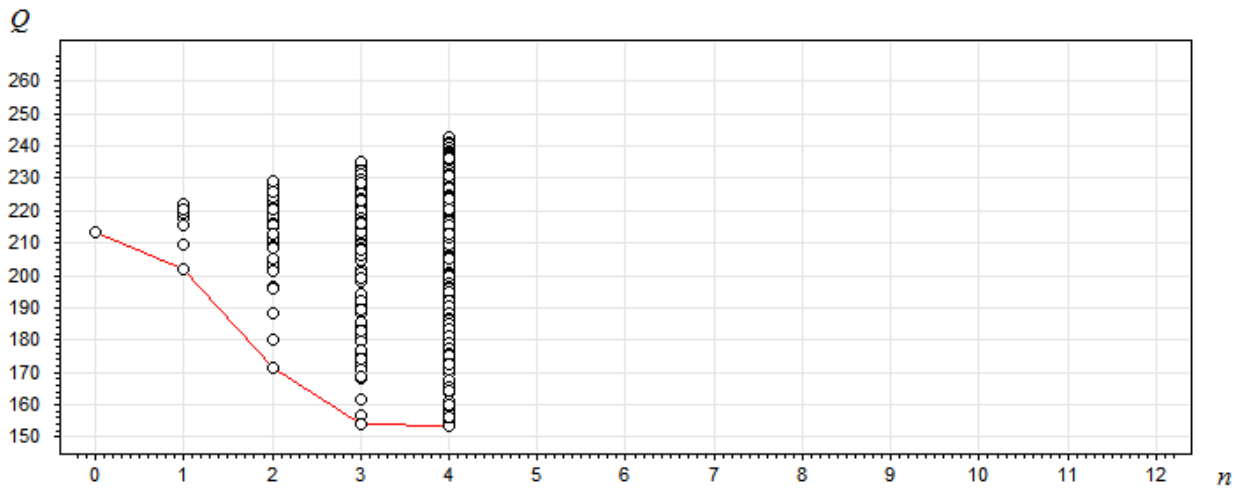


$d = 3$
$j = 3$
$j^* = 3$

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)

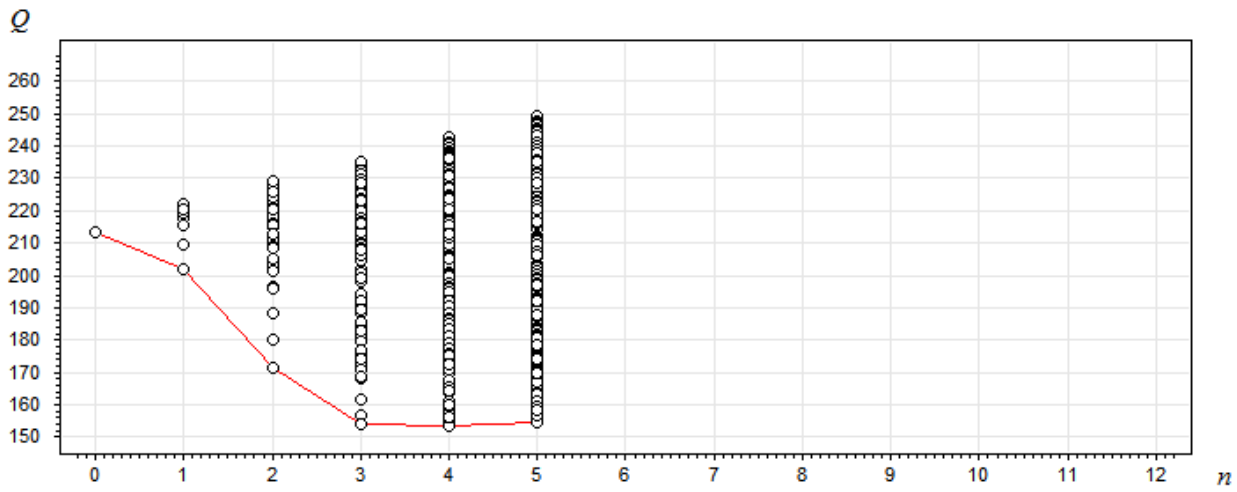


$d = 3$
$j = 4$
$j^* = 4$

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)

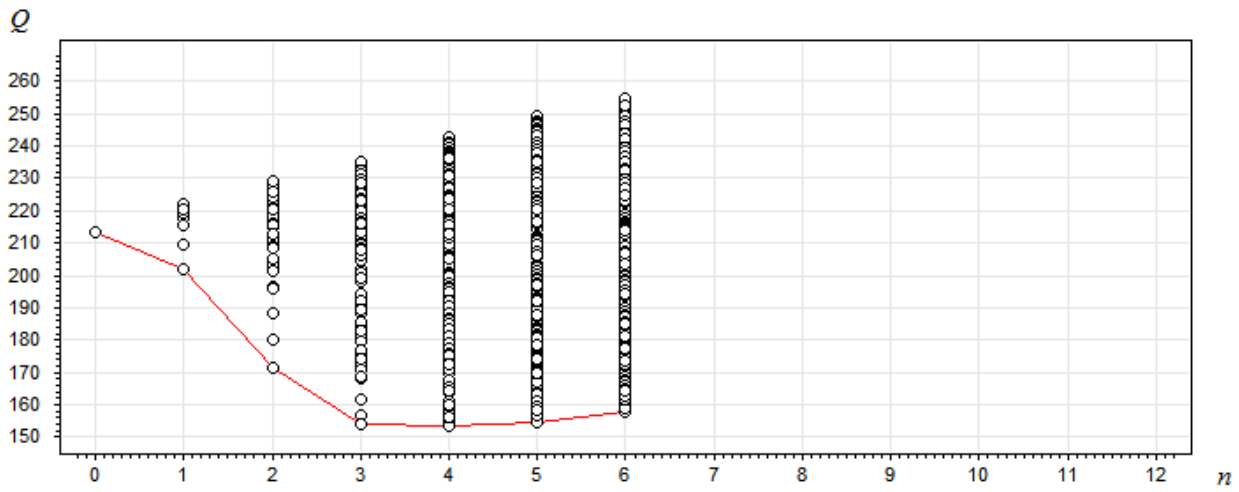


$d = 3$
$j = 5$
$j^* = 4$

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)

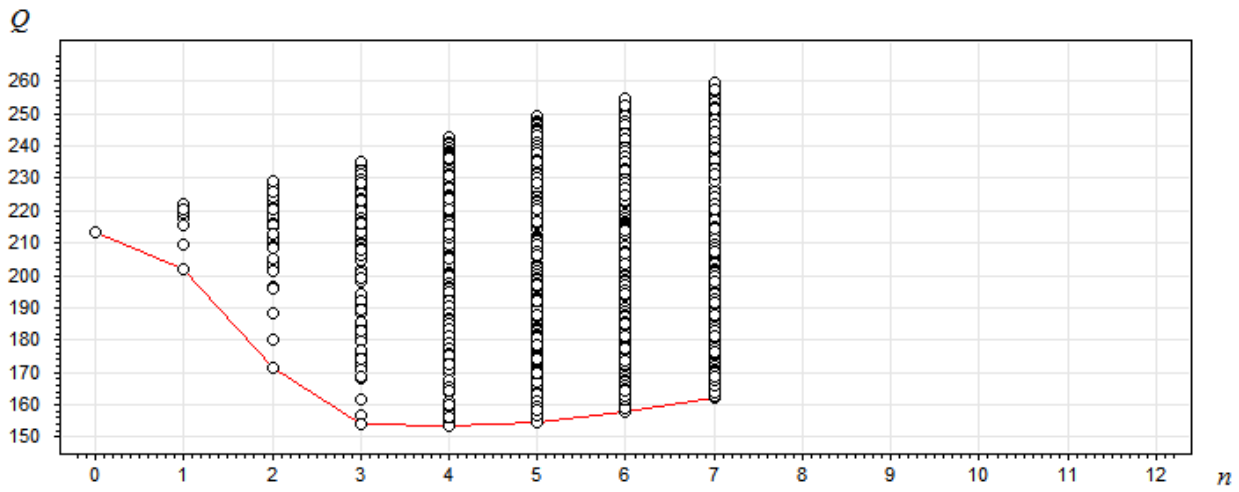


$d = 3$
$j = 6$
$j^* = 4$

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:

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

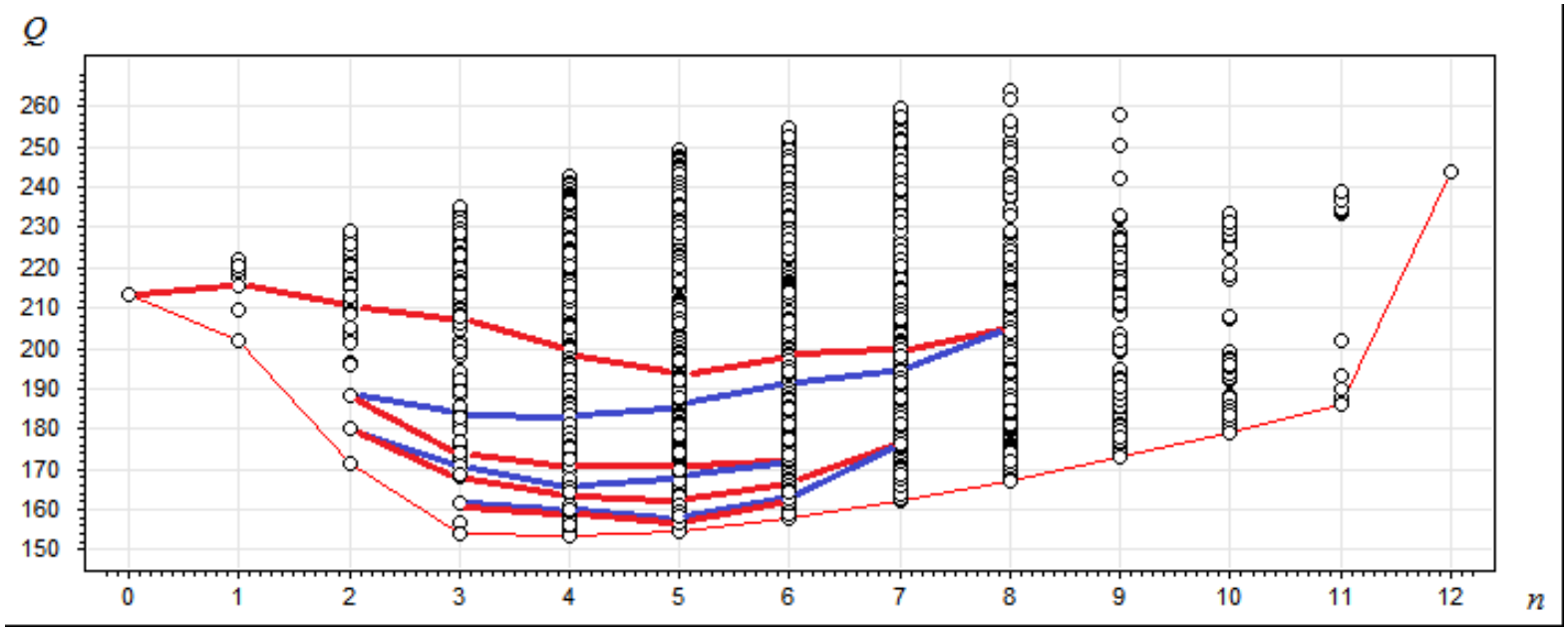
Алгоритм поочерёдного добавления и удаления (Add-Del)

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)

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

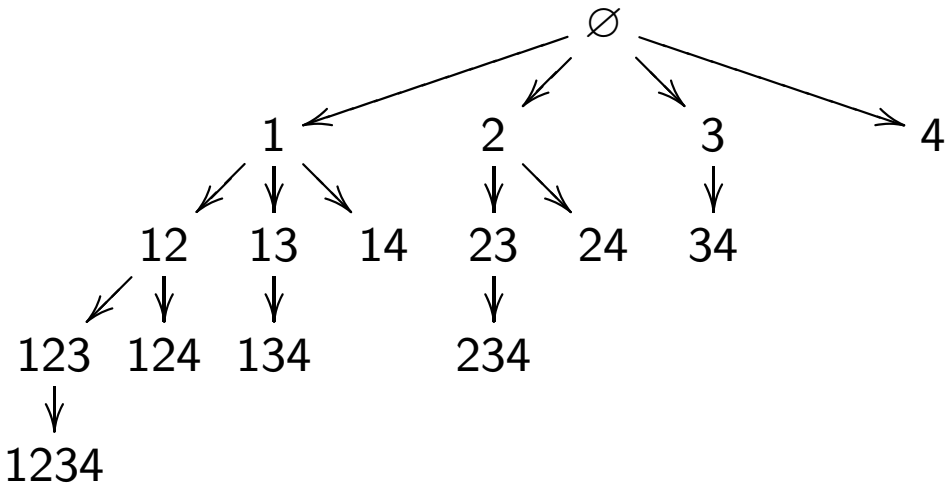
пока $|J_t| < n$ добавлять признаки (Add):
  $t := t + 1$; — началась следующая итерация;
  $f^* := \arg\min_{f \in F \setminus J_{t-1}} Q\left(J_{t-1} \cup \{f\}\right)$; $J_t := J_{t-1} \cup \{f^*\}$;
  если $Q(J_t) < Q^*$ **то** $t^* := t$; $Q^* := Q(J_t)$;
  если $t - t^* \geqslant d$ **то прервать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

пока $|J_t| > 0$ удалять признаки (Del):
  $t := t + 1$; — началась следующая итерация;
  $f^* := \arg\min_{f \in J_{t-1}} Q\left(J_{t-1} \setminus \{f\}\right)$; $J_t := J_{t-1} \setminus \{f^*\}$;
  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.

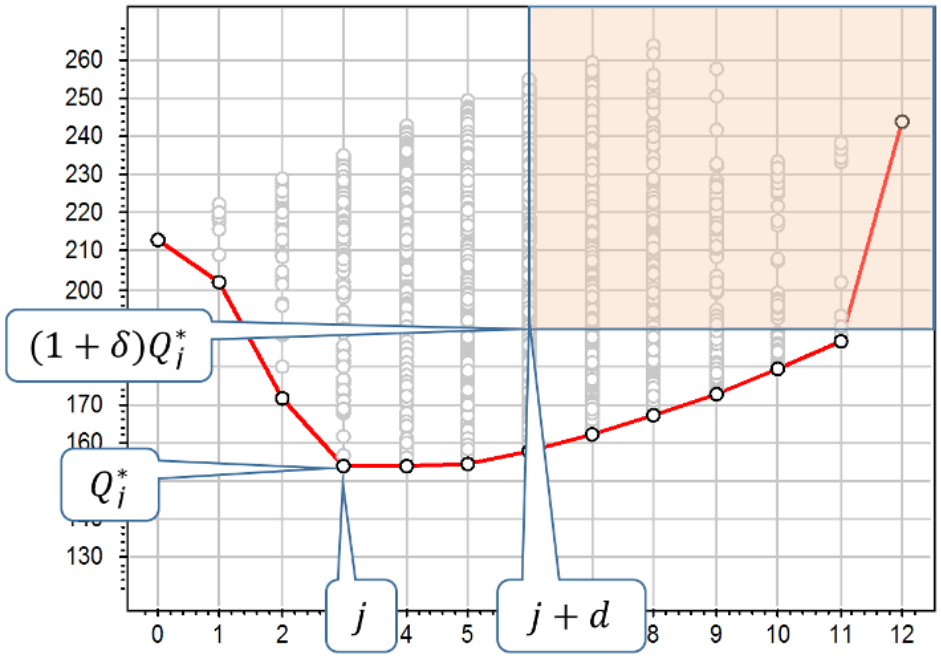
Поиск в глубину (DFS, метод ветвей и границ)

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);
если найдётся $j \leqslant |J| - d$ такое, что $Q(J) \geqslant (1+\delta)Q_j^*$, **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*.

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

Поиск в ширину (BFS)

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 всех $j = 1, \ldots, n$, 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**
    $R_j := \{J_j^1, \ldots, J_j^B\}$; — 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:
  $R_{j+1} := \{J \cup \{f\} \mid J \in R_j,\ f \in F \setminus J\}$;

Поиск в ширину (BFS)

- **Трудоёмкость:**
  $O(Bn^2)$, точнее $O(Bn(j^* + d))$.
- **Проблема дубликатов:**
  после сортировки (шаг 3) проверить на совпадение соседние наборы с равными значениями 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»);

$R_t := \{J_t^1, \ldots, J_t^{B_t}\}$ — *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:
$B_1 := B$; $R_1 := \{J_1^1, \ldots, J_1^{B_1}\}$; $Q^* := Q(\varnothing)$;
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: $R_t := \{J_t^1, \ldots, J_t^B\}$;
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:
$R_{t+1} := \{\sim(J' \times J'') \mid J', J'' \in R_t\} \cup R_t$;

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).

Случайный поиск с адаптацией (СПА)

Вместо скрещивания наборов попарно $\sim(J' \times J'')$,
можно генерировать наборы из распределения $(p_1, \ldots, p_n)$,
постепенно повышая вероятности p_j удачных признаков.

Вход: множество F, критерий Q, параметры d, j_0, T, r;

1: $p_1 = \cdots = p_n := 1/n$; — равномерная инициализация;
2: **для всех** $j = j_0, \ldots, n$, где j — сложность наборов:
3: **для всех** $t = 1, \ldots, T$, где t — номер итерации:
4: генерация $J_1, \ldots, J_r$ из распределения $(p_1, \ldots, p_n)$;
5: ранжирование: $Q(J_1) \leqslant \ldots \leqslant Q(J_r)$;
6: увеличить p_j признаков, попавших в топовые наборы;
7: **если** $Q(J_1) < Q^*$ **то** $j^* := j$; $J^* := J_1$; $Q^* := Q(J^*)$;
8: **если** $j - j^* \geqslant d$ **то вернуть** J^*;

Лбов Г. С. Выбор эффективной системы зависимых признаков. Вычислительные системы, 1965.

Преимущества и недостатки эволюционных алгоритмов

Преимущества:

- it's fun: «трудно быть богом»
- возможность введения различных эвристик
- решает задачи даже с очень большим числом признаков

Недостатки:

- относительно медленная сходимость
- отсутствие теоретических гарантий
- подбор параметров — непростое искусство

Резюме. Методы отбора признаков

- Критерий $Q(J)$ должен иметь оптимум по сложности:
  — внешний критерий — оценка обобщающей способности,
  — информативность — для логических правил.
- Большинство эвристик эксплуатируют две основные идеи:
  — признаки ранжируются по их полезности;
  — $Q(J)$ изменяется не сильно при небольшом изменении J.
- МГУА, ЭА, СПА очень похожи, и нетрудно придумать много аналогичных алгоритмов–*метаэвристик*.
- L_1-регуляризация в линейных моделях бесперебороная? Нет! Перебор при поиске подмножества активных ограничений есть, но он эффективно управляется условиями KKT.

Метод главных компонент: постановка задачи

$f_1(x), \ldots, f_n(x)$ — исходные числовые признаки;
$g_1(x), \ldots, g_m(x)$ — новые числовые признаки, $m \leqslant n$;

Требование: старые признаки должны линейно восстанавливаться по новым:

$$\hat{f}_j(x) = \sum_{s=1}^{m} g_s(x) u_{js}, \quad j = 1, \ldots, n, \quad \forall x \in X,$$

как можно точнее на обучающей выборке $x_1, \ldots, x_\ell$:

$$\sum_{i=1}^{\ell} \sum_{j=1}^{n} \big(\hat{f}_j(x_i) - f_j(x_i)\big)^2 \to \min_{\{g_s(x_i)\}, \{u_{js}\}}$$

Матричные обозначения

Матрицы «объекты–признаки», старая и новая:

$$\underset{\ell\times n}{F} = \begin{pmatrix} f_1(x_1) & \ldots & f_n(x_1) \\ \ldots & \ldots & \ldots \\ f_1(x_\ell) & \ldots & f_n(x_\ell) \end{pmatrix}; \quad \underset{\ell\times m}{G} = \begin{pmatrix} g_1(x_1) & \ldots & g_m(x_1) \\ \ldots & \ldots & \ldots \\ g_1(x_\ell) & \ldots & g_m(x_\ell) \end{pmatrix}.$$

Матрица линейного преобразования новых признаков в старые:

$$\underset{n\times m}{U} = \begin{pmatrix} u_{11} & \ldots & u_{1m} \\ \ldots & \ldots & \ldots \\ u_{n1} & \ldots & u_{nm} \end{pmatrix}; \qquad \hat{F} = GU^{\mathsf{T}} \overset{\text{хотим}}{\approx} F.$$

Найти: и новые признаки G, и преобразование U:

$$\sum_{i=1}^{\ell}\sum_{j=1}^{n}\big(\hat{f}_j(x_i) - f_j(x_i)\big)^2 = \big\|GU^{\mathsf{T}} - F\big\|^2 \to \min_{G,U},$$

Основная теорема метода главных компонент

Теорема

Если $m \leqslant \operatorname{rk} F$, то минимум $\left\|GU^{\mathsf{T}} - F\right\|^2$ достигается, когда столбцы U — это с.в. матрицы $F^{\mathsf{T}}F$, соответствующие m максимальным с.з. $\lambda_1, \ldots, \lambda_m$, а матрица $G = FU$.

При этом:

1. матрица U ортонормирована: $U^{\mathsf{T}}U = I_m$;
2. матрица G ортогональна: $G^{\mathsf{T}}G = \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_m)$;
3. $U\Lambda = F^{\mathsf{T}}FU$; $G\Lambda = FF^{\mathsf{T}}G$;
4. $\left\|GU^{\mathsf{T}} - F\right\|^2 = \|F\|^2 - \operatorname{tr}\Lambda = \sum\limits_{j=m+1}^{n} \lambda_j$.

Связь с сингулярным разложением

Если взять $m = n$, то:

1. $\left\| GU^{\mathsf{T}} - F \right\|^2 = 0$;
2. представление $\hat{F} = GU^{\mathsf{T}} = F$ точное и совпадает с сингулярным разложением при $G = V\sqrt{\Lambda}$:
$$F = GU^{\mathsf{T}} = V\sqrt{\Lambda}U^{\mathsf{T}}; \quad U^{\mathsf{T}}U = I_m; \quad V^{\mathsf{T}}V = I_m.$$
3. линейное преобразование U работает в обе стороны:
$$F = GU^{\mathsf{T}}; \quad G = FU.$$

Поскольку новые признаки некоррелированы ($G^{\mathsf{T}}G = \Lambda$), преобразование U называется *декоррелирующим* (или преобразованием Карунена–Лоэва).

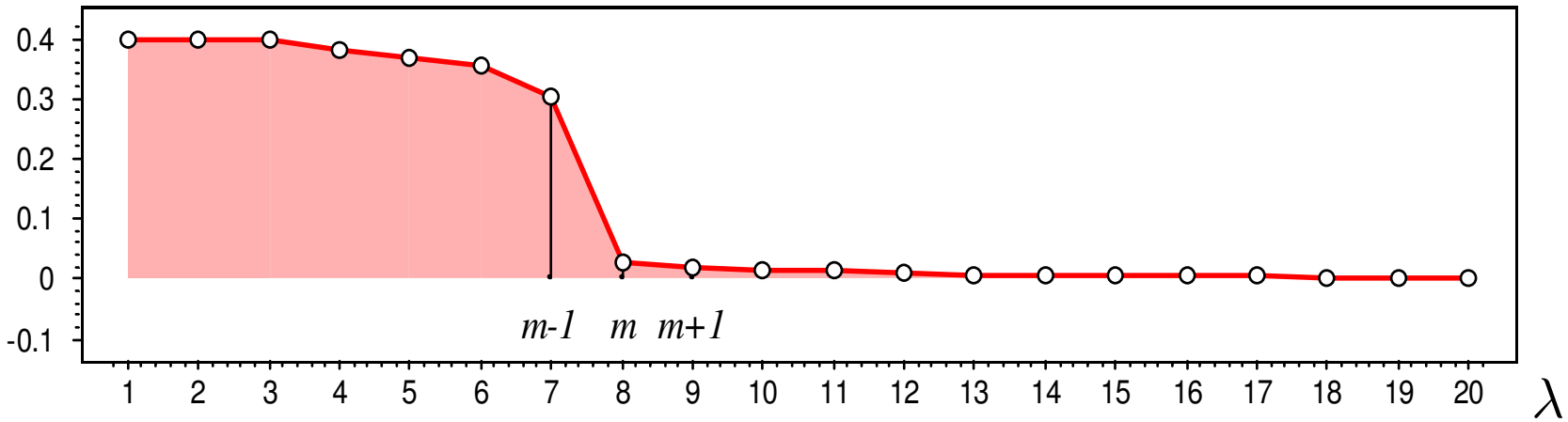
Эффективная размерность выборки

Упорядочим с.з. $F^\mathsf{T}F$ по убыванию: $\lambda_1 \geqslant \ldots \geqslant \lambda_n \geqslant 0$.

Эффективная размерность выборки — это наименьшее целое m, при котором

$$E_m = \frac{\|GU^\mathsf{T} - F\|^2}{\|F\|^2} = \frac{\lambda_{m+1} + \cdots + \lambda_n}{\lambda_1 + \cdots + \lambda_n} \leqslant \varepsilon.$$

Критерий «крутого склона»: находим m: $E_{m-1} \gg E_m$:



Применение метода главных компонент для регрессии

Заменим $\underset{\ell\cdot n}{F}$ на её приближение $\underset{\ell\cdot m}{G}\cdot\underset{m\cdot n}{U^{\mathsf{T}}}$, предполагая $m \leqslant n$:

$$\|G\underbrace{U^{\mathsf{T}}\alpha}_{\beta} - y\|^2 = \|G\beta - y\|^2 \to \min_{\beta}.$$

Связь нового и старого вектора коэффициентов:

$$\beta = U^{\mathsf{T}}\alpha; \qquad \alpha = U\beta.$$

Решение задачи наименьших квадратов относительно β:

$$\beta^* = D^{-1}V^{\mathsf{T}}y; \qquad \alpha^* = UD^{-1}V^{\mathsf{T}}y = \sum_{j=1}^{m}\frac{1}{\sqrt{\lambda_j}}u_j(v_j^{\mathsf{T}}y);$$

$$G\beta^* = VV^{\mathsf{T}}y = \sum_{j=1}^{m} v_j(v_j^{\mathsf{T}}y);$$

Отличается от решения исходной задачи заменой n на m.

Обобщение. Спектральный метод наименьших квадратов

1. Построить SVD-разложение: $\lambda_1 \geqslant \cdots \geqslant \lambda_m \geqslant \lambda_{m+1} \geqslant \cdots \geqslant \lambda_n$.
2. Игнорировать $n - m$ наименьших собственных значений, или иным способом отделить с. з. от нуля: $\lambda'_j := f(\lambda_j)$.

Частные случаи:

- $\lambda'_j := \lambda_j + \tau$ — гребневая регрессия
- $\lambda'_j := \lambda_j [j \leqslant m]$ — метод главных компонент

3. Применить формулы SVD для модификации МНК-решения:

$$\alpha^* = \sum_{j=1}^{n} \frac{1}{\sqrt{\lambda_j}} u_j (v_j^{\mathsf{T}} y) \quad \longrightarrow \quad \alpha^* = \sum_{j=1}^{m} \frac{\sqrt{\lambda_j}}{\lambda'_j} u_j (v_j^{\mathsf{T}} y)$$

$$F\alpha^* = \sum_{j=1}^{n} v_j (v_j^{\mathsf{T}} y) \quad \longrightarrow \quad F\alpha^* = \sum_{j=1}^{m} \frac{\lambda_j}{\lambda'_j} v_j (v_j^{\mathsf{T}} y)$$

Задачи низкорангового матричного разложения

Дано: матрица $Z = \|z_{ij}\|_{\ell \times n}$, $(i,j) \in \Omega \subseteq \{1..\ell\} \times \{1..n\}$

Найти: матрицы $X = \|x_{it}\|_{\ell \times k}$ и $Y = \|y_{tj}\|_{k \times n}$ такие, что

$$\|Z - XY\|_{\Omega}^2 = \sum_{(i,j) \in \Omega} \left(z_{ij} - \sum_{t=1}^{k} x_{it} y_{tj} \right)^2 \to \min_{X,Y}$$

Цели низкоранговых матричных разложений:

- понижение размерности пространства признаков, $k \ll n$
- формирование сжатого представления данных

Обобщения, вынуждающие отказаться от SVD:

- неполные ($|\Omega| < \ell n$) или разреженные ($|\Omega| \ll \ell n$) данные
- неотрицательное матричное разложение: $x_{it} \geqslant 0$, $y_{tj} \geqslant 0$
- неквадратичная функция потерь

Примеры прикладных задач матричного разложения

1. **Выявление интересов в рекомендательных системах (recommender systems, collaborative filtering)**

$$z_{ui} = \sum_t p_{tu} q_{ti}$$

дано: z_{ui} — рейтинги товаров i, поставленные пользователем u;
найти: p_{tu} — профиль интересов пользователя u;
q_{ti} — профиль интересов товара i.

2. **Латентный семантический анализ коллекций текстов (topic modeling, тематическое моделирование)**

$$z_{wd} = \sum_t \varphi_{wt} \theta_{td}$$

дано: $z_{wd} = p(w|d)$ — частоты слов w в документах d;
найти: $\varphi_{wt} = p(w|t)$ — распределения слов w в темах t,
$\theta_{td} = p(t|d)$ — распределения тем t в документах d.

Примеры прикладных задач матричного разложения

3. **Разделение смеси химических веществ по данным жидкостной хроматографии**

$$z_{t\lambda} = \sum_i x_{ti} y_{i\lambda}$$

дано: $z_{t\lambda}$ — выход сканирующего УФ-детектора;
найти: x_{ti} — хроматограмма i-го вещества, t — время;
$y_{i\lambda}$ — спектр i-го вещества, λ — длина волны.

4. **Оценивание экспрессии генов по данным ДНК-микрочипов с учётом кросс-гибридизации**

$$z_{pk} = \sum_g a_{pg} c_{gk}$$

дано: z_{pk} — интенсивность свечения p-й пробы на k-м чипе;
найти: a_{pg} — коэффициент сродства p-й пробы g-му гену,
c_{gk} — концентрация g-го гена на k-м чипе.

Постановка задачи коллаборативной фильтрации

U — множество субъектов (users/пользователей/клиентов)
I — множество объектов (items/предметов/товаров)
$Z = \|z_{ui}\|_{U \times I}$ — матрица кросс-табуляции

Типы данных — бинарные, порядковые или количественные:

- $z_{ui} =$ [клиент u использовал товар i]
- $z_{ui} =$ оценка, поставленная товару i клиентом u
- $z_{ui} =$ объём потребления товара i клиентом u

Задачи:

- прогнозирование незаполненных ячеек $\hat{z}_{ui}$
- формирование списка рекомендаций для клиента u
- поиск *коллабораций* клиентов со схожими предпочтениями

Пример. Конкурс NetflixPrize (www.netflixprize.com)

- z_{ui} = рейтинг фильма i, поставленный пользователем u
- рейтинги — целые числа $\{1, 2, 3, 4, 5\}$
- пользователей $|U| = 0.48\text{M}$, фильмов $|I| = 17\text{K}$
- обучение: $|\Omega| = 100\text{M}$ рейтингов, разреженность 98.7%
- тестовая выборка: $|\Omega'| = 2.8\text{M}$ рейтингов
- точность прогнозов оценивается по тестовой выборке Ω'

$$\text{RMSE}^2 = \frac{1}{|\Omega'|} \sum_{(u,i)\in\Omega'} \left(z_{ui} - \hat{z}_{ui}\right)^2;$$

- **задача:** уменьшить RMSE с 0.9514 до 0.8563 (на 10%)
- главный приз — \$1M
- конкурс длился со 2 октября 2006 по 21 сентября 2009

Латентные модели и матричные разложения

Латентная модель: по данным Z оцениваются векторы:

$p_u = (p_{tu})_{t \in T}$ — профиль интересов клиента $u \in U$

$q_i = (q_{ti})_{t \in T}$ — профиль интересов товара $i \in I$

T — множество латентных интересов/тем (topic)

Задача: найти представление $z_{ui} \approx \hat{z}_{ui} = \sum_{t \in T} p_{tu} q_{ti} = \langle p_u, q_i \rangle$

Матричная запись: $\hat{Z} = P^{\mathsf{T}} Q; \quad P = \|p_{tu}\|_{T \times U}; \quad Q = \|q_{ti}\|_{T \times I}$

Вероятностная интерпретация: $\underbrace{p(i|u)}_{z_{ui}} = \sum_{t \in T} \underbrace{q(i|t)}_{q_{ti}} \cdot \underbrace{p(t|u)}_{p_{tu}}$

Методы решения:

LFM — разреженный аналог SVD

NNMF — неотрицательное матричное разложение: $p_{tu} \geqslant 0$, $q_{ti} \geqslant 0$

PLSA — вероятностный латентный семантический анализ

Модель латентных факторов (LFM, Latent Factor Model)

Недостатки постановки задачи SVD $\|Z - P^{\mathsf{T}}\Delta Q\|^2 \to \min\limits_{P,Q,\Delta}$:

- если z_{ui} не известно, то полагают $z_{ui} = 0$
- лишнее требование ортогональности векторов p_t, q_t
- компоненты векторов p_u, q_i не интерпретируемы

Модификация задачи SVD для случая разреженных данных:

$$\sum_{(u,i)\in\Omega} \Big(\underbrace{z_{ui} - \bar{z}_u - \bar{z}_i - \langle p_u, q_i \rangle}_{\varepsilon_{ui}} \Big)^2 \to \min_{P,Q}$$

Метод стохастического градиента: для задачи $\varepsilon_{ui}^2 \to \min\limits_{p_u,q_i}$ градиентный шаг по случайной паре $(u,i) \in \Omega$:

$$p_{tu} := p_{tu} + \eta\varepsilon_{ui} q_{ti} \quad \text{для всех } t \in T$$

$$q_{ti} := q_{ti} + \eta\varepsilon_{ui} p_{tu} \quad \text{для всех } t \in T$$

Tacáks G., Pilászy I., Németh B., Tikk D. Scalable collaborative filtering approaches for large recommendation systems. JMLR, 2009.

Модель латентных факторов (LFM, Latent Factor Model)

Преимущества метода стохастического градиента:

- легко вводится регуляризация:
$$\varepsilon_{ui}^2 + \lambda\|p_u\|^2 + \mu\|q_i\|^2 \to \min_{p_u, q_i};$$
- легко вводятся ограничения неотрицательности:
$$p_{tu} \geqslant 0, \quad q_{ti} \geqslant 0 \quad \text{(метод проекции градиента)};$$
- легко вводится обобщение для ранговых данных:
$$\sum_{(u,i)\in\Omega} \left(z_{ui} - \bar{z}_u - \bar{z}_i - \beta\left(\sum_{t\in T} p_{tu} q_{ti}\right)\right)^2 \to \min_{P,Q,\beta}.$$
- легко реализуются все виды инкрементности:
  – добавление ещё одного клиента u,
  – добавление ещё одного объекта i,
  – добавление ещё одного значения z_{ui}.
- высокая численная эффективность на больших данных;

NNMF (Non-Negative Matrix Factorization)

Метод чередующихся наименьших квадратов (Alternating Least Squares, ALS):

$$D = \left\|Z - \sum_{t\in T} p_t q_t^{\mathsf{T}}\right\|^2 = \left\|Z_t - p_t q_t^{\mathsf{T}}\right\|^2 \to \min_{\{p_t\geqslant 0,\, q_t\geqslant 0\}}$$

Идея: искать поочерёдно то строки p_t, то строки q_t при фиксированных остальных $s \neq t$, $Z_t = Z - \sum_{s\in T\setminus t} p_s q_s^{\mathsf{T}}$.

$$\frac{\partial D}{\partial p_t} = 0 \quad\Rightarrow\quad (p_t^{\mathsf{T}} q_t - Z_t) q_t^{\mathsf{T}} = 0 \quad\Rightarrow\quad p_t = \left(\frac{q_t Z_t^{\mathsf{T}}}{q_t q_t^{\mathsf{T}}}\right)_+$$

$$\frac{\partial D}{\partial q_t} = 0 \quad\Rightarrow\quad p_t (p_t^{\mathsf{T}} q_t - Z_t) = 0 \quad\Rightarrow\quad q_t = \left(\frac{p_t Z_t}{p_t p_t^{\mathsf{T}}}\right)_+$$

Cichocki A., Zdunek R., Amari S. Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization. 2007.

Модель с учётом неявной информации (implicit feedback)

Явные (explicit) предпочтения z_{ui}, более качественные данные:

- покупка товара в интернет-магазине
- оценка, рейтинг, комментарий, лайк

Неявные (implicit) предпочтения y_{ui}, больший объём данных:

- посещение страницы товара
- просмотр (какой-то части) фильма

Идея: предсказываем y_{ui} с весом $c_{ui} = 1 + \alpha z_{ui}$:

$$\sum_{(u,i)\in\Omega} c_{ui}\big(y_{ui} - \bar{y}_u - \bar{y}_i - \langle p_u, q_i\rangle\big)^2 + \lambda \sum_{u\in U} \|p_u\|^2 + \mu \sum_{i\in I} \|q_i\|^2 \to \min_{P,Q}$$

Модель с неявными предпочтениями победила в Netflix Prize.

Bell R. M., Koren Y., Volinsky C. The BellKor 2008 solution to the Netflix Prize.

Измерение качества рекомендаций

RMSE — точность предсказания рейтингов:

$$\text{RMSE}^2 = \sum_{(u,i)\in\Omega} \left(z_{ui} - \hat{z}_{ui}\right)^2$$

Точность предсказаний не гарантирует хороших рекомендаций.

$R_u(k) \subset I$ — первые k рекомендаций для u;
$L_u \subset I$ — наблюдаемые предпочтения u.

Более адекватные метрики качества рекомендаций:

- $\text{precision@}k = \frac{|R_u(k) \cap L_u|}{|R_u(k)|}$ — точность
- $\text{recall@}k = \frac{|R_u(k) \cap L_u|}{|L_u|}$ — полнота
- меры качества ранжирования: MAP, NDCG и др.

Измерение качества рекомендаций

Многокритериальная система оценивания качества:

- *Разнообразие* (diversity):
  — число рекомендаций из разных категорий,
  — различность рекомендаций между сессиями пользователя
- *Новизна* (novelty): сколько среди рекомендаций объектов, которые данный пользователь ещё не видел
- *Покрытие* (coverage): доля объектов, которые хотя бы раз побывали среди рекомендованных
- *Догадливость* (serendipity): способность угадывать неожиданные нетривиальные предпочтения пользователей

Можно оптимизировать линейную комбинацию критериев, либо оптимизировать один при ограничениях на остальные.

Резюме

Методы понижения размерности

- Отбор признаков:
  — не столь важно, *как*,
  — важно, по какому критерию — *внешнему*,
  — и лучше даже по нескольким критериям сразу.
- Преобразование признаков:
  — метод главных компонент
  — неотрицательные матричные разложения

Низкоранговые матричные разложения

- Сжатие данных
- Построение латентных векторных представлений
- Много приложений:
  — линейные модели регрессии и классификации
  — коллаборативная фильтрация
  — тематическое моделирование
  — и т.д.