Министерство образования и науки Российской Федерации

Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

И.И. Кручинин
(к.т.н. доцент)

лекция «Метрические методы классификации многомерных объектов пересекающихся классов» по курсу «Введение в машинное обучение»

Калуга - 2018

Краткое содержание:

1. Метрические методы классификации
2. Метод ближайшего соседа

3. Метод ближайших соседей
4. Метод парзеновского окна
5. Метод потенциальных функций
6. Отбор эталонных объектов
7. Алгоритм STOLP
8. Задача построения оптимального метрического классификатора
9. Алгоритмы поиска схожих объектов в крупных массивах данных.

Метрические методы классификации

Во многих прикладных задачах измерять степень сходства объектов существенно проще, чем формировать признаковые описания. Если мера сходства объектов введена достаточно удачно, то, как правило, оказывается, что схожим объектам очень часто соответствуют схожие ответы. В задачах классификации это означает, что классы образуют компактно локализованные подмножества. Это предположение принято называть гипотезой компактности[1]. Для формализации понятия «сходства» вводится функция расстояния в пространстве объектов X. Методы обучения, основанные на анализе сходства объектов, будем называть метрическими, даже если функция расстояния не удовлетворяет всем аксиомам метрики (в частности, аксиоме треугольника).

Метод ближайшего соседа и его обобщения

Пусть на множестве объектов X задана функция расстояния ρ: $X \times X \to [0,\infty)$. Существует целевая зависимость $y* : X \to Y$, значения которой известны только на объектах обучающей выборки $X^\ell = (x_i, y_i)_{i=1}^\ell,\ y_i = y^*(x_i)$. Множество классов Y конечно. Требуется построить алгоритм классификации a: $X \to Y$, аппроксимирующий целевую зависимость $y*(x)$ на всём множестве X.

Обобщённый метрический классификатор

Для произвольного объекта $u \in X$ расположим элементы обучающей выборки $x_1,\ldots,x_\ell$ в порядке возрастания расстояний до u:

$$\rho(u, x_u^{(1)}) \leqslant \rho(u, x_u^{(2)}) \leqslant \cdots \leqslant \rho(u, x_u^{(\ell)}),$$

где через $x_u^{(i)}$ обозначается i-й сосед объекта u. Соответственно, ответ на i-м соседе объекта u есть $y_u^{(i)} = y*(x_u^{(i)})$. Таким образом, любой объект $u \in X$ порождает свою перенумерацию выборки.

Опр. 3.1. Метрический алгоритм классификации с обучающей выборкой X^ℓ относит объект u к тому классу $y \in Y$, для которого суммарный вес ближайших обучающих объектов $\Gamma_y(u,X^\ell)$ максимален:

[1] В математике компактными принято называть ограниченные замкнутые множества. Гипотеза компактности не имеет ничего общего с этим понятием, и должна пониматься скорее в «бытовом» смысле этого слова.

$$a(u; X^\ell) = \arg\max_{y \in Y} \Gamma_y(u, X^\ell);$$

$$\Gamma_y(u, X^\ell) = \sum_{i=1}^{\ell} [y_u^{(i)} = y]\, w(i, u); \qquad (3.1)$$

где весовая функция $w(i,u)$ оценивает степень важности i-го соседа для классификации объекта u. Функция $\Gamma_y(u,X^\ell)$ называется оценкой близости объекта u к классу y.

Метрический классификатор определён с точностью до весовой функции $w(i,u)$. Обычно она выбирается неотрицательной, не возрастающей по i. Это соответствует гипотезе компактности, согласно которой чем ближе объекты u и $x_u^{(i)}$, тем выше шансы, что они принадлежат одному классу.

Обучающая выборка X^ℓ играет роль параметра алгоритма a. Настройка сводится к запоминанию выборки, и, возможно, оптимизации каких-то параметров весовой функции, однако сами объекты не подвергаются обработке и сохраняются «как есть». Алгоритм $a(u;X^\ell)$ строит локальную аппроксимацию выборки X^ℓ, причём вычисления откладываются до момента, пока не станет известен объект u. По этой причине метрические алгоритмы относятся к методам ленивого обучения (lazy learning), в отличие от усердного обучения (eager learning), когда на этапе обучения строится функция, аппроксимирующая выборку.

Метрические алгоритмы классификации относятся также к методам рассуждения по прецедентам (case-based reasoning, CBR). Здесь действительно можно говорить о «рассуждении», так как на вопрос «почему объект u был отнесён к классу y?» алгоритм может дать понятное экспертам объяснение: «потому, что имеются схожие с ним прецеденты класса y», и предъявить список этих прецедентов.

Выбирая весовую функцию $w(i,u)$, можно получать различные метрические классификаторы, которые подробно рассматриваются ниже.

Метод ближайших соседей

Алгоритм ближайшего соседа (nearest neighbor, NN) относит классифицируемый объект $u \in X^\ell$ к тому классу, которому принадлежит ближайший обучающий объект:

$w(i,u) = [i = 1]; \qquad a(u;X^\ell) = y_u^{(1)}.$

Этот алгоритм является, по всей видимости, самым простым классификатором. Обучение NN сводится к запоминанию выборки X^ℓ. Единственное достоинство этого алгоритма — простота реализации. Недостатков гораздо больше:

- Неустойчивость к погрешностям. Если среди обучающих объектов есть выброс — объект, находящийся в окружении объектов чужого класса, то не только он сам будет классифицирован неверно, но и те окружающие его объекты, для которых он окажется ближайшим.
- Отсутствие параметров, которые можно было бы настраивать по выборке.

  Алгоритм полностью зависит от того, насколько удачно выбрана метрика ρ.
- В результате — низкое качество классификации.

Алгоритм k ближайших соседей (k nearest neighbors, kNN). Чтобы сгладить влияние выбросов, будем относить объект u к тому классу, элементов которого окажется больше среди ближайших соседей.

$$w(i,u) = [i \leqslant k]; \qquad a(u; X^\ell, k) = \arg\max_{y \in Y} \sum_{i=1}^{k} [y_u^{(i)} = y].$$

При $k = 1$ этот алгоритм совпадает с предыдущим, следовательно, неустойчив к шуму. При $k = \ell$, наоборот, он чрезмерно устойчив и вырождается в константу.
Таким образом, крайние значения k нежелательны. На практике оптимальное значение параметра k определяют по критерию скользящего контроля с исключением объектов по одному (leave-one-out, LOO). Для каждого объекта $x_i \in X^\ell$ проверяется, правильно ли он классифицируется по своим k ближайшим соседям.

$$\text{LOO}(k, X^\ell) = \sum_{i=1}^{\ell} \Big[a\big(x_i; X^\ell \backslash \{x_i\}, k\big) \neq y_i \Big] \to \min_k .$$

Заметим, что если классифицируемый объект x_i не исключать из обучающей выборки, то ближайшим соседом x_i всегда будет сам x_i, и минимальное (нулевое) значение функционала LOO(k) будет достигаться при $k = 1$.

Существует и альтернативный вариант метода kNN: в каждом классе выбирается k ближайших к u объектов, и объект u относится к тому классу, для которого среднее расстояние до k ближайших соседей минимально.

Алгоритм k взвешенных ближайших соседей. Недостаток kNN в том, что максимум может достигаться сразу на нескольких классах. В задачах с двумя классами этого можно избежать, если взять нечётное k. Более общая тактика, которая годится и для случая многих классов — ввести строго убывающую последовательность вещественных весов w_i, задающих вклад i-го соседа в классификацию:

$$w(i,u) = [i \leqslant k]\, w_i; \qquad a(u; X^\ell, k) = \arg\max_{y \in Y} \sum_{i=1}^{k} [y_u^{(i)} = y] w_i.$$

Выбор последовательности w_i является эвристикой. Если взять линейно убывающие веса $w_i = \frac{k+1-i}{k}$, то неоднозначности также могут возникать, хотя и реже (пример: классов два; первый и четвёртый сосед голосуют за класс 1, второй и третий — за класс 2; суммы голосов совпадают). Неоднозначность устраняется окончательно, если взять нелинейно убывающую последовательность, скажем, геометрическую прогрессию: $w_i = q^i$, где знаменатель прогрессии $q \in (0,1)$ является параметром алгоритма. Его можно подбирать по критерию LOO, аналогично числу соседей k.

Недостатки простейших метрических алгоритмов типа kNN.

- Приходится хранить обучающую выборку целиком. Это приводит к неэффективному расходу памяти и чрезмерному усложнению решающего правила. При наличии погрешностей (как в исходных данных, так и в модели сходства ρ) это может приводить к понижению точности классификации вблизи границы классов. Имеет

смысл отбирать минимальное подмножество эталонных объектов, действительно необходимых для классификации.

- Поиск ближайшего соседа предполагает сравнение классифицируемого объекта со всеми объектами выборки за $O(\ell)$ операций. Для задач с большими выборками или высокой частотой запросов это может оказаться накладно. Проблема решается с помощью эффективных алгоритмов поиска ближайших соседей, требующих в среднем $O(\ln\ell)$ операций.
- В простейших случаях метрические алгоритмы имеют крайне бедный набор параметров, что исключает возможность настройки алгоритма по данным.

Метод парзеновского окна

Ещё один способ задать веса соседям — определить w_i как функцию от расстояния $\rho(u, x_u^{(i)})$, а не от ранга соседа i. Введём функцию ядра $K(z)$, невозрастающую на $[0,\infty)$. Положив $w(i,u) = K\left(\frac{1}{h}\rho(u, x_u^{(i)})\right)$ в общей формуле

$$a(u; X^\ell, h) = \arg\max_{y\in Y} \sum_{i=1}^{\ell} [y_u^{(i)} = y]\, K\left(\frac{\rho(u, x_u^{(i)})}{h}\right). \qquad (3.2)$$

Параметр h называется шириной окна и играет примерно ту же роль, что и число соседей k. «Окно» — это сферическая окрестность объекта u радиуса h, при попадании в которую обучающий объект x_i «голосует» за отнесение объекта u к классу y_i. Мы пришли к этому алгоритму чисто эвристическим путём, однако он имеет более строгое обоснование в байесовской теории классификации, и, фактически, совпадает с методом парзеновского окна.

Параметр h можно задавать априори или определять по скользящему контролю. Зависимость LOO(h), как правило, имеет характерный минимум, поскольку слишком узкие окна приводят к неустойчивой классификации; а слишком широкие — к вырождению алгоритма в константу.

Фиксация ширины окна h не подходит для тех задач, в которых обучающие объекты существенно неравномерно распределены по пространству X. В окрестности одних объектов может оказываться очень много соседей, а в окрестности других — ни одного. В этих случаях применяется окно переменной ширины. Возьмём финитное ядро — невозрастающую функцию $K(z)$, положительную на отрезке $[0,1]$, и равную нулю вне его. Определим h как наибольшее число, при котором ровно k ближайших соседей объекта u получают ненулевые веса: $h(u) = \rho(u, x_u^{(k+1)})$. Тогда алгоритм принимает вид

$$a(u; X^\ell, k) = \arg\max_{y\in Y} \sum_{i=1}^{k} [y_u^{(i)} = y]\, K\left(\frac{\rho(u, x_u^{(i)})}{\rho(u, x_u^{(k+1)})}\right). \qquad (3.3)$$

Заметим, что при финитном ядре классификация объекта сводится к поиску его соседей, тогда как при не финитном ядре (например, гауссовском) требуется перебор всей обучающей выборки. Выбор ядра обсуждается также в разделе 2.2.2.

Метод потенциальных функций

В методе парзеновского окна центр радиального ядра $K_h(u,x) = K\left(\frac{1}{h}\rho(u,x)\right)$ помещается в классифицируемый объект u. В силу симметричности функции расстояния $\rho(u,x)$ возможен и другой, двойственный, взгляд на метрическую классификацию. Допустим, что ядро помещается в каждый обучающий объект x_i и «притягивает» объект u к классу y_i, если он попадает в его окрестность радиуса h_i:

$$a(u;X^\ell) = \arg\max_{y\in Y} \sum_{i=1}^{\ell} [y_i = y]\, \gamma_i\, K\left(\frac{\rho(u,x_i)}{h_i}\right), \qquad \gamma_i \geqslant 0,\ h_i > 0 \quad . \tag{3.4}$$

По сути, эта формула отличается от (3.3) только тем, что здесь ширина окна h_i зависит от обучающего объекта x_i, а не от классифицируемого объекта u.

Алгоритм 3.1. Метод потенциальных функций

Вход:
X^ℓ — обучающая выборка;
Выход:
Коэффициенты $\gamma_i,\ i = 1,\ldots,\ell$ в (3.4);

1: Инициализация: $\gamma_i = 0;\ \ i = 1,\ldots,\ell;$
2: **повторять**
3: выбрать объект $x_i \in X^\ell$;
4: **если** $a(x_i) \neq y_i$ **то**
5: $\gamma_i := \gamma_i + 1;$
6: **пока** число ошибок на выборке не окажется достаточно мало

Данная идея лежит в основе метода потенциальных функций [3] и имеет прямую физическую аналогию с электрическим потенциалом. При $Y = \{-1,+1\}$ обучающие объекты можно понимать как положительные и отрицательные электрические заряды; коэффициенты γ_i — как абсолютную величину этих зарядов; ядро $K(z)$ — как зависимость потенциала от расстояния до заряда; а саму задачу классификации — как ответ на вопрос: какой знак имеет электростатический потенциал в заданной точке пространства u. Заметим, что в электростатике $K(z) = \frac{1}{z}$ либо $\frac{1}{z+a}$, однако для наших целей совершенно не обязательно брать именно такое ядро.

Алгоритм (3.4) имеет достаточно богатый набор из 2ℓ параметров γ_i, h_i. Простейший и исторически самый первый метод их настройки представлен в Алгоритме 3.1. Он настраивает только веса γ_i, предполагая, что радиусы потенциалов h_i и ядро K выбраны заранее. Идея очень проста: если обучающий объект x_i классифицируется неверно, то потенциал класса y_i недостаточен в точке x_i, и вес γ_i увеличивается на единицу. Выбор объектов на шаге 3 лучше осуществлять не подряд, а в случайном порядке. Этот метод не так уж плох, как можно было бы подумать.

Достоинство Алгоритма 3.1 в том, что он очень эффективен, когда обучающие объекты поступают потоком, и хранить их в памяти нет возможности или необходимости. В те

годы, когда метод потенциальных функций был придуман, хранение выборки действительно было большой проблемой. В настоящее время такой проблемы нет, и Алгоритм 3.1 представляет скорее исторический интерес.

Недостатков у Алгоритма 3.1 довольно много: он медленно сходится; результат обучения зависит от порядка предъявления объектов; слишком грубо (с шагом 1) настраиваются веса γ_i; центры потенциалов почему-то помещаются только в обучающие объекты; задача минимизации числа потенциалов (ненулевых γ_i) вообще не ставится; вообще не настраиваются параметры h_i. В результате данный алгоритм не может похвастаться высоким качеством классификации.

Рассмотрим множество прецедентов. Пусть каждый из них имеет *поле* притяжения. Берем новый *объект* и смотрим, каким классом притягивается.

Пусть $L = 2$ – *число классов*. Обозначим эти классы через K_1 и K_2 соответственно. Рассмотрим обучающую последовательность $x_1, \ldots, x_{r_1}, x_{r_1+1}, \ldots, x_{r_2}$. Без ограничения общности будем считать, что $x_1, \ldots, x_{r_1} \in K_1$ и $x_{r_1+1}, \ldots, x_{r_2} \in K_2$.

Каждая точка образует в *пространстве признаков* X некоторое *поле* притяжения. Например, можно рассматривать каждую точку как единичный заряд. *Поле* описывается потенциалом, создаваемым системой зарядов во всем пространстве.

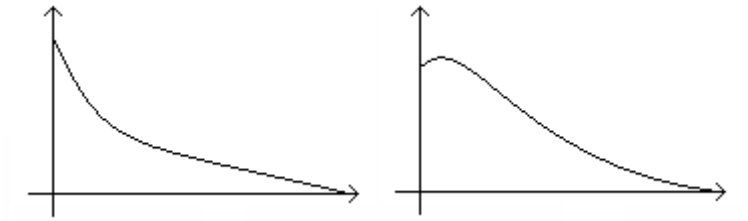
В пространстве задана потенциальная *метрика*: $K(x, y)$ – потенциальная *функция*, $x, y \in X$ такая, что

$$K(x, y) > 0, \text{ при } x \neq y,$$
$$K(x, y) = K(x, x + \mu(y - x)) = \widetilde{K}(\mu),$$

где $\widetilde{K}(\mu)$ – *монотонно убывающая функция* и $\widetilde{K}(0)$ – ее максимальное *значение*.

Пример. Пусть $d(x, y)$ – *расстояние* в R^2. Рассмотрим функцию $K(x, y) = K(d(x, y))$. Пусть α – *параметр* функции. Рассмотрим два примера функций $k(x, y)$:

$$K(x, y) = e^{-\alpha^2 d^2(x,y)} \text{ рис. слева},$$
$$K(x, y) = \frac{1}{1 + \alpha^2 d^2(x, y)} \text{ рис. справа}$$



Пусть X и $\overline{X}$ – прецеденты первого и второго класса соответственно, y – пробный образ. Тогда потенциалы, создаваемые в пространстве точками из классов X и $\overline{X}$ будут иметь вид:

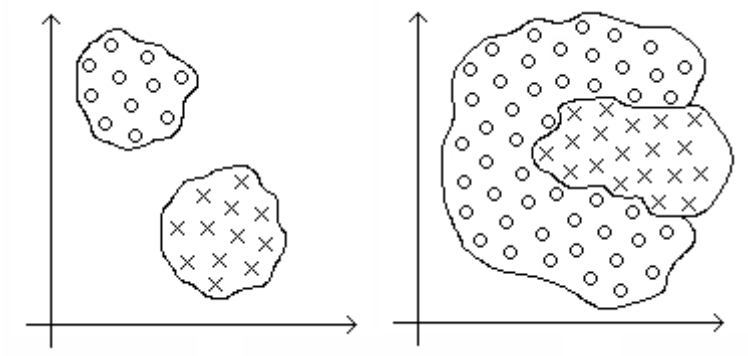
$$K_X(y) = \sum_{x \in X} K(x, y),\ K_{\overline{X}}(y) = \sum_{\overline{x} \in \overline{X}} K(\overline{x}, y)$$

соответственно.

Тогда *правило классификации* можно записать следующим образом: Если $K_X(y) > K_{\overline{X}}(y)$ то пробный образ y относится к классу X, иначе к классу $\overline{X}$, при равенстве выдавать ответ "не знаю" (отказ от распознавания).

Если рассмотреть дискриминантную функцию $\Phi(x) = K_X(x) - K_{\overline{X}}(x)$, то задача сводится к поиску этой функции по обучающей последовательности.

Рассмотрим вариант метода потенциальных функций называется "наивным". Он имеет следующие недостатки. Если, например, в первом классе точек много больше, чем в остальных, то голоса первого класса подавят голоса других классов, то есть *потенциал* o больше потенциала x (рис. слева). Но даже, если *множества* соизмеримы, могут возникнуть проблемы другого характера, например, погружение одних точек в другие (рис. справа). Поэтому наивный метод подлежит усовершенствованию.



Общая рекуррентная процедура

Пусть $\{\varphi_i(x)\}$ – конечная или бесконечная система функций на X. Будем искать дискриминантную функцию в виде

$$\Phi(x) = \sum_i c_i \varphi_i(x).$$

Требования к рассматриваемому ряду:

Для бесконечного ряда требуем поточечную *сходимость*.

Также желательно, чтобы c_i убывали быстро с ростом i. Это необходимо для обеспечения хорошего совпадения "обрезанного" бесконечного ряда с $\Phi(x)$.

Итак, пусть $\{\varphi_i(x)\}$ – *базовая* система функций. В качестве потенциальной функции будем рассматривать функцию вида

$$K(x,y) = \sum_i \lambda_i^2 \varphi_i(x)\varphi_i(y),$$

где λ_i удовлетворяет условиям: $\sum_i \lambda_i^2 < \infty$ и $\lambda_i \neq 0$.

Обозначим $\psi_i(x) = \lambda_i \varphi_i(x)$. Тогда

$$K(x,y) = \sum_i \psi_i(x)\varphi_i(y).$$

Предположим, что

$$K(x,x) = \sum_i \varphi_i^2(x) \leq M = const,$$

Тогда

$$K(x,y) = \sum_i \psi_i(x)\varphi_i(y) \leq M.$$

Для приближения $\Phi(x)$ предлагается рекуррентная процедура, называемая общей рекуррентной процедурой:

$$\Phi_{n+1} = q_n * \Phi_n(x) + r_n * K(x_{n+1}, x),\ n = 1, 2, \ldots$$

Пусть

- $\{x_{n+1}\}$ – обучающая последовательность прецедентов;
- q_n, r_n – некоторые числовые последовательности, которые должны задаваться так, чтобы обеспечить сходимость $\Phi_n(x)$ к $\Phi(x)$ при $n \to \infty$ в том или ином смысле.

Зададим начальное приближение $\Phi_0(x) = 0$. Как уже отмечалось, мы ищем функцию $\Phi(x)$ в виде:

$$\Phi(x) = \sum_{i=1}^{\infty} c_i \varphi_i(x).$$

Мы сделали достаточно сильное допущение, сказав, что наше решение будем выражать через базовую систему функций. Т.е. мы априорно предполагаем, что $\Phi_n(x)$ разложимо по системе функций $\{\varphi_i(x)\}$:

$$K(x,y) = \sum_{i=1}^{\infty} \lambda_i^2 \varphi_i(x)\varphi_j(x) \text{ и } \Phi_k(x) = \sum_{i=1}^{\infty} c_i^k \varphi_i(x).$$

Тогда, учитывая, что $K(x_{n+1}, x) = \sum_i \psi_i(x_{n+1})\psi_i(x)$, получаем:

$$\sum_i c_i^{n+1}\varphi_i(x) = q_n \sum_i c_i^n \varphi_i(x) + r_n \sum_i \psi_i(x_{n+1})\psi_i(x).$$

Обозначим через

$$\overline{c}_i^k = \frac{c_i^k}{\lambda_i},\ i,k = 1,2\ldots$$

Тогда

$$\overline{c}_i^{n+1}\psi_i(x) = q_n c_i^n \psi_i(x) + r_n \psi_i(x_{n+1})\psi_i(x).$$

Откуда получаем вторую форму общей рекуррентной процедуры:

$$\overline{c}_i^{n+1} = q_n c_i^n + r_n \psi_i(x_{n+1}).$$

Для нахождения связи коэффициентов $\{c_i^k\}$ и $\{c_i^{k+1}\}$ воспользуемся второй формой для формулы общей рекуррентной процедуры и соотношением

$$\overline{c}_i^k = \frac{c_i^k}{\lambda_i},\ i,k = 1,2,\ldots$$

Получим соотношение, связывающее *коэффициенты* $\{c_i^k\}$ и $\{c_i^{k+1}\}$:

$$\frac{c_i^{n+1}}{\lambda_i} = q_n c_i^n + r_n \psi_i(x_{n+1}),\ \text{где } \psi_i = \lambda_i \varphi_i(x)$$

Для возможности итерационных вычислений необходимо понять, как вычислять параметры q_n и r_n, а также начальное приближение c_i^0.

Зададим функцию

$$\Phi_i(x) = \begin{cases} K(x, x_i), \text{ при } x_i \in X \\ -K(x, x_i), \text{ при } x_i \in \overline{X} \end{cases}$$

где

$$K(x, x_m) = \sum_{i=1}^{\infty} \lambda_i^2 \varphi_i(x) \varphi_i(x_m) \text{ и } c_i^0 = \pm(\lambda_i^2 \varphi_i(x_1))$$

Тогда процесс перехода от Φ_n к Φ_{n+1} суть процесс подсчета коэффициентов. Обычно $q_n = 1,\ r_n$, вычисляется по следующему правилу:

$$r_n = \begin{cases} 0, \text{ если } \Phi_n(x_{n+1}) > 0 \text{ и } x_{n+1} \in X \\ 0, \text{ если } \Phi_n(x_{n+1}) < 0 \text{ и } x_{n+1} \in \overline{X} \\ 1, \text{ если } \Phi_n(x_{n+1}) < 0 \text{ и } x_{n+1} \in X \\ -1, \text{ если } \Phi_n(x_{n+1}) > 0 \text{ и } x_{n+1} \in \overline{X} \end{cases}.$$

Возьмем следующее начальное приближение:

$$\Phi_1(x) = \begin{cases} K(x, x), \text{ при } x_1 \in X \\ -K(x, x), \text{ при } x_1 \in \overline{X} \end{cases}$$

Таким образом, при правильном определении Φ_{n+1} получаем, что $\Phi_{n+1}(x) = \Phi_n(x)$; а в случае ошибки

$$\Phi_{n+1}(x) \begin{cases} \Phi_n(x) + K(x_{n+1}, x) \text{ при } x_{n+1} \in X \\ \Phi_n(x) - K(x_{n+1}, x) \text{ при } x_{n+1} \in \overline{X} \end{cases}$$

Данный процесс напоминает обучение в алгоритме персептрона.

Возникают следующие естественные вопросы:

Есть ли поточечная *сходимость* функции $\Phi_n(x)$ к $\Phi(x)$?

Где взять *базисные функции* $\varphi_i(x)$ в многомерном пространстве?

Попробуем ответить на эти вопросы.

Рассмотрим аналогию данного алгоритма с алгоритмом персептрона. Для функции

$$\Phi_n(x) = \sum_{i+1}^{\infty} c_i^k \varphi_i(x)$$

произведем замену $\varphi_i(x) = z_i$ и $z = (z_1, z_2, \ldots),\ z \in Z$ – это *вектор* в бесконечномерном пространстве, тогда

$$\Phi(x) = \sum_{i+1}^{\infty} c_i \varphi_i(x) = \sum_{i+1}^{\infty} c_i z_i,$$

где z – спрямляющее *пространство*. Таким образом, если $\Phi(x) > 0$ или $\Phi(x) < 0$, то $\sum_{i+1}^{\infty} c_i z_i > 0$ или $\sum_{i+1}^{\infty} c_i z_i < 0$ соответственно. Пусть $x_k \in X \bigcup \overline{X}$, тогда $x_k \to (z_1^k, z_2^k, \ldots)$ и $z_1^k = \varphi_i(x_k)$.

Выбор системы функций

$\{\varphi_i(x)\}$. Система функций $\{\varphi_i(x)\}$ задается априорно. Обычно используют некую полную систему функций, например, на конечном отрезке можно взять систему тригонометрических функций. Эта система к тому же ортогональна.

Утверждение. *Если задана полная ортогональная система функций одной переменной, то можно построить полную ортогональную систему функций любого числа переменных*.

Доказательство. Пусть $\{\varphi_i(x)\}$ – полная ортогональная система функций на конечном интервале I. Рассмотрим систему

$$\{\varphi_{i_1,\ldots,i_m}(x_1, \ldots, x_m) = \varphi_{i_1}(x_1), \ldots, \varphi_{i_m}(x_m)\},\ i_1, \ldots, i_m = 0, 1, 2, \ldots$$

Эта система полна и ортогональна на декартовом произведении m экземпляров I, то есть на $\underbrace{I \times I \times \ldots \times I}_{m \text{ раз}}$..

Проверим ортогональность. В скалярном произведении двух различных функций $\varphi_{i_1,\ldots,i_m}$ и $\varphi_{j_1,\ldots,j_m}$:

$$\int_I \ldots \int_I (\varphi_{i_1,\ldots,i_m}, \varphi_{j_1,\ldots,j_m}) dx_1 \ldots dx_m$$

всегда найдется такое k, что $\varphi_{i_k}(x_k) \neq \varphi_{j_k}(x_k)$ и, в силу ортогональности системы $\{\varphi_i(x)\}$, имеем:

$$\int_I (\varphi_{i_k}(x_k), \varphi_{j_k}(x_k))dx_k = 0$$

Далее, пусть $F(x_1, \ldots, x_m)$ – произвольная *функция* m переменных. Фиксируем все переменные, кроме x_1, и получаем разложение функции F:

$$F = \sum_{i_1} c_{i_1}(x_2, \ldots, x_m)\varphi_{i_1}$$

Повторяем это рассуждение для c_{i_1} последовательности $m-1$ раз:

$$F(x_1, \ldots, x_m) = \sum_{i_1,\ldots,i_m} c_{i_1\ldots i_m}\varphi_{i_1}(x_1)\ldots\varphi_{i_m}(x_m),$$

что и доказывает полноту системы

$$\{\varphi_{i_1,\ldots,i_m}(x_1, \ldots, i_m)\},\ i_1, \ldots, i_m = 0, 1, 2, \ldots$$

Сходимость общей рекуррентной процедуры

Предположим, что обучающая последовательность есть *выборка* конечного объема из пространства $\widetilde{X}$ ($\widetilde{X}$ – *пространство признаков*). Тогда последовательность $\{\Phi_n(x)\}$ есть последовательность случайных функций, и последовательность c_i – последовательность случайных чисел. Поэтому будем говорить о сходимости $\{\Phi_n(x)\}$ в вероятностном смысле, то есть либо по вероятности, либо с вероятностью равной 1, либо в среднем.

Пусть x – случайные величины из $\widetilde{X}$, а $X, \overline{X}$ – *выборка* для конечного объекта.

Теорема. *Пусть заданы два множества* X *и* $\overline{X}$ *и выполнены следующие условия*.

- *Существует функция* $\Phi(x)$ *такая, что* $\Phi(x) \geq \varepsilon$, *при* $x \in X$, $\Phi(x) \leq -\varepsilon$ *при* $x \in \overline{X}$, *где константа* $\varepsilon > 0$.
- *Задана система функций* $\{\varphi_i(x)\},\ i = 1, 2, \ldots$ *такая, что* $\Phi(x) = \sum_i c_i\varphi_i(x)$, $K(x, y) = \sum_i \lambda_i^2\varphi_i(x)\varphi_i(y)$, $\sum_i \left(\frac{c_i}{\lambda_i}\right)^2 < \infty$

- *Точки из обучающей последовательности независимые случайные величины, с одной и той же плотностью* $p(x)$. *Тогда общая рекуррентная процедура, определяемая формулой* $\Phi_{n+1} = q_n\Phi_n + r_nK(x, x_{n+1})$, *где* $\Phi_1(x) = 0,\ q_n = 1$ *и*

$$r_n = \begin{cases} 0,\ \textit{если}\ \Phi_n(x_{n+1}) > 0\ \textit{и}\ x_{n+1} \in X \\ 0,\ \textit{если}\ \Phi_n(x_{n+1}) < 0\ \textit{и}\ x_{n+1} \in \overline{X} \\ 1,\ \textit{если}\ \Phi_n(x_{n+1}) < 0\ \textit{и}\ x_{n+1} \in X \\ -1,\ \textit{если}\ \Phi_n(x_{n+1}) > 0\ \textit{и}\ x_{n+1} \in \overline{X} \end{cases}$$

сходится в следующем смысле: $\mathbb{E}\left[|sign\Phi(x) - sign\Phi_n(x)|\right] \to 0$, *при* $n \to \infty$.

Теорема. *Пусть выполнены все условия предыдущей теоремы. Пусть также на каждом* n *-ом шаге работы общей рекуррентной процедуры существует строго положительная вероятность исправления ошибки, если функция* $\Phi_n(x)$ *к* n *-ому шагу еще не разделила классы* K_1 *и* K_2. *Пусть с вероятностью единица для каждой реализации процедуры существует конечное число* l *такое, что*

$\Phi_l(x)$ *– правильно разделяет* X *и* $\overline{X}$,

Z *– конечный интервал на прямой* $(Z \in R^1)$,

$\{\varphi_i(x)\}$ *– полная ортогональная система функций,* $\varphi_1(x) \to R^1$,

$\int\limits_Z \varphi_i(x)\varphi_j(x)dx = 0$ *при* $i \neq j$.

Тогда система функций $\{\varphi_{i_1,\ldots,i_m}(x_1, \ldots, x_m) = \varphi_{i_1}(x_1) * \ldots * \varphi_{i_m}(x_m)\}$ *полна и ортогональна на пространстве* $\underbrace{Z \times \ldots \times Z}_{m \text{ раз}}$.

Доказательство. Ортогональность очевидна. Докажем полноту. Пусть $F(x_1, \ldots, x_m))$ – произвольная *функция* в Z^m такая, что $Z^m \to R^1$. Фиксируем переменные начиная с x_2, тогда

$$F = F(x_1) = \sum_{i=1}^{\infty} c_i(x_1, \ldots, x_m)\cdot\varphi_i(x_1),\ \text{где}\ c_1(x_2, \ldots, x_m) = \sum_{i=1}^{\infty} c_i(x_3, \ldots, x_m)\cdot\varphi_i(x_l).$$

Функции Эрмита

Если в качестве системы функций $\{\varphi_i(x)\}$ взять функции вида

$$\varphi_i(x) = \frac{1}{(2_i i! \sqrt{\pi})^{1/2}} e^{-\frac{x^2}{2}} H_i(x),$$

где $H_i(x) = (-1)^i e^{x^2} \left(\frac{d}{dx}\right)^i e^{-x^2}$ – *полином* Эрмита.

Тогда

$$K(x,y) = \sum_{i=0}^{\infty} \lambda_i^2 \varphi_i(x) \varphi_j(x).$$

Обозначим $\alpha^i = \lambda_i^2$, где $|\alpha| < 1$, тогда

$$K(x,y) = \frac{1}{\sqrt{\pi(1-\alpha)^2}} \exp\left[\frac{2xy\alpha - (x^2+y^2)\alpha^2}{1-\alpha^2}\right].$$

Отбор эталонных объектов

Обычно объекты обучения не являются равноценными. Среди них могут находиться типичные представители классов — эталоны. Если классифицируемый объект близок к эталону, то, скорее всего, он принадлежит тому же классу. Ещё одна категория объектов — неинформативные или периферийные. Они плотно окружены другими объектами того же класса. Если их удалить из выборки, это практически не отразится на качестве классификации. Наконец, в выборку может попасть некоторое количество шумовых выбросов — объектов, находящихся «в толще» чужого класса. Как правило, их удаление только улучшает качество классификации.

Эти соображения приводят к идее исключить из выборки шумовые и неинформативные объекты, оставив только минимальное достаточное количество эталонов. Этим достигается несколько целей одновременно — повышается качество и устойчивость классификации, сокращается объём хранимых данных и уменьшается время классификации, затрачиваемое на поиск ближайших эталонов. Кроме того, выделение небольшого числа эталонов в каждом классе позволяет понять структуру класса.

В первую очередь введём функцию отступа, которая позволит оценивать степень типичности объекта.

Понятие отступа объекта

Общая формула (3.1) позволяет ввести характеристику объектов, показывающую, насколько глубоко объект погружён в свой класс.

Опр. 3.2. *Отступом (margin) объекта $x_i \in X^\ell$ относительно алгоритма классификации, имеющего вид $a(u) = \arg\max_{y \in Y} \Gamma_y(u)$, называется величина*

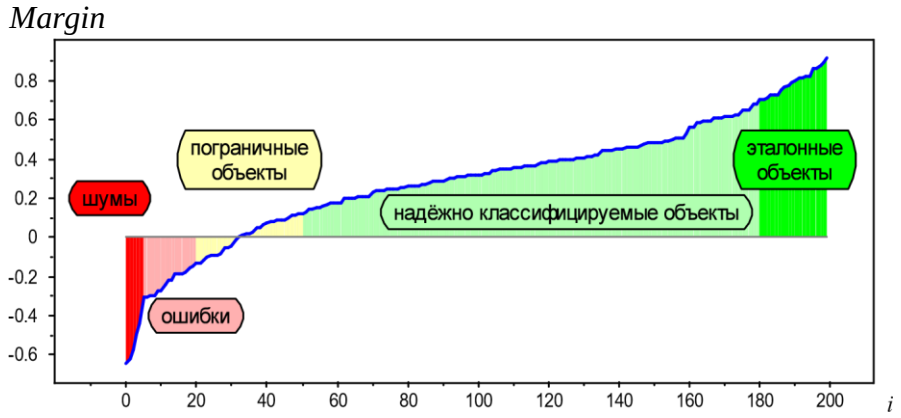
$$M(x_i) = \Gamma_{y_i}(x_i) - \max_{y \in Y \setminus y_i} \Gamma_y(x_i).$$

Отступ показывает степень типичности объекта. Отступ отрицателен тогда и только тогда, когда алгоритм допускает ошибку на данном объекте.

В зависимости от значений отступа обучающие объекты условно делятся на пять типов, в порядке убывания отступа: эталонные, неинформативные, пограничные, ошибочные, шумовые.

Эталонные объекты имеют большой положительный отступ, плотно окружены объектами своего класса и являются наиболее типичными его представителями.

- Неинформативные объекты также имеют положительный отступ. Изъятие этих объектов из выборки (при условии, что эталонные объекты остаются), не влияет на качество классификации. Фактически, они не добавляют к эталонам никакой новой информации. Наличие неинформативных объектов характерно для выборок избыточно большого объёма.



Упорядоченные по возрастанию отступов M_i объекты выборки, i = 1,...,200. Условное деление объектов на пять типов.

- Пограничные объекты имеют отступ, близкий к нулю. Классификация таких объектов неустойчива в том смысле, что малые изменения метрики или состава обучающей выборки могут изменять их классификацию.
- Ошибочные объекты имеют отрицательные отступы и классифицируются неверно. Возможной причиной может быть неадекватность алгоритмической модели (3.1), в частности, неудачная конструкция метрики ρ.
- Шумовые объекты или выбросы — это небольшое число объектов с большими отрицательными отступами. Они плотно окружены объектами чужих классов и классифицируются неверно. Они могут возникать из-за грубых ошибок или пропусков в исходных данных, а также по причине отсутствия важной информации, которая позволила бы отнести эти объекты к правильному классу.

Приведённая типизация условна. Не существует чёткого различия между «соседними» типами объектов. В частности, легко строятся примеры выборок, содержащих такие пары близких объектов, что любой из них может быть объявлен эталонным, а второй — неинформативным.

Шумовые и неинформативные целесообразно удалять из выборки. Соответствующий эвристический алгоритм будет описан ниже.

Распределение значений отступов в выборке даёт полезную дополнительную информацию не только об отдельных объектах, но и о выборке в целом. Если основная масса объектов имеет положительные отступы, то разделение выборки можно считать успешным. Если в выборке слишком много отрицательных отступов, то гипотеза компактности не выполняется, и в данной задаче при выбранной метрике применять алгоритмы типа kNN нецелесообразно. Если значения отступов концентрируются вблизи нуля, то ждать надёжной классификации не приходится, так как слишком много объектов оказываются в пограничной «зоне неуверенности».

Алгоритм STOLP для отбора эталонных объектов

Идея отбора эталонов реализована в алгоритме STOLP. Мы рассмотрим его обобщённый вариант с произвольной весовой функцией $w(i,u)$. Будем строить метрический алгоритм $a(u;\Omega)$ вида (3.1), где $\Omega \subseteq X^\ell$ — множество эталонов.

Алгоритм 3.2. Отбор эталонных объектов STOLP

Вход:
X^ℓ — обучающая выборка;
δ — порог фильтрации выбросов;
ℓ_0 — допустимая доля ошибок;

Выход:
Множество опорных объектов $\Omega \subseteq X^\ell$;

1: **для всех** $x_i \in X^\ell$ проверить, является ли x_i выбросом:
2: **если** $M(x_i, X^\ell) < \delta$ **то**
3: $X^{\ell-1} := X^\ell \setminus \{x_i\}$; $\ell := \ell - 1$;
4: Инициализация: взять по одному эталону от каждого класса:
$\Omega := \left\{ \arg\max_{x_i \in X^\ell_y} M(x_i, X^\ell) \;\middle|\; y \in Y \right\}$;
5: **пока** $\Omega \neq X^\ell$;
6: Выделить множество объектов, на которых алгоритм $a(u;\Omega)$ ошибается:
$E := \{x_i \in X^\ell \setminus \Omega : M(x_i, \Omega) < 0\}$;
7: **если** $|E| < \ell_0$ **то**
8: **выход**;
9: Присоединить к Ω объект с наименьшим отступом:
$x_i := \arg\min_{x \in E} M(x, \Omega)$; $\Omega := \Omega \cup \{x_i\}$;

Обозначим через $M(x_i,\Omega)$ отступ объекта x_i относительно алгоритма $a(x_i;\Omega)$. Большой отрицательный отступ свидетельствует о том, что объект x_i окружён объектами чужих классов, следовательно, является выбросом. Большой положительный отступ означает, что

объект окружён объектами своего класса, то есть является либо эталонным, либо периферийным.

Алгоритм 3.2 начинает с отсева выбросов (шаги 1–3). Из выборки X^ℓ исключаются все объекты x_i с отступом $M(x_i,X^\ell)$, меньшим заданного порога δ. Если взять $\delta = 0$, то оставшиеся объекты будут классифицированы верно. Вместо δ можно задавать долю исключаемых объектов с наименьшими значениями отступа.

Затем формируется начальное приближение — в Ω заносится по одному наиболее типичному представителю от каждого класса (шаг 4).

После этого начинается процесс последовательного «жадного» наращивания множества Ω. На каждом шаге к Ω присоединяется объект x_i, имеющий минимальное значение отступа. Так продолжается до тех пор, пока число ошибок не окажется меньше заданного порога ℓ_0. Если положить $\ell_0 = 0$, то будет построен алгоритм $a(u;\Omega)$, не допускающий ошибок на обучающих объектах, за исключением заранее исключённых выбросов.

В результате каждый класс будет представлен в Ω одним «центральным» эталонным объектом и массой «приграничных» эталонных объектов, на которых отступ принимал наименьшие значения в процессе итераций. Параметр δ позволяет регулировать ширину зазора между эталонами разных классов. Чем больше δ, тем дальше от границы классов будут располагаться «приграничные» эталоны, и тем более простой, менее «изрезанной» будет граница между классами.

В описанном варианте алгоритм STOLP имеет относительно низкую эффективность. Для присоединения очередного эталона необходимо перебрать множество новобъектов $X\ell \setminus \Omega$, и для каждого вычислить отступ относительно множества этало-$_{O}(|\Omega|^2\ell)$. Для ускорения алгоритма можно

$_{\Omega}$. Общее число операций составляет добавлять сразу по несколько эталонов, не пересчитывая отступов. Если при этом выбирать добавляемые эталоны достаточно далеко друг от друга, то добавление одного из них практически не будет влиять на отступы остальных. Аналогично, на этапе отсева выбросов можно вычислить отступы только один раз, и потом отбросить все объекты с отступами ниже δ. Реализация этих идей не показана в Алгоритме 3.2, чтобы не загромождать его техническими подробностями.

Результатом работы алгоритма STOLP является разбиение обучающих объектов на три категории: шумовые, эталонные и неинформативные. Если гипотеза компактности верна и выборка достаточно велика, то основная масса обучающих объектов окажется неинформативной и будет отброшена. Фактически, этот алгоритм выполняет сжатие исходных данных.

Задача построения метрического классификатора

Пусть

- Ω – пространство образов,
- X – признаковое пространство,
- $g(\omega),\ \omega \in \Omega$ – индикаторная функция,
- M – множество признаков.

Тогда $g : \Omega \to M$.

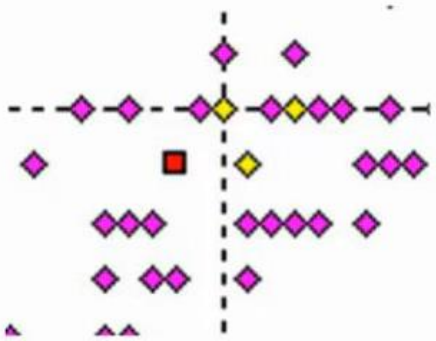
Пусть также

- $X = \langle x(\omega_i), g(\omega_i) \rangle,\ i = 1, \ldots, N$ – множество прецедентов,
- $\widehat{g}(x)$ – решающее правило.

Тогда $\widehat{g} : X \to M$

Выбор решающего правила исходит из минимизации $d(g, \widehat{g}) \to \min$, где d – *метрика*, *мера* близости функций $g(\omega)$ и $\widehat{g}(x(\omega))$. Построение $\widehat{g}$ называют задачей обучения. $\widehat{g}$ – это ученик, процедура формирования – это учитель, прецеденты – это обучающая последовательность.

Случай классификации по 5 ближайшим соседям



$$\rho_1 = \rho\big((5.9,1.5),(5.8,1.4)\big) = \sqrt{(5.9-5.8)^2 + (1.5-1.4)^2} = 0.1414$$
$$\rho_2 = \rho\big((5.7,1.3),(5.8,1.4)\big) = \sqrt{(5.7-5.8)^2 + (1.3-1.4)^2} = 0.1414$$
$$\rho_3 = \rho\big((6,1.5),(5.8,1.4)\big) = \sqrt{(6-5.8)^2 + (1.5-1.4)^2} = 0.2236$$
$$\rho_4 = \rho\big((5.6,1.5),(5.8,1.4)\big) = \sqrt{(5.6-5.8)^2 + (1.5-1.4)^2} = 0.2236$$
$$\rho_5 = \rho\big((5.7,1.3),(5.8,1.4)\big) = \sqrt{(5.7-5.8)^2 + (1.3-1.4)^2} = 0.3$$

$1/\rho_1 = 1/0.1414 = 7.0710$
$1/\rho_2 = 1/0.1414 = 7.0710$
$1/\rho_3 = 1/0.2236 = 4.4721$
$1/\rho_4 = 1/0.2236 = 4.4721$
$1/\rho_5 = 1/0.3 = 3.3333$

1 класс	*2 класс*	*3 класс*
0	$1/\rho_1 + 1/\rho_2 + 1/\rho_4 =$ $= 7.0710 + 7.0710 +$ $+ 4.4721 = 18.6141$	$1/\rho_3 + 1/\rho_5 =$ $= 4.4721 + 3.3333 =$ $= 7.8054$

Ответ: 2 класс

Качество обучения классификатора

Относительная доля несовпадений классификации с учителем для решающего правила есть: $K = \frac{m}{N}$, где $m = |\{\omega_i : g(\omega_i) \neq \widehat{g}(x(\omega_i)),\ i = 1, 2, \ldots, N\}|$.
Надежность обучения классификатора – это *вероятность* получения решающего правила с заданным качеством.

Пусть $f(x, \alpha)$ – *класс* дискриминантных функций, где $\alpha \in A$ – *параметр*. Число степеней свободы при выборе конкретной функции в классе определяется количеством параметров в векторе α, т.е. размерностью A.

Например, для классов линейных и квадратичных функций имеем:

Линейная дискриминантная *функция*: $f(x,\alpha) = \sum_{i=1}^{n} \alpha_i x_i + \alpha_0$. В таком случае имеем $n+1$ степень свободы.

Квадратичная
дискриминантная *функция*:
$f(x,\alpha) = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_{ij} x_i x_j + \sum_{i=1}^{n} \beta_i x_i + \beta_0$. В таком случае имеем n^2+n+1 степеней свободы.

С увеличением степеней свободы увеличивается способность классификатора по разделению.

Вероятностная модель

Пусть прецеденты – это результат реализации случайных величин. Рассмотрим *величину риска* (т.е. ошибки) связанной с классификацией. Определим понятия риск среднего и риска эмпирического.

Пусть на Ω заданы σ -алгебра и *мера* P. Пусть также

- x – вектор признаков,
- $\widetilde{f}$ – класс функций, из которых выбирается решающее правило,
- $f(x,\alpha)$ – решающее правило (результат классификации), которое принимает значение 0 или 1 при фиксированном векторе параметра,
- χ – характеристическая функция множества,
- A – множество параметров, описывающие различные функции в $\widetilde{f}$.

Тогда $\widehat{g} = f(x,\alpha)$, где $f \in \widetilde{f}$ и $f : X \times A \to M$, $y = g(\omega)$.

В данных обозначениях средний риск выглядит следующим образом:

$$K(\alpha) = \int_X \chi\{y \neq f(x,\alpha)\} dP.$$

Для случая двух классов, при $M = \{0,1\}$, имеем:

$$K(\alpha) = \int_\Omega (y - f(x,\alpha))^2 dP$$

или

$$K(\alpha) = \int\limits_{(X,M)} (y - f(x,\alpha))^2 dP(x,y),$$

где dP – это вероятностная *мера* на пространстве X.

Задача поиска наилучшего классификатора

Рассмотрим минимизацию функционала:

$$K(\alpha) \to \min_{\alpha \in A}$$

Задача же поиска наилучшего классификатора состоит в нахождении α^* такого, что

$$K(\alpha^*) = \min_{\alpha \in A} K(\alpha)$$

Если же минимума не существует, то надо найти α^* такое, что

$$\left| K(\alpha^*) - \inf_{\alpha \in A} K(\alpha) \right| < \delta.$$

Другими словами, необходимо решить задачу минимизации среднего риска.

Поскольку dP неизвестно, будем решать задачу минимизации *эмпирического риска*. Пусть l – число прецедентов. Тогда эмпирический риск задается выражением:

$$K_{эмп}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} |y - f(x,\alpha)| .$$

Таким образом, задача минимизации *эмпирического риска* выглядит так:

$$K_{эмп}(\alpha) \to \min_{\alpha \in A},$$

где случайные величины мы минимизируем по параметру α – любой возможный *параметр*.

В идеале надо получить взаимосвязанные оценки эмпирического и среднего риска.

Отметим, что чем меньше l, тем легче построить $f(x,\alpha)$ такую, что $K_{эмп}(\alpha)$ обращается в ноль, либо очень мало. Но при этом истинное *значение* $K(\alpha)$ может сильно отличаться от $K_{эмп}(\alpha)$. Необходимо

выбрать $f(x,\alpha)$ такую, чтобы имела *место* равномерная *сходимость* по α выражения:

$$P\left\{\sup_{\alpha}|K_{эмп}(\alpha)-K(\alpha)|>\varepsilon\right\}\xrightarrow[l\to\infty]{}0.$$

Фактически это есть *сходимость* частот к математическому ожиданию.

В дальнейшем будем считать, что в зависимости от конкретного набора прецедентов можем получить любые α. Но необходимо, чтобы полученные эмпирическое решающее хорошо работало (отражало общие свойства) для всех образов. Поэтому в формуле присутствует равномерная *сходимость*.

Сходимость эмпирического риска к среднему. Случай конечного числа решающих правил.

Пусть

- $K(\alpha)$ – математическое ожидание ошибки классификатора $f(x,\alpha)$,
- A – событие – ошибка классификатора при решающем правиле $f(x,\alpha)$,
- $P(A)$ – вероятность,
- $v(A)$ – частота в l испытаниях.

Воспользуемся неравенством Бернштейна, тогда

$$P\{|v(A)-P(A)|>\varepsilon\}\le e^{-2e^2l}$$

есть оценка – соотношение между частотой и вероятностью при заданном количестве испытаний.

Пусть ξ_j – *случайная величина*. Тогда $E(\xi_j)=0$ – математическое ожидание ξ_j, $E\xi_j^2=\delta^2$ – *дисперсия*, причем $|\xi_j|\le L$.
Обозначим $S_0=\xi_1+\xi_2+\ldots+\xi_n$. Тогда соответствующая оценка имеет вид:

$$P\left\{|S_n|>t\delta\sqrt{n}\right\}\le 2\cdot\exp\left\{-\frac{t^2}{2\cdot\left(1+\frac{a}{3}\right)}\right\},\ \text{где } a=\frac{L\cdot t}{\sqrt{n\delta}}.$$

$$l=\frac{\ln N-\ln\eta}{2e^2}\ \text{и}\ e=\sqrt{\frac{\ln N-\ln\eta}{2l}},$$

где l – необходимое количество прецедентов для обеспечения близости.

Теорема. *Пусть из множества, состоящего из* N *решающих правил, выбирается правило, частота ошибок которого на прецедентах составляет* v. *Тогда с*

вероятностью $1-\eta$ *можно утверждать, что вероятность ошибочной классификации с помощью данного правила* $f(x\alpha)$ *составит величину, меньшую* $v+e$*, если длина обучающей последовательности не меньше* $l=\frac{\ln N-\ln\eta}{2e^2}$*, где* $e=\sqrt{\frac{\ln N-\ln\eta}{2l}}$*,* η *и* e *заданы и последовательность независима*.

Данная теорема справедлива для случая конечного числа решающих правил. Вапник и Червоненкис смогли обобщить эти оценки на случай бесконечного числа решающих правил.

Случай бесконечного числа решающих правил

Введем понятие "разнообразия класса функций для бесконечного *множества*". Пусть $x_1, x_2, \ldots, x_l$ – прецеденты.

Определение. *Дихотомией называется разбиение множества на два подмножества*.

В нашем случае имеем 2^l дихотомий. Итак, пусть $f(x,\alpha),\ \alpha\in A$ – это *класс* решающих правил, причем $f(x,\alpha)=\{0,1\}$. Пусть $\Delta(x_1,x_2,\ldots,x_l)$ есть количество дихотомий на классе решающих правил. Тогда зададим энтропию следующим образом:

$$H(l)=E\{\log_2\Delta(x_1,x_2,\ldots,x_l)\},$$

где математическое ожидание берется по всем выборкам $(x_1,x_2,\ldots,x_l)$. Тогда

$$H^S(l)=E\{\log_2\Delta^S(x_1,x_2,\ldots,x_l)\}$$

есть *энтропия* класса S решающих правил на выборках длины l.

Критерий равномерной сходимости $v(\alpha)$ **к вероятностям** $P(\alpha)$

Теорема. *Для равномерной сходимости* $v(\alpha)=K_{эмп}(\alpha)$ *к* $K(\alpha)=P(\alpha)$ *по классу* $\alpha\in A$ *необходимо и достаточно, чтобы* $\frac{H(l)}{l}\xrightarrow[l\to\infty]{}0$.

Суть данного критерия – не пытаться выделить очень точный *классификатор*, так как это отдаляет от общности.

Сразу же возникает проблема необходимость перехода к бесконечным системам решающих правил. Существенно, что *значение* имеет лишь конечное *подмножество* систем решающих правил, необходимое для разделения конечного числа прецедентов.

Достаточное условие равномерной сходимости

Проверка условия критерия равномерной сходимости по вероятности затрудняется неопределенностью распределения выборки. Поэтому достаточные условия формулируются таким образом, чтобы не зависеть от распределения и при этом гарантировать равномерную *сходимость*. В таком случае вместо энтропии рассматривается величина:

$$m^S(l) = \max_{x_1,\ldots,x_l} \Delta^S(x_1, x_2, \ldots, x_l),$$

где $m^S(l)$ – это *функция* роста класса решающих функций $f(x, \alpha)$.

Т.к. логарифм максимума равен максимуму логарифмов, что, в свою *очередь*, не меньше математического ожидания от логарифма, то

$$\log_2 m^S(l) \geq H^S(l).$$

Если

$$\lim_{l\to\infty} \left\lfloor \log_2 m^S(l)/l \right\rfloor \to 0,$$

то по свойствам пределов

$$\lim_{l\to\infty} \frac{H^S(l)}{l} \to \infty.$$

Данное условие легко проверятся для различных классов решающих правил.

Другими словами $m^S(l)$ можно трактовать как максимальное число способов разделения l точек на два класса с помощью решающих правил $f(x, \alpha),\ \alpha \in A$.

Теорема. *Функция роста либо тождественно равна* 2^l *, либо, мажорируется функцией* $\sum_{i=0}^{n-1} C_l^i$ *, где* n *– минимальное значение* l *, при котором* $m^S(l) \neq 2^l$ *, т.е. либо* $m^S(l) = 2^l$ *, либо* $m^S(l) \leq \sum_{i=0}^{n-1} C_l^i$.

В свою *очередь*

$$\sum_{i=0}^{n-1} C_l^i \leq 1,5 \cdot \frac{l^{n-1}}{(n-1)!}.$$

Значит

$$m^S(l) \leq 1,5 \cdot \frac{l^{n-1}}{(n-1)!},$$

где $l = 1, 2, \ldots, n$, и $\frac{l^{n-1}}{(n-1)!}$ – степенная *функция*, мажорирующая $m^S(l)$.

Существует *максимум* $n-1$ точка, которая еще разбивается всеми возможными способами с помощью правила $f(x,\alpha)$, но никакие n точек этим свойством не обладают.

Определение. $n-1$ *называется емкостью класса решающих функций или мера разнообразия решающих правил в классе* $f()x,\alpha$ *или* VC *– размерностью класса – универсальная характеристика класса решающих функций*.

Отметим, что если $m^S(l)=2^l$ для всех l, то емкость бесконечна.

Теорема. *Если емкость класса решающих функций конечна, то всегда имеет место равномерная сходимость частот к вероятностям такое, что*

$$\lim_{l\to\infty}\left(\frac{\log_2 m^S(l)}{l}\right)\le\lim_{l\to\infty}\left(\frac{(n-1)\log l+\log 1.5}{l}\right)=0$$

и достаточное условие выполнено.

Скорость сходимости

Запишем оценку для бесконечного числа решающих правил. Ее вид аналогичен случаю конечного числа решающих правил:

$$P\left\{\sup_\alpha|P(\alpha)-v(\alpha)|>\varepsilon\right\}<3m^S(2l)\cdot e^{\frac{\varepsilon^2(l-1)}{4}}.$$

Если емкость бесконечна, то оценка тривиальная (не больше единицы). Пусть r – конечная емкость класса решающих функций. Тогда

$$P\left\{\sup_\alpha|P(\alpha)-v(\alpha)|>\varepsilon\right\}<4,5\cdot\frac{(2l)^r}{r}\cdot e^{\frac{\varepsilon^2(l-1)}{4}}.$$

Введем обозначение: $\eta=4,5\cdot\frac{(2l)^r}{r}\cdot e^{\frac{\varepsilon^2(l-1)}{4}}$,
Тогда $P\left\{\sup_\alpha|P(\alpha)-v(\alpha)|>\varepsilon\right\}<\eta$.

Отсюда следует, что $\varepsilon=\sqrt{\frac{r\left(\ln\frac{2l}{r}+1\right)-\ln\frac{\eta}{5}}{l-1}}$.

Значит, с вероятностью, превышающей $1-\eta$ качество эмпирического оптимального решающего правила отличается от истинно оптимально решающего правила не более чем на величину $\Delta=2\varepsilon$.

	Малая емкость класса решающих функций (бедный)	Большая емкость класса решающих функций (богатый)
Близость эмпирического решающего правила к оптимальному решающему правилу	Хорошая	Плохая
Качество разделения (минимизация ошибки)	Низкое	Высокое

Таким образом, необходимо минимизировать степени свободы.

Случай класса линейных решающих функций

Пусть $f(x, \alpha)$ – линейная решающая *функция*, m – *размерность* пространства.

Как уже отмечалось выше, имеем 2^l дихотомий, где l – *длина* выборки. Хотим выяснить, какое количество дихотомий реализуется с помощью гиперплоскостей?

Максимальное число точек в пространстве размерности Ь, которое с помощью гиперплоскостей можно разбить всеми возможными способами на два класса есть $\text{Ь} + 1$. Если $m^S(l) \leq 1,5 \cdot \frac{l^{m+1}}{(m+1)!}$, то линейный риск будет равномерно сходиться к среднему риску. Емкость класса конечна и равна $m + 1$.

Алгоритмы поиска схожих объектов в крупных массивах данных.

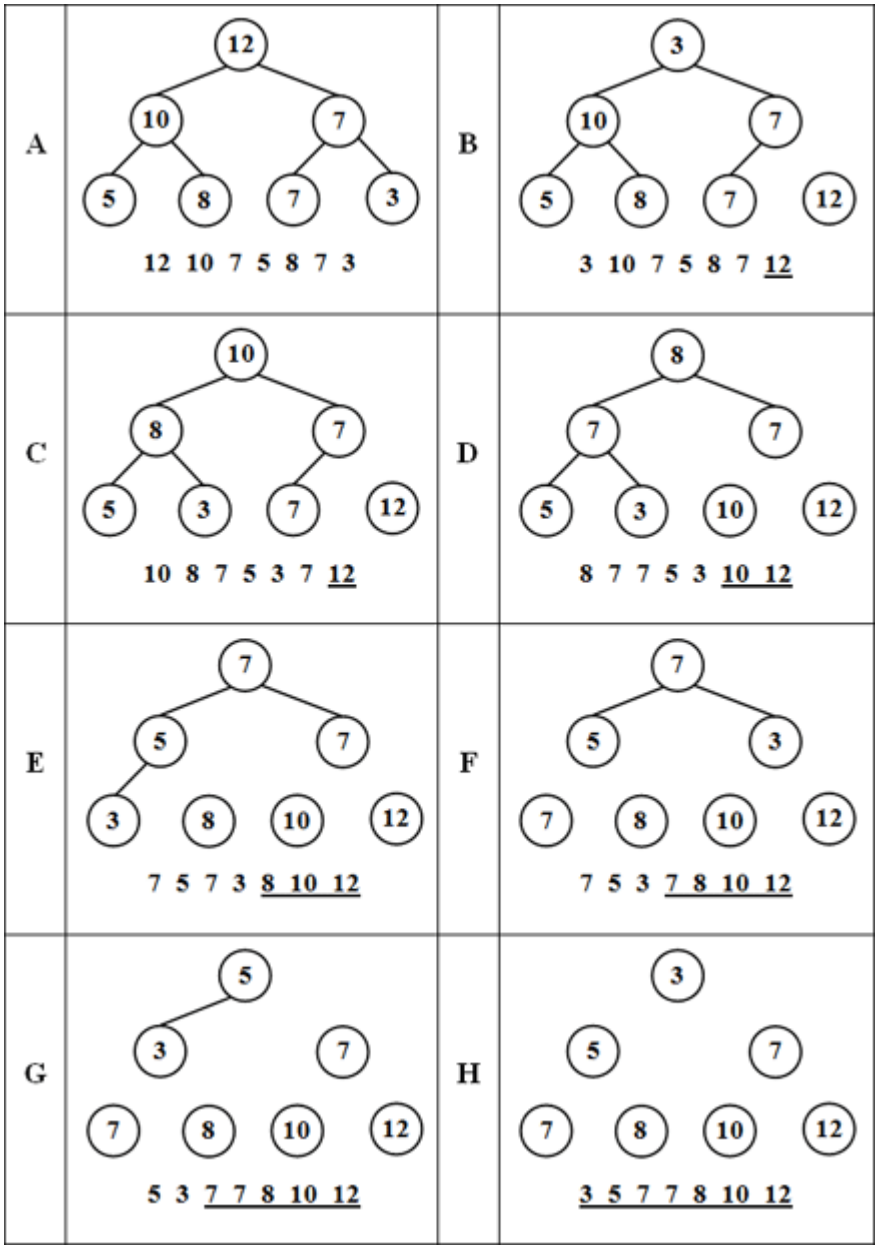
Вспомним недостатки алгоритма К-ближайших соседей, когда приходится решать сложные задачи для хранения обучающих выборок целиком, организовывать поиск ближайшего соседа путем сравнения классифицируемого объекта со всеми объектами выборки за $O(\ell)$ операций. Для задач с большими выборками или высокой частотой запросов это может оказаться сложно с точки зрения вычислительных ресурсов. Для выхода из подобных ситуаций следует разработать эффективные алгоритмы поиска ближайших соседей, основанные на известных методах сортировки сведений.

Зададим параметры, с помощью которых можно оценивать алгоритмы поиска соседей:

- *Время поиска* – основной параметр, характеризующий быстродействие алгоритма.

- *Память* – один из параметров, который характеризуется тем, что ряд алгоритмов поиска требуют выделения дополнительной памяти под временное хранение данных. При оценке используемой памяти не будет учитываться место, которое занимает исходный массив данных и независящие от входной последовательности затраты, например, на хранение кода программы.
- *Устойчивость* – это параметр, который отвечает за то, что поиск не меняет взаимного расположения равных элементов.
- *Естественность поведения* – параметр, которой указывает на эффективность метода при обработке уже отобранных, или частично пересортированных данных. Алгоритм ведет себя естественно, если учитывает эту характеристику входной последовательности и работает лучше.

Можно использовать поиск данных на основе сортировки пирамиды.



Пирамида (сортирующее дерево) – двоичное дерево с упорядоченными листьями (корень дерева – наименьший или наибольший элемент). Пирамиду можно представить в виде массива. Первый элемент пирамиды является наименьшим или наибольшим, что зависит от ключа поиска.

Алгоритм пирамидальной сортировки.

Шаг 1. Преобразовать массив в пирамиду.

Шаг 2. Использовать алгоритм сортировки пирамиды.

Алгоритм преобразования массива в пирамиду (построение пирамиды). Пусть дан массив x[1],x[2],...,x[n].

Шаг 1. Устанавливаем k=n/2.

Шаг 2. Перебираем элементы массива в цикле справа налево для i=k,k-1,...,1. Если неравенства $x_i > x_{2i}$, $x_i > x_{2i+1}$ не выполняются, то повторяем перестановки x_i с наибольшим из потомков. Перестановки завершаются при выполнении неравенств $x_i > x_{2i}$, $x_i > x_{2i+1}$.

Алгоритм сортировки пирамиды.

Рассмотрим массив размерности n, который представляет пирамиду x[1],x[2],...,x[n].

Шаг 1. Переставляем элементы x[1] и x[n].

Шаг 2. Определяем n=n-1. Это эквивалентно тому, что в массиве из дальнейшего рассмотрения исключается элемент x[n].

Шаг 3. Рассматриваем массив x[1],x[2],...,x[n-1], который получается из исходного за счет исключения последнего элемента. Данный массив из-за перестановки элементов уже не является пирамидой. Но такой массив легко преобразовать в пирамиду. Это достигается повторением перестановки значения элемента из x[1] с наибольшим из потомков. Такая перестановка продолжается до тех пор, пока элемент из x[1] не окажется на месте элемента x[i] и при этом будут выполняться неравенства x[i] > x[2i], x[i] > x[2i+1]. Тем самым определяется новое место для значения первого элемента из x[1].

Шаг 4. Повторяем шаги 2, 3, 4 до тех пор, пока не получим n=1. Произвольный массив можно преобразовать в пирамиду.

Построение пирамиды, ее сортировка и "просеивание" элементов реализуются с помощью рекурсии. Базой рекурсии при этом выступает пирамида из одного элемента, а сортировка и просеивание элементов сводятся посредством декомпозиции к аналогичным действиям с пирамидой из n-1 элемента.

Время работы алгоритма пирамидальной сортировки без учета времени построения пирамиды будет определяться по формуле $T_1(n)=O(n\log n)$.

Построение пирамиды занимает $T_2(n)=O(n)$ операций за счет того, что реальное время выполнения функции построения зависит от высоты уже созданной части пирамиды.

Тогда общее время сортировки (с учетом построения пирамиды) будет равно: $T(n)=T_1(n)+T_2(n)=O(n)+O(n\log n)=O(n\log n)$.

Пирамидальная сортировка не использует дополнительной памяти. Метод не является устойчивым: по ходу работы массив так "перетряхивается", что исходный порядок элементов может измениться случайным образом. Поведение неестественно: частичная упорядоченность массива никак не учитывается. Данная сортировка на почти отсортированных массивах работает также долго, выигрыш ее получается только на больших n.

Поиск схожих объектов путем слияния массивов

Алгоритм был изобретен Джоном фон Нейманом в 1945 году. Он является одним из самых быстрых способов сортировки.

Слияние – это объединение двух или более упорядоченных массивов в один упорядоченный.

Сортировка слиянием является одним из самых простых алгоритмов сортировки (среди быстрых алгоритмов). Особенностью этого алгоритма является то, что он работает с элементами массива преимущественно последовательно, благодаря чему именно этот алгоритм используется при сортировке в системах с различными аппаратными ограничениями (например, при сортировке данных на жестком диске). Кроме того, сортировка слиянием является алгоритмом, который может быть эффективно использован для сортировки таких структур данных, как связанные списки.

Данный алгоритм применяется тогда, когда есть возможность использовать для хранения промежуточных результатов память, сравнимую с размером исходного массива. Он построен на принципе "разделяй и властвуй". Сначала задача разбивается на несколько подзадач меньшего размера. Затем эти задачи решаются с помощью рекурсивного вызова или непосредственно, если их размер достаточно мал. Далее их решения комбинируются, и получается решение исходной задачи.

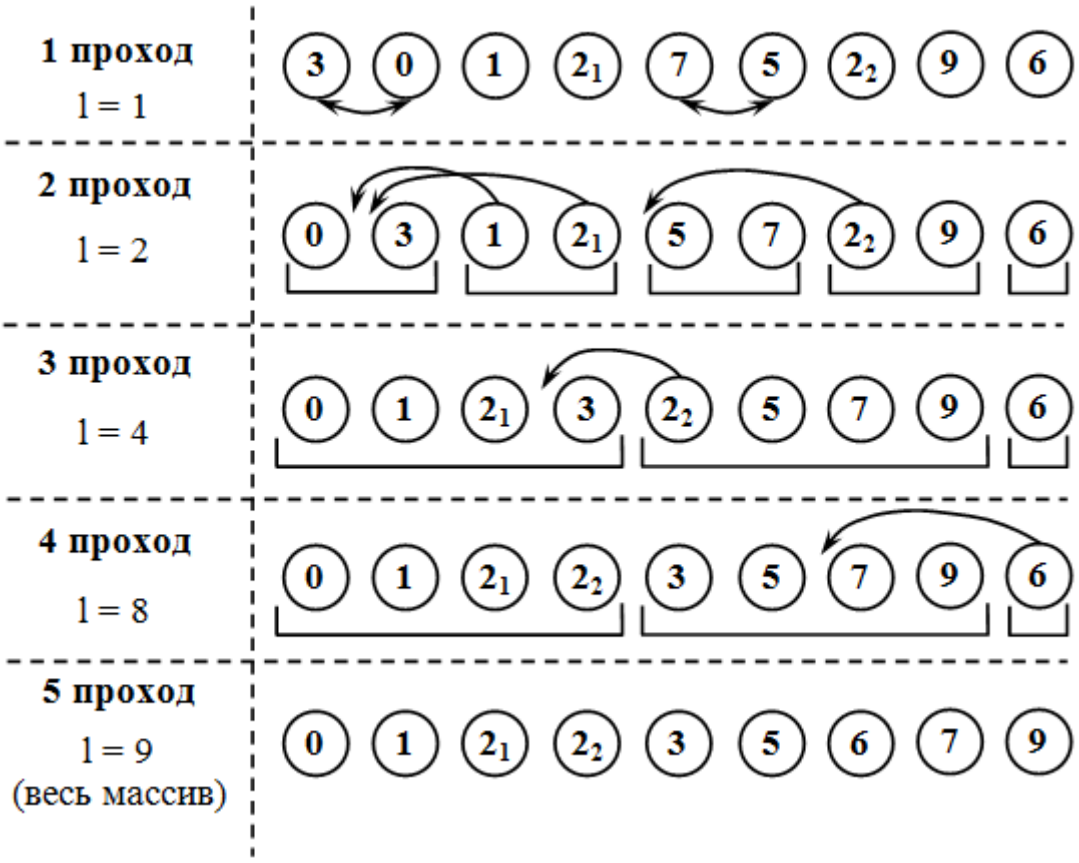
Процедура слияния требует два отсортированных массива. Заметим, что массив из одного элемента по определению является отсортированным.

Алгоритм сортировки слиянием

Шаг 1. Разбить имеющиеся элементы массива на пары и осуществить слияние элементов каждой пары, получив отсортированные цепочки длины 2 (кроме, быть может, одного элемента, для которого не нашлось пары).

Шаг 2. Разбить имеющиеся отсортированные цепочки на пары, и осуществить слияние цепочек каждой пары.

Шаг 3. Если число отсортированных цепочек больше единицы, перейти к шагу 2.



Недостаток алгоритма заключается в том, что он требует дополнительную память размером порядка n (для хранения вспомогательного массива). Кроме того, он не гарантирует сохранение порядка элементов с одинаковыми значениями. Но его временная сложность всегда пропорциональна O(n log n).

Быстрая сортировка Хоара

Метод быстрой сортировки был впервые описан Ч.А.Р. Хоаром в 1962 году. *Быстрая сортировка* – это общее название ряда алгоритмов, которые отражают различные подходы к получению критичного параметра, влияющего на производительность метода.

При общем рассмотрении алгоритма быстрой сортировки, отметим, что этот метод основывается на последовательном разделении сортируемого набора данных на блоки меньшего размера таким образом, что между значениями разных блоков обеспечивается отношение упорядоченности (для любой пары блоков все значения одного из этих блоков не превышают значений другого блока).

Опорным (ведущим) элементом называется некоторый элемент массива, который выбирается определенный образом. С точки зрения корректности алгоритма выбор опорного элемента безразличен. С точки зрения повышения эффективности алгоритма выбираться должна медиана, но без дополнительных сведений о сортируемых данных ее обычно невозможно получить. Необходимо выбирать постоянно один и тот же элемент (например, средний или последний по положению) или выбирать элемент со случайно выбранным индексом.

Алгоритм быстрой сортировки Хоара

Пусть дан массив x[n] размерности n.

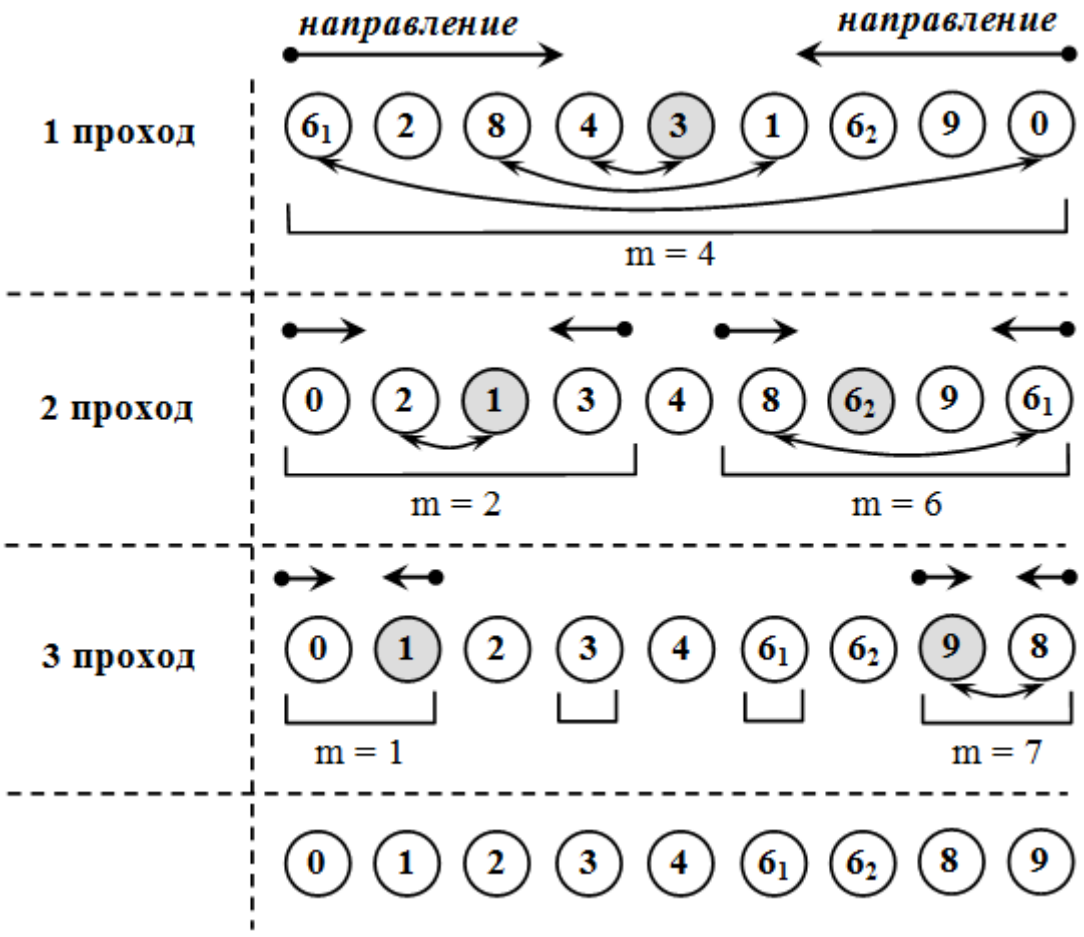
Шаг 1. Выбирается опорный элемент массива.

Шаг 2. Массив разбивается на два – левый и правый – относительно опорного элемента. Реорганизуем массив таким образом, чтобы все элементы, меньшие опорного элемента, оказались слева от него, а все элементы, большие опорного – справа от него.

Шаг 3. Далее повторяется шаг 2 для каждого из двух вновь образованных массивов. Каждый раз при повторении преобразования очередная часть массива разбивается на два меньших и т. д., пока не получится массив из двух элементов.

Быстрая сортировка стала популярной прежде всего потому, что ее нетрудно реализовать, она хорошо работает на различных видах входных данных и во многих случаях требует меньше затрат ресурсов по сравнению с другими методами сортировки.

Выберем в качестве опорного элемент, расположенный на средней позиции.



Эффективность быстрой сортировки в значительной степени определяется правильностью выбора опорных (ведущих) элементов при формировании блоков. В худшем случае трудоемкость метода имеет ту же сложность, что и пузырьковая сортировка, то есть порядка $O(n^2)$. При оптимальном выборе ведущих элементов, когда разделение каждого блока происходит на равные по размеру части, трудоемкость алгоритма совпадает с быстродействием наиболее эффективных способов сортировки, то

есть порядка O(n log n). В среднем случае количество операций, выполняемых алгоритмом быстрой сортировки, определяется выражением T(n) = O(1.4n log n)

Быстрая сортировка является наиболее эффективным алгоритмом из всех известных методов сортировки, но все усовершенствованные методы имеют один общий недостаток – невысокую скорость работы при малых значениях n.

Рекурсивная реализация быстрой сортировки позволяет устранить этот недостаток путем включения прямого метода сортировки для частей массива с небольшим количеством элементов. Анализ вычислительной сложности таких алгоритмов показывает, что если подмассив имеет девять или менее элементов, то целесообразно использовать прямой метод (сортировку простыми вставками).

Список литературы

1 Гладков Л. А., Курейчик В. В., Курейчик В. М. Генетические алгоритмы. — М.: Физматлит, 2006.— 320 с.

2 Ермаков С. М., Михайлов Г. А. Курс статистического моделирования. — М.: Наука, 1976.

3 Колмогоров А. Н. О представлении непрерывных функций нескольких переменных в виде суперпозиции непрерывных функций одного переменного // Докл. АН СССР. — 1958. — Т. 114, № 5. — С. 953–956.

4 Мерков А. Б. О статистическом обучении. — Лаборатория распознавания образов МЦНМО. — 2006. http://www.recognition.mccme.ru/pub/RecognitionLab.html/slt.pdf.

5 Шурыгин А. М. Прикладная стохастика: робастность, оценивание, прогноз. — М.: Финансы и статистика, 2000.

6 Норушис А. Построение логических (древообразных) классификаторов методами нисходящего поиска (обзор) // Статистические проблемы управления. Вып. 93 / Под ред. Ш. Раудис. — Вильнюс, 1990.— С. 131–158.

7 Cohen W. W., Singer Y. A simple, fast and effective rule learner // Proc. of the 16 National Conference on Artificial Intelligence. — 1999.— Pp. 335–342.

8 http://citeseer.ist.psu.edu/198445.html.

9 Ивахненко А. Г., Юрачковский Ю. П. Моделирование сложных систем по экспериментальным данным. — М.: Радио и связь, 1987.

10 Рудаков К. В. Универсальные и локальные ограничения в проблеме коррекции эвристических алгоритмов // Кибернетика.— 1987.— № 2. — С. 30–35. http://www.ccas.ru/frc/papers/rudakov87universal.pdf.

11 Роберт Седжвик, Алгоритмы на C++. Фундаментальные алгоритмы и структуры данных / пер. с англ. (Algorithms in C++) — М.: Вильямс. — 2011 г. — 1056 с.

12