

# Метрические алгоритмы классификации

К. В. Воронцов

23 февраля 2010 г.

Материал находится в стадии разработки, может содержать ошибки и неточности. Автор будет благодарен за любые замечания и предложения, направленные по адресу [vokov@forecsys.ru](mailto:vokov@forecsys.ru), либо высказанные в обсуждении страницы «Машинное обучение (курс лекций, К.В.Воронцов)» вики-ресурса [www.MachineLearning.ru](http://www.MachineLearning.ru).

Перепечатка фрагментов данного материала без согласия автора является плагиатом.

## Содержание

<b>1</b>	<b>Метрические алгоритмы классификации</b>	<b>2</b>
1.1	Метод ближайшего соседа и его обобщения	2
1.1.1	Обобщённый метрический классификатор	2
1.1.2	Метод ближайших соседей	3
1.1.3	Метод парзеновского окна	5
1.1.4	Метод потенциальных функций	6
1.2	Отбор эталонных объектов	7
1.2.1	Понятие отступа объекта	8
1.2.2	Алгоритм STOLP для отбора эталонных объектов	9
1.3	Оптимизация метрик	10

# 1 Метрические алгоритмы классификации

Во многих прикладных задачах измерять степень сходства объектов существенно проще, чем формировать признаковые описания. Например, такие сложные объекты, как фотографии лиц, подписи, временные ряды или первичные структуры белков естественнее сравнивать непосредственно с друг с другом путём некоторого «наложения с выравниванием», чем изобретать какие-то признаки и сравнивать признаковые описания. Если мера сходства объектов введена достаточно удачно, то, как правило, оказывается, что *схожим объектам чаще соответствуют схожие ответы*. В задачах классификации это означает, что классы образуют компактно локализованные подмножества. Это предположение принято называть *гипотезой компактности*<sup>1</sup>. 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). В англоязычной литературе употребляются термины *similarity-based learning* или *distance-based learning*.

## §1.1 Метод ближайшего соседа и его обобщения

Пусть на множестве объектов  $X$  задана функция расстояния  $\rho: X \times X \rightarrow [0, \infty)$ . Существует целевая зависимость  $y^*: X \rightarrow Y$ , 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  $a: X \rightarrow Y$ , аппроксимирующий целевую зависимость  $y^*(x)$  на всём множестве  $X$ .

### 1.1.1 Обобщённый метрический классификатор

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

$$\rho(u, x_u^{(1)}) \leq \rho(u, x_u^{(2)}) \leq \dots \leq \rho(u, x_u^{(\ell)}),$$

где через  $x_u^{(i)}$  обозначается  $i$ -й сосед объекта  $u$ . Соответственно, ответ на  $i$ -м соседе объекта  $u$  есть  $y_u^{(i)} = y^*(x_u^{(i)})$ . Фактически, любой объект  $u \in X$  порождает свою перенумерацию выборки  $X^\ell = \{(x_u^{(1)}, y_u^{(1)}), \dots, (x_u^{(\ell)}, y_u^{(\ell)})\}$ .

**Опр. 1.1.** *Метрический алгоритм классификации с обучающей выборкой  $X^\ell$  относит объект  $u$  к тому классу  $y \in Y$ , для которого суммарный вес ближайших обучающих объектов  $\Gamma_y(u, X^\ell)$  максимален:*

$$a(u; X^\ell) = \arg \max_{y \in Y} \Gamma_y(u, X^\ell); \quad \Gamma_y(u, X^\ell) = \sum_{i=1}^{\ell} [y_u^{(i)} = y] w(i, u); \quad (1.1)$$

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  $y$ .

<sup>1</sup> 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.

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  $X^\ell$  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  $X^\ell$ , 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 метод обучения заранее строит функцию, глобально аппроксимирующую всю 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:

$$\begin{aligned}
 w(i, u) &= [i = 1] && \text{— метод ближайшего соседа (1NN);} \\
 w(i, u) &= [i \leq k] && \text{— метод } k \text{ ближайших соседей (kNN);} \\
 w(i, u) &= [i \leq k] q^i && \text{— метод } k \text{ взвешенных ближайших соседей;} \\
 w(i, u) &= K \left( \frac{\rho(u, x_u^{(i)})}{h} \right) && \text{— метод парзеновского окна ширины } h; \\
 w(i, u) &= K \left( \frac{\rho(u, x_u^{(i)})}{\rho(u, x_u^{(k+1)})} \right) && \text{— метод парзеновского окна переменной ширины;} \\
 w(i, u) &= \gamma_u^{(i)} K \left( \frac{\rho(u, x_u^{(i)})}{h_u^{(i)}} \right) && \text{— метод потенциальных функций.}
 \end{aligned}$$

### 1.1.2 Метод ближайших соседей

**Алгоритм ближайшего соседа** (nearest neighbor, NN) является самым простым алгоритмом классификации. Он относит классифицируемый объект  $u \in X^\ell$  к тому классу, которому принадлежит ближайший обучающий объект:

$$a(u; X^\ell) = y_u^{(1)}.$$

Обучение NN сводится к запоминанию выборки  $X^\ell$ . 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 те окружающие его объекты, для которых он окажется ближайшим, также будут классифицированы неверно.

- Отсутствие параметров, которые можно было бы настраивать по выборке. Алгоритм полностью зависит от того, насколько удачно выбрана метрика  $\rho$ .
- В результате — низкое качество классификации.

**Алгоритм  $k$  ближайших соседей** ( $k$  nearest neighbors,  $k$ NN). Чтобы сгладить влияние шумовых выбросов, будем классифицировать объекты путём *голосования* по  $k$  ближайшим соседям. Каждый из соседей  $x_u^{(i)}$ ,  $i = 1, \dots, k$  голосует за отнесение объекта  $u$  к своему классу  $y_u^{(i)}$ . Алгоритм относит объект  $u$  к тому классу, который наберёт большее число голосов:

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

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.

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

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  $\text{LOO}(k)$  будет достигаться при  $k = 1$ .

Как правило, функционал  $\text{LOO}(k)$  имеет чётко выраженный минимум, Рис. ??.

Существует и альтернативная трактовка метода  $k$ NN: в каждом классе выбирается  $k$  ближайших к  $u$  объектов, и объект  $u$  относится к том классу, для которого среднее расстояние до  $k$  ближайших соседей минимально.

**Алгоритм  $k$  взвешенных ближайших соседей.** Недостаток  $k$ NN в том, что максимальная сумма голосов может достигаться на нескольких классах одновременно. В задачах с двумя классами этого можно избежать, если брать только нечётные значения  $k$ . 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:

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

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 нелинейную 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  $k$ .

## Достоинства метрических алгоритмов.

- Простота реализации и возможность введения различных модификаций весовой функции  $w(i, u)$ .
- Возможность интерпретировать классификацию объекта путём предъявления пользователю ближайшего объекта или нескольких. «Прецедентная» логика работы алгоритма хорошо понятна экспертам в таких предметных областях, как медицина, биометрия, юриспруденция, и др.

## Недостатки простейших метрических алгоритмов типа $k$ NN.

- 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  $\rho$ ) 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.

### 1.1.3 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  $[0, \infty)$ , и рассмотрим алгоритм

$$a(u; X^\ell, h, K) = \arg \max_{y \in Y} \sum_{i=1}^{\ell} [y_u^{(i)} = y] K\left(\frac{\rho(u, x_u^{(i)})}{h}\right). \quad (1.2)$$

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 обучающего объекта  $x_i$  объект  $u$  «притягивается» к классу  $y_i$ . Мы пришли к этому алгоритму чисто эвристическим путём, однако он имеет более строгое обоснование в байесовской теории классификации, и тесно связан с непараметрическим оцениванием плотности распределения по Парзену-Розенблатту, см. ???. Фактически, формула (1.2) 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

$$a(u; X^\ell, k, K) = \arg \max_{y \in Y} \sum_{i=1}^k [y_u^{(i)} = y] K \left( \frac{\rho(u, x_u^{(i)})}{\rho(u, x_u^{(k+1)})} \right). \quad (1.3)$$

Выбор ядра  $K$  слабо влияет на качество классификации. На практике ядро либо задаётся априори, либо выбирается по скользящему контролю из нескольких стандартных ядер. Более подробно выбор ядра обсуждается в разделе ???. Сейчас отметим лишь то, что выбор финитного ядра позволяет свести классификацию объекта  $u$  к поиску  $k$  его ближайших соседей, тогда как при не финитном ядре (например, гауссовском) требуется перебор всей обучающей выборки, что может приводить к неприемлемым затратам времени на больших выборках.

#### 1.1.4 Метод потенциальных функций

Пойдём по пути дальнейшей модификации парзеновского алгоритма (1.2). До сих пор мы рассматривали метрические алгоритмы, в которых центр радиального ядра  $K_h(u, x) = K(\frac{1}{h}\rho(u, x))$  помещался в классифицируемый объект  $u$ , затем суммировались вклады ближайших к нему обучающих объектов  $x = x_u^{(i)}$ . В силу симметричности функции расстояния  $\rho(u, x)$  возможен и другой, двойственный, взгляд на метрическую классификацию. Допустим теперь, что ядро помещается в каждый обучающий объект  $x_i$  и «притягивает» объект  $u$  к классу  $y_i$ , если он попадает в его окрестность радиуса  $h_i$ :

$$a(u; X^\ell) = \arg \max_{y \in Y} \sum_{i=1}^{\ell} [y_i = y] \gamma_i K \left( \frac{\rho(u, x_i)}{h_i} \right), \quad \gamma_i \geq 0, h_i > 0. \quad (1.4)$$

По сути, эта формула отличается от (1.3) только тем, что здесь ширина окна  $h_i$  зависит от обучающего объекта  $x_i$ , а не от классифицируемого объекта  $u$ .

Данная идея лежит в основе *метода потенциальных функций* [1] и имеет прямую физическую аналогию с электрическим потенциалом. При  $Y = \{-1, +1\}$  обучающие объекты можно понимать как положительные и отрицательные электрические заряды; коэффициенты  $\gamma_i$  — 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 (1.4) имеет достаточно богатый набор из  $2\ell$  параметров  $\gamma_i, h_i$ . Простейший и исторически самый первый метод их настройки представлен в Алгоритме 1.1. Он настраивает только веса  $\gamma_i$ , предполагая, что радиусы потенциалов  $h_i$

---

**Алгоритм 1.1.** Метод потенциальных функций
 

---

**Вход:**
 $X^\ell$  — обучающая выборка;
**Выход:**
 Коэффициенты  $\gamma_i$ ,  $i = 1, \dots, \ell$  в (1.4);
 

---

- 1: Инициализация:  $\gamma_i = 0$ ;  $i = 1, \dots, \ell$ ;
  - 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
- 

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  $\gamma_i$  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. Условия его сходимости и многочисленные вариации исследованы в [1].

**Достоинство Алгоритма 1.1** 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 1.1 представляет скорее исторический интерес.

**Недостатков у Алгоритма 1.1** довольно много: он медленно сходится; результат обучения зависит от порядка предъявления объектов; слишком грубо (с шагом 1) настраиваются веса  $\gamma_i$ ; центры потенциалов почему-то помещаются только в обучающие объекты; задача минимизации числа потенциалов (ненулевых  $\gamma_i$ ) вообще не ставится; вообще не настраиваются параметры  $h_i$ . В результате данный алгоритм не может похвастаться высоким качеством классификации.

Более современный метод настройки линейных комбинаций потенциальных функций, основанный на EM-алгоритме, рассматривается в ???. Он позволяет оптимизировать и ширину каждого потенциала, и положения центров, и даже количество потенциалов. Изменилось и название метода: теперь потенциальные функции предпочитают называть *радиальными базисными функциями*. Формально этот метод не подчиняется общей формуле (1.1), так как классифицируемый объект  $u$  сравнивается не с обучающими объектами, а с центрами потенциалов, которые в общем случае не совпадают ни с одним из обучающих объектов.

## §1.2 Отбор эталонных объектов

Обычно объекты обучения не являются равноценными. Среди них могут находиться типичные представители классов — *эталонные*. 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.

### 1.2.1 Понятие отступа объекта

Общая формула (1.1) позволяет ввести характеристику объектов, показывающую, насколько глубоко объект погружён в свой класс.

**Опр. 1.2.** *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.
- *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 (1.1), в частности, неудачная конструкция метрики  $\rho$ .
- *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  $k$ NN нецелесообразно. Если значения отступов концентрируются вблизи нуля, то ждать надёжной классификации не приходится, так как слишком много объектов оказываются в пограничной «зоне неуверенности».

### 1.2.2 Алгоритм STOLP для отбора эталонных объектов

Идея отбора эталонов реализована в алгоритме STOLP [2]. Мы рассмотрим его обобщённый вариант с произвольной весовой функцией  $w(i, u)$ . Будем строить метрический алгоритм  $a(u; \Omega)$  вида (1.1), где  $\Omega \subseteq X^\ell$  — множество эталонов.

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

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

После этого начинается процесс последовательного «жадного» наращивания множества  $\Omega$ . На каждом шаге к  $\Omega$  присоединяется объект  $x_i$ , имеющий минимальное значение отступа. Так продолжается до тех пор, пока число ошибок не окажется меньше заданного порога  $\ell_0$ . 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  $\Omega$  одним «центральным» эталонным объектом и массой «приграничных» эталонных объектов, на которых отступ принимал наименьшие значения в процессе итераций. Параметр  $\delta$  позволяет регулировать ширину зазора между эталонами разных классов. Чем больше  $\delta$ , тем дальше от границы классов будут располагаться «приграничные» эталоны, и тем более простой, менее «изрезанной» будет граница между классами.

---

**Алгоритм 1.2.** Отбор эталонных объектов STOLP
 

---

**Вход:**

- $X^\ell$  — обучающая выборка;
- $\delta$  — порог фильтрации выбросов;
- $\ell_0$  — 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:  
 $\Omega := \{\arg \max_{x_i \in X_y^\ell} M(x_i, X^\ell) \mid y \in Y\}$ ;
  - 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  $\Omega$  объект с наименьшим отступом:  
 $x_i := \arg \min_{x \in E} M(x, \Omega)$ ;    $\Omega := \Omega \cup \{x_i\}$ ;
- 

В описанном варианте алгоритм STOLP имеет относительно низкую эффективность. Для присоединения очередного эталона необходимо перебрать множество объектов  $X^\ell \setminus \Omega$ , и для каждого вычислить отступ относительно множества эталонов  $\Omega$ . Общее число операций составляет  $O(|\Omega|^2 \ell)$ . Для ускорения алгоритма можно 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  $\delta$ . При программной реализации имеет смысл определить отдельную процедуру для обновления значений всех отступов  $M_i = M(x_i, \Omega)$  при текущем наборе эталонов  $\Omega$ ; а в самом алгоритме гибко принимать решение о том, в какие моменты вызывать эту процедуру. Реализация этих идей не показана в Алгоритме 1.2, 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.

### §1.3 Оптимизация метрик

Выбор метрики  $\rho(x, x')$  в пространстве объектов  $X$  является серьёзной проблемой в задачах классификации, кластеризации и непараметрической регрессии. Метрика — это математическая модель сходства объектов, и её выбор во многих случаях

не однозначен. В то же время, в большинстве метрических алгоритмов предполагается, что метрика фиксирована. В последнее время всё чаще применяются методы, в которых метрика настраивается по обучающей выборке.

Если объекты описываются набором признаков  $f_1(x), \dots, f_n(x)$ , то естественным обобщением евклидовой метрики является *взвешенная метрика Минковского*:

$$\rho(x, x') = \left( \sum_{j=1}^n w_j |f_j(x) - f_j(x')|^p \right)^{\frac{1}{p}},$$

где  $w_j$  — веса признаков, показатель степени  $p$  может принимать значения от 0 до  $\infty$  включительно. В пределе при  $p \rightarrow \infty$  получаем взвешенную равномерную метрику

$$\rho(x, x') = \max_{j=1, \dots, n} w_j |f_j(x) - f_j(x')|.$$

Веса признаков часто задают из соображений нормировки:  $M_j = \max_{i=1, \dots, \ell} |f_j(x_i)|$ ,  $w_j = M_j^{-p}$ , однако такой способ не позволяет учитывать относительную важность (ценность, информативность) признаков, а в многомерных пространствах приводит к проблеме «*проклятия размерности*» (curse of dimensionality) — чем выше размерность пространства  $n$ , тем более устойчивой становится сумма разностей  $\sum_j |f_j(x) - f_j(x')|$ . Этот факт аналогичен закону больших чисел в теории вероятностей. Увеличение размерности в пределе  $n \rightarrow \infty$  приводит к тому, что все точки выборки становятся почти одинаково далеки друг от друга. Становится невозможно выделить локальную окрестность объекта, теряется информация о структуре метрического пространства, что затрудняет решение задач классификации, кластеризации и непараметрической регрессии.

К настоящему времени разработаны различные методы отбора информативных признаков и оптимизации коэффициентов  $w_j$  во взвешенных метриках.

## Список литературы

- [1] Айзерман М. А., Браверман Э. М., Розоноэр Л. И. Метод потенциальных функций в теории обучения машин. — М.: Наука, 1970. — 320 pp.
- [2] Загоруйко Н. Г. Прикладные методы анализа данных и знаний. — Новосибирск: ИМ СО РАН, 1999.