Лекции по метрическим алгоритмам классификации

К. В. Воронцов

16 апреля 2008 г.

Материал находится в стадии разработки, может содержать ошибки и неточности. Автор будет благодарен за любые замечания и предложения, направленные по адресу `voron@ccas.ru`. Перепечатка любых фрагментов данного материала без согласия автора является плагиатом.

Содержание

1 Метрические алгоритмы классификации

Во многих прикладных задачах измерять степень сходства объектов существенно проще, чем формировать признаковые описания. Например, гораздо легче сравнить две фотографии и сказать, что они принадлежат одному человеку, чем понять, на основании каких признаков они схожи. Такие ситуации часто возникают при распознавании изображений, временных рядов или символьных последовательностей. Они характеризуются тем, что «сырые» исходные данные не годятся в качестве признаковых описаний, но в то же время, существуют эффективные и содержательно обоснованные способы оценить степень сходства любой пары «сырых» описаний.

Есть ещё одна характерная особенность этих задач. Если мера сходства введена достаточно удачно, то оказывается, что *схожим объектам, как правило, соответствуют схожие ответы*. В задачах классификации это означает, что схожие объекты гораздо чаще лежат в одном классе, чем в разных. Если задача в принципе поддаётся решению, то граница между классами не может «проходить повсюду»; классы образуют компактно локализованные подмножества в пространстве объектов. Это предположение принято называть *гипотезой компактности*[1].

Для формализации понятия «сходства» вводится функция расстояния или метрика $\rho(x, x')$ в пространстве объектов X. Алгоритмы, основанные на анализе сходства объектов, часто называют *метрическими*, даже в тех случаях, когда функция ρ не удовлетворяет всем аксиомам метрики (например, аксиоме треугольника).

Различные метрические алгоритмы классификации рассматриваются в §1.1. Основной вопрос §1.2 — откуда берётся метрика, и как строить «хорошие» метрики в конкретных задачах. В §1.3 выводятся оценки обобщающей способности метрических алгоритмов, и на их основе строится ещё несколько эффективных алгоритмов.

§1.1 Метрические алгоритмы классификации

Напомним основные обозначения и постановку задачи классификации.

Имеется пространство объектов X и конечное множество имён классов Y. На множестве X задана функция расстояния $\rho\colon X \times X \to [0, \infty)$. Существует целевая зависимость $y^*\colon X \to Y$, значения которой известны только на объектах обучающей выборки $X^\ell = (x_i, y_i)_{i=1}^\ell$, $y_i = y^*(x_i)$. Требуется построить алгоритм классификации $a\colon X \to Y$, аппроксимирующий целевую зависимость $y^*(x)$ на всём множестве X.

1.1.1 Метод ближайших соседей и его обобщения

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

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

где через $x_{i,u}$ обозначается i-й сосед объекта u. Аналогичное обозначение введём и для ответа на i-м соседе: $y_{i,u} = y^*(x_{i,u})$. Каждый объект $u \in X$ порождает свою перенумерацию выборки $x_{1,u}, \ldots, x_{\ell,u}$.

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

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:

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

Обучение NN сводится к элементарному 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 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 ρ.
- В результате — низкое качество классификации.

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

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

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.

$$\mathrm{LOO}(k, X^\ell) = \sum_{i=1}^{\ell} \Bigl[a\bigl(x_i; X^\ell \backslash \{x_i\}, k\bigr) \neq y_i \Bigr] \to \min_k .$$

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

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

Недостаток kNN 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:

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

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.

Метод парзеновского окна. Ещё один способ задать веса соседям — определить w_i как функцию от расстояния $\rho(u, x_{i,u})$, а не от ранга соседа i. Введём *функцию ядра* $K(z)$, невозрастающую на $[0, \infty)$, и рассмотрим алгоритм

$$a(u; X^\ell, h, K) = \arg\max_{y \in Y} \sum_{i=1}^{\ell} [y_{i,u} = y]\, K\left(\frac{\rho(u, x_{i,u})}{h}\right). \tag{1.1}$$

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. **??**.

Параметр h можно задавать априори или определять по скользящему контролю. Зависимость $\mathrm{LOO}(h)$, 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: $h(u) = \rho(u, x_{k+1,u})$. Тогда алгоритм принимает вид

$$a(u; X^\ell, k, K) = \arg\max_{y \in Y} \sum_{i=1}^{k} [y_{i,u} = y]\, K\left(\frac{\rho(u, x_{i,u})}{\rho(u, x_{k+1,u})}\right). \tag{1.2}$$

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 требуется полный перебор всей обучающей выборки. На сверхбольших выборках данных полный перебор может приводить к неприемлемым затратам времени.

Обобщённый метрический классификатор. Описанные выше алгоритмы классификации являются частными случаями одной общей формулы.

Пусть задана весовая функция $w(i,u)$, которая оценивает степень важности i-го соседа для классификации объекта u. Естественно полагать, что эта функция неотрицательна и не возрастает по i. Согласно гипотезе компактности чем меньше i, тем ближе объекты u и $x_{i,u}$, тем выше шансы, что они принадлежат одному классу.

Опр. 1.1. *Метрическим алгоритмом классификации с обучающей выборкой X^ℓ будем называть отображение вида*

$$a(u;X^\ell) = \arg\max_{y\in Y} \underbrace{\sum\nolimits_{i=1}^{\ell} [y_{i,u}=y]\, w(i,u)}_{\Gamma_y(u,X^\ell)}. \tag{1.3}$$

Алгоритм a относит объект u к тому классу, для которого суммарный вес ближайших обучающих объектов $\Gamma_y(u) \equiv \Gamma_y(u,X^\ell)$ максимален.

Выбирая весовую функцию $w(i,u)$, можно получать различные типы метрических алгоритмов классификации:

$w(i,u) = [i=1]$ — ближайший сосед;

$w(i,u) = [i \leqslant k]$ — k ближайших соседей;

$w(i,u) = [i \leqslant k]\, q^i$ — k взвешенных ближайших соседей;

$w(i,u) = K\bigl(\rho(u,x_{i,u})/h\bigr)$ — парзеновское окно фиксированной ширины h;

$w(i,u) = K\left(\dfrac{\rho(u,x_{i,u})}{\rho(u,x_{k+1,u})}\right)$ — парзеновское окно переменной ширины.

Обучающая выборка X^ℓ играет роль параметра алгоритма a. Настройка сводится к запоминанию выборки, и, возможно, оптимизации ещё каких-то параметров, однако сами объекты не подвергаются обработке и сохраняются «как есть». По этой причине метрические алгоритмы относятся к методам *вывода на основе прецедентов* (case-based reasoning, CBR). Здесь действительно можно говорить о «выводе», так как на вопрос «почему объект u был отнесён к классу y?» алгоритм может дать понятный эксперту ответ: «потому, что имеются прецеденты — схожие с ним объекты, принадлежащие классу y», и предъявить список этих прецедентов.

Понятие отступа объекта. Общая формула (1.3) 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 вида $a(u) = \arg\max_{y\in Y}\Gamma_y(u)$ называется величина $M(x_i) = \Gamma_{y_i}(x_i) - \max_{y\in Y\setminus y_i}\Gamma_y(x_i)$.*

Понятие «отступ» можно трактовать как «расстояние от объекта до поверхности, отделяющей свой класс от всех остальных». Величина отступа показывает *степень граничности* объекта.

Отступ $M(x_i)$ отрицателен тогда и только тогда, когда алгоритм допускает ошибку: $a(x_i) \neq y_i$. Большой отрицательный отступ свидетельствует о том, что объект x_i окружён объектами чужих классов — такие объекты называют *шумовыми* или *выбросами*. Большой положительный отступ означает, что объект окружён объектами своего класса — это наиболее типичные, *эталонные* представители классов. Отступ, близкий к нулю, означает, что объект x_i является *пограничным* — на таких объектах классификация неустойчива в том смысле, что малые изменения в составе обучающей выборки могут приводить к ошибочной классификации объекта x_i. В выборках избыточно большого объёма выделяется масса объектов с большим положительным отступом — это *неинформативные* объекты, которые правильно классифицируются по ближайшим к ним эталонам и фактически не несут никакой новой информации. Таким образом, распределение отступов позволяет выделить четыре категории объектов: шумовые, пограничные, неинформативные и эталонные. Из них шумовые и неинформативные целесообразно удалять из выборки. Соответствующий алгоритм будет описан ниже в разделе 1.1.3.

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 произошло успешно. Если же значения отступов концентрируются вблизи нуля, значит, почти все объекты являются граничными, и гипотеза компактности не выполняется. Это означает, что в данной задаче при выбранной метрике применять алгоритмы типа kNN бесполезно.

Достоинства простых метрических алгоритмов типа kNN.

- Простота реализации и возможность введения различных модификаций.
- 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 простых 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 *опорных объектов*, действительно необходимых для классификации.
- Поиск ближайшего соседа «в лоб» требует сравнения классифицируемого объекта со всеми объектами выборки за $O(\ell)$ операций. Для задач с большими выборками это может оказаться накладно. Проблема решается с помощью эффективных алгоритмов поиска ближайших соседей, которые требуют в среднем $O(\ln \ell)$ операций.
- В исходном виде модель алгоритмов kNN крайне бедна. Она имеет только свободный параметр k, да и тот дискретный с небольшим числом разумных альтернатив. Для обогащения модели необходимо вводить веса объектов и/или параметризовать способ вычисления метрики.

Продолжим рассмотрение различных видов метрических алгоритмов, в которых делается попытка устранить обозначенные выше недостатки.

Алгоритм 1.1. Метод потенциальных функций

Вход:

X^ℓ — обучающая выборка;

Выход:

Коэффициенты $\gamma_i,\ i = 1, \ldots, \ell$ в (1.4);

1: Инициализация: $\gamma_i = 0;\ i = 1, \ldots, \ell$;
2: **для всех** $i = 1, \ldots, \ell$
3:   **если** $a(x_i) \neq y_i$ **то**
4:     $\gamma_i := \gamma_i + 1$;

1.1.2 Метод потенциальных функций

Пойдём по пути дальнейшей модификации парзеновского алгоритма (1.1). До сих пор мы рассматривали метрические алгоритмы, в которых центр ядра K помещался в классифицируемый объект u, и суммировались вклады ближайших к нему обучающих объектов. В силу симметричности функции расстояния $\rho(x, x')$ 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:

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

По сути, эта формула отличается от (1.2) 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 γ_i — как абсолютную величину этих зарядов; ядро $K(z) = \frac{1}{z}$ — 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 для наших целей совершенно не обязательно брать именно такое ядро.

Алгоритм (1.4) имеет достаточно богатый набор из 2ℓ параметров γ_i, h_i. Простейший и исторически самый первый метод их настройки представлен в Алгоритме 1.1. 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. Этот метод не так уж плох, как можно было бы подумать. Условия его сходимости подробно исследованы в [1].

Возможны многочисленные вариации этого алгоритма. Перебор объектов можно осуществлять либо в случайном порядке, либо каждый раз брать объект с наименьшим значением отступа $M(x_i)$. Шаги 2–4 можно повторять в цикле до тех пор, пока происходят коррекции весов. На шаге 4 вместо увеличения веса на 1 можно оптимизировать вес γ_i и заодно ширину окна h_i по критерию максимума среднего отступа $\sum_{i=1}^{\ell} M(x_i)$.

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:

- Результат обучения зависит от порядка предъявления объектов.
- Медленная сходимость.
- Слишком грубо (с шагом 1) настраиваются веса γ_i.
- Слишком грубо настраиваются центры потенциалов; вообще говоря, их оптимальные положения не обязаны совпадать с объектами обучения.
- Вообще не ставится задача минимизации числа потенциалов (ненулевых γ_i).
- Вообще не настраиваются параметры h_i.
- Как следствие — относительно невысокое качество классификации.

Более современный метод настройки линейных комбинаций потенциальных функций основан на EM-алгоритме и 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*.

1.1.3 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 гуще» 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 классификации, сокращается объём хранимых данных и уменьшается время классификации, затрачиваемое на поиск ближайших эталонов.

Идея отбора эталонов реализована в алгоритме STOLP [3]. Мы рассмотрим его обобщённый вариант с произвольной весовой функцией $w(i, u)$. Будем строить метрический алгоритм $a(u; \Omega)$ вида (1.3), 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

Алгоритм 1.2. 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:
$\Omega := \bigl\{\arg\max\limits_{x_i \in X^\ell_y} M(x_i, X^\ell) \bigm| y \in Y\bigr\}$;
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:
$x_i := \arg\min\limits_{x \in E} M(x, \Omega)$; $\Omega := \Omega \cup \{x_i\}$;

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^ℓ исключаются все объекты x_i с отступом $M(x_i, X^\ell)$, меньшим заданного порога δ, например, можно положить $\delta = 0$. Как вариант, можно исключать заданную долю 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 отрицательное значение отступа. Так продолжается до тех пор, пока число ошибок не окажется пренебрежимо малым. Если положить $\ell_0 = 0$, то будет построен алгоритм $a(u;\Omega)$, не допускающий ошибок на обучающих объектах, за исключением, разве что, объектов-выбросов.

В результате каждый класс будет представлен в Ω одним «центральным» эталонным объектом и массой «пограничных» объектов, на которых отступ принимал наименьшие значения в процессе итераций.

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 Ω. 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 δ. При программной реализации имеет смысл определить отдельную процедуру для обновления значений всех отступов $M_i = M(x_i, \Omega)$ при текущем наборе эталонов Ω. Тогда в самом алгоритме можно будет 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.

1.1.4 Взвешенный kNN

Альтернативный подход заключается в том, чтобы не отбирать объекты жёстко, а присвоить каждому обучающему объекту x_i некоторый неотрицательный вес $\gamma(x_i)$. Чем более полезен объект для классификации других объектов, тем больше должен быть его вес.

Определим весовую функцию $w(i,u)$ так, чтобы каждый объект x_i учитывался со своим весом $\gamma(x_i)$:

$$w(i,u) = \gamma(x_{i,u})\,\tilde{w}(i,u),$$

где функция $\tilde{w}(i,u)$ неотрицательна и не возрастает по i. В частности, можно положить $\tilde{w}(i,u) = 1$. Тогда вес $w(i,u)$ будет зависеть только от самого соседа $x_{i,u}$, но ни от его ранга i, ни от расстояния $\rho(u, x_{i,u})$.

Чтобы настроить значения параметров $\gamma(x_i)$ по обучающей выборке, выпишем условие корректности алгоритма на объектах обучения:

$$a(x_i) = y_i, \quad i = 1, \ldots, \ell.$$

Исходя из общего представления метрического алгоритма (1.3), легко сделать вывод, что эта система из ℓ равенств равносильна системе из $\ell\bigl(|Y| - 1\bigr)$ линейных неравенств относительно весов $\gamma(x_i)$:

$$\sum_{s=2}^{\ell} \gamma(x_{s,x_i})\,\tilde{w}(s,x_i)\bigl([y_{s,x_i} = y_i] - [y_{s,x_i} = y]\bigr) > 0, \quad (i,y) \in T, \tag{1.5}$$

где суммирование начинается с 2, чтобы сам объект x_i не попадал в число своих ближайших соседей; множество T определяется как множество всех пар «номер объекта — номер класса, которому он не принадлежит»:

$$T = \bigl\{(i,y) \bigm| i = 1, \ldots, \ell;\; y \in Y \setminus \{y_i\}\bigr\}.$$

Очевидно, $|T| = \ell\bigl(|Y| - 1\bigr)$.

Запишем систему неравенств (1.5) в матричном виде. Введём вектор-столбец весов $\gamma = \bigl(\gamma(x_1), \ldots, \gamma(x_\ell)\bigr)^{\intercal}$. Определим матрицу $B = (b_{(i,y)j})$ с $|T|$ строками и j столбцами. Строки будем нумеровать парами (i,y) из T. Заполним матрицу B следующим образом. Для каждой тройки индексов (i, y, s), в которой $(i, y) \in T$, $s = 2, \ldots, \ell$, найдём объект $x_j = x_{s,x_i}$, который является s-м соседом объекта x_i. Положим

$$b_{(i,y)j} = \begin{cases} \tilde{w}(s, x_i), & \text{если } y_j = y_i; \\ -\tilde{w}(s, x_i), & \text{если } y_j = y; \\ 0, & \text{в противном случае.} \end{cases}$$

Тогда в матричном представлении система неравенств (1.5) относительно неизвестного неотрицательного вектора γ приобретёт совсем простой вид:

$$B\gamma > 0; \quad \gamma \geqslant 0.$$

В общем случае эта система несовместна. Нарушение неравенства, соответствующего паре индексов (i, y), означает, что алгоритм $a(u)$ допускает ошибку на обучающем объекте x_i, выдавая ответ y вместо y_i. Минимизация числа ошибок на обучающей выборке сводится к задаче поиска максимальной совместной подсистемы в системе неравенств $B\gamma > 0$ при обязательном выполнении ограничений $\gamma \geqslant 0$.

Таким образом, задача определения весов объектов в метрическом классификаторе сводится к построению линейного разделяющего правила. Фактически, это новая задача классификации, в которой объектами являются всевозможные пары $(i, y) \in T$, а признаковые описания задаются векторами $\bigl(b_{(i,y)1}, \ldots, b_{(i,y)\ell}\bigr)$.

Для решения данной задачи применимы методы построения линейной разделяющей поверхности с неотрицательными коэффициентами, в частности, модификация метода опорных векторов Non-Negative SVM, описанная в разделе ??.??.

В результате решения системы неравенств веса некоторых объектов могут принять нулевые значения. Обычно ими оказываются шумовые выбросы, только ухудшающие качество классификации. Обнуление веса γ_i означает, что объект x_i не будет учитываться при классификации. Таким образом, данный алгоритм автоматически исключает выбросы из обучающей выборки.

1.1.5 Быстрый поиск ближайших соседей.

1.1.6 Сравнение метрических методов классификации

§1.2 Синтез и оптимизация метрик

1.2.1 Проблема выбора метрики

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 $f_1(x), \ldots, f_n(x)$, первое, что приходит в голову — применить евклидову метрику (именно её все проходили в школе):

$$\rho(x, x') = \left(\sum_{j=1}^{n} \bigl(f_j(x) - f_j(x')\bigr)^2\right)^{\frac{1}{2}}.$$

Её обобщением является *взвешенная метрика Минковского*:

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

где c_j — веса признаков, показатель степени p может принимать значения от 1 до ∞ включительно. При $p = \infty$ имеем взвешенную равномерную метрику:

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

Веса признаков часто задают из соображений нормировки: $M_j = \max\limits_{i=1,\dots\ell} \left| f_j(x_i) \right|$, $c_j = M_j^{-p}$, 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 $\sum_j \left| f_j(x) - f_j(x') \right|$. Этот факт аналогичен закону больших чисел в теории вероятностей. Увеличение размерности в пределе $n \to \infty$ 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. Это существенно затрудняет решение задач классификации, кластеризации и непараметрической регрессии.

Решение заключается в том, чтобы применять более тонкие методы для настройки весов c_j. Причём в пространствах с избыточным количеством признаков значительная часть весов должна обнуляться.

1.2.2 Оптимизация весов признаков

1.2.3 Беспризнаковая классификация

1.2.4 Линейные комбинации метрик

§1.3 Обобщающая способность метрических алгоритмов

Напомним некоторые определения из вводной части.

Методом обучения μ называется отображение, которое по произвольной обучающей выборке X^ℓ строит некоторый алгоритм $a\colon X \to Y$, $a = \mu(X^\ell)$. В частности, для алгоритма ближайшего соседа метод обучения сводится к элементарному запоминанию обучающей выборки. *Частота ошибок* алгоритма a на произвольной выборке X^ℓ по определению есть

$$\nu(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [a(x_i) \neq y_i].$$

Пусть задана выборка прецедентов $X^L = (x_i, y_i)_{i=1}^L$. Обозначим через (X_n^ℓ, X_n^k), $n = 1, \dots, N$ всевозможные разбиения выборки X^L на обучающую и контрольную подвыборки длиной ℓ и k соответственно. Будем предполагать, что все $N = C_L^\ell = \frac{L!}{\ell!\,k!}$

разбиений равновероятны. Определим *обобщающую способность* метода μ на конечной выборке X^L как матожидание частоты ошибок на контроле. Оно совпадает с функционалом *полного скользящего контроля* (complete cross-validation):

$$Q_c(\mu, X^L) = \mathsf{E}_n \nu\bigl(\mu(X_n^\ell), X_n^k\bigr) = \frac{1}{N}\sum_{n=1}^{N} \nu\bigl(\mu(X_n^\ell), X_n^k\bigr).$$

Оказывается, для алгоритма ближайшего соседа существует точное выражение функционала Q_c, не требующее полного перебора всех разбиений [4].

1.3.1 Скользящий контроль и профиль компактности

Рассмотрим алгоритм ближайшего соседа $a(u; X^\ell) = y_{1,u}$.

Для произвольного объекта $x_i \in X^L$ через $x_{m,i}$ будем обозначать m-го соседа объекта x_i в полной выборке X^L. Аналогичное обозначение введём и для ответа на m-м соседе: $y_{m,i} = y^*(x_{m,i})$. Нумерацию соседей начнём с нуля, чтобы каждый объект являлся нулевым соседом самого себя, $x_i \equiv x_{0,i}$.

Обозначим через $r_m(x_i)$ бинарную величину, принимающую значение 1 тогда и только тогда, когда ответ на объекте x_i не совпадает с ответом на его m-м соседе:

$$r_m(x_i) = \bigl[y_i \neq y_{m,i}\bigr]; \quad i = 1, \dots, L; \quad m = 1, \dots, L-1.$$

Опр. 1.3. *Профилем компактности выборки X^L называется функция $R(m, X^L)$, выражающая долю объектов выборки $x_i \in X^L$, для которых ответ y_i не совпадает с ответом на m-ом соседе $y_{m,i}$:*

$$R(m, X^L) = \frac{1}{L}\sum_{i=1}^{L} r_m(x_i); \quad m = 1, \dots, L-1.$$

Профиль компактности является формальным выражением гипотезы компактности. Чем проще задача, то есть чем чаще близкие объекты оказываются в одном классе, тем сильнее «прижимается к нулю» начальный участок профиля. И наоборот, в сложных задачах, где ближайшие объекты практически не несут информации о классах, профиль вырождается в константу, близкую к 0.5, см. Рис. 1.

Интуитивно очевидная связь профиля компактности с качеством классификации формально подтверждается следующей теоремой.

Теорема 1.1 (Воронцов, [2]). *Для метода классификации по ближайшему соседу справедливо следующее выражение функционала Q_c:*

$$Q_c(\mu, X^L) = \sum_{m=1}^{k} R(m, X^L) \frac{C_{L-1-m}^{\ell-1}}{C_{L-1}^{\ell}}. \tag{1.6}$$

Доказательство.

Запишем в функционале скользящего контроля частоту ошибок через сумму индикаторов ошибки и переставим знаки суммирования:

$$Q_c = \frac{1}{k}\sum_{i=1}^{L} \underbrace{\frac{1}{N}\sum_{n=1}^{N} \bigl[x_i \in X_n^k\bigr] \bigl[y_i \neq \mu(X_n^\ell)(x_i)\bigr]}_{N_i}. \tag{1.7}$$

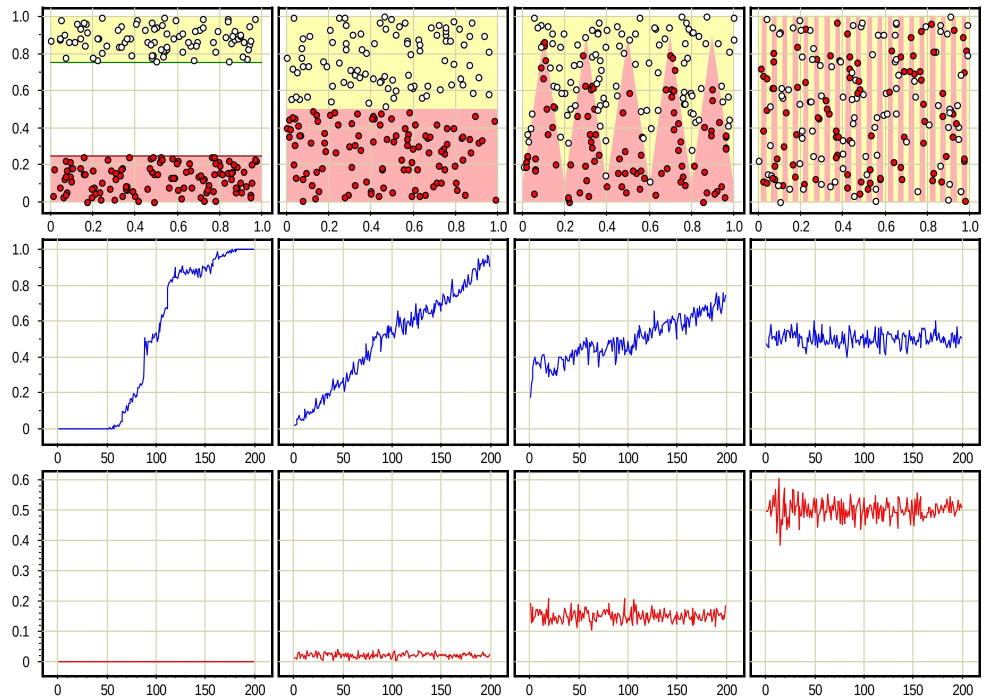

Рис. 1. Верхний ряд: 4 модельные задачи классификации, в порядке возрастания трудности, $L = 200$. Средний ряд: профили компактности этих задач. Чем ниже проходит начальный участок профиля, тем «проще» задача для алгоритма 1NN, и тем выше обобщающая способность. Нижний ряд: зависимость качества Q_c от длины контрольной выборки k при фиксированной длине обучения $\ell = 200$.

Внутренняя сумма, обозначенная через N_i, выражает число разбиений выборки X^L, при которых объект x_i оказывается в контрольной подвыборке и алгоритм $\mu(X_n^\ell)$ допускает на нём ошибку. Данная ситуация реализуется для таких разбиений, при которых m первых объектов из последовательности $x_{i,0}, x_{i,1}, \ldots, x_{i,L-1}$ попадают в контрольную подвыборку, m-ый сосед $x_{i,m}$ находится в обучающей подвыборке и принадлежит другому классу, то есть $r_m(x_i) = 1$. Число таких разбиений равно числу способов выбрать $(\ell - 1)$ обучающих объектов из оставшихся $(L-1-m)$ объектов полной выборки, и равно в точности $C_{L-1-m}^{\ell-1}$. Поскольку m может принимать значения от 1 до k, общее число разбиений составляет

$$N_i = \sum_{m=1}^{k} r_m(x_i) C_{L-1-m}^{\ell-1}.$$

Подставляя N_i в (1.7), и используя определение профиля компактности, получаем требуемое выражение функционала Q_c. ■

Обсудим некоторые свойства формулы (1.6).

Комбинаторный множитель $C_{L-1-m}^{\ell-1}/C_{L-1}^{\ell}$ быстро убывает с ростом m. Поэтому для обеспечения малого значения функционала Q_c достаточно потребовать, чтобы профиль $R(m)$ принимал малые значения при малых m, то есть чтобы близкие объекты лежали преимущественно в одном классе. При больших m рост $R(m)$ компенсируется убыванием комбинаторного множителя, поэтому классификации далёких друг от друга объектов не влияют на значение функционала Q_c.

Вычисление профиля компактности требует $O(\ell^2)$ операций, без учёта упорядочивания объектов по близости. После этого вычисление Q_c производится за $O(k)$ операций. Это гораздо быстрее, чем производить суммирование по всем разбиениям (что практически нереально уже при $k > 2$).

Быстрое вычисление Q_c при любых k позволяет проверить экспериментально, существует ли зависимость Q_c от k. Нижний ряд графиков на Рис. 1 показывает, что её нет. Таким образом, увеличение длины контроля k не влияет на точность оценивания, и на практике можно обходиться малыми значениями $k = 1, 2, 3$.

При $k = 1$ профиль компактности вырождается в точку и совпадает с самим функционалом, который в этом случае называют скользящим контролем с *исключением объектов по одному* (leave-one-out, LOO). При $k = 2$ и 3 профиль состоит из двух и трёх точек соответственно, причём относительный вклад $R(m)$ быстро уменьшается из-за убывания комбинаторных множителей по m:

$$
\begin{aligned}
k = 1:&\quad Q_c(\mu, X^L) = R(1);\\
k = 2:&\quad Q_c(\mu, X^L) = R(1)\tfrac{\ell}{\ell+1} + R(2)\tfrac{1}{\ell+1};\\
k = 3:&\quad Q_c(\mu, X^L) = R(1)\tfrac{\ell}{\ell+2} + R(2)\tfrac{2\ell}{(\ell+1)(\ell+2)} + R(3)\tfrac{2}{(\ell+1)(\ell+2)}.
\end{aligned}
$$

1.3.2 Стабильность обучения

Список литературы

[1] *Айзерман М. А., Браверман Э. М., Розоноэр Л. И.* Метод потенциальных функций в теории обучения машин. — М.: Наука, 1970. — P. 320.

[2] *Воронцов К. В.* Комбинаторный подход к оценке качества обучаемых алгоритмов // Математические вопросы кибернетики / Под ред. О. Б. Лупанов. — М.: Физматлит, 2004. — Т. 13. — С. 5–36.
http:// www.ccas.ru/frc/papers/voron04mpc.pdf.

[3] *Загоруйко Н. Г.* Прикладные методы анализа данных и знаний. — Новосибирск: ИМ СО РАН, 1999.

[4] *Mullin M., Sukthankar R.* Complete cross-validation for nearest neighbor classifiers // Proceedings of International Conference on Machine Learning. — 2000.
http:// citeseer.ist.psu.edu/309025.html.