Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)
Факультет управления и прикладной математики
Вычислительный центр им. А. А. Дородницына РАН
Кафедра «Интеллектуальные системы»

Романов Лев Юрьевич

О согласованных оценках сложности задач и алгоритмов классификации

511656 - Математические и информационные технологии

Выпускная квалификационная работа магистра

Научный руководитель:
чл.-корр. РАН, проф.
Рудаков Константин Владимирович

Москва
2007

Содержание

Аннотация

В работе рассмотрены два независимо определенных понятия сложности: геометрическая сложность конфигурации объектов выборки и функциональная сложность выборки как сложность разделяющей поверхности. Сформулирована гипотеза о возможной корреляции сложности конфигурации объектов и сложности разделяющей их поверхности. Проведены вычислительные эксперименты, в ходе которых гипотеза проверена и подтверждена. В результате получена экспериментальная зависимость, позволяющая по одной из величин произвести оценку другой.

1 Введение

Существует множество определений понятия «сложности» задачи распознавания. Под этим словом можно понимать вычислительную сложность, когда мерой сложности считают число элементарных шагов алгоритма настройки решающего правила. За сложность задачи распознавания можно принять и сложность разделяющей поверхности. Тут можно рассматривать два альтернативных подхода: принять за сложность поверхности ее «геометрическую сложность» или некоторым образом определенную сложность функции, задающей эту поверхность.

Также говорят о геометрической сложности обучающей выборки. Под этим термином понимают некоторую числовую характеристику, отражающую то, насколько сильно «перемешаны» объекты выборки, принадлежащие разным классам.

Интуитивно понятно, что чем сильнее взаимопроникновение классов обучающей выборки, тем сложнее окажется разделяющая поверхность, и наоборот. Отсюда возникает гипотеза о возможной корреляции сложности конфигурации объектов и сложности разделяющей их поверхности. Поэтому в настоящей работе предлагается исследовать наличие зависимости между геометрической сложностью конфигурации объектов обучающей выборки и сложностью разделяющей поверхности (как функции). Чтобы определить наличие зависимости между этими двумя характеристиками обучающей выборки, мы будем проводить вычислительные эксперименты, на наборе модельных задач. Для каждой задачи мы будем находить две указанные характеристики и наносить их на график в виде точки. Таким образом можно будет исследовать наличие зависимости между этими видами сложности.

2 Постановка задачи

Задано евклидово пространство размерности n. Рассматривается классическая задача классификации с двумя непересекающимися классами.

Мы будем изучать возможную зависимость сложностей на примере алгоритма SVM [1, 2]. Этот алгоритм строит в заданном пространстве линейную разделяющую поверхность — гиперплоскость. Для построения нелинейного решающего правила используется следующая схема: исходное пространство расширяется дополнительными осями таким образом, чтобы SVM разделял обучающую выборку без ошибок. Значения координат по каждой из дополнительных осей являются нелинейными функциями от исходных координат: $x_{n+k} = g_k(x_1, \ldots, x_n)$. Построенное таким образом пространство называют *спрямляющим*. После построения решающего правила в расширенном пространстве, полученная гиперплоскость проектируется на исходное пространство. В нем разделяющая поверхность оказывается нелинейной и разделяет обучающую выборку без ошибок.

Основной проблемой при построении спрямляющего пространства оказывается нахождение оптимального набора дополнительных осей. Поэтому очень полезной для исследователя может оказаться возможность оценить сложность спрямляющего пространства до его построения, имея в наличии только обучающую выборку и вычислив ее геометрическую сложность.

Изучению вопроса о зависимости геометрической и функциональной сложностей обучающей выборки и посвящена настоящая работа.

3 Геометрическая сложность обучающей выборки

Геометрическую сложность можно определить различными способами. Наибольший интерес представляет степень взаимного проникновения классов друг в друга. Для оценки этой величины можно разными способами измерять расстояние между классами.

Мы будем оценивать расстояние между классами как минимальное расстояние между точками разных классов. Приведем теперь строгое определение выбранного нами способа оценки геометрической сложности.

Пусть мы имеем обучающую выборку в n-мерном пространстве. Выборка конечна, поэтому она ограничена. Опишем около выборки n-мерный куб. Свяжем понятие геометрической сложности выборки с расстоянием между классами. Интуитивно понятно, что если расстояние между классами внутри куба велико, то сложность мала, а если расстояние мало, то сложность велика. Поэтому введем геометрическую сложность следующим образом:

Определение 1. *Геометрической сложностью обучающей выборки в n-мерном пространстве называется отношение стороны описанного около выборки n-мерного куба к минимальному расстоянию между объектами двух классов.*

Примеры обучающих выборок с различной геометрической сложностью приведены на рис. 1 и 2.

4 Функциональная сложность обучающей выборки

Основная идея нахождения сложности разделяющей поверхности состоит в том, что за сложность поверхности принимается определенная некоторым образом сложность спрямляющего пространства, достаточного для разделения обучающей выборки без ошибок. Это оправдано потому, что от сложности пространства, в котором строится линейный классификатор, будет зависеть то, насколько сложной окажется разделяющая поверхность при проектировании линейного классификатора из расширенного в исходное пространство.

4.1 Определения и понятия

Будем говорить, что дополнительная ось задается функционалом $g(x_1, \ldots, x_k)$, если вводится новая координата, значения которой у рассматриваемого множества точек задаются указанным функционалом.

Предположим, что оси, расширяющие исходное признаковое пространство, задаются некоторыми функциями от исходных координат:

$$\mathscr{G} = \left\{ g_i(x_1, x_2, \ldots, x_n) : \mathbb{R}^n \to \mathbb{R}^1 \right\}, \quad i \in \{1, \ldots, l\}. \tag{1}$$

Введем множество функционалов $\mathscr{F}$:

$$\mathscr{F} = \left\{ f_i(x_1, x_2, \ldots, x_{k_i}) : \mathbb{R}^{k_i} \to \mathbb{R}^1 \right\}, \quad i \in \{1, \ldots, m\}. \tag{2}$$

Мы будем использовать их в качестве базиса для построения спрямляющего пространства. Таким образом, каждая функция g_i задается суперпозицией функционалов из базиса.

Будем считать, что каждая функция f_i из базиса $\mathscr{F}$ обладает некоторой сложностью c_i, то есть ее вхождение в формулу, определяющую значения координат дополнительной оси, вносит некоторый вклад в общую сложность оси. Сами стоимости задаются исследователем эвристически.

Также задается сложность операции суперпозиции функций из базиса. Например, сложность суперпозиции функций можно определить, как суммарную сложность входящих в нее функций.

Определение 2. *Под сложностью оси будем понимать сложность суперпозиции функций из базиса, определяющей значения координат оси.*

Определение 3. *Сложностью совокупности осей назовем суммарную сложность всех составляющих ее осей.*

Определение 4. *Сложностью спрямляющего пространства назовем сложность совокупности всех его осей.*

Определение 5. *Под сложностью разделяющей поверхности будем понимать сложность спрямляющего пространства, которое явилось достаточным для разделения выборки.*

Определение 6. *Функциональной сложностью обучающей выборки назовем минимальную сложность разделяющей ее поверхности.*

Действительно, чем больше осей потребовалось добавить к исходному пространству для построения спрямляющего пространства, и чем более сложными оказались суперпозиции функций из $\mathscr{F}$, определяющие значения координат новых осей, тем более сложной окажется разделяющая поверхность в исходном пространстве, и наоборот.

5 Описание алгоритма нахождения функциональной сложности

5.1 Алгоритм

В общем случае, суперпозицию функций многих переменных можно представить в виде дерева. В нем в качестве листьев будем рассматривать переменные из исходного евклидова пространства, в качестве ребер — функционалы из $\mathscr{F}$, а в качестве узлов — суперпозиции функций.

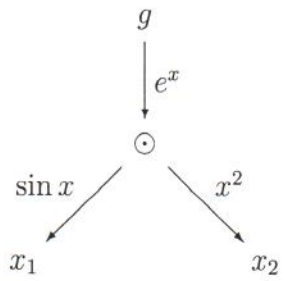
Пример.

Пусть $\mathscr{F} = \{f_i(x)\}_{i=1}^{3} = \{e^x, \sin x, x^2\}$. Определим сложности каждой функции из $\mathscr{F}$. Пусть $C\left(e^x\right) = 3$, $C\left(\sin x\right) = 4$, $C\left(x^2\right) = 2$, $C\left(f_1 \cdot f_2\right) = C(f_1) + C(f_2)$.

Рассмотрим функцию $g(x_1, x_2) = e^{\sin(x_1)\cdot x_2^2}$. Ее сложность равна

$$C(g) = C\left(e^x\right) + \left(C\left(\sin x\right) + C\left(x^2\right)\right) = 9.$$

В виде дерева она будет выглядеть следующим образом:



▲

Будем составлять список функций, образованных суперпозицией функционалов из $\mathscr{F}$, в порядке неубывания сложности. Для этого будем перебирать соответствующие деревья так, чтобы сложности образованных ими функций не убывали.

Пример.

Путь, аналогично предыдущему примеру, $\mathscr{F} = \{f_i(x)\}_{i=1}^{3} = \{e^x, \sin x, x^2\}$, а сложности определены как $C(e^x) = 3$, $C(\sin x) = 4$, $C(x^2) = 2$, $C(f_1 \cdot f_2) = C(f_1) + C(f_2)$. Рассмотрим двумерный случай.

Тогда список всевозможных функций, образованных суперпозициями функций из базиса $\mathscr{F}$, упорядоченный по неубыванию сложности, будет выглядеть следующим образом:

Сложность	Функция
2	x_1^2
2	x_2^2
3	e^{x_1}
3	e^{x_2}
4	$\sin x_1$
4	$\sin x_2$
4	$x_1^2 \cdot x_2^2$
4	$(x_1^2)^2 = x_1^4$
4	$(x_2^2)^2 = x_2^4$
5	$(e^{x_1})^2 = e^{2x_1}$
5	$(e^{x_2})^2 = e^{2x_2}$
5	$e^{(x_1^2)} = e^{x_1^2}$
5	$e^{(x_2^2)} = e^{x_2^2}$
...	...

▲

Из имеющихся теперь функций составим следующую таблицу Ω. Каждая строка в ней соответствует некоторой сложности пространства и содержит список наборов функций, которые в сумме эту сложность дают.

Пример.

Для определенного в предыдущем примере базиса $\mathscr{F}$ построим таблицу Ω.

Сложность набора	Наборы функций
2	x_1^2
2	x_2^2
3	e^{x_1}
3	e^{x_2}
4	$\sin x_1$
4	$\sin x_2$
4	$x_1^2 \cdot x_2^2$
4	$(x_1^2)^2 = x_1^4$
4	$(x_2^2)^2 = x_2^4$
4	$[x_1^2; x_2^2]$
5	$(e^{x_1})^2 = e^{2x_1}$
5	$(e^{x_2})^2 = e^{2x_2}$
5	$e^{(x_1^2)} = e^{x_1^2}$
5	$e^{(x_2^2)} = e^{x_2^2}$
5	$[x_1^2; e^{x_1}]$
5	$[x_2^2; e^{x_1}]$
5	$[x_1^2; e^{x_2}]$
5	$[x_2^2; e^{x_2}]$
...	...

▲

Таким образом, появляется возможность, последовательно перебирая наборы функций из построенной таблицы, монотонно повышать сложность спрямляющего пространства до получения линейного разделения.

Пусть мы имеем некоторую обучающую выборку. Алгоритм 1 описывает процедуру нахождения ее функциональной сложности.

Алгоритм 1: Нахождение функциональной сложности обучающей выборки.

1: **для** строк таблицы Ω, начиная с минимальной сложности
2: **для всех** наборов функций, содержащихся в текущей строке
// каждый набор имеет одинаковую суммарную сложность
3: для каждого набора функций, задающих дополнительные оси спрямляющего пространства, строим линейное разделение в расширенном ими пространстве с помощью алгоритма SVM.
4: **если** добились линейного разделения **то**
5: функциональная сложность найдена — **выход** из обоих циклов
6: **иначе** *// присутствуют ошибки классификации*
7: следующая итерация

Теперь можно исследовать зависимость геометрической и функциональной сложностей. Для этого строятся обучающие выборки с заданной геометрической сложностью, для каждой из них с помощью приведенного алгоритма вычисляется функциональная сложность. Имея эти две величины для каждой выборки, мы можем нанести их в виде точек на график для последующего анализа.

5.2 Описание экспериментов

Для построения выборок применяется следующая процедура. Задается минимальное расстояние между точками классов $\varepsilon \propto \frac{1}{\nu}$, где ν - геометрическая сложность данной выборки. Поскольку всякая выборка конечна, она всегда ограничена некоторым d-мерным кубом, где d — размерность пространства. Поэтому в двумерном случае ограничивается квадрат, в котором датчиком случайных чисел генерируются объекты поочередно для каждого из двух классов таким образом, чтобы расстояние от вновь полученного объекта до всех объектов другого класса не нарушает выбранного зазора между классами (см. алгоритм 2). После этого вычисляется сторона описанного куба, отношение которой к ε дает значение геометрической сложности.

Алгоритм 2: Генерация обучающей выборки (MATLAB)

```
function [data, newstate] = sample_generator(n1, n2, eps, rndstate)
if nargin == 4
    rand('state', rndstate);
end;
if nargin < 2
    n1 = 50;
    n2 = 50;
end
if nargin == 0
    eps = 0.1;
end
  % get_new_dot() — алгоритм 3
c1 = get_new_dot([1000, 1000], eps);
c2 = get_new_dot(c1, eps);
for i = 2:min(n1,n2)
    c1 = [c1; get_new_dot(c2, eps)];
    c2 = [c2; get_new_dot(c1, eps)];
end
if n1>n2
    for i = n2+1 : n1
        c1 = [c1; get_new_dot(c2, eps)];
    end
elseif n1<n2
    for i = n1+1 : n2
        c2 = [c2; get_new_dot(c1, eps)];
    end
end
data = [[c1,  ones(size(c1, 1), 1)];
        [c2, -ones(size(c2, 1), 1)]];
if nargout > 1
    newstate = rand('state');
end;
return
```

Алгоритм 3: Генерация очередного объекта (MATLAB)

```
function dot = get_new_dot(class, eps)
% 'class' is the other class
% eps is a distance new dot to fit
while true
% new dot generating
    x = rand;
    y = rand;
% calculating of squared distances from (x,y) to the all class points
    dist = (x-class(:,1)).^2 + (y-class(:,2)).^2;
% check epsilon fitting
    fitting = find(dist < eps^2);
    s = size(fitting(:));
    if s(1) == 0
        dot = [x, y];
        break;
    end
end
return
```

Результаты работы алгоритма в двумерном пространстве для разных величин ε приведены на рис. 1 и 2.

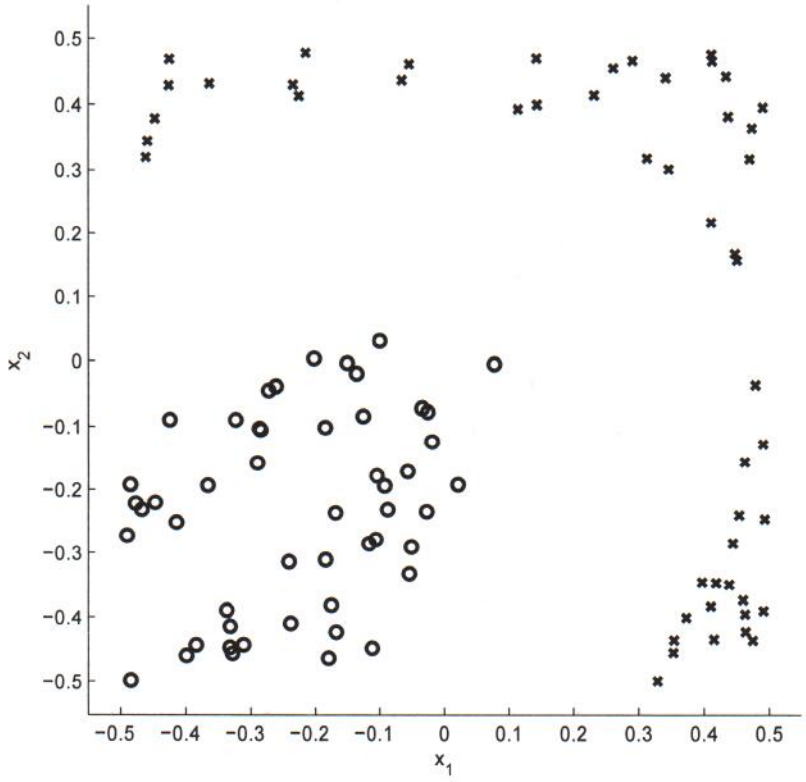


Рис. 1: Обучающая выборка с двумя классами в двумерном пространстве, $\varepsilon = 0.4$

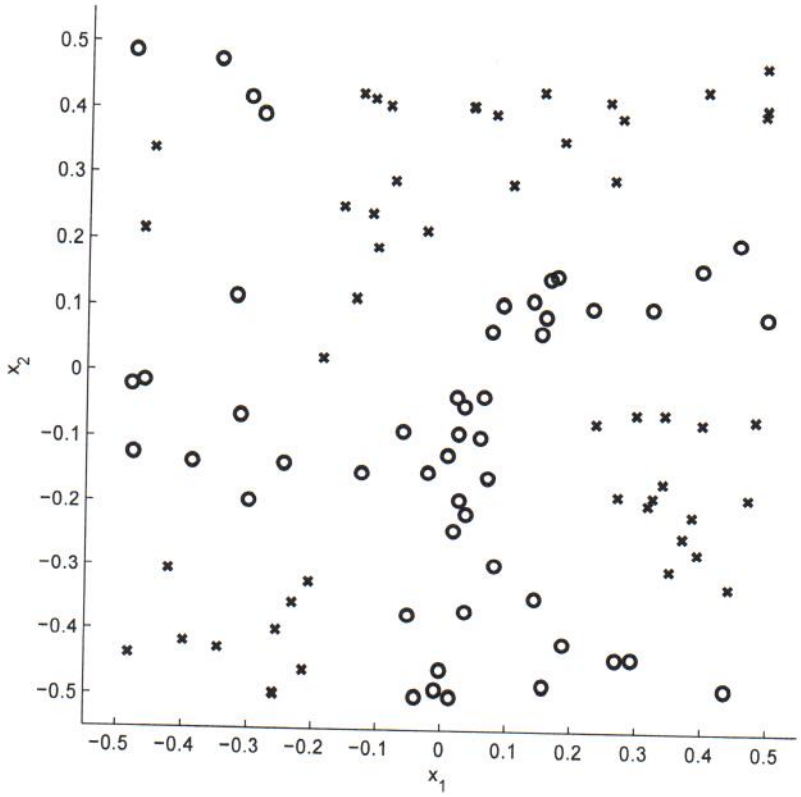


Рис. 2: Обучающая выборка с двумя классами в двумерном пространстве, $\varepsilon = 0.15$

При реализации метода SVM для решения возникающей в нем задачи квадратичной оптимизации используется алгоритм «Sequential Minimal Optimization» (SMO), разработанный J. C. Platt [4]. Этот алгоритм является одним из самых эффективных алгоритмов решения задачи квадратичной оптимизации для SVM, значительно превосходя по скорости алгоритм «chunking», предложенный В. Н. Вапником [1] и алгоритм, предложенный E. Osuna [3].

В настоящей работе описанный выше метод анализа проводится при некоторых упрощениях. Во-первых, будем рассматривать двумерные выборки. Во-вторых, в качестве функций базиса $\mathscr{F}$ рассмотрим функции

$$\left\{ x_i^{\alpha}, x_i^{-\alpha}, x_i^{\frac{1}{\alpha}}, x_i^{-\frac{1}{\alpha}} : i = 1, 2, \quad \alpha = 1, 2, \ldots \right\}.$$

Суперпозицией двух функций из базиса будем считать их произведение:

$$f_1 \circ f_2 = f_1 \cdot f_2.$$

Тогда дополнительные оси имеют вид $g_i = x_1^{\alpha_1^i} x_2^{\alpha_2^i}$.

Сложности функций из базиса $\mathscr{F}$ определим следующим образом:

$$C(f) = C(x^{\alpha}) = \begin{cases} \alpha, & \text{если } \alpha = 1, 2, \ldots; \\ \beta + 1, & \text{если } \alpha = -\beta, \quad \beta = 1, 2, \ldots; \\ \beta, & \text{если } \alpha = \frac{1}{\beta}, \quad \beta = 1, 2, \ldots; \\ \beta + 1, & \text{если } \alpha = -\frac{1}{\beta}, \quad \beta = 1, 2, \ldots. \end{cases}$$

Сложность функции $g = x_1^{\alpha_1} x_2^{\alpha_2}$, задающей значения оси, расширяющей исходное признаковое пространство, определяется как $C(g) = C(x_1^{\alpha_1}) + C(x_2^{\alpha_2})$.

При перечислении возможных осей надо учитывать, что сложность оси не должна быть равна 1, поскольку такую сложность имеют все оси исходного признакового пространства и только они, и эти оси всегда входят в спрямляющее пространство.

Рассматривая наборы дополнительных осей указанного вида в порядке неубывания сложности набора, применим описанный ранее алгоритм.

6 Результаты экспериментов

6.1 Перебор спрямляющих пространств

В процессе работы алгоритма 1 обрабатывается список Ω, содержащий наборы дополнительных осей в порядке неубывания сложности этих наборов. Приведем полученный список наборов осей сложности не более 4.

Сложность	Набор дополнительных осей
2	$[x_2^2]$
2	$[x_2^{-1}]$
2	$[x_2^{\frac{1}{2}}]$
2	$[x_1 x_2]$
2	$[x_1^2]$
2	$[x_1^{-1}]$
2	$[x_1^{\frac{1}{2}}]$

Сложность	Набор дополнительных осей
3	$[x_2^3]$
3	$[x_2^{-2}]$
3	$[x_2^{-\frac{1}{2}}]$
3	$[x_2^{\frac{1}{3}}]$
3	$[x_1 x_2^2]$
3	$[x_1 x_2^{-1}]$
3	$[x_1 x_2^{\frac{1}{2}}]$
3	$[x_1^2 x_2]$
3	$[x_1^{-1} x_2]$
3	$[x_1^{\frac{1}{2}} x_2]$
3	$[x_1^3]$
3	$[x_1^{-2}]$
3	$[x_1^{-\frac{1}{2}}]$
3	$[x_1^{\frac{1}{3}}]$

Сложность	Набор дополнительных осей
4	$\left[x_2^{-1}; x_2^2\right]$
4	$\left[x_2^{\frac{1}{2}}; x_2^2\right]$
4	$\left[x_1x_2; x_2^2\right]$
4	$\left[x_1^2; x_2^2\right]$
4	$\left[x_1^{-1}; x_2^2\right]$
4	$\left[x_1^{\frac{1}{2}}; x_2^2\right]$
4	$\left[x_2^{\frac{1}{2}}; x_2^{-1}\right]$
4	$\left[x_1x_2; x_2^{-1}\right]$
4	$\left[x_1^2; x_2^{-1}\right]$
4	$\left[x_1^{-1}; x_2^{-1}\right]$
4	$\left[x_1^{\frac{1}{2}}; x_2^{-1}\right]$
4	$\left[x_1x_2; x_2^{\frac{1}{2}}\right]$
4	$\left[x_1^2; x_2^{\frac{1}{2}}\right]$
4	$\left[x_1^{-1}; x_2^{\frac{1}{2}}\right]$
4	$\left[x_1^{\frac{1}{2}}; x_2^{\frac{1}{2}}\right]$
4	$\left[x_1^2; x_1x_2\right]$
4	$\left[x_1^{-1}; x_1x_2\right]$
4	$\left[x_1^{\frac{1}{2}}; x_1x_2\right]$
4	$\left[x_1^{-1}; x_1^2\right]$
4	$\left[x_1^{\frac{1}{2}}; x_1^2\right]$
4	$\left[x_1^{\frac{1}{2}}; x_1^{-1}\right]$
4	$\left[x_2^4\right]$
4	$\left[x_2^{-3}\right]$
4	$\left[x_2^{-\frac{1}{3}}\right]$
4	$\left[x_2^{\frac{1}{4}}\right]$
4	$\left[x_1x_2^3\right]$
4	$\left[x_1x_2^{-2}\right]$
4	$\left[x_1x_2^{-\frac{1}{2}}\right]$
4	$\left[x_1x_2^{\frac{1}{3}}\right]$
4	$\left[x_1^2x_2^2\right]$
4	$\left[x_1^2x_2^{-1}\right]$
4	$\left[x_1^2x_2^{\frac{1}{2}}\right]$
4	$\left[x_1^{-1}x_2^2\right]$
4	$\left[x_1^{-1}x_2^{-1}\right]$
4	$\left[x_1^{-1}x_2^{\frac{1}{2}}\right]$
4	$\left[x_1^{\frac{1}{2}}x_2^2\right]$

Сложность	Набор дополнительных осей
4	$\left[x_1^{\frac{1}{2}} x_2^{-1}\right]$
4	$\left[x_1^{\frac{1}{2}} x_2^{\frac{1}{2}}\right]$
4	$[x_1^3 x_2]$
4	$\left[x_1^{-2} x_2\right]$
4	$\left[x_1^{-\frac{1}{2}} x_2\right]$
4	$\left[x_1^{\frac{1}{3}} x_2\right]$
4	$[x_1^4]$
4	$\left[x_1^{-3}\right]$
4	$\left[x_1^{-\frac{1}{3}}\right]$
4	$\left[x_1^{\frac{1}{4}}\right]$

6.2 Анализ графика

Вычислительные эксперименты проводились следующим образом. В двумерном пространстве был выбран квадрат $[0;1] \times [0;1]$, в котором датчиком случайных чисел генерировалась случайная выборка из 100 объектов, 50 из которых принадлежат первому классу, а остальные 50 — второму.

В цикле равномерно уменьшалось значение минимального расстояния между объектами разных классов ε и для каждого ε строились 10 случайных выборок, удовлетворяющих заданному ε. После этого для каждой построенной выборки определялись ее геометрическая и функциональная сложности. Геометрическая сложность вычислялась как $\nu = \frac{1}{\varepsilon}$, функциональная сложность μ — по алгоритму 1. Далее точка (μ, ν) наносилась на график. После этого производилось усреднение полученных 10 значений μ для каждого значения ν, а также была подсчитана их дисперсия. Полученный график изображен на рис. 3.

Также производилось усреднение 30 значений μ, полученных для каждого значения ν и ближайшего большего и меньшего значений ν. Полученный график изображен на рис. 4.

Кроме того, производилось усреднение 50 значений μ, полученных для каждого значения ν и двух его ближайших больших и меньших значений ν. Полученный график изображен на рис. 5.

Таким образом, в результате проведенных вычислительных экспериментов были получены графики зависимости функциональной сложности от геометрической, приведенные на рис. 3, 4 и 5. Для отображения функциональной сложности естественным образом выбрана логарифмическая шкала ввиду того, что абсолютная величина изменения сложности спрямляющего пространства при малых величинах сложности связана со значительно более сильным изменением геометрической сложности, чем при больших величинах.

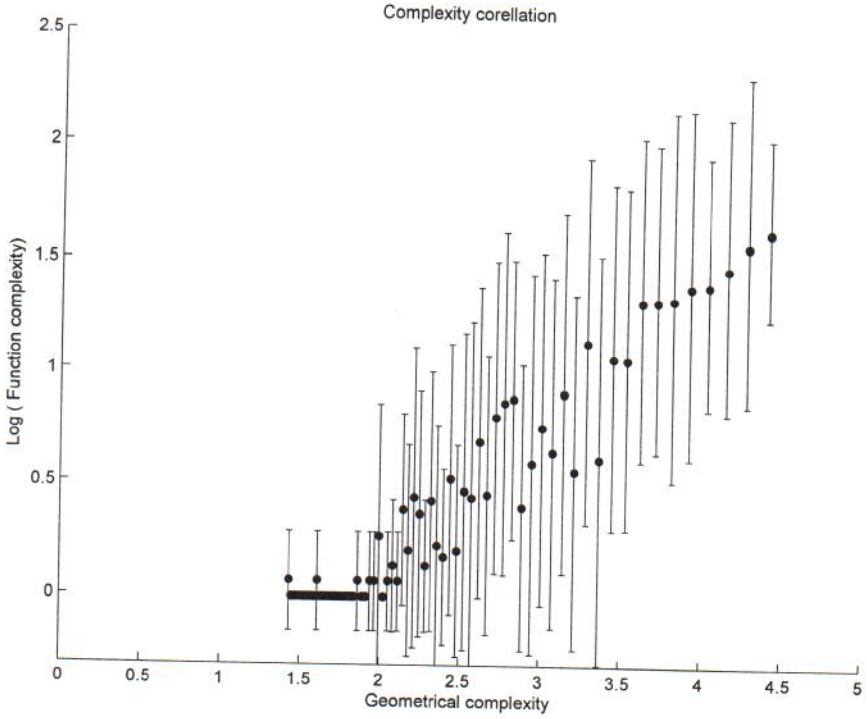


Рис. 3: Зависимость функциональной сложности от геометрической при усреднении по 10 точкам

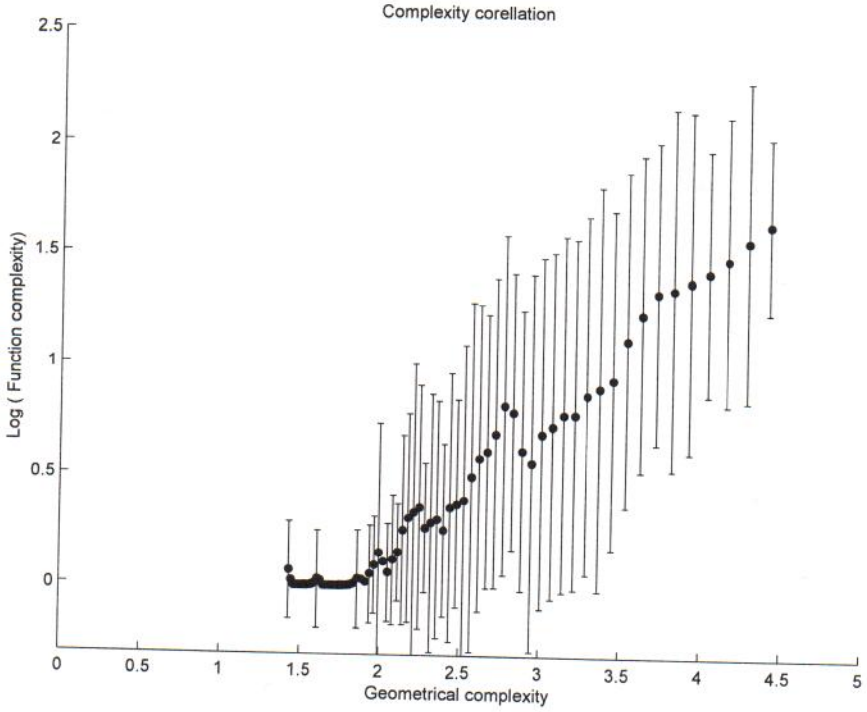


Рис. 4: Зависимость функциональной сложности от геометрической при усреднении по 30 точкам

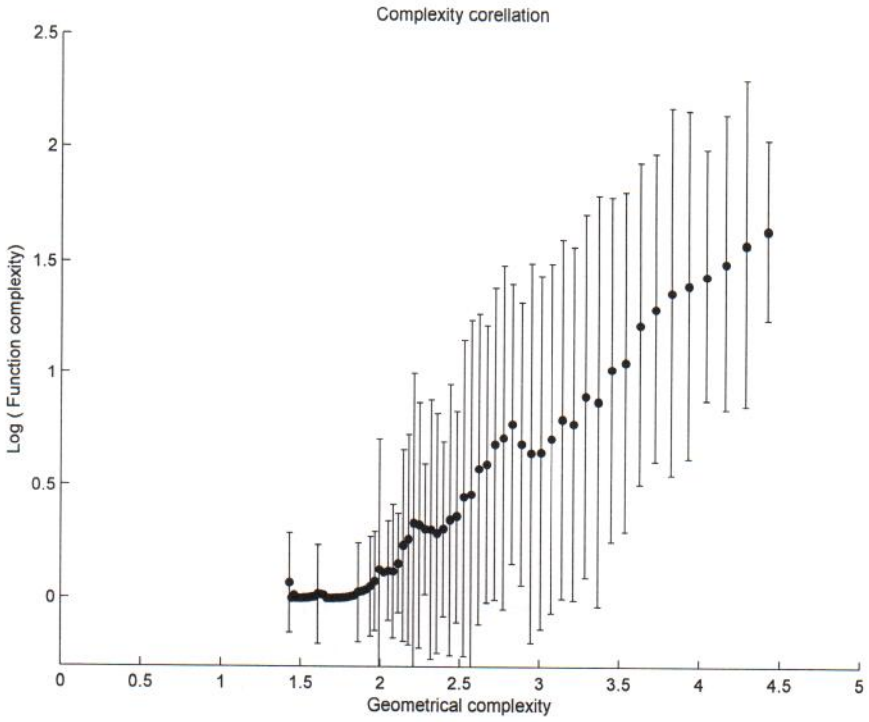


Рис. 5: Зависимость функциональной сложности от геометрической при усреднении по 50 точкам

7 Выводы

В работе рассмотрены различные подходы к понятию сложности задачи классификации. Более подробно рассмотрены две независимо определенные сложности — геометрическая и функциональная сложности. Исследован вопрос об их взаимной зависимости. Проведены эксперименты, в которых для разделения выборок использовался алгоритм SVM. В результате получена экспериментальная зависимость, позволяющая по одной из величин произвести оценку другой.

8 Литература

[1] *Vapnik V.* Estimation of Dependences Based on Empirical Data — Springer–Verlag, 1982.

[2] *Burges C. J. C.* A tutorial on support vector machines for pattern recognition. — Data Mining and Knowledge Discovery, 2(2), 1998.

[3] *Osuna E., Freund R., and Girosi F.* Improved training algorithm for support vector machines. — In *Proc. IEEE Neural Networks in Signal Processing '97*, 1997.

[4] *Platt J. C.* Sequential minimal optimization: A fast algorithm fo training support vector machines. — Technical Report MSR–TR–98–14, Microsoft Research, 1998.
`http://www.research.microsoft.com/~jplatt/smo.html`.

K-L Divergency