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)

М.Г. Семененко

(к.ф.-м.н. доцент)

**ЛАБОРАТОРНАЯ РАБОТА № 3**

**по курсу «Методы машинного обучения»**

## Вычисление отступов по выборке

Калуга

2025

# 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. Для формализации понятия «сходства» вводится функция расстояния в пространстве объектов *X*. Методы обучения, основанные на анализе сходства объектов, называют 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*×*X* → [0*,*∞). Существует целевая зависимость *y*∗ : *X* → *Y* , значения которой известны только на объектах обучающей выборкиИзображение выглядит как черный, текст  Содержимое, созданное искусственным интеллектом, может быть неверным.. Множество классов *Y* конечно. Требуется построить алгоритм классификации *a*: *X* → *Y* , аппроксимирующий целевую зависимость *y*∗(*x*) на всём множестве *X*.

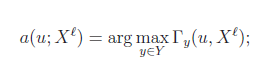
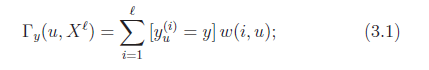
### Обобщённый метрический классификатор

Для произвольного объекта *u* ∈ *X* расположим элементы обучающей выборки *x*1*,...,xℓ* в порядке возрастания расстояний до *u*:

*,*

где через  обозначается *i*-й сосед объекта *u*. Соответственно, ответ на *i*-м соседе объекта *u* есть *yu*(*i*) = *y*∗(*x*(*ui*)). Таким образом, любой объект *u* ∈ *X* порождает свою перенумерацию выборки.

Опр. 3.1. Метрический алгоритм классификации с обучающей выборкой *Xℓ* относит объект *u* к тому классу *y* ∈ *Y* , для которого суммарный вес ближайших обучающих объектов Γ*y*(*u,Xℓ*) максимален:

  (**1**)

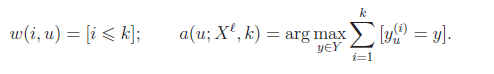
где весовая функция *w*(*i,u*) оценивает степень важности *i*-го соседа для классификации объекта *u*. Функция Γ*y*(*u,Xℓ*) называется оценкой близости объекта *u* к классу *y*.

Весовая функция *w*(*i,u*) обычно выбирается неотрицательной и не возрастающей по *i*. Это соответствует гипотезе компактности, согласно которой чем ближе объекты

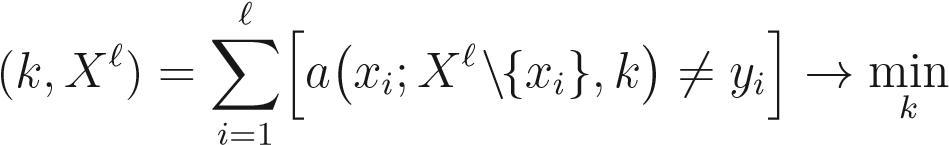
, тем выше шансы, что они принадлежат одному классу. Выбирая весовую функцию w(i,u), можно получать различные метрические классификаторы.

Алгоритм *k* ближайших соседей (*k* nearest neighbors, *k*NN).

Чтобы сгладить влияние выбросов, будем относить объект *u* к тому классу, элементов которого окажется больше среди ближайших соседей.



На практике оптимальное значение параметра *k* определяют по критерию скользящего контроля с исключением объектов по одному (leave-one-out, LOO). Для каждого объекта *xi* ∈ *Xℓ* проверяется, правильно ли он классифицируется по своим *k* ближайшим соседям.

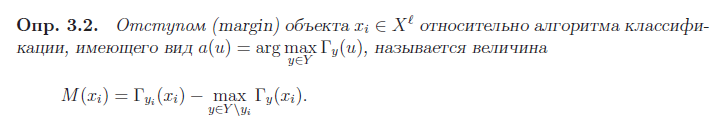
LOO*.*

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

Существует и альтернативный вариант метода *k*NN: 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.

Понятие отступа объекта

Общая формула (1) позволяет ввести характеристику объектов, показывающую, насколько глубоко объект погружён в свой класс.



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.

*Margin*



Упорядоченные по возрастанию отступов *Mi* объекты выборки, *i* = 1*,...,*200.

Условно объекты делят 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 (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 *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».

                        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*;Ω) вида (3.1), где Ω ⊆ *Xℓ* — множество эталонов.

Алгоритм 3.2. Отбор эталонных объектов STOLP



Обозначим через *M*(*xi,*Ω) отступ объекта *xi* относительно алгоритма *a*(*xi*;Ω). Большой отрицательный отступ свидетельствует о том, что объект *xi* 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 *Xℓ* исключаются все объекты *xi* с отступом *M*(*xi,Xℓ*), меньшим заданного порога *δ*. Если взять *δ* = 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 *xi*, имеющий минимальное значение отступа. Так продолжается до тех пор, пока число ошибок не окажется меньше заданного порога ℓ0. Если положить ℓ0 = 0, то будет построен алгоритм *a*(*u*;Ω), 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 объектов *Xℓ* \ Ω, и для каждого вычислить отступ относительно множества эталонов-*O*(|Ω|2ℓ). 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*. 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.

**Задание**

**1.** Сгенерировать случайную выборку из 10 объектов с классами +1 и -1 и одним количественным признаком.

2. Выбрать один из объектов в сгенерированной выборке и рассчитать для него все отступы для алгоритма 2NN. Результаты вычислений представить в виде таблицы.

3. Применить алгоритм STOLP для отбора эталонных объектов.

**Образец программы на Phyton**

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, число

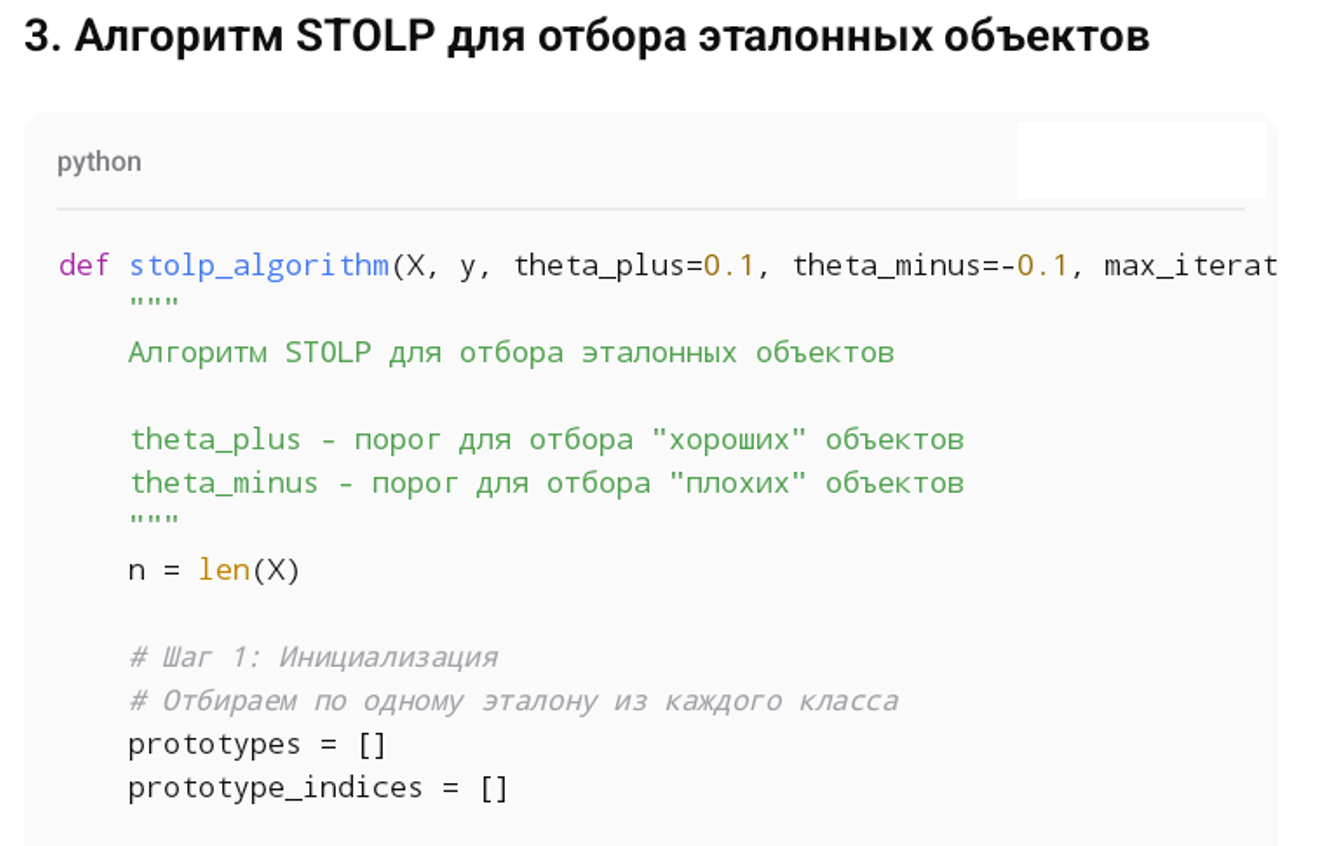
Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.





Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, Шрифт, снимок экрана, чек

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.