

ЦИФРОВАЯ
КУЛЬТУРА
УНИВЕРСИТЕТ ИТМО

Лекция «Наивный байесовский классификатор.
Метод k -ближайших соседей»

Бойцов Антон Александрович
Волчек Дмитрий Геннадьевич
Романов Алексей Андреевич

Санкт-Петербург
2019

Содержание

1	Наивный байесовский классификатор	2
1.1	Построение классификатора	2
1.2	Пример классификации	7
2	Метод k-ближайших соседей	10
2.1	Метрики	11
2.2	Построение классификатора	13
2.3	Пример классификации	15
2.4	Взвешенный k-NN	20
2.5	Дilemma смещения–дисперсии	20
2.6	Кросс-валидация	22
2.7	Проклятие размерностей	25

1 Нaивный байесовский классификатор

В прошлой лекции мы изучили алгоритм логистической регрессии, позволяющий решать задачи классификации, назначая один из двух классов новому объекту. В этой лекции мы продолжим рассматривать задачи такого рода и познакомимся с двумя простыми, но от того не менее эффективными алгоритмами.

1.1 Построение классификатора

Начнем с алгоритма Байеса, концепция которого довольно стара и существует с 18-го века. Именно тогда были разработаны основополагающие математические принципы определения вероятности какого-либо события при условии, что произошло другое, взаимосвязанное с ним событие.

Например, нам известно, что все яблоки красного цвета имеют средний диаметр 7 сантиметров. Мы же берем наугад из корзины некий фрукт: он оказывается красным с диаметром в 6 сантиметров. Какова вероятность того, что этот конкретно взятый фрукт будет яблоком? Похоже, что у нас возникает условная вероятность – вероятность события, что наудачу взятый фрукт – это яблоко при условии, что он красный и имеет конкретный диаметр. Из теории вероятностей известно определение условной вероятности. Пусть событие B таково, что $P(B) > 0$, тогда условной вероятностью события A при условии, что произошло событие B , называется число

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Иными словами, условная вероятность – это отношение вероятности события $A \cap B$, то есть события «произошло и A , и B », к вероятности события B . Иначе – ищется отношение вероятности доли события A , попавшего в B , к вероятности события B . Из этого определения совершенно ясно, что $P(A \cap B) = P(A|B)P(B)$. Если, к тому же, $P(A) > 0$, то мы получаем так называемые правила умножения:

$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A).$$

Заменив теперь в определении условной вероятности числитель на второе соотношение, приходим к так называемой формуле Байеса, широко используемой в машинной обучении:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

Обратите внимание, в этой формуле аргументы условных вероятностей «поменялись местами». Часто вероятность $P(B|A)$ называют правдоподобием,

вероятности $P(A)$ и $P(B)$ – априорными вероятностями, а вероятность $P(A|B)$ – апостериорной вероятностью.

Как нам может помочь эта формула, и при чем тут вообще машинное обучение? Рассмотрим жизненную и хорошо известную проблему – задачу классификации входящих писем на «спам» и «не спам». Многие почтовые сервисы решают эту задачу, используя какие-то алгоритмы, при этом результаты бывают разные: бывает, что все хорошо, все письма классифицированы верно, но иногда важное для вас письмо относится к категории «спам», а иногда наоборот: откровенный мусор проходит проверку и оказывается во входящих. Итак, перед нами – не что иное, как задача двухклассовой классификации.

Для решения этой задачи вооружимся теорией вероятностей и построим наивный байесовский классификатор. Конечно, для решения такого рода задач требуется наличие исходного (обучающего) набора данных. Пример таких данных представлен в таблице, где единица указывает на вхождение слова «приз» в текст письма и отнесение письма в категорию спам.

Текст письма	«приз»	Спам
Вы выиграли приз!	1	1
Оплатите первый месяц услуг и получите приз.	1	1
Получите приз за рекомендацию банковских услуг.	1	0
Вам достался дом в наследство.	0	1
.....

Например, из анализа всего набора данных объемом 80 писем, слово «приз» встретилось среди пяти писем, помеченных как спам, и среди трех, помеченных как не спам. В других 72 письмах такого слова не было, при этом 27 из них были помечены как спам.

Вспомним, на основе приведенных данных, как рассчитываются вероятности событий. Конечно, в нашем случае вероятности есть не что иное, как частоты. Например, вероятность того, что письмо, попавшее в спам, содержит слово «приз», составит $5/32$, ведь всего спам писем 32, из них ровно 5 содержат слово «приз»:

$$P(\text{«приз»} \mid \text{спам}) = \frac{5}{32}.$$

В свою очередь, вероятность того, что письмо относится к спаму, равна $\frac{32}{80}$, а вероятность того, что письмо содержит слово «приз» равна $\frac{8}{80}$:

$$P(\text{спам}) = \frac{32}{80}, \quad P(\text{«приз»}) = \frac{8}{80}.$$

Аналогично можно рассчитать вероятности и других событий:

$$P(\text{не спам}) = \frac{48}{80}, P(\text{«приз»} \mid \text{не спам}) = \frac{3}{48}.$$

$$P(\text{нет слова «приз»}) = \frac{72}{80}, P(\text{нет слова «приз»} \mid \text{спам}) = \frac{45}{48}.$$

В результате, используя формулу Байеса, мы можем найти вероятность того, что письмо является спамом, если оно содержит слово «приз», она составит 0.625:

$$P(\text{спам} \mid \text{«приз»}) = \frac{P(\text{«приз»} \mid \text{спам}) \cdot P(\text{спам})}{P(\text{«приз»})} = \frac{\frac{5}{32} \cdot \frac{32}{80}}{\frac{8}{80}} = \frac{5}{8} = 0.625.$$

Как искать вероятность мы вспомнили, но как применить полученные результаты для решения задачи классификации? Цель классификации состоит в том, чтобы понять, к какому классу принадлежит объект. В нашем случае объект – это письмо, которое мы хотим отнести к наиболее вероятному классу.

Задачу можно поставить следующим образом. Пусть X «проверяемый» объект с p атрибутами (предикторами) X_1, X_2, \dots, X_p , для которого необходимо найти наиболее вероятный класс y^* из множества Y . Воспользуемся уже знакомым оператором $\arg \max$ для математической постановки задачи:

$$y^* = \arg \max_{y \in Y} P(\text{Класс} = y \mid X_1, X_2, \dots, X_p).$$

Запись X_1, X_2, \dots, X_p на языке событий означает, что все эти события произошли одновременно, то есть рассматривается условная вероятность события Класс = y при условии, что произошло событие $X_1 \cap X_2 \cap \dots \cap X_p$. Иными словами, что рассматривается объект X , обладающий атрибутами X_1, X_2, \dots, X_p . Таким образом, наивный байесовский классификатор использует оценку апостериорного максимума (maximum a posteriori hypothesis) для определения наиболее вероятного класса, согласно соотношению, которое моментально получается из формулы Байеса:

$$y^* = \arg \max_{y \in Y} \left(\frac{P(X_1, X_2, \dots, X_p \mid \text{Класс} = y) P(\text{Класс} = y)}{P(X_1, X_2, \dots, X_p)} \right).$$

В качестве Y выступают классы, в нашем примере их два: «спам» и «не спам», для каждого из которых вычисляется вероятность и выбирается тот, на котором достигается максимальное значение вероятности. В общем случае число классов может быть любым.

Выражение для y^* можно упростить, так как вероятность в знаменателе не зависит от y и ее можно не рассматривать:

$$y^* = \arg \max_{y \in Y} (\mathsf{P}(X_1, X_2, \dots, X_p | \text{Класс} = y) \mathsf{P}(\text{Класс} = y)).$$

Не предполагая независимости признаков, способ вычисления условной вероятности требует знания совместного распределения, что обычно невозможно. Почему? Да потому что вероятность $\mathsf{P}(X_1, X_2, \dots, X_p | \text{Класс} = y)$ – это вероятность ровно такого набора предикторов в ровно таком классе. В нашем примере – это вероятность ровно такого набора слов в одном и другом классах. Откуда взять эту статистику? Очевидно, что неоткуда.

А вот если предположить «наивность», то есть независимость событий, дело обстоит совершенно по-другому и вероятность представляется, как произведение всех условных вероятностей для X_1, X_2, \dots, X_p из класса y :

$$\mathsf{P}(X_1, X_2, \dots, X_p | \text{Класс} = y) = \mathsf{P}(X_1 | \text{Класс} = y) \cdot \mathsf{P}(X_2 | \text{Класс} = y) \cdot \dots \cdot$$

$$\mathsf{P}(X_p | \text{Класс} = y) = \prod_{i=1}^p \mathsf{P}(X_i | \text{Класс} = y).$$

Итого, задача сводится к нахождению:

$$y^* = \arg \max_{y \in Y} \left(\mathsf{P}(\text{Класс} = y) \prod_{i=1}^p \mathsf{P}(X_i | \text{Класс} = y) \right).$$

Отметим, что на этом этапе нет проблем с математикой, но вот компьютер может не справиться. Например, при большой длине письма, условные вероятности будут малы, а результат их перемножения компьютерными средствами, имеющими ограниченную точность, приведет нас к нулю. Для решения этой проблемы воспользуемся нашими любимыми логарифмами, ведь как мы уже знаем, логарифмирование не влияет на нахождение максимума, так как логарифм (натуральный) является монотонно возрастающей функцией:

$$y^* = \arg \max_{y \in Y} \left(\ln \left(\mathsf{P}(\text{Класс} = y) \prod_{i=1}^p \mathsf{P}(X_i | \text{Класс} = y) \right) \right).$$

Теперь представим выражение в виде суммы, так как логарифм произведения равен сумме логарифмов:

$$y^* = \arg \max_{y \in Y} \left(\ln \mathsf{P}(\text{Класс} = y) + \sum_{i=1}^p \ln \mathsf{P}(X_i | \text{Класс} = y) \right).$$

Применим описанный алгоритм к задаче классификации писем, так как в ней есть свои нюансы.

В рамках рассматриваемой задачи, $P(X_i|\text{Класс} = y)$ – вероятность встретить слово X_i в письмах, отнесенных к классу y . Она может быть найдена, как отношение числа $W_{X_i}(\text{Класс} = y)$ вхождений слова X_i в письмах класса y , к суммарному количеству слов во всех письмах этого класса:

$$P(X_i|\text{Класс} = y) = \frac{W_{X_i}(\text{Класс} = y)}{\sum_{j=1}^p W_{X_j}(\text{Класс} = y)}.$$

Важный момент! Что, если в письмах будут слова, которых не было в тестовой выборке? К чему это приведет? Такой документ попросту нельзя будет классифицировать, так как логарифм от нуля не существует. Если бы мы не использовали логарифмирование, то мы бы получили просто 0, что опять таки не даст классифицировать письмо. Как решить такую проблему? Все слова все равно не учтены в выборке, ведь возможны как опечатки, так и новые формы слов. На практике часто используют размытие по Лапласу (additive smoothing). Идея очень простая, и заключается в том, что мы предполагаем, как будто видели каждое слово на один раз больше, в результате чего получаем смещенную частоту. Таким образом числитель увеличен на единицу, а знаменатель на число W всех уникальных слов набора данных:

$$P(X_i|\text{Класс} = y) = \frac{W_{X_i}(\text{Класс} = y) + 1}{W + \sum_{j=1}^p W_{X_j}(\text{Класс} = y)},$$

при этом, введенное определение корректно, так как при суммировании всех числителей накопится ровно p единиц, что соответствует числу W уникальных слов набора:

$$\sum_{i=1}^p (W_{X_i}(\text{Класс} = y) + 1) = \sum_{i=1}^p W_{X_i}(\text{Класс} = y) + p = \sum_{i=1}^p W_{X_i}(\text{Класс} = y) + W.$$

Такое преобразование приведет к тому, что вероятности для часто встречающихся слов немного снижаются, а для редких слов, наоборот, повышаются.

Окончательно, оценка апостериорного максимума в рамках примера классификации писем, может быть найдена по следующей формуле, ее мы и будем использовать в практической части лекции:

$$y^* = \arg \max_{y \in Y} \left(\ln P(\text{Класс} = y) + \sum_{i=1}^p \ln \frac{W_{X_i}(\text{Класс} = y) + 1}{W + \sum_{j=1}^p W_{X_j}(\text{Класс} = y)} \right).$$

Но что, если важна вероятностная оценка исхода, а не сам факт? Мы же не хотим лишиться письма, если, например, фильтр уверен в этом лишь с

вероятностью 0.51? Чтобы это сделать, получившиеся оценки должны соответствовать вероятностным, то есть быть от 0 до 1, а их сумма должна быть равна единице. Формально решить надо две задачи: это обратно избавиться от логарифмов и нормировать результаты.

Для каждого класса $y \in Y$ вычислим значение

$$F(y) = \ln P(\text{Класс} = y) + \sum_{i=1}^p \ln \frac{W_{X_i}(\text{Класс} = y) + 1}{W + \sum_{j=1}^p W_{X_j}(\text{Класс} = y)}.$$

Тогда, чтобы получить вероятность отнесения объекта X к какому-то классу $y^* \in Y$, остается составить отношение следующего вида:

$$P(\text{Класс} = y^* | X_1, X_2, \dots, X_p) = \frac{e^{F(y^*)}}{\sum_{y \in Y} e^{F(y)}}.$$

А чтобы сократить число операций с экспонентой, можно сделать следующее преобразование:

$$P(\text{Класс} = y^* | X_1, X_2, \dots, X_p) = \frac{1}{e^{-F(y^*)} \sum_{y \in Y} e^{F(y)}} = \frac{1}{1 + \sum_{y \in Y, y \neq y^*} e^{F(y) - F(y^*)}},$$

где последняя сумма пробегает все классы, кроме класса y^* . Запишем упрощенную формулу для двух классов, обозначив оценку апостериорного максимума, как y^* , а для противоположного класса, как y , тогда:

$$P(\text{Класс} = y^* | X_1, X_2, \dots, X_p) = \frac{1}{1 + e^{F(y) - F(y^*)}}.$$

1.2 Пример классификации

Мы с вами в самом начале оттолкнулись от задачи фильтрации писем и пришли к модели наивного байесовского классификатора, которая оказалась крайне проста в реализации. Давайте теперь рассмотрим классификатор в деле. Нашими тренировочными данными будут три письма с текстом:

- Win a million rubles – СПАМ;
- Ruble drops again – НЕ СПАМ;
- A million ways to get rich – СПАМ.

Сформируем следующую сводную таблицу, опишем количество писем и количество слов, входящих в группы спам и не спам. Выпишем слова без повторений и укажем количество упоминаний их в письмах (слова rubles и ruble

будем считать за одно, так как на практике применяются методы обработки языка, а артикли и частицы опускаются).

	Спам	Не спам
Письма	2	1
Слова	7	3
win	1	0
million	2	0
ruble	1	1
again	0	1
drops	0	1
ways	1	0
get	1	0
rich	1	0

Поставим задачу классифицировать письмо с фразой «Get rich with ruble» к одной из групп.

Произведем расчет для класса спам. Вероятность того, что письмо является спамом, исходя из тренировочного набора, составит $\frac{2}{3}$, так как всего три письма, и два отмечены как спам. Найдем сразу натуральный логарифм от вероятности, так как в модели мы переходили от вероятностей к логарифмам:

$$\ln P(\text{спам}) = \ln \frac{2}{3}.$$

Далее запишем частоты. Слово «ruble» входит один раз в спам, следовательно, смещенная частота равна двум:

$$W_{ruble}(\text{Класс} = \text{спам}) + 1 = 1 + 1 = 2.$$

Слово «with» не упоминалось в тренировочных данных, значит его частота равна единице:

$$W_{with}(\text{Класс} = \text{спам}) + 1 = 0 + 1 = 1.$$

Для слов «get» и «rich», частота равна двум, так как они по одному разу входят в категорию спам:

$$W_{get}(\text{Класс} = \text{спам}) = 2, W_{rich}(Y = \text{спам}) = 2.$$

Знаменатель дробей состоит из числа уникальных слов по всем письмам – таких восемь, и из суммарного числа слов класса «Спам» – таких семь.

$$W + \sum_j W_{X_j}(\text{Класс} = \text{спам}) = 8 + 7 = 15.$$

Запишем логарифмы отношений и вычислим сумму. При выполнении заданий, рекомендуем это делать в любых математических пакетах или в Excel, а округлять только итоговый результат:

$$y_{\text{спам}}^* = \ln \frac{2}{3} + \ln \frac{2}{15} + \ln \frac{1}{15} + \ln \frac{2}{15} + \ln \frac{2}{15} \approx -9.158.$$

Для класса «не спам» расчет аналогичен, предлагаем выполнить его самостоятельно. Заметим что значение получится меньше и составит $y_{\text{не спам}}^* = -9.997$. А так как необходимо выбрать наибольшее, то тестовая фраза будет отнесена к классу «спам» с вероятностью около 0.613:

$$P(\text{спам} \mid \text{письмо}) = \frac{1}{1 + e^{-9.997 - (-9.158)}} \approx 0.698.$$

2 Метод k -ближайших соседей

Поговорим далее про метод k -ближайших соседей (k -nearest neighbors algorithm, k -NN), еще один алгоритм классификации, и, пожалуй, самый простой. Обучение модели, основанной на данном алгоритме, часто называют «ленивым обучением», так как на этапе обучения модель ничему не учится, в отличие от, скажем, регрессии. Вместо этого модель начинает работать только на этапе тестирования, для сравнения данных тестовых наблюдений с ближайшими обучающими наблюдениями, что занимает значительное время при сравнении каждой тестовой точки данных с каждой точкой тренировочного набора. Таким образом, можно отметить, что этот метод не эффективен на больших данных.

Метод ближайших соседей хорошо объясняется следующим коротким примером. Цель – определить за кого проголосует избиратель на выборах, исходя из своего окружения: близкие родственники, лучшие друзья и приятели. Мы предполагаем, что можем определить, за кого он проголосует, исходя из большинства голосов, отданных за конкретного депутата среди его круга общения. Важным, конечно, является значение k – числа, среди кого следует учитывать большинство голосов. Выбор этого k является вопросом на миллион.

Например, если мы разместим все голоса наших знакомых на диаграмме (рисунок 1) исходя из степени близости, и очертиим круг, в который попадут $k = 5$ соседей, то в получившейся ситуации мы отдадим предпочтение депутату – пятиугольнику, так как голосов за него оказалось большинство. Если же взять число k , равным единице, то голос будет отдан квадрату, ведь именно он является ближайшим соседом.

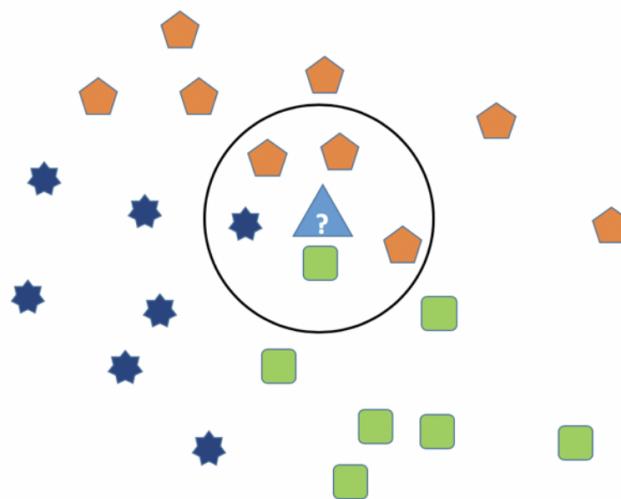


Рис. 1: За кого проголосовать?

В примере мы заговорили о близости объектов, так что резонно задаться вопросом, как эту близость измерять? Это действительно важный вопрос, давайте рассмотрим его отдельно, и уже потом перейдем к самому алгоритму k -NN классификатора.

2.1 Метрики

Когда все говорят о расстоянии, первое, что приходит голову – это взять линейку и произвести замер по прямой (по кратчайшему пути), получив заветное число. Будут ли это метры, футы не важно. В то же время ясно, что измерение по прямой не всегда дает нужный результат. Представьте себе, условно, глобус, и пингвина на южном полюсе. Пусть глобус имеет форму шара, а радиус шара равен, скажем, трем метрам. Пингвин хочет кратчайшим образом попасть на северный полюс, какое при этом расстояние он пройдет? Просто соединить две точки и получить 6 метров не выйдет – пингвин же не пройдет сквозь глобус, верно? А как тогда? Смекалистые, конечно, догадались, что пингвину нужно пройти половину длины окружности центрального сечения шара. Так как это сечение – круг радиуса 3, то половина длины окружности, его ограничивающей, это $3\pi \approx 9.4$ метра. Согласитесь, намного больше, чем 6. А если бы было какое-то сложное тело, а не шар?

Можно привести и другой жизненный пример. Представьте, что вы приехали на экскурсию в Санкт-Петербург и планируете свой маршрут по осмотру достопримечательностей. Расстояние от Зимнего дворца до Петропавловской крепости очень маленькое – рукой подать, да только Нева мешает. А чтобы дойти или доехать, придется сделать нехилый крюк: пройти по Дворцовому мосту, обогнуть здание биржи, пересечь Биржевой мост, еще чуть-чуть по набережной – и опять, расстояние в несколько раз больше!

В частности, по этим причинам рассматривают разные расстояния. В более общих понятиях, такие расстояния называют метриками, которые, формально, являются некоторыми математическими моделями сходства объектов. Рассмотрим основные метрики, которые встретятся нам далее:

1. Начнем с известной большинству слушателей евклидовой метрики или евклидова расстояния, являющейся, по сути, обобщением теоремы Пифагора. Так, расстояние d_E между точками $x = (x_1, x_2, \dots, x_n)$ и $x' = (x'_1, x'_2, \dots, x'_n)$ вычисляется, как корень из суммы квадратов разностей соответствующих координат:

$$d_E(x, x') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \dots + (x_n - x'_n)^2},$$

$$d_E(x, x') = \sqrt{\sum_{i=1}^n (x_i - x'_i)^2}.$$

2. На самом деле, евклидово расстояние – это лишь частный случай вот какого расстояния:

$$d_p(x, x') = \sqrt[p]{|x_1 - x'_1|^p + |x_2 - x'_2|^p + \dots + |x_n - x'_n|^p}, \quad p \geq 1,$$

$$d_p(x, x') = \sqrt[p]{\sum_{i=1}^n |x_i - x'_i|^p}, \quad p \geq 1.$$

Таким образом, расстояние $d_E = d_2$, при $p = 2$.

3. Если взять $p = 1$, получим следующее расстояние, которое называется манхэттенским или расстоянием городских кварталов:

$$d_1(x, x') = \sum_{i=1}^n |x_i - x'_i|.$$

Это расстояние, по сути, решает вопрос с передвижением по городу, если все улицы города строго перпендикулярны. Тогда произвольный «адекватный» маршрут, соединяющий две выбранные точки, имеет одинаковую длину, совпадающую с нашими о ней представлениями.

4. Если же число $p \rightarrow +\infty$ получим расстояние Чебышёва:

$$d_\infty(x, x') = \max_{1, \dots, n} |x_i - x'_i|.$$

Расстояние Чебышёва можно применять, например, вот в какой ситуации. Представьте, что вы совершаете замеры воды в водохранилищах $1, 2, \dots, n$, причем норма уровня воды в каждом из них равна x'_1, x'_2, \dots, x'_n . Тогда метрика Чебышёва показывает максимальное отклонение от нормы среди всех водохранилищ.

Что же, любая ли функция может быть метрикой? Конечно нет. В математике метрикой на множестве X называют такую функцию $d(x, y)$, что для любых $x, y, z \in X$ она удовлетворяет трем свойствам:

1. Она неотрицательна, то есть $d(x, y) \geq 0$, причем $d(x, y) = 0 \Leftrightarrow x = y$.
2. Она симметрична, то есть $d(x, y) = d(y, x)$.
3. Выполняется неравенство треугольника, то есть $d(x, y) \leq d(x, z) + d(y, z)$.

Первое требование очень логично: расстояние должно быть неотрицательным, причем оно равно нулю между объектами тогда и только тогда, когда объекты совпадают. Второе требование говорит, что мерить от точки x до y то же самое, что от y до x . Это вполне себе интуитивно, не так ли? Третье же есть не что иное, как неравенство треугольника, утверждающее, что если вершины треугольника находятся в точках x, y, z , то длина любой стороны не больше суммы длин двух других.

Можно проверить, что функции, которые мы обсуждали ранее, и правда являются метриками. Так как мы будем использовать метрики для сравнения «близости» объектов, немаловажно понимать их геометрическую интерпретацию. Для наглядности, сравним расстояния на плоскости, изобразив множества точек, которые удалены от центра $O(0, 0)$ на расстояние не больше чем 1, т.е. те множества $x \in \mathbb{R}^2$, для которых $d_p(O, x) \leq 1$ (рисунок 2).

В случае с евклидовым расстоянием, достаточно возвести выражение в квадрат и получить внутренность окружности, то есть круг:

$$d(O, x) = \sqrt{x_1^2 + x_2^2} \leq 1 \Rightarrow x_1^2 + x_2^2 \leq 1.$$

Для манхэттенского расстояния получаем внутренность ромба (на самом деле повернутого квадрата):

$$|x_1| + |x_2| \leq 1.$$

И для расстояния Чебышёва внутренность квадрата со стороной два:

$$\max_{i=1,2} |x_i| \leq 1 \Rightarrow |x_1| \leq 1, |x_2| \leq 1.$$

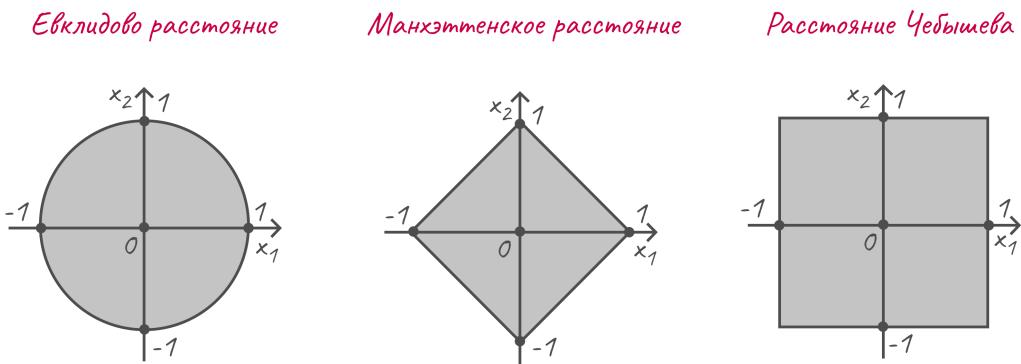


Рис. 2: Расстояния слева направо: евклидово, манхэттенское, Чебышёва.

2.2 Построение классификатора

Вооружившись метриками, мы можем перейти к построению алгоритма классификатора. Для нашего алгоритма важно, чтобы объекты, которые

меньше похожи друг на друга, были дальше друг от друга. Тут и пригодятся разные расстояния. Определить, какое из них подходит лучше, можно практическим путем, оценив полученный результат модели.

Давайте перейдем к алгоритму. Пусть имеется обучающая выборка $X = (X_1, \dots, X_n)$ объема n с откликами $y \in Y$, а на множестве объектов задана некоторая функция расстояния $d_p(x, x')$, далее d . При этом d достаточно качественно отображает сходство объектов, то есть чем больше значение d , тем менее схожими являются объекты x и x' . Итак, алгоритм состоит из следующих шагов:

1. Для нового объекта Z вычисляется расстояние d до каждого объекта X_i , $i \in \{1, 2, \dots, n\}$.
2. Исходный набор данных сортируется по возрастанию значений d для рассматриваемого Z , что приводит к перенумерованной выборке $X^{(Z)} = (X_1^{(Z)}, X_2^{(Z)}, \dots, X_n^{(Z)})$. В таких обозначениях первый сосед – это самый близкий к объекту Z объект исходной выборки, он становится элементом $X_1^{(Z)}$. Второй по близости объект становится элементом $X_2^{(Z)}$ и так далее.
3. Отклики также перенумеруем для каждого i -го соседа, обозначим их $y_i^{(Z)}$.
4. И, наконец, среди ближайших k соседей, найдем такой класс $y \in Y$, который встречается большее число раз:

$$a(Z; X^{(Z)}; k) = \arg \max_{y \in Y} \sum_{i=1}^k \mathbf{I}(y_i^{(Z)} = y),$$

где функция $\mathbf{I}(A)$ – индикатор события A , равна единице, если событие A произошло (истинно), а иначе нуль. В рассматриваемом случае индикатор дает единицу, если класс $y_i^{(Z)}$ равен классу y .

Отметим, что на практике признаки могут иметь разные единицы измерения, что может искажать реальное расстояние между объектами. В таких случаях производят нормализацию данных. Напомним, что один из самых простых способов – переход к относительным значениям с помощью линейной нормировки:

$$\overline{X}_i = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}},$$

где X_{\min} – минимальное значение рассматриваемого предиктора, X_{\max} – максимальное значение.

2.3 Пример классификации

В качестве примера возьмем следующие входные данные из таблицы. Может показаться, что они немного притянуты за уши, но хорошо покажут идею метода.

Продукт	Сладость	Хруст	Класс
банан	10	1	фрукт
апельсин	7	4	фрукт
виноград	8	3	фрукт
креветка	2	2	протеин
бекон	1	5	протеин
орехи	3	3	протеин
сыр	2	1	протеин
рыба	3	2	протеин
огурец	2	8	овощ
яблоко	9	8	фрукт
морковь	4	10	овощ
сельдерей	2	9	овощ
салат айсберг	3	7	овощ
груша	8	7	фрукт

Итак, начнем с того, что эти данные легко можно разместить на плоскости (рисунок 3). По горизонтали отметим сладость, по вертикали хрусткость. Таким образом каждому фрукту, овощу или протеину соответствует точка на плоскости.

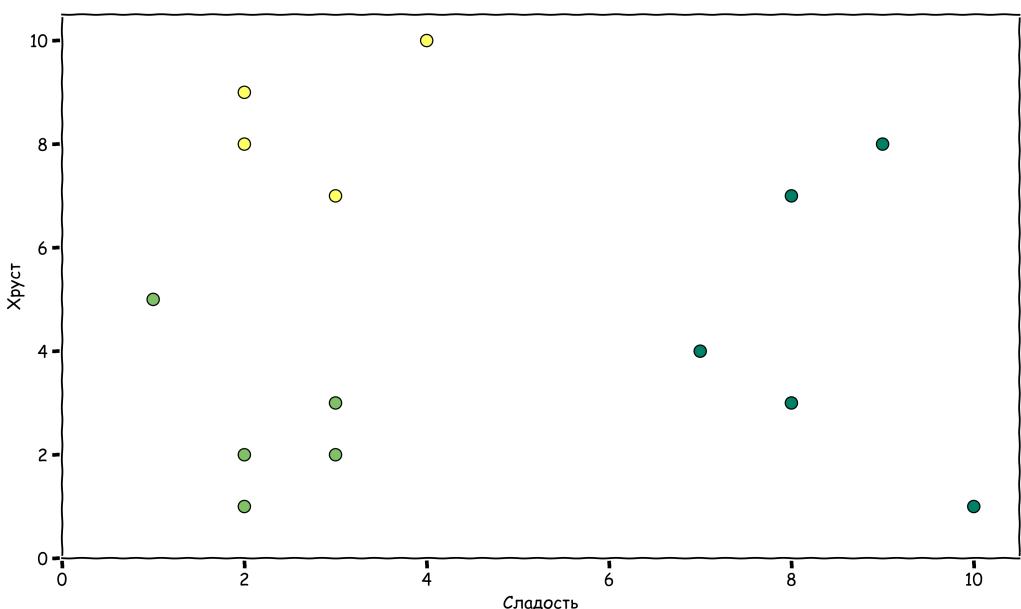


Рис. 3: Визуализация значений таблицы.

Кроме того, как видим, данные легко выделить в группы (рисунок 4).

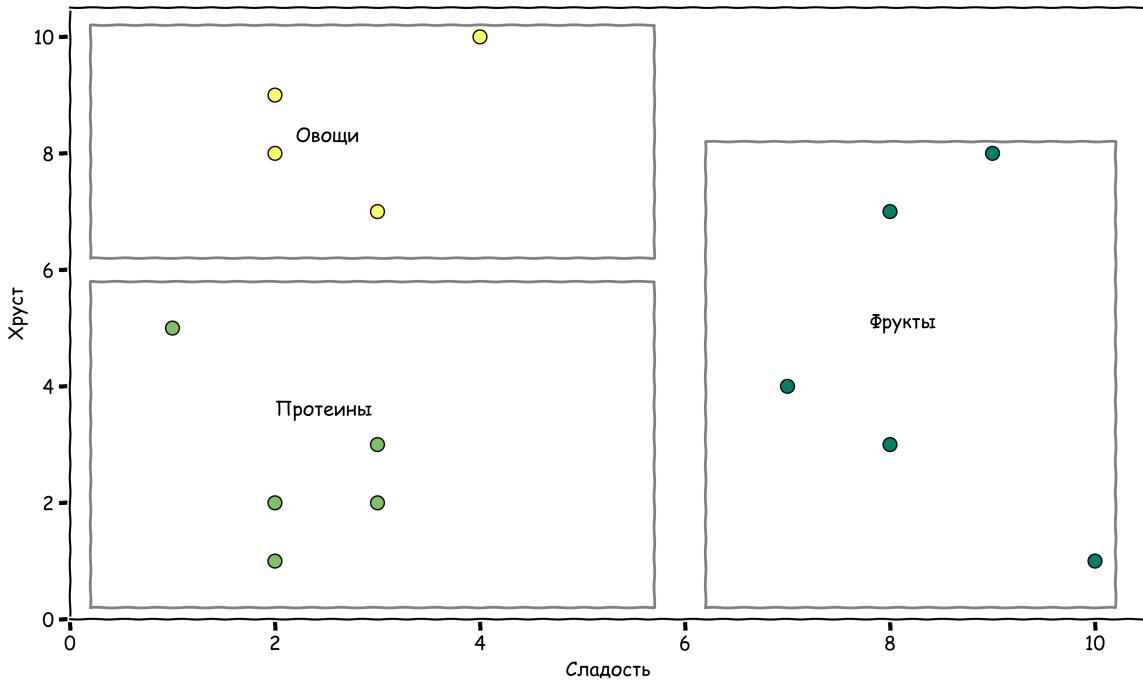


Рис. 4: Объединение в группы.

Теперь предположим, что нам необходимо классифицировать новый объект, обладая только значениями сладости и хруста. Допустим, это перец со значениями сладость — 6, хруст — 9. Согласно методу, нам необходимо рассчитать расстояние до других объектов. Воспользуемся для начала евклидовой метрикой. В нашем примере все переменные будем считать нормированными по шкале от 0 до 10. Произведем расчет длины между точкой «морковь» и точкой «перец», расстояние составит около 2.24:

$$d_E(\text{морковь}, \text{перец}) = \sqrt{(6 - 4)^2 + (9 - 10)^2} \approx 2.24.$$

Аналогичный расчет произведем для всех точек:

Продукт	Расстояние
банан	8.94
апельсин	5.10
виноград	6.32
креветка	8.06
бекон	6.4
орехи	6.71
сыр	8.94
рыба	7.62
огурец	4.12
яблоко	3.16
морковь	2.24
сельдерей	4
салат айсберг	3.61
груша	2.83

Отсортируем таблицу по возрастанию расстояния:

Продукт	Расстояние
морковь	2.24
груша	2.83
яблоко	3.16
салат айсберг	3.61
сельдерей	4
огурец	4.12
апельсин	5.10
виноград	6.32
бекон	6.4
орехи	6.71
рыба	7.62
креветка	8.06
банан	8.94
сыр	8.94

Теперь следует выбрать число k . Начнем с единицы, то есть выберем ближайший объект – это морковь, принадлежащая классу овощей, и, следовательно, перец также будет классифицирован этим классом, то есть классом овощ. При $k = 3$, самыми близкими окажутся яблоко, груша и морковь, тогда перец будет отнесен к классу фруктов, так как их большинство. При $k = 4$ в выборку добавится салат, и вероятность отнесения к каждому из классов составит 0.5, что с равными шансами позволяет присвоить объекту как класс фрукт, так и овощ. Именно по этой причине как правило игнорируют четные

значения k , чтобы избежать равенства классов, однако об этом мы еще поговорим подробней. При $k = 5, 6, 7$, перец будет уверенно классифицироваться, как овощ.

Продукт	Класс	Расстояние
морковь	овощ	2.24
груша	фрукт	2.83
яблоко	фрукт	3.16
салат айсберг	овощ	3.61
сельдерей	овощ	4
огурец	овощ	4.12
апельсин	фрукт	5.10
виноград	фрукт	6.32
бекон	протеин	6.4
орехи	протеин	6.71
рыба	протеин	7.62
креветка	протеин	8.06
банан	фрукт	8.94
сыр	протеин	8.94

Что, если взять другую метрику? Давайте попробуем манхэттенскую, которая вычисляется, как сумма модулей разности координат точек x и x' :

$$d_1(x, x') = \sum_{i=1}^n |x_i - x'_i|.$$

Возьмем все тот же перец, сладость – 6, хруст – 9. Найдем расстояние между точкой «морковь» и точкой «перец», которое составит ровно 3:

$$d_1(\text{морковь}, \text{перец}) = |6 - 4| + |9 - 10| = 3.$$

Различия в метриках видны на рисунке 5. Так, если евклидово расстояние d_E – это кратчайший путь по прямой, то манхэттенское расстояние d_1 строится отрезками под углом в 90 градусов.

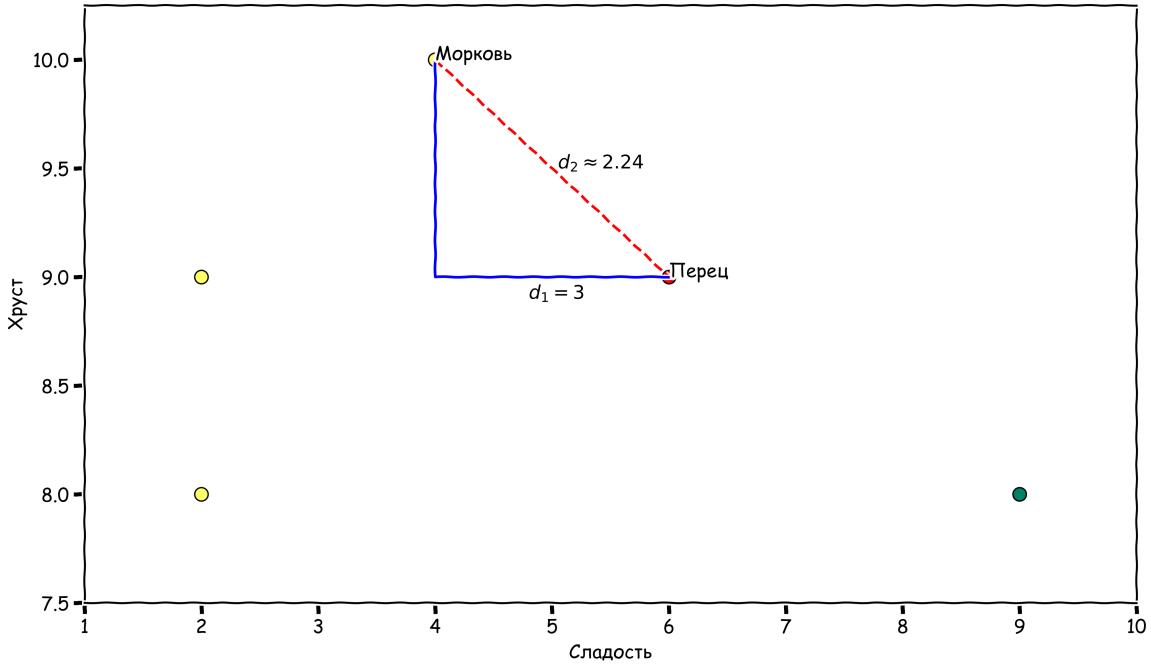


Рис. 5: Отличие в расстояниях.

Аналогичный расчет произведем для всех точек и сразу отсортируем продукты по возрастанию расстояний.

Продукт	Класс	Расстояние
морковь	овощ	3
яблоко	фрукт	4
сельдерей	овощ	4
груша	фрукт	4
огурец	овощ	5
салат айсберг	овощ	5
апельсин	фрукт	6
виноград	фрукт	8
бекон	протеин	9
орехи	протеин	9
рыба	протеин	10
креветка	протеин	11
банан	фрукт	12
сыр	протеин	12

Теперь мы не сможем уверенно классифицировать перец при $k = 2, 3, 4$, но сможем отнести его к его классу овощей при $k = 1, 5, 6, 7$. Также можно обратить внимание, что манхэттенское расстояние дает нам целые значения в рамках рассматриваемого примера и имеется очень много одинаковых расстояний.

2.4 Взвешенный k-NN

На текущий момент мы рассмотрели простейший случай алгоритма k -NN, однако столкнулись с проблемой равного числа вхождений элементов класса. Как же решить эту проблему? В таком случае проводится так называемое взвешенное голосование, где каждому соседу i соотносится вес w_i . Тогда объект относится к тому классу, который набирает наибольший суммарный вес среди k соседей. Один из способов задать веса основан на близости «проверяемого» объекта по отношению к другим. Идея проста, чем меньше расстояние, тем более значимым является объект. Например, вес w_i можно взять обратно пропорциональным квадрату длины между объектами x и x' :

$$w_i(x, x') = \frac{1}{d^2(x, x')}.$$

Тогда, возвращаясь к алгоритму k -NN, мы будем искать такой класс среди ближайших k соседей, который имеет наибольший суммарный вес:

$$a(Z; X^{(Z)}; k) = \arg \max_{y \in Y} \sum_{i=1}^k \mathbf{1}(y_i^{(Z)} = y) \cdot w_i(Z, X_i^{(Z)}).$$

Далее мы рассмотрим пример, но еще немного поговорим про число k .

2.5 Дilemma смещения–дисперсии

Вы справедливо задаетесь важным вопросом, как выбрать переменную k и каковы последствия для вашего классификатора с большим числом данных. Как и в большинстве алгоритмов машинного обучения, параметр k – это гиперпараметр, который вы должны выбрать самостоятельно, чтобы получить наилучшее соответствие набору данных.

Когда k мало, мы ограничиваем область данного прогноза и, по сути, заставляем наш классификатор быть «более слепым» к общему распределению. С другой стороны, это защищает от «выбросов» – объектов с неправильно определенным классом или просто аномальных значений в данных, так как вероятность попадания такой записи в число k ближайших соседей обычно мала. Однако может так получиться, что «выброс» окажется рядом и повлияет на последующую классификацию. С другой стороны, более высокое значение k «откроет глаза» нашему классификатору и, следовательно, сделает его более устойчивым к таким «выбросам», если они рядом.

Такая проблема называется компромиссом или дилеммой между смещением и дисперсией (Bias–Variance Tradeoff). Лучшего всего эта проблема визуализируется в виде мишени.

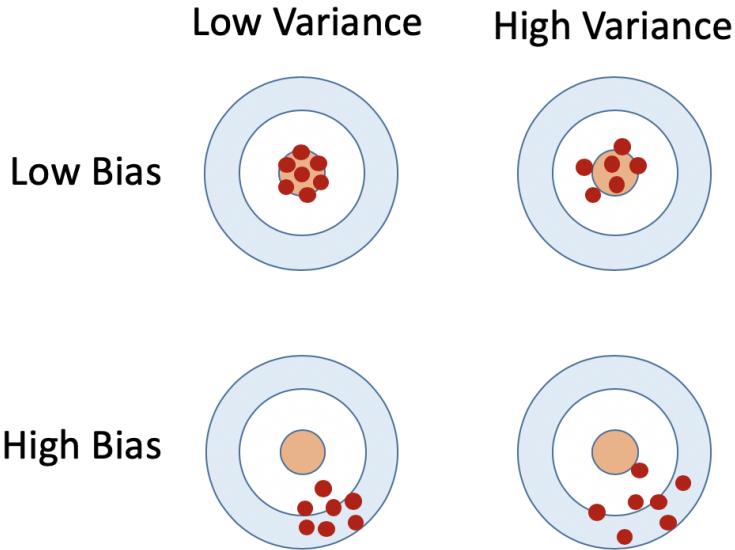


Рис. 6: Определение порога отсечения. Потом взять из презентации картинку!!!

Конечно, в результате обучения и выбора параметров мы стремимся к низкой дисперсии и смещению, то есть стремится попасть в яблочко! Однако, как правило, стремясь уменьшить один параметр, возрастает другой и наоборот. При малых значениях k будет низкое смещение и высокая дисперсия, при больших значениях k – наоборот. Это можно проследить по следующим рисункам, где для каждой точки плоскости было вычислено расстояние до всех точек тренировочного набора данных, а затем, в зависимости от значения k , каждая точка была классифицирована тем или иным образом. То есть, если мы поместим новый объект в зеленую область, он будет классифицирован как класс, соответствующий этой области.

Обратимся к рисунку 7, где видно, что граница классов будет более неровной при малых значениях k , имеются даже некоторые «островки», что с точки зрения модели соответствует низкому смещению. С другой стороны, при увеличении k граница разделения становится настолько гладкой, что в данном случае сравнима с линейной регрессией, которой как правило характерно высокое смещение.

Так какое значение k лучше? Конечно то, которое соответствует самой низкой частоте ошибок теста. Самым разумным подходом является оценка частоты ошибок теста путем разделения исходного набора данных на тестовую и обучающую выборки. Существуют различные способы, которые используются на практике, мы рассмотрим k -блочная или k -fold кросс-проверка.

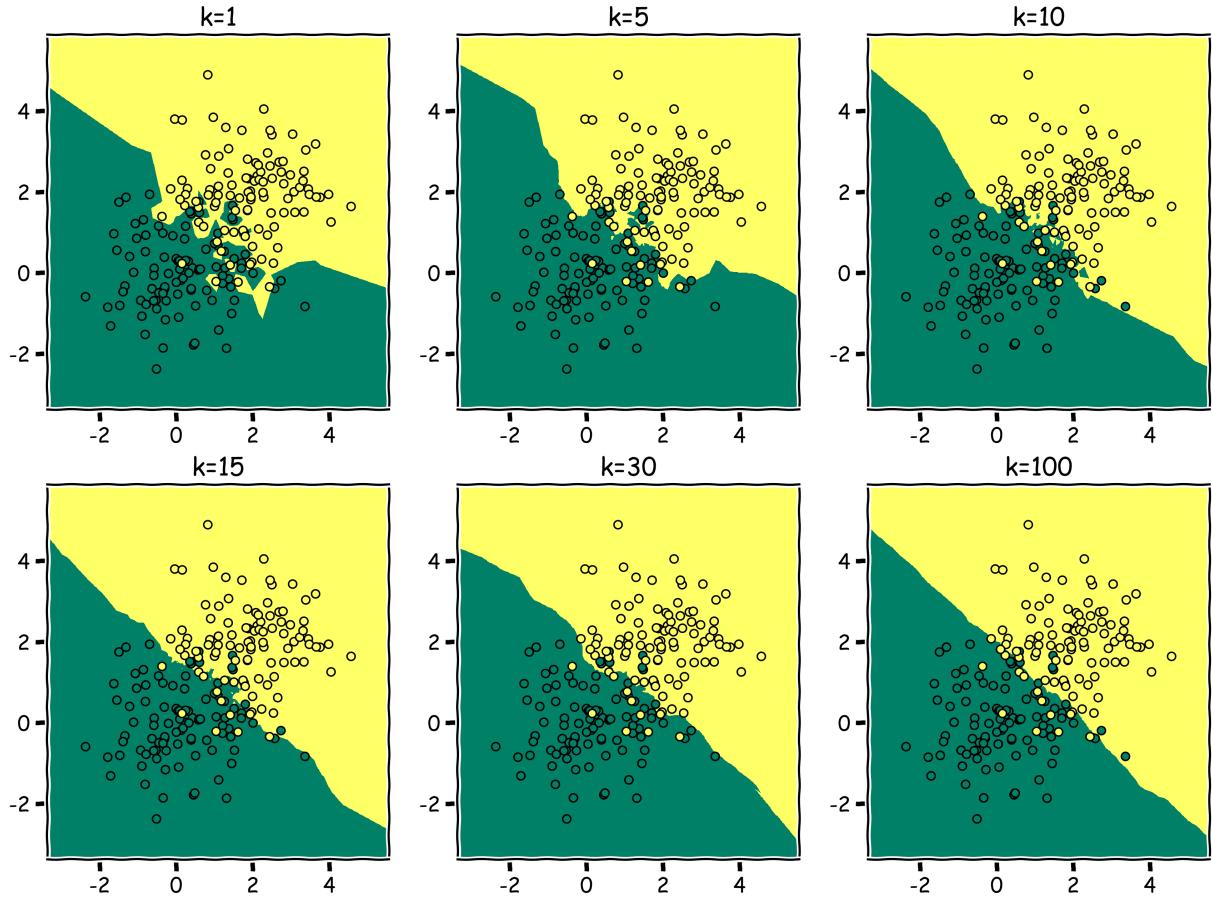


Рис. 7: Влияние числа k .

2.6 Кросс-валидация

Кросс-валидация или скользящий контроль – алгоритм, позволяющий оценить обобщающую способность модели за счет выделения тестовой выборки из обучающего набора данных. Алгоритм k -fold кросс-проверки крайне прост и состоит из следующих шагов:

1. Исходный набор данных X объема n разбивается на k непересекающихся одинаковых по размеру блоков $X = X_1 \cup X_2 \cup \dots \cup X_k$, если n кратно числу k . В противном случае размер определяется почти одинаково. Например, если исходный объем 10, а $k = 4$, блоки могут быть размерности 3, 3, 2, 2 соответственно.
2. Выполняется k итераций, на каждой из которых:
 - (a) Модель обучается на $k - 1$ блоках.
 - (b) Модель тестируется на блоке, не участвующем в обучении, с номером соответствующей итерации $i \in \{1, 2, \dots, k\}$. Таким образом, модель можно оценить любой метрикой качества и найти k статистик accuracy_i . Это может быть и уже знакомое значение AUC,

площади под ROC-кривой, или, например, частота ошибок:

$$\text{ErrorRate} = \frac{1}{m} \cdot \sum_{i=1}^m \mathbb{I}(y_i \neq y_i^*),$$

где y_i – исходный класс объекта, y_i^* – назначенный класс, m – объем тестовой выборки.

3. В конце все полученные оценки усредняются:

$$\text{Accuracy} = \frac{1}{k} \cdot \sum_{i=1}^k \text{accuracy}_i.$$

Вернемся к нашему примеру с продуктами и применим алгоритм кросс-валидации, при $k = 3$. Объем выборки $n = 14$, а значит блоки будут почти одинакового размера 5, 5, 4. На первом этапе будем использовать блок 1 как тестовый (таблица 1), а блоки 2 и 3 как тренировочные (таблица 2).

Продукт	Сладость	Хруст	Класс
банан	10	1	фрукт
апельсин	7	4	фрукт
виноград	8	3	фрукт
креветка	2	2	протеин
бекон	1	5	протеин

Таблица 1: Итерация 1. Тестовые данные.

Продукт	Сладость	Хруст	Класс
орехи	3	3	протеин
сыр	2	1	протеин
рыба	3	2	протеин
огурец	2	8	овощ
яблоко	9	8	фрукт
морковь	4	10	овощ
сельдерей	2	9	овощ
салат айсберг	3	7	овощ
груша	8	7	фрукт

Таблица 2: Итерация 1. Тренировочные данные.

Отобразим на плоскости данные из тренировочного набора (рисунок 8) и проведем классификацию продуктов из тестового набора методом взвешенных k -NN, при $k = 4$. В качестве метрики используем евклидово расстояние.

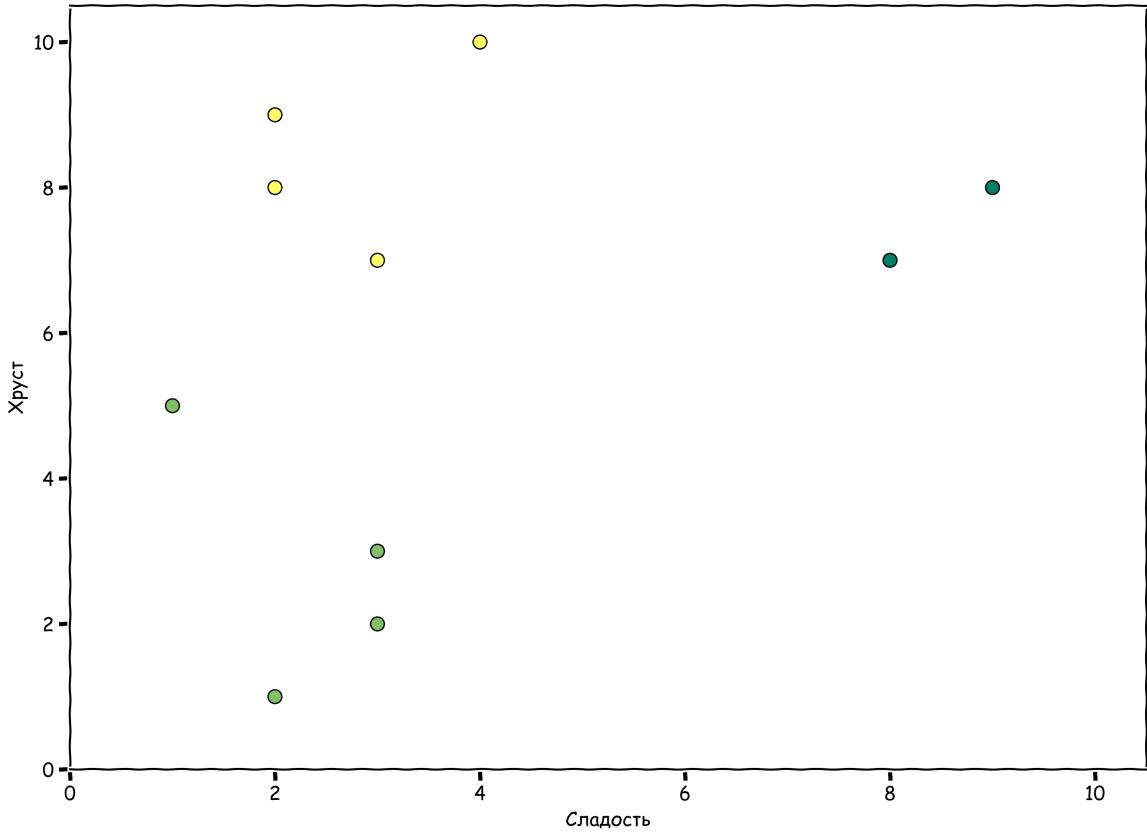


Рис. 8: Итерация 1. Тренировочные данные.

Найдем расстояние от продукта «банан» до всех объектов. Напомним, как вычисляется расстояние на примере сыра:

$$d_E(\text{банан}, \text{сыр}) = \sqrt{(10 - 2)^2 + (1 - 1)^2} = 8.$$

Так как мы будем использовать взвешенный k -NN, найдем вес

$$w_{\text{банан, сыр}} = \frac{1}{d_E^2(\text{банан, сыр})} = \frac{1}{64} \approx 0.016.$$

Аналогично подсчитаем расстояния и веса до остальных объектов, данные представлены в таблице 3. Назначим класс на основе четырех ближайших соседей, которыми окажутся груша, яблоко, рыба и орехи. Имеем два класса, фрукты и протеины, к одному из которых будет произведена классификация. Сумма весов по протеинам составит 0.039, а по фруктам 0.045. Таким образом банан будет классифицирован как фрукт.

Продукт	Класс	Расстояние	Вес
орехи	протеин	7.28	0.019
сыр	протеин	8	0.016
рыба	протеин	7.071	0.020
огурец	овощ	10.63	0.009
яблоко	фрукт	7.071	0.020
морковь	овощ	10.817	0.009
сельдерей	овощ	11.314	0.008
салат айсберг	овощ	9.22	0.012
груша	фрукт	6.325	0.025

Таблица 3: Расстояние и веса для продукта «банан»

Аналогичные вычисления выполняются для каждого продукта из тестового набора (таблица 4). Вы можете проверить себя и выполнить расчеты, мы предоставим итоговый результат классификации.

Продукт	Сладость	Хруст	Исходный класс	Назначенный класс
банан	10	1	фрукт	фрукт
апельсин	7	4	фрукт	фрукт
виноград	8	3	фрукт	фрукт
креветка	2	2	протеин	протеин
бекон	1	5	протеин	овощ

Таблица 4: Итерация 1. Тестовые данные.

Как видим из таблицы, имеет место 1 ошибка, а значит частота ошибок составит 0.2:

$$\text{ErrorRate}_1 = \frac{1}{5} = 0.2.$$

Далее вы можете ознакомиться с результатами итераций 2 и 3, для которых повторяются аналогичные шаги (таблицы 5, 6). В этих итерациях назначенные классы полностью совпадают с исходными. Таким образом частота ошибок будет нулевой, а усреднение всех ошибок по трем итерациям составит около 0.07:

$$\text{ErrorRate} = \frac{1}{3} \sum_{i=1}^3 \text{ErrorRate}_i = \frac{1}{15} \approx 0.07.$$

2.7 Проклятие размерностей

Разобравшись с методом, теперь нам стоит осознать так называемое проклятие размерностей. Суть в том, чтобы разобраться, когда метод ближай-

Продукт	Сладость	Хруст	Исходный класс	Назначенный класс
орехи	3	3	протеин	протеин
сыр	2	1	протеин	протеин
рыба	3	2	протеин	протеин
огурец	2	8	овощ	овощ
яблоко	9	8	фрукт	фрукт

Таблица 5: Итерация 2. Тестовые данные.

Продукт	Сладость	Хруст	Исходный класс	Назначенный класс
морковь	4	10	овощ	овощ
сельдерей	2	9	овощ	овощ
салат айсберг	3	7	овощ	овощ
груша	8	7	фрукт	фрукт

Таблица 6: Итерация 3. Тестовые данные.

ших соседей ухудшает свою предсказательную способность.

Проклятие размерностей может быть объяснено на следующем примере. Допустим, вы обронили монетку на беговой дорожке длиной в 100 метров. Ее не будет трудно найти, достаточно пройти вдоль дорожки, что займет около минуты. Теперь предположим, что вы обронили монетку в поле размером 100 на 100 метров. Поиск уже затруднится, так как придется пройти каждый сантиметр площадки, сопоставимый по размерам с футбольным полем. Если мы перейдем в трехмерное пространство, то есть будем иметь дело с кубом со сторонами по 100 метров, это будет сопоставимо с поисками монетки по внутреннему торговому центру. Сложность поиска в N -мерном пространстве становится еще выше. Все это и есть проклятие размерности.

Что же произойдет, если монеток больше одной? Окажутся ли они близко друг к другу, или нам придется изучить каждый уголок пространства?

Представим себе куб объемом 1, в пространстве, размерности N , внутри которого n маленьких кубиков объема $\frac{1}{n}$ распределенных равномерно, в каждом из которых содержится по одной точке. Тогда если мы хотим для новой тестовой точки, найти k соседей, то они должны находиться в некотором кубе со стороной $l \leq 1$, объем которого, с одной стороны, вычисляется как l^N , а с другой стороны, должен быть около $\frac{k}{n}$:

$$l^N \approx \frac{k}{n}.$$

Таким образом, сторона такого куба может быть вычислена через размер-

ность пространства и объем выборки:

$$l \approx \left(\frac{k}{n}\right)^{\frac{1}{N}}.$$

Что интересно, если $N \rightarrow \infty$ сторона маленького куба стремится к единице, а значит мы можем забить куб единичного объема бесконечным числом кубов единичного объема, парадокс! :)

Например, если объем выборки $n = 1000$, число соседей $k = 5$, оказывается, что сторона куба $l \approx 0.948$ в стомерном пространстве ($N = 100$):

$$\left(\frac{5}{1000}\right)^{\frac{1}{100}} \approx 0.948,$$

а это значит что нужно изучить почти все пространство, чтобы найти 5 соседей, что полностью рушит идею алгоритма k -NN, ведь объекты со схожим классом должны располагаться близко. Тут вы можете сказать, что надо добавить данных. Это, конечно, верно, однако такое количество данных надо еще найти. Например, если мы зафиксируем размер нашего маленького куба, окажется, что объем выборки n должен расти экспоненциально ($l \leq 1$):

$$l^N \approx \frac{k}{n} \Rightarrow n \approx kl^{-N}.$$

Гистограммы на рисунке 9, хорошо демонстрируют, что с увеличением размерности, попарные расстояния между 1000 случайно распределенных точек в пределах куба объема 1 начинают концентрироваться в малой области (обратите внимание на значения, отмеченные на горизонтальных осях). Таким образом выходит, что точки находятся далеко друг от друга, но при этом в среднем на одинаковом расстоянии.

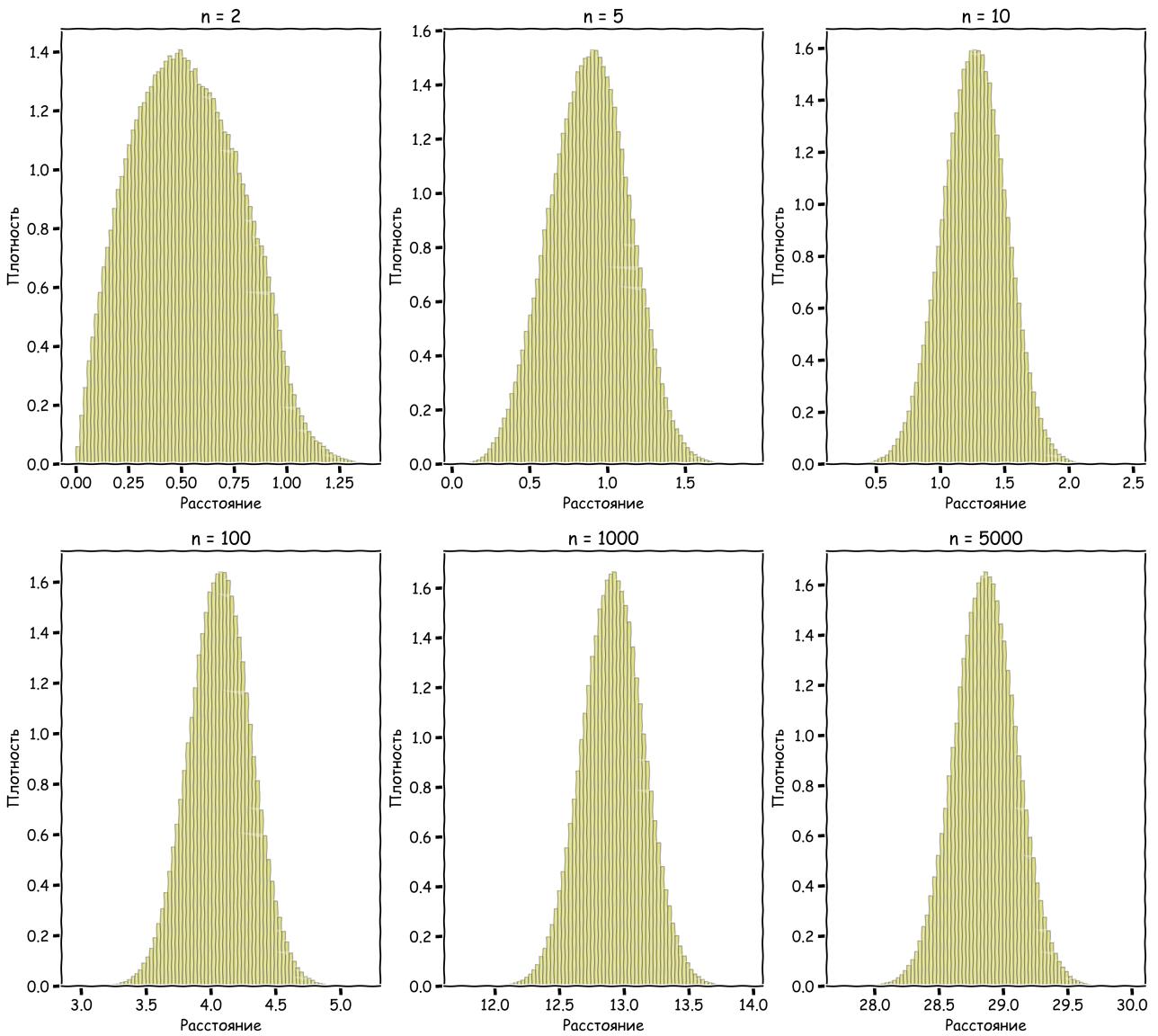


Рис. 9: Зависимость попарных расстояний от размерности пространства.

Все это приводит нас к основной идее – количество признаков выборки не должно быть большим. Но если вы имеете дело с такими данными, их необходимо обработать так, чтобы размерность пространства снизилась.

На этом мы заканчиваем знакомство с алгоритмами, основанными на обучении с учителем, и в следующей лекции начнем знакомство с обучением без учителя на примере кластеризации набора данных.