GeekBrains

Классификация. Задачи Классификации. Логистическая регрессия

Библиотеки Python для Data Science

Оглавление

Введение

Сегодня мы с вами продолжим знакомиться с линейными моделями и с типом задач Обучения с учителем. И посмотрим на другую группу задач — задачи классификации. Посмотрим, чем задача классификации отличается от задачи регрессии. Какие проблемы вы сможете решить с помощью задачи классификации.

На этой лекции вы найдете ответы на такие вопросы как:

- Какие задачи решает классификация
- Что такое логистическая регрессия
- Метрики качества классификации

Термины, используемые в лекции

Задача классификации – это задача обучения с учителем (supervised learning), при которой модель должна классифицировать объекты (например, фотографии,

текстовые документы или звуковые сигналы) на несколько заранее определенных категорий (классов).

Модель SVM (Support Vector Machine, машина опорных векторов) – это алгоритм машинного обучения, который подходит для задачи классификации данных. Он использует опорные векторы, которые представляют точки данных в пространстве признаков, чтобы разделить разные классы данных.

Линейная разделимость – относится к возможности линейно разделить элементы векторного пространства на две или более группы с помощью гиперплоскости.

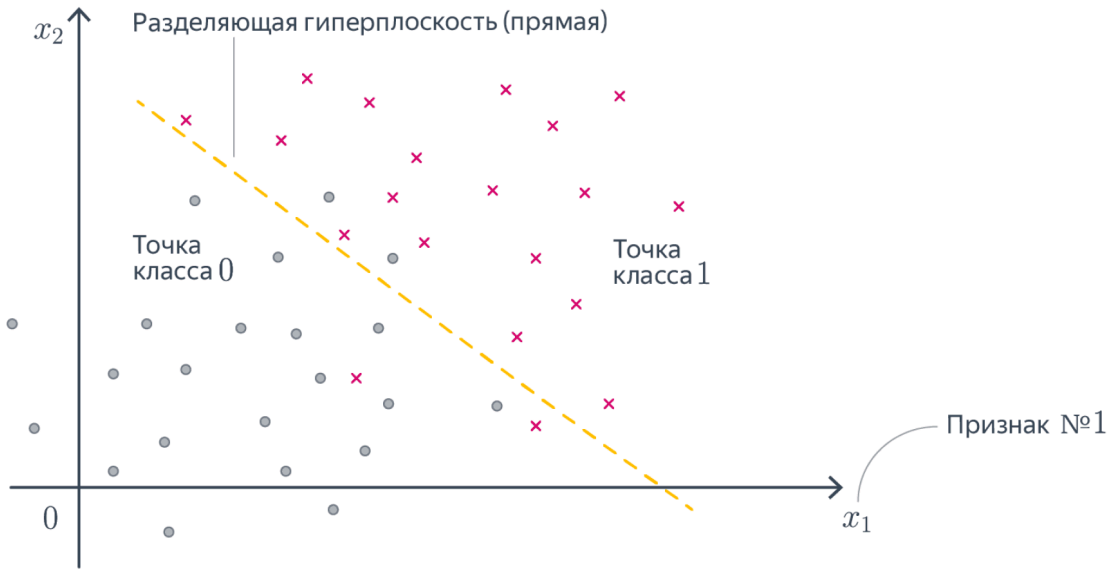
Классификация. Задача Классификации

Задача классификации в машинном обучении заключается в определении к какому классу (или категории) относится некоторый объект на основе имеющихся данных. Примерами задач классификации могут быть определение: раковой опухоли по результатам медицинских анализов, письма спам или не спам, наличия или отсутствия дефектов на производственных деталях, фотографии животных по их виду и т.д.

Для решения задачи классификации используются различные алгоритмы машинного обучения, например: логистическая регрессия, решающие деревья, случайный лес, нейронные сети и т.д. Алгоритмы классификации обучаются на наборе данных, где уже известно, к какому классу принадлежит каждый объект, затем полученная модель может быть применена для определения класса новых объектов.

Теперь я расскажу о задаче классификации. Для начала мы поговорим о бинарной классификации, которая предполагает классификацию двух классов. Обобщить эту задачу до задачи классификации на n классов не составит большого труда.

Предположим, что **y** - целевая переменная принадлежит положительному или отрицательному классу, т.е. множеству -1,1 (мы договорились представлять классы таким образом, но на практике часто встречаются метки 0,1), а **x** по-прежнему является вектором из пространства признаков. Мы хотим обучить линейную модель таким образом, чтобы плоскость, которую она описывает, как можно лучше отделяла объекты одного класса от объектов другого класса.



В идеальной ситуации существует плоскость, разделяющая классы, с положительными классами по одну сторону и отрицательными - по другую. Случаи, когда это возможно, называются линейно разделимыми случаями. В реальности, однако, такое случается редко.

Разделимость данных является важным понятием в машинном обучении, используемым для определения возможности классификации объектов в заданных наборах данных. Разделимость означает, что объекты могут быть разделены на два или более класса таким образом, чтобы объекты внутри каждого класса были похожи между собой, но отличались от объектов в другом классе.

Допустим, рассмотрим набор данных, состоящий из нескольких столбцов - "имя", "возраст" и "зарплата". Если мы хотим исследовать, есть ли различия в зарплате в зависимости от возраста, мы можем использовать способ разделимости данных.

Примерно половина людей в наборе данных моложе 30 лет, а другая половина старше 30 лет. Мы можем разделить данные на две группы - молодежь (возраст до 30 лет) и взрослых (возраст 30 и выше) - и сравнить их средние зарплаты.

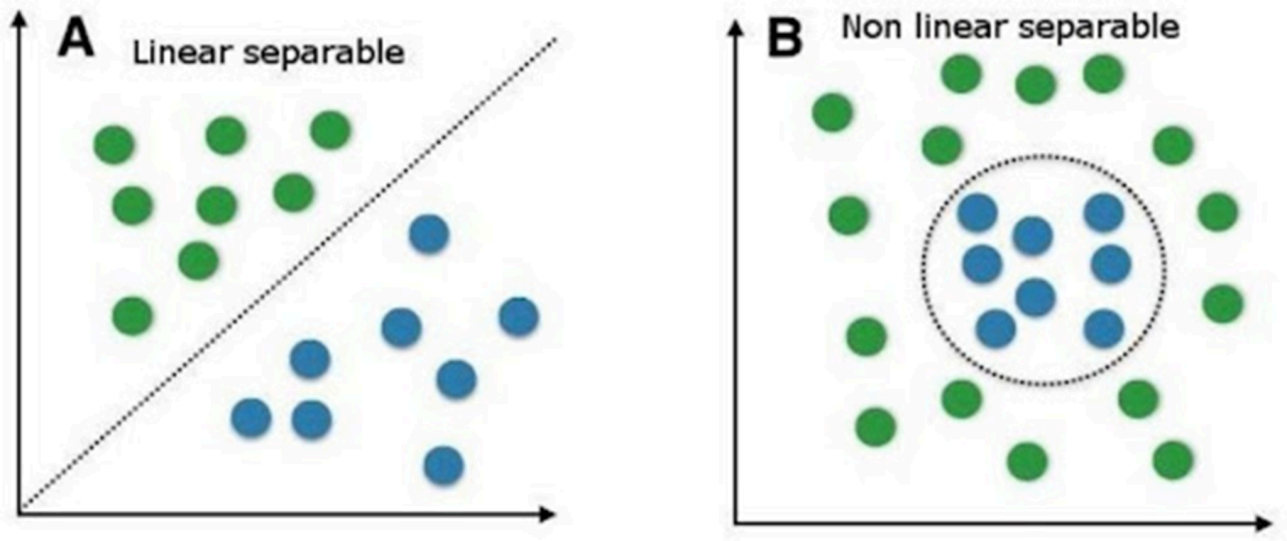
Если у молодежи средняя зарплата явно ниже, чем у взрослых, то можно сделать вывод о неравенстве зарплат в зависимости от возраста. Это может указывать на возможные проблемы с возрастной дискриминацией или различиями в опыте и квалификации работников разного возраста.

Это пример использования разделимости данных для анализа и сравнения различных групп в наборе данных. Он позволяет выявить возможные

закономерности и различия между группами, что может быть полезно для принятия решений и анализа данных.

Визуально разделить объекты на два класса можно с помощью графика, который отображает признаки каждого объекта в двух измерениях. График может представлять собой точки, расположенные на плоскости, и каждая точка представляет собой один объект с его признаками. Если точки одного класса находятся по одну сторону гиперплоскости (линии в двумерном пространстве), а точки другого класса — по другую сторону, то можно говорить об условной разделимости данных.

Однако, часто данных не разделить обычной прямой линией или гиперплоскостью. В таких случаях можно использовать изогнутые границы, такие как кривые Безье, параболы и окружности. Однако это уже будет говорить о безусловной разделимости данных.



Задача определения разделимости данных является важной, так как от нее зависит, можно ли использовать простые модели машинного обучения, такие как логистическая регрессия или метод опорных векторов (SVM Support Vector Machine), для классификации объектов в данном наборе данных. Если данные разделены условно или не разделены вовсе, то необходимо использовать более сложные модели машинного обучения, такие как деревья решений, нейронные сети, глубокое обучение и т.д.

Кроме того, для определения разделимости данных широко используется техника визуального анализа данных, такая как scatter plot и heatmap. Scatter plot

отображает каждый объект в двумерном пространстве, которое основано на двух признаках, тогда как heatmap представляет собой таблицу, где каждая ячейка представляет собой пару признаков, которые сравниваются между собой.

На практике, чтобы определить разделимость данных, часто используются статистические методы.

1. Метод главных компонент (PCA Principal Component Analysis): позволяет уменьшить размерность данных и визуализировать их на плоскости или в пространстве. Если классы данных хорошо разделимы, то они будут наблюдаться как отдельные скопления точек.
2. Метод опорных векторов (SVM Support Vector Machine): позволяет построить гиперплоскости, разделяющие классы данных. Если данные легко разделимы, то гиперплоскости будут иметь хороший запас отступа от точек каждого класса.
3. Алгоритм кластеризации (например, k-средних): позволяет разделить данные на группы на основе их сходства. Если классы данных хорошо разделимы, то кластеры будут отдельными и компактными.
4. Алгоритмы машинного обучения (например, линейная регрессия, дерево решений, случайный лес - о них мы поговорим позже на уроке 7) можно использовать эти алгоритмы для предсказания меток классов на основе доступных данных. Если точность предсказаний высока, то данные можно считать разделимыми.
5. Визуализация данных: визуальный анализ данных может помочь определить, насколько классы данных разделимы. Если точки одного класса сосредоточены в отдельных областях и между классами есть значительные различия, то данные можно считать разделимыми.

Однако следует отметить, что определение разделимости данных является относительным и может зависеть от выбранного метода и алгоритма. Некоторые данные могут быть легко разделимыми для одного алгоритма, но трудно разделимыми для другого. Предположим, у нас есть набор данных, содержащий информацию о покупках клиентов в интернет-магазине. Каждая запись содержит такие признаки, как сумма покупки, время покупки, местоположение клиента и категория товара.

Допустим, мы хотим разделить эти данные на два класса: покупки клиентов, которые были совершены в рабочие дни и покупки, которые были совершены в выходные дни. Мы хотим использовать два разных алгоритма машинного обучения, логистическую регрессию (LR) и случайный лес (RF), чтобы сделать это разделение.

В данном случае, данные можно легко разделить с помощью логистической регрессии. Логистическая регрессия может использовать временные признаки и категориальные признаки (например, день недели) для предсказания класса покупки. Например, мы можем наблюдать, что покупки совершаются чаще в выходные дни, и на основе этого составить правило, которое классифицирует покупки в выходные дни.

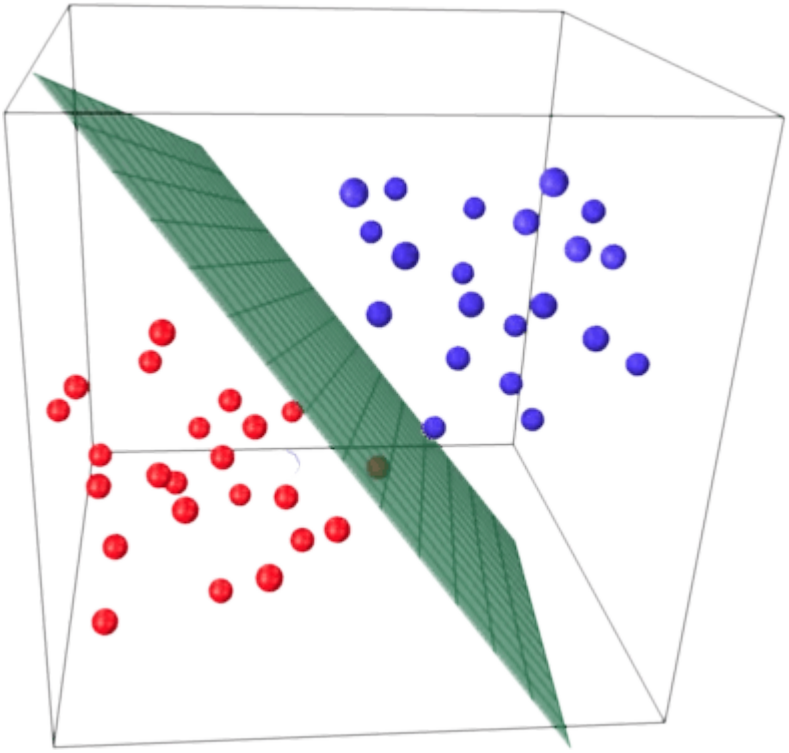
Однако, если мы используем случайный лес вместо логистической регрессии, может быть трудно разделить данные. Случайный лес состоит из нескольких решающих деревьев, и каждое дерево использует только поднабор признаков для принятия решений. В данном случае, временные признаки и категориальные признаки могут быть распределены по разным деревьям, что делает сложным принятие решения о классификации покупок в выходные и рабочие дни.

Таким образом, данные, которые могут быть легко разделимыми для логистической регрессии, могут оказаться трудно разделимыми для случайного леса. Это пример ситуации, когда данные могут быть легко разделимыми для одного алгоритма, но трудно разделимыми для другого.

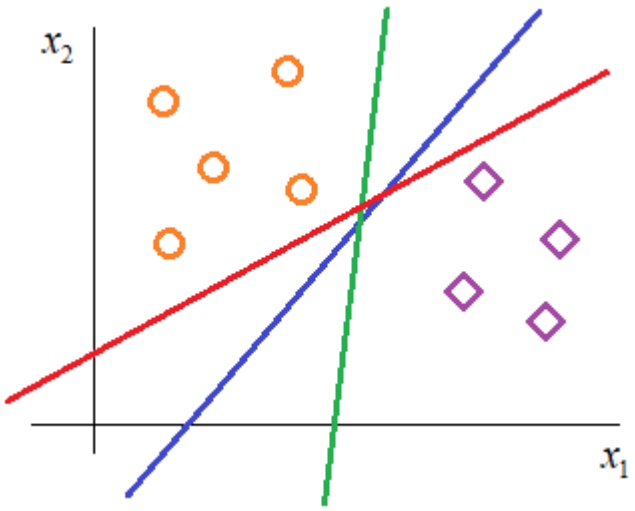
В заключение, разделимость данных является важным понятием в машинном обучении, она позволяет определить возможность классификации объектов в заданных наборах данных. Для определения разделимости данных нужно использовать различные методы, такие как визуальный анализ данных, статистические методы, методы машинного обучения и т.д. Важно также понимать, что нет одного универсального способа определения разделимости, и выбор метода будет зависеть от структуры и количества данных и требуемой точности решения.

Рассмотрим, пример реализации линейной классификации.

Основная идея линейного классификатора заключается в том, что пространство признаков делится на два полупространства гиперплоскостью, и в каждом полупространстве предсказывается одно из двух значений целевого класса. Если это можно сделать без ошибок, то обучающие выборки называются линейно разделяемыми.

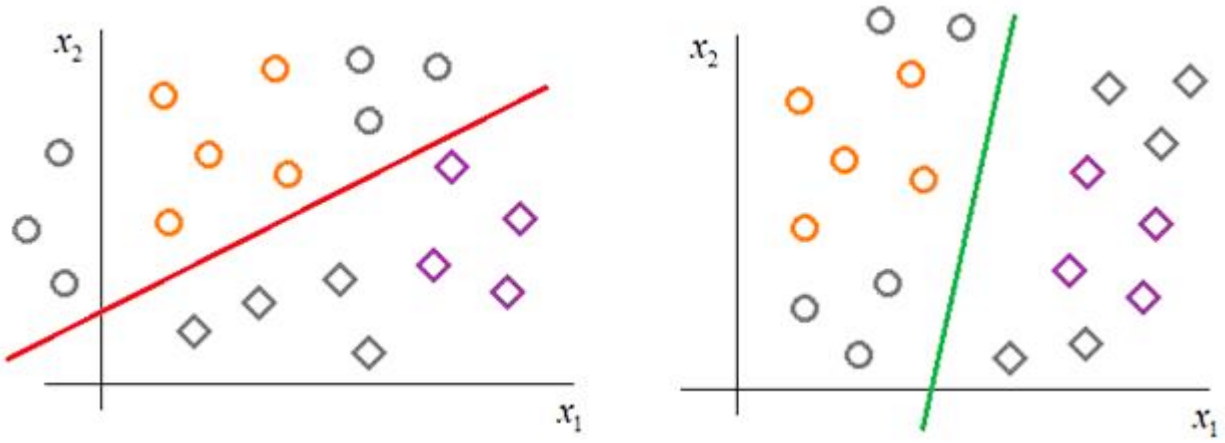


Давайте снова рассмотрим проблему бинарной классификации объектов на основе положения гиперплоскости разделения. Для простоты предположим двумерное пространство признаков. Каждый объект, принадлежащее классу, может быть представлено как точка на плоскости. Объекты обучающей выборки распределены следующим образом:



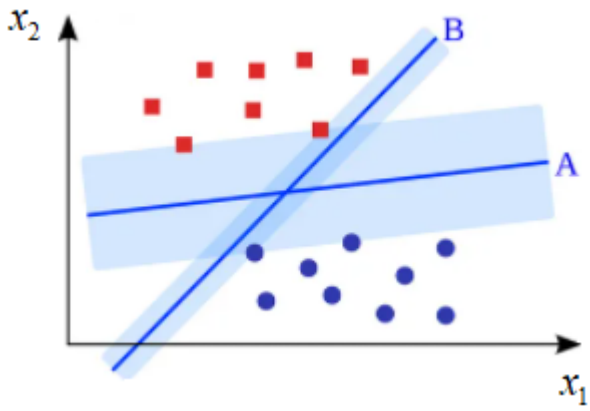
Как видите, в этом примере можно создать множество различных разделительных линий (обычно гиперплоскостей), каждая из которых правильно отделяет один класс от другого. Возникает вопрос, какое разделение лучше. В машинном обучении предполагается, что модель, обученная на выборке, должна хорошо работать на любом другом наборе из того же распределения. Другими

словами, модель должна обладать хорошей обобщаемостью (не быть переобученной). Когда я смотрю на разделительную линию, проведенную с этой точки зрения, чисто интуитивно я бы выбрала синюю. Почему так? Видите ли, красная и зеленая линии на самом деле делают дополнительные предположения о распределении объектов в обоих классах:



Красная линия "предполагает", что объекты распределено несколько горизонтально, а зеленая линия "предполагает", что объекты распределены более вертикально. Конечно, синяя линия также "предполагает", но только сохраняет более наглядное представление исходного распределения. Метод опорных векторов (SVM) основаны на этой идее разделяющей гиперплоскости и, по возможности, ориентируются только на распределение обучающих образцов и не делают никаких дополнительных предположений о распределении объектов внутри класса.

Давайте теперь формализуем эту чисто интуитивную идею на уровне математики. Затем ответим на первый вопрос. Что такое разделительная гиперплоскость, которая "делает" меньше всего предположений о распределении классов и, таким образом, приводит к лучшим обобщающим способностям алгоритмов классификации С точки зрения SVM, оптимальная разделительная гиперплоскость - это гиперплоскость, которая образует самую широкую полосу между двумя классами объектов. Сама разделительная гиперплоскость располагается в центре этой полосы:



В чем причина этого? Потому что чем шире полоса пропускания, тем надежнее классификатор отличает один класс объектов от другого.

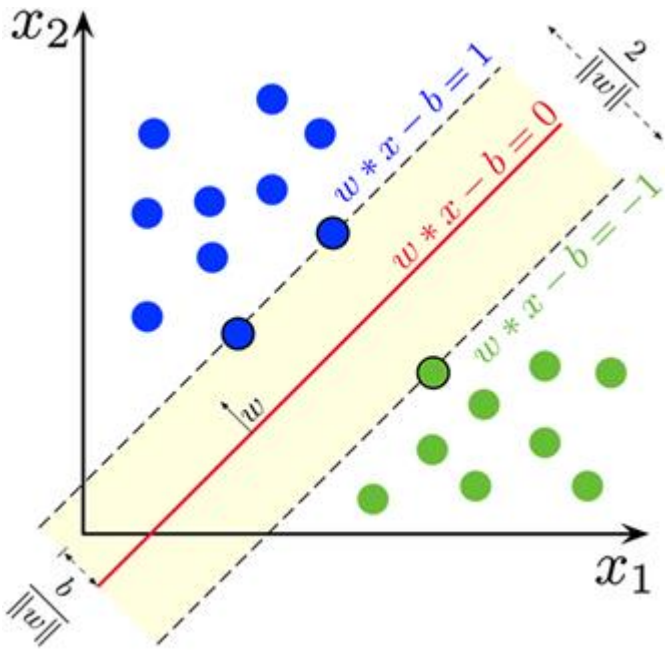
Чтобы объяснить эту идею на математическом уровне, мы должны сначала определить модель классификатора, которая фактически описывает уравнения гиперплоскости пространства признаков. Здесь выбрана простейшая линейная модель:

$$a(x) = sign(w^T x - b) = sign(\langle w, x \rangle - b) = sign(w * x - b)$$

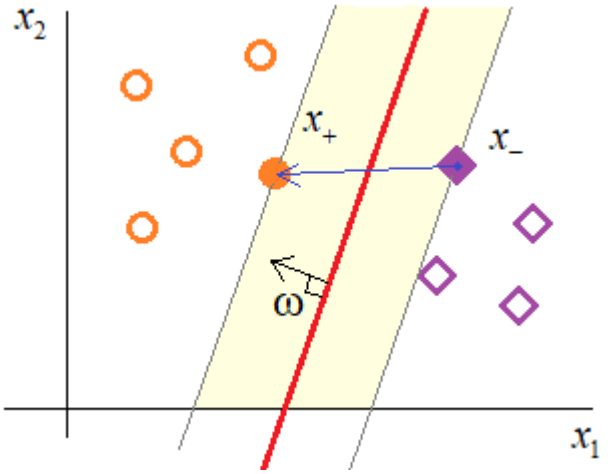
Есть три различных способа записать это, все они выражают линейную комбинацию вектора параметров w и объекта x, плюс смещение -b. Модель выдает значение:

$$a(x) \in \{-1; +1\}$$

Предполагается, что обучающая выборка состоит из линейно разделяемых объектов (позже это обобщается до нелинейно разделяемых). Полоса пропускания определяется положением граничного вектора x в пространстве признаков:



Учитывая любые два изображения разных классов, которые находятся ближе всего к границе разделения (т.е. границе полосы), ширина полосы может быть рассчитана как проекция вектора x+-x- на вектор w.



То есть, при минимизации квадратичной нормы весов необходимо найти w и b таким образом, чтобы все отступы гарантированно были больше единицы, за исключением объектов, расположенных непосредственно на краях ленты (где отступы должны быть равны единице).

Эта задача называется квадратичным программированием и минимизирует квадраты весов при линейных ограничениях неравенства.

Эту задачу решает модель SVM (Support Vector Machine, машина опорных векторов) - это алгоритм машинного обучения, который подходит для задачи

классификации данных. Он использует опорные векторы, которые представляют точки данных в пространстве признаков, чтобы разделить разные классы данных.

В контексте задачи классификации, SVM создает гиперплоскость в многомерном пространстве, которая разделяет данные разных классов на две стороны. Гиперплоскость строится таким образом, чтобы максимизировать расстояние между ней и ближайшими опорными векторами (то есть ближайшими точками от каждого класса).

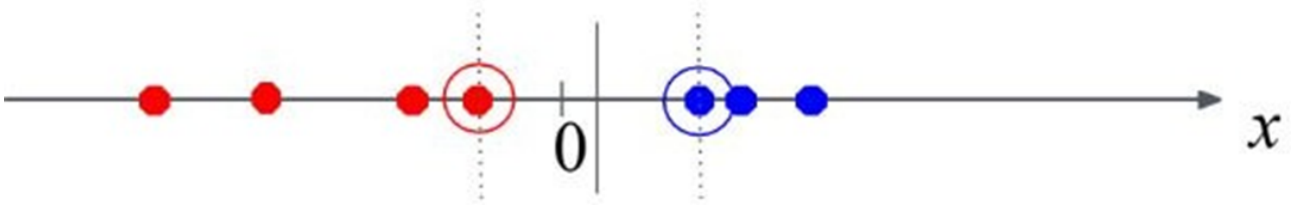
Этот алгоритм построения гиперплоскости также позволяет классифицировать новые точки данных. Если новая точка находится по одну сторону от гиперплоскости, она будет отнесена к одному классу, а если по другую - к другому классу. Это делается на основе положения точки относительно гиперплоскости и расстояния от нее до нее.

Цель использования модели SVM в задаче классификации состоит в построении оптимальной гиперплоскости, которая максимально точно разделяет данные разных классов. SVM позволяет учесть как опорные вектора, которые наиболее важны для разделения классов, так и внутренние точки каждого класса при построении гиперплоскости.

В итоге, модель SVM может быть использована для классификации новых точек данных на основе изученных свойств гиперплоскости и положения этих точек относительно нее.

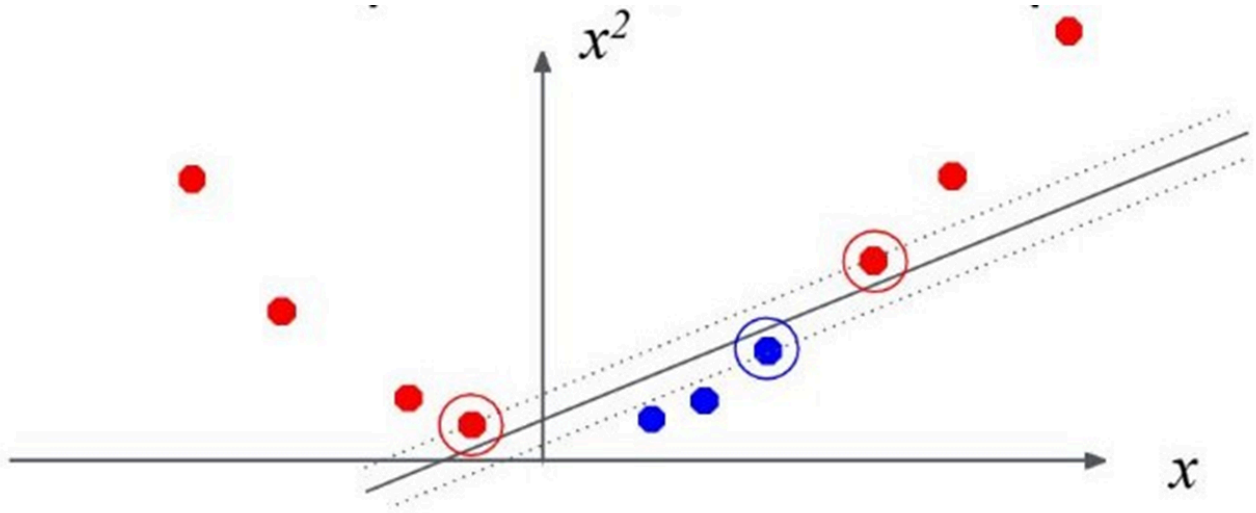
Ядро распределения или еще раз о разделимости данных

Линейно разделимые датасеты хорошо классифицируются



Но что делать, если они не линейно разделимы?

Можно попробовать отобразить данные в пространство более высокой размерности



Ядро распределения данных в машинном обучении — это функция, которая измеряет сходство между двумя данными или объектами в пространстве признаков, которая используется в методе ядерного трюка (kernel trick) для анализа данных. Ядерная функция принимает на вход два вектора и возвращает их скалярное произведение в пространстве более высокой размерности. После этого значение ядерной функции может быть использовано для классификации или регрессии данных.

Ядро используется в алгоритмах машинного обучения, таких как метод опорных векторов (Support Vector Machine), чтобы преобразовать данные из исходного пространства в новое пространство более высокой размерности, где их классы линейно разделимы.

Преобразование данных с помощью ядерной функции позволяет выполнить нелинейную классификацию или регрессию в пространстве более высокой размерности, тогда как в исходном пространстве эти задачи могли бы оказаться сложными или невозможными. Основная идея состоит в том, что ядро определяет сходство между объектами, основываясь на их признаках, и позволяет алгоритму машинного обучения учитывать нелинейные зависимости в данных.

The «Kernel Trick»

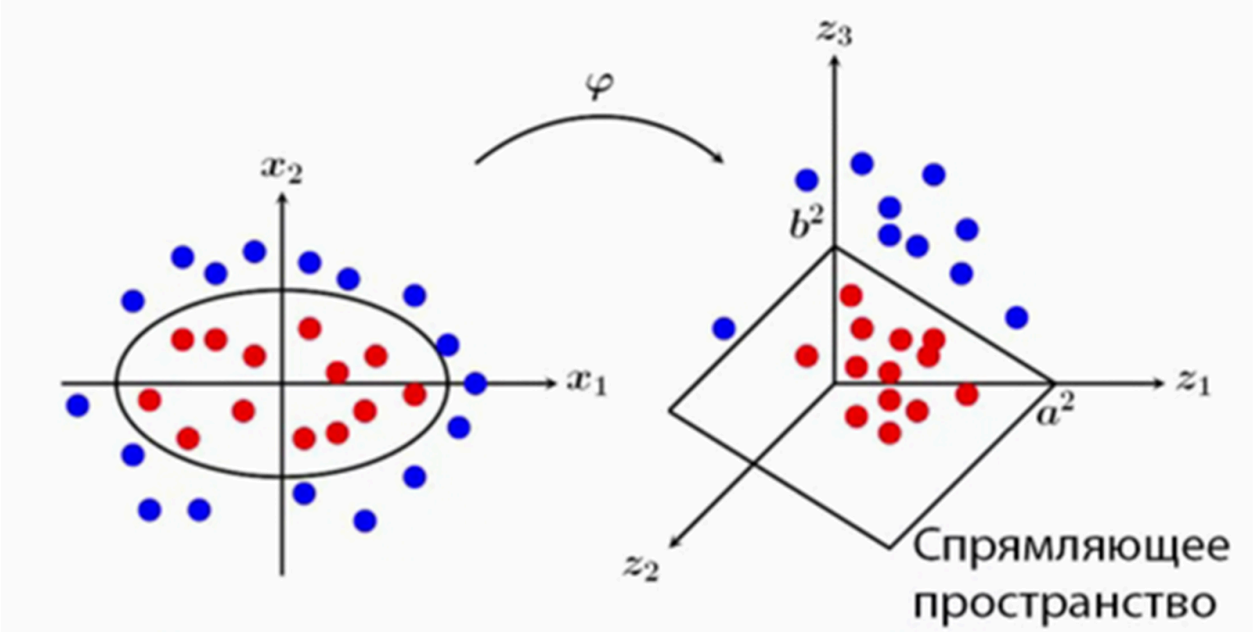
SVM зависит от скалярного произведения

$$K(x_i, x_j) = x_i^T x_j$$

Если каждая точка отображается в пр-во более высокой размерности при помощи Ф: x → φ(x), тогда скалярное произведение становится:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

Функция ядра — это функция, соответствующая скалярному произведению в производстве более высокой размерности

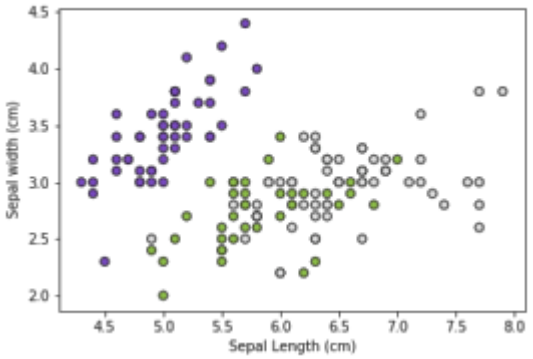


Примерами ядерных функций являются полиномиальное ядро, гауссово ядро (RBF) и сигмоидное ядро. В зависимости от выбранного ядра и его параметров, алгоритмы машинного обучения могут более точно моделировать сложные структуры данных и достигать лучшей производительности.

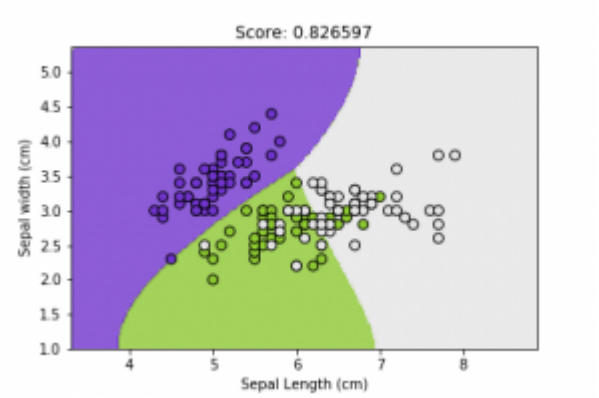
Преимуществом использования ядерных функций является то, что они избегают необходимости явного преобразования данных в более высокую размерность, что может быть вычислительно затратно и требовать большого объема памяти. Вместо этого, ядерные функции позволяют алгоритмам работать с данными в исходном пространстве, используя преимущества работы в пространствах более высокой размерности.

Выбор ядра для модели SVM зависит от типа данных и задачи, с которыми вы работаете. Вот некоторые из самых популярных и часто используемых ядер для SVM:

1. Линейное ядро: Линейное ядро является наиболее простым и прямолинейным. Оно хорошо подходит для задач классификации, когда данные линейно разделимы.



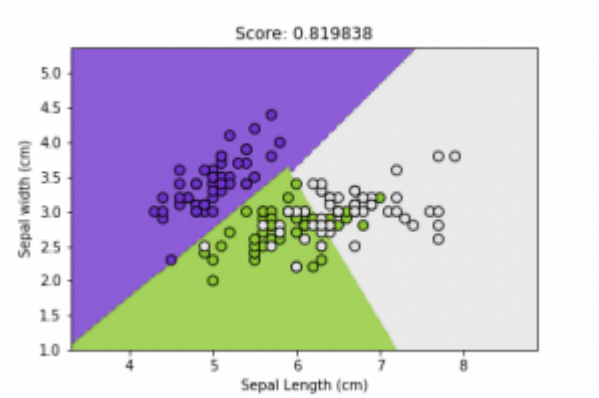
2. Полиномиальное ядро: Полиномиальное ядро позволяет построить нелинейную границу классов. Оно подходит для задач, где данные имеют нелинейные зависимости. В случае полиномиального ядра, данные проецируются в более высокую размерность с использованием полиномиальной функции. Например, если у нас есть двумерные данные (x, y), полиномиальное ядро может преобразовать их в трехмерные данные, добавляя квадраты исходных переменных: (x, y, x^2, y^2, xy).



3. Радиальная базисная функция (RBF) ядро: RBF ядро является одним из наиболее широко используемых ядер для SVM. Оно создает нелинейные разделяющие поверхности и позволяет моделировать сложные зависимости в данных, используя подход гауссовской радиальной базисной функции. RBF ядро определяется как экспонентой отрицательного квадрата Евклидового расстояния между двумя точками в исходном пространстве данных. Формула для RBF ядра выглядит следующим образом:

$$K(x, y) = \exp(-\gamma \| x - y \|^2)$$

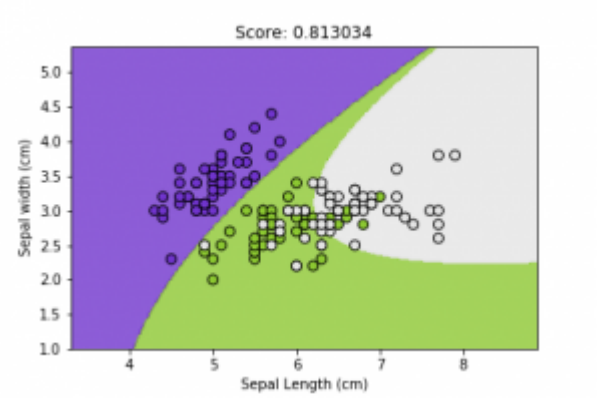
Здесь x и y представляют собой две точки данных, γ (гамма) - параметр, определяющий ширину радиуса ядра. Большие значения γ приводят к узкому ядру и модели с меньшей комплексностью, в то время как маленькие значения γ создают более широкое ядро и более сложную модель.



4. Сигмоидное ядро: Сигмоидное ядро преобразует данные в вероятности и может быть полезно для бинарной классификации. Функция значения сигмоидного ядра определяется следующим образом:

K(x, y) = tanh(αx • y + c),

где α и c - настраиваемые параметры, x и y - векторы признаков.

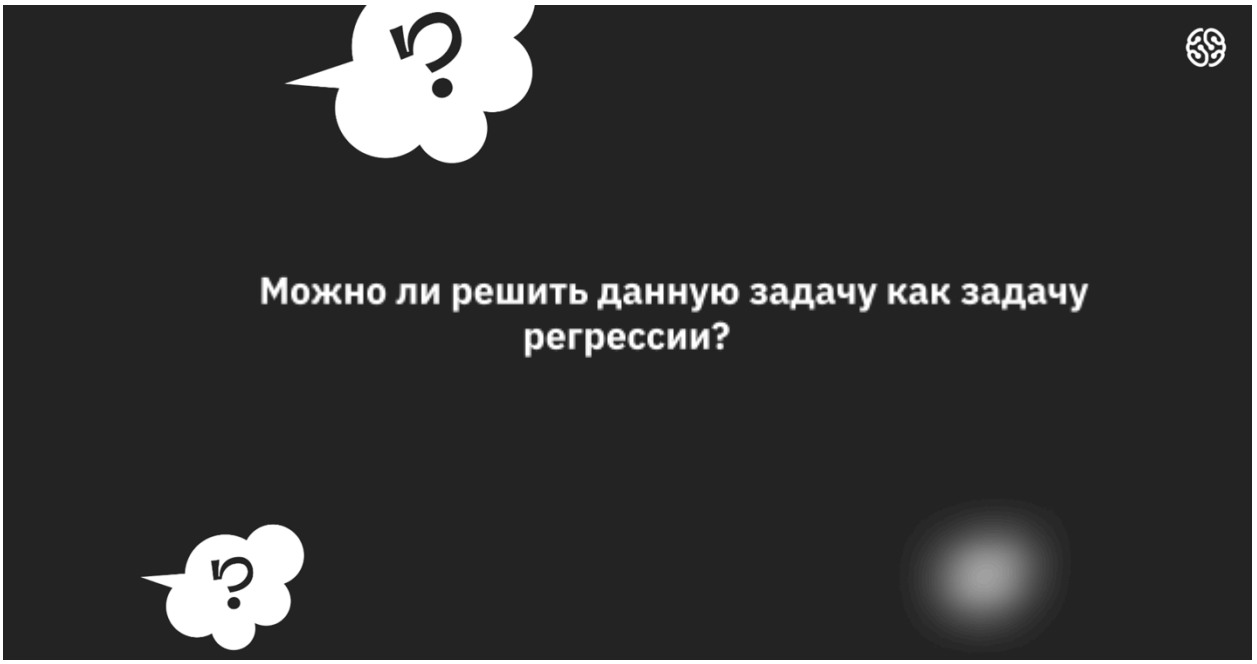


Кроме того, существуют и другие ядра, такие как радиально-базовая функция сигмы, полиномиальные линейные комбинации и аппроксимации низкого порядка.

Выбор ядра для модели SVM может быть основан на эмпирическом опыте, анализе данных или применении метода перекрестной проверки для оценки производительности модели с различными ядрами.

Пока нам неизвестно, как обучается модель линейной классификации, но уже ясно, что окончательные прогнозы можно рассчитать с помощью следующей формулы:

$$y = \mathbf{sign}\langle w, x_i \rangle$$



Задание: Можно ли решить данную задачу, как задачу регрессии?

Если мы попробуем предсказать числа -1 и 1, минимизируя MSE, учитывая знак, но результаты получатся плохими. Во-первых, регрессия дает очень маленькую ошибку для объектов, которые находятся близко к *плоскости разделения*, но не с той стороны. Во-вторых, предсказание, например, 5 вместо 1 - это ошибка. Однако, если знак правильный, то модуль числа не имеет значения.

То есть, нам нужна прямая, которая разделяет эти точки, а не проходит через них!

💡 SVM чувствителен к выбросам, которые могут исказить обучение модели. Проведите анализ выбросов и принимайте меры для их обработки, например, удаление или замена выбросов, или использование более устойчивых к выбросам ядер (например, RBF)

🔥 Если у вас есть большой объем данных, обучение модели SVM может требовать значительного времени. Используйте возможности параллельной обработки, такие как распределенное обучение или использование графических процессоров (GPU), чтобы ускорить процесс обучения.

Логистическая регрессия

Другой интересный подход вытекает из желания рассматривать классификацию как задачу оценки вероятности. Хорошим примером является прогнозирование кликов в Интернете (например, в рекламе и поиске). Наличие клика в обучении не означает, что пользователь снова кликнет на объект, если условия эксперимента будут идеально повторены. Скорее, объект обладает определенной "кликабельностью", т.е. реальной вероятностью того, что объект будет кликнут. Мы считаем, что каждый клик в обучающей выборке является реализацией этой вероятностной переменной, и что в пределе в каждой точке соотношение положительных и отрицательных примеров должно сходиться к этой вероятности.

Проблема в том, что вероятность по определению является значением между 0 и 1, и нет простого способа обучить линейную модель, чтобы она удовлетворяла этому ограничению. Способ преодоления этой проблемы заключается в обучении линейной модели правильному прогнозированию объектов, связанных с вероятностями в диапазоне (-∞, ∞).

Затем ответ модели преобразуется в вероятность. Одним из таких объектов является logit или log odds, который представляет собой логарифм отношения вероятностей положительных и отрицательных событий $log(p/1 - p)$

Если ответом нашей модели является logit, то искомую вероятность посчитать нетрудно:

$$\langle w, x_i \rangle = \log\left(\frac{p}{1-p}\right)$$

$$e^{\langle w, x_i \rangle} = \frac{p}{1-p}$$

$$p = \frac{1}{1 + e^{-\langle w, x_i \rangle}}$$

Функция в правой части называется **сигмоидой** и обозначается

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Таким образом,

$$p = \sigma(\langle w, x_i \rangle)$$

Как теперь научиться оптимизировать *w* так, чтобы модель как можно лучше предсказывала логиты? Нужно применить метод максимума правдоподобия. Правдоподобие позволяет понять, насколько, вероятно, получить данные значения таргета *y* при данных *X* и весах *w*. Оно имеет вид

$$p(y \mid X, w) = \prod_i p(y_i \mid x_i, w)$$

Оптимизировать произведение неудобно, мы хотим иметь дело с суммой, поэтому переходим к логарифмическому правдоподобию и подставляем формулу вероятности

$$\ell(w, X, y) = \sum_i \left(y_i \log(p_i) + (1 - y_i) \log(1 - p_i)\right) = \sum_i \left(y_i \log(\sigma(\langle w, x_i \rangle)) + (1 - y_i) \log(1 - \sigma(\langle w, x_i \rangle))\right)$$

Учитывая, что

$$\sigma(-z) = \frac{1}{1 + e^{z}} = \frac{e^{-z}}{e^{-z} + 1} = 1 - \sigma(z),$$

Нас интересует w, которое максимизирует вероятность. Умножим на минус один, чтобы получить минимизирующую функцию потерь:

$$L(w, X, y) = -\sum_i \left(y_i \log(\sigma(\langle w, x_i \rangle)) + (1 - y_i) \log(\sigma(-\langle w, x_i \rangle))\right)$$

Предсказание модели будет вычисляться, как мы договаривались, следующим образом:

$$p = \sigma(\langle w, x_i \rangle)$$

Это вероятность того, что класс положительный, и как нам перейти от этого к оценке самого класса? Все предсказания положительны и находятся в диапазоне от 0 до 1. Интуитивно, хотя и не совсем точно, можно ответить так: "Возьмите порог 0,5". Более точным является индивидуальный выбор этого порога для уже построенных регрессий, которые минимизируют требуемые метрики на отложенной тестовой выборке. Например, доли положительных и отрицательных классов должны примерно соответствовать истинным долям.

Кроме того, обратите внимание, что данный метод называется логистической регрессией, а не логистической классификацией.

Многоклассовая классификация

Пусть каждый объект в примере принадлежит к одному из K классов.
Чтобы предсказать эти классы с помощью линейной модели, задачу многоклассовой классификации необходимо свести к набору бинарных задач. Ниже описаны два наиболее распространенных способа решения этой задачи — один против всех *one-vs-all* и все против всех *all-vs-all* .

Один против всех (one-versus-all)

Обучим K линейных классификаторов $b_1(x), \ldots, b_K(x)$, выдающих оценки принадлежности классам $1, \ldots, K$ соответственно. В случае с линейными моделями эти классификаторы будут иметь вид

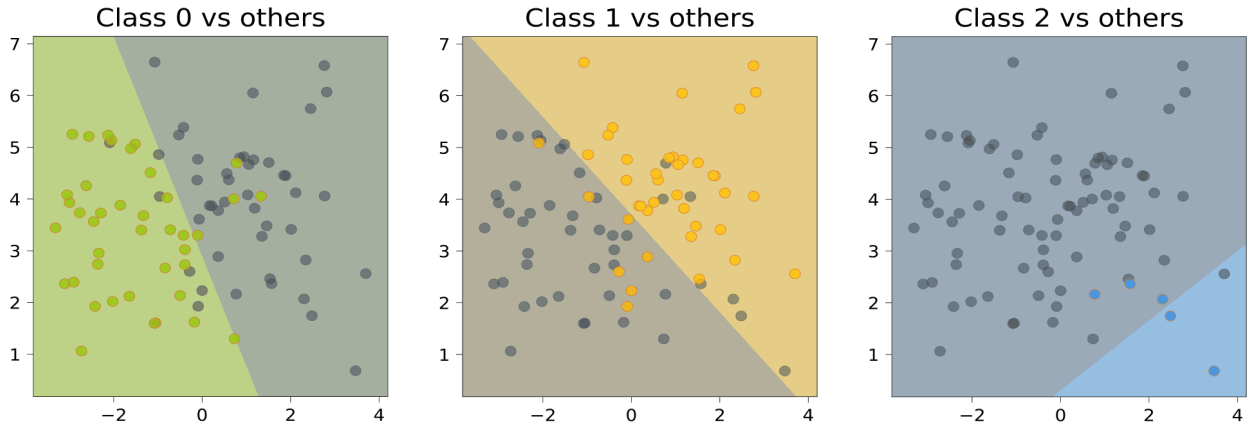
$$b_k(x) = \operatorname{sgn}\left(\langle w_k, x\rangle + w_{0k}\right)$$

Мы будем учить классификатор отличать k-й класс от всех остальных.

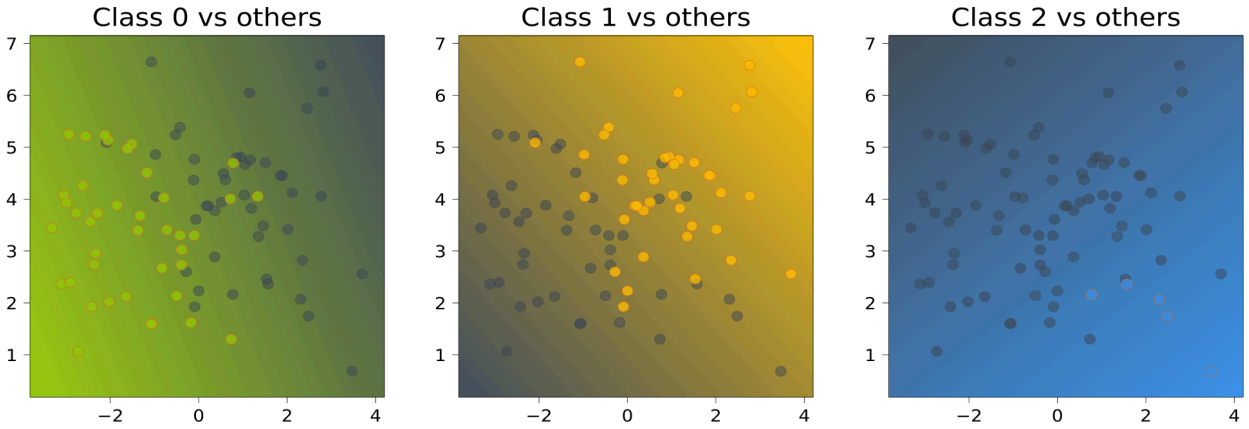
Логично, что окончательный классификатор выдает класс, соответствующий самому уверенному из бинарных алгоритмов. В некотором смысле, уверенность может быть измерена значениями линейной функции.

$$a(x) = \operatorname{argmax}_k\left(\langle w_k, x\rangle + w_{0k}\right)$$

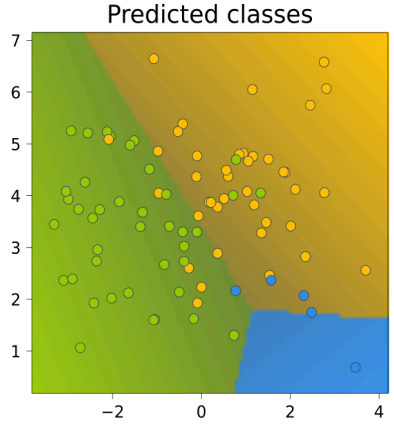
Обучим три линейных модели, отличающих один класс от остальных:



Теперь сравним значения линейных функций



И для каждой точки выберем тот класс, которому соответствует большее значение, то есть самый «уверенный» классификатор



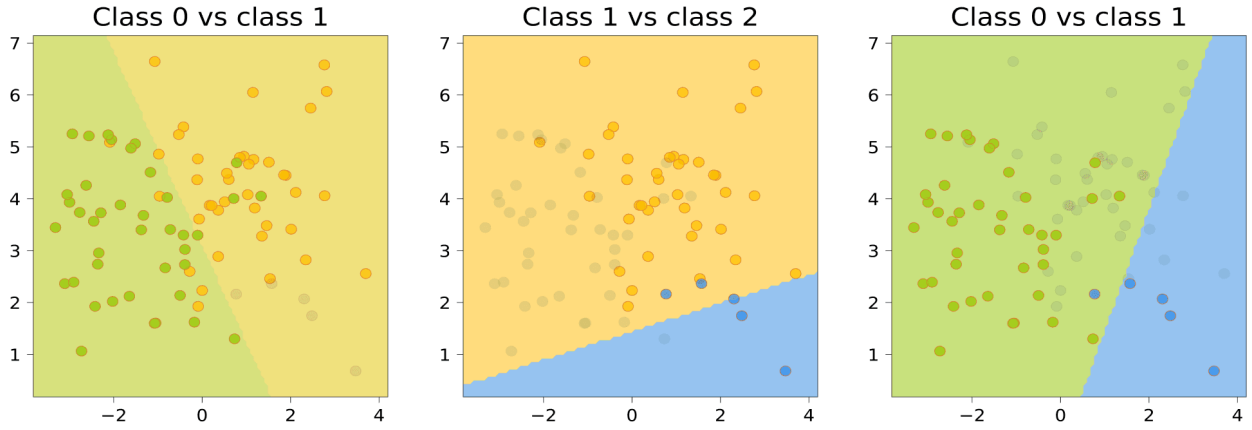
Проблема этого подхода заключается в том, что каждый классификатор обучается на своей выборке, и значения линейных функций или, проще говоря, "выходы" классификаторов могут иметь разные масштабы.

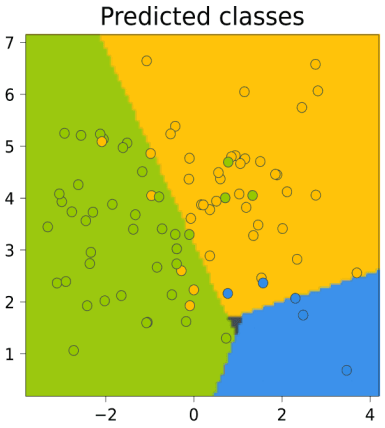
Поэтому, сравнивать их некорректно. В случае SVM нормализация весового вектора уже не является решением проблемы, так как нормализация изменит норму весов.

Все против всех (all-versus-all)

Подход "Все против всех" (all-versus-all) в многоклассовой классификации предполагает создание отдельных классификаторов для каждой пары классов и последующее голосование для определения окончательного класса.

Метод "Все против всех" начинается с того, что каждая пара классов сравнивается между собой. Для каждой пары классов обучается отдельный бинарный классификатор, который должен определить, к какому классу принадлежит та или иная точка данных. После обучения всех двухклассовых классификаторов производится голосование для каждой точки данных. Класс, набравший наибольшее количество голосов, становится окончательным классом для данной точки данных.





Преимуществом подхода "Все против всех" является то, что он может быть применен для классификации в случае, когда количество классов велико и

скомпоновать все классификаторы в матрицу попарных классов затруднительно или невозможно. Также этот подход имеет меньшую вычислительную сложность, поскольку классификаторы обучаются для каждой пары классов, вместо обучения единственного многоклассового классификатора.

Однако подход "Все против всех" может быть более подвержен ошибкам и шуму, так как каждый двухклассовый классификатор обучается только по части данных, относящихся к двум конкретным классам. Кроме того, алгоритм может столкнуться с проблемой несбалансированности классов, когда некоторые пары классов имеют гораздо большее количество обучающих данных, чем другие пары.

Метрики качества

Перейдём к обзору метрик.

Метрики качества моделей в машинном обучении используются для оценки и сравнения производительности различных моделей и алгоритмов. Они помогают нам понять, насколько хорошо модель решает поставленную задачу и насколько точны ее прогнозы.

1. Оценка точности: Метрики позволяют нам измерить, насколько точно модель предсказывает значения целевой переменной. Например, метрика среднеквадратической ошибки (MSE) показывает, насколько среднеквадратичное отклонение прогнозов модели от фактических значений. Более низкое значение MSE указывает на более точные прогнозы.

2. Выбор модели: Метрики помогают нам выбрать наилучшую модель из нескольких альтернатив. Сравнение метрик для различных моделей позволяет определить, какая из них дает наилучшие результаты. Например, метрика точности (accuracy) показывает, какой процент примеров модель классифицирует правильно.

3. Оптимизация модели: Метрики качества помогают нам оптимизировать модель путем настройки ее гиперпараметров (настраиваемые параметры, которые определяют конфигурацию алгоритма обучения модели. Они отличаются от обычных модельных параметров, которые определяются во время обучения). Например, метрика F1-меры комбинирует точность и полноту в одну метрику и может быть использована для настройки порога классификации модели.

4. Мониторинг и оценка производительности: Метрики позволяют нам мониторить производительность модели на продакшене и оценивать ее работу в реальном времени. Например, метрики точности и полноты могут быть использованы для оценки работы модели на новых данных и выявления проблем, связанных с переобучением или недообучением модели.

В целом, использование метрик качества моделей в машинном обучении позволяет нам объективно оценить и сравнить производительность моделей, выбрать наилучшую модель и оптимизировать ее работу.

Первым критерием качества, который приходит в голову, является **accuracy** – доля объектов, для которых мы правильно предсказали класс:

$$\text{Accuracy}(y, y^{pred}) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[y_i = f(x_i)]$$

Более пристальный взгляд на этот показатель выявляет некоторые недостатки:

Он не учитывает дисбаланс классов. Например, в задаче диагностики редких заболеваний классификатор, предсказывающий, что не все пациенты больны, будет иметь довольно высокую точность, потому что в выборке гораздо меньше больных людей;

Кроме того, не учитывается стоимость ошибок для различных классов объектов. Ложноположительный диагноз для здорового пациента потребует лишь проведения еще одного теста, в то время как ложноотрицательный диагноз может иметь фатальные последствия.

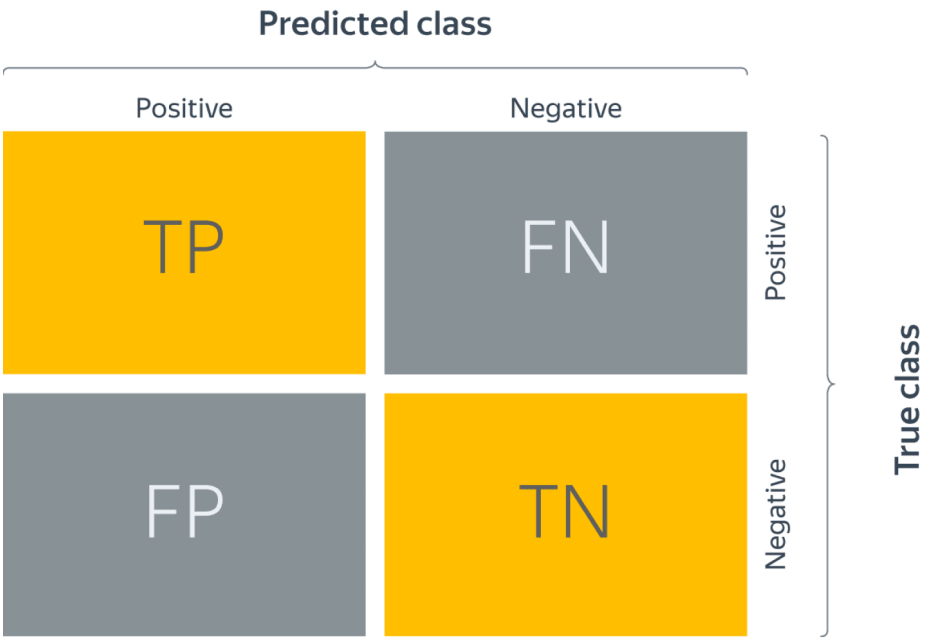
Confusion matrix (матрица ошибок)

Исторически проблема бинарной классификации — это проблема обнаружения необычных объектов в большом потоке объектов, например, обнаружение больного туберкулезом с помощью рентгеноскопии. Или проблема распознавания точки на экране радарного приемника как бомбардировщика, представляющего угрозу охраняемой цели (в отличие от стаи гусей).

Поэтому мы называем интересующие нас классы "положительными", а остальные классы - "отрицательными".

Обратите внимание, что для каждого объекта в выборке возможны четыре случая:

- Мы угадали положительную метку и удалили ее. Мы помещаем такие объекты в группу истинно положительных (TP) (истинных, потому что мы угадали правильно, положительных, потому что мы угадали положительную метку);
- Мы угадали положительную метку, но угадали неправильно - ложноположительные (ЛП) (ложные, потому что угадали неправильно); и
- Предсказал отрицательную метку и оценил ее как истинно отрицательную (TN);
- И, наконец, предсказал отрицательную метку, но угадал неправильно - ложноотрицательный результат (FN). Для удобства все эти четыре числа показаны в таблице, называемой матрицей ошибок:



Исходя из матрицы ошибок accuracy можно рассчитать:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Точность и полнота

Accuracy - это мера, которая описывает качество комбинированной модели по всем классам. Это полезно, если классы имеют одинаковую ценность для нас. В противном случае точность может вводить в заблуждение.

Рассмотрим случай, когда положительный класс является редким событием. Возьмем в качестве примера поисковую систему: В хранилище хранятся миллиарды документов, но только на несколько порядков меньше релевантных определенному поисковому запросу.

Предположим, мы хотим решить задачу бинарной классификации вида "документ d релевантен запросу q". Из-за большого дисбаланса точность фиктивного классификатора, который делает все документы нерелевантными, близка к единице. Высокая точность TP более важна для пользователя.

Поэтому при наличии дисбаланса классов можно использовать метрики, ориентированные на TP, игнорируя TN.

Учет доли правильно предсказанных положительных объектов среди всех объектов, предсказанных положительным классом, дает метрику, называемую **точностью (precision).**

$$\text{Precision} = \frac{TP}{TP + FP}$$

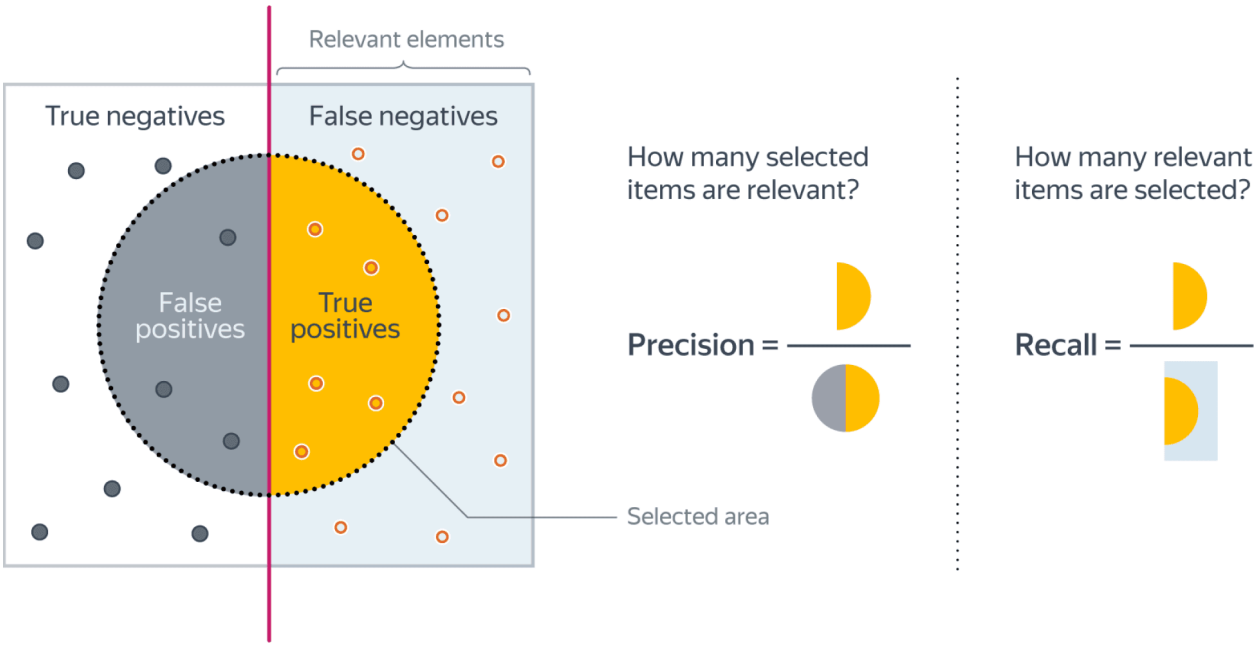
Интуитивно понятно, что этот показатель показывает процент релевантных документов среди всех документов, найденных классификатором. Чем меньше ложных срабатываний допускает модель, тем выше ее Precision.

Учитывая долю правильно найденных положительных элементов среди всех элементов положительного класса, получается метрика, называемая **полнотой (recall)**.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Интуитивно метрика показывает долю найденных документов из всех релевантных. Чем меньше ложно отрицательных срабатываний, тем выше recall модели.

Например, в задаче прогнозирования злокачественности опухолей точность показывает, сколько из опухолей, идентифицированных как злокачественные, на самом деле являются злокачественными, а полнота показывает, сколько из злокачественных опухолей идентифицировано.



F1-мера

Как упоминалось ранее, модели очень полезны для сравнения, когда их качество представлено одним числом; в случае пар Precision и Recall существует обычный способ объединить их в одну меру — взять их среднее гармоническое. Эта мера производительности исторически называется F1-мерой.

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} =$$

$$= 2\frac{Recall \cdot Precision}{Recall + Precision} = \frac{TP}{TP + \frac{FP+FN}{2}}$$

Стоит отметить, что критерий F1 предполагает, что значения Precision и Recall имеют одинаковое значение.

Если одна из этих метрик для вас приоритетнее, то можно воспользоваться F_β мерой:

$$F_\beta = (\beta^2 + 1)\frac{Recall \cdot Precision}{Recall + \beta^2 Precision}$$

F-мера объединяет точность и полноту в одну метрику, позволяя оценить их совместно. Коэффициент бетта в F-мере, обозначаемый как β, контролирует баланс между точностью и полнотой. Значение β определяет вес точности в соотношении к полноте.

В частности, когда β = 1, F-мера является сбалансированной метрикой, которая учитывает и точность, и полноту одинаково. Коэффициенты β < 1 делают F-меру более взвешенной в пользу точности, в то время как β > 1 делает ее более взвешенной в пользу полноты.

Итоги лекции:

1. Классификация — это задача отнесения объектов к заранее определенным классам на основе имеющихся данных. Она является одной из самых распространенных и важных задач в области машинного обучения.

2. Модели SVM (Support Vector Machines) и логистической регрессии - две популярные модели для решения задач классификации. SVM строит гиперплоскость, разделяющую объекты разных классов, а логистическая регрессия использует логистическую функцию для вероятностного подхода к классификации.

3. Многоклассовая классификация — это задача классификации, в которой объекты могут быть отнесены к одному из нескольких классов. Для решения такой задачи можно использовать методы один против всех, один против одного или иерархические методы.

4. Метрики качества классификации — это меры, используемые для оценки эффективности модели классификации. Некоторые из них включают точность (accuracy), полноту (recall), точность (precision), F-меру (F1-score).

5. Информация о задачах классификации полезна для понимания процесса решения задачи и выбора наиболее подходящей модели. Знание различных моделей, их особенностей и алгоритмов поможет в применении их к разным типам данных и нахождении оптимального решения задачи классификации.

6. В результате лекции мы получили навыки по выбору модели классификации, обработке и анализу данных, а также понимание метрик качества классификации и их применения для оценки результатов моделей. Мы также научились проводить многоклассовую классификацию и использовать различные методы для решения этой задачи.

Понимание задачи классификации может дать следующие полезные навыки:

1. Умение выбирать подходящие методы и алгоритмы классификации: понимание задачи поможет определить, какие методы и алгоритмы классификации

лучше всего подходят для решения данной задачи. Например, для решения задачи бинарной классификации можно использовать логистическую регрессию или метод опорных векторов, а для многоклассовой классификации — методы, такие как случайные леса или градиентный бустинг.

2. Навыки предобработки данных: понимание задачи помогает определить, какие признаки могут быть полезны для классификации и как обрабатывать или очищать данные, чтобы улучшить процесс классификации.

3. Умение работы с несбалансированными данными: задачи классификации могут столкнуться с проблемой несбалансированных данных, где классы имеют различное количество образцов. Понимание задачи поможет разработать стратегии для борьбы с этими проблемами, такие как использование взвешивания классов или изменение порога принятия решения.

4. Навыки оценки модели: понимание задачи поможет в выборе подходящих метрик оценки качества модели классификации. Например, для задачи бинарной классификации могут использоваться метрики, такие как точность (accuracy), полнота (recall) и F1-мера, а для многоклассовой классификации — метрики, такие как макро- и микро-усреднение.

5. Навыки настройки гиперпараметров модели: понимание задачи позволит определить, какие гиперпараметры модели необходимо настроить для достижения наилучшей производительности. Например, в моделях градиентного бустинга можно настраивать гиперпараметры, такие, как количество деревьев, глубина деревьев и скорость обучения, чтобы улучшить результаты классификации.

6. Умение интерпретировать результаты: понимание задачи поможет интерпретировать результаты классификации и понять, какие признаки наиболее важны для разделения классов. Например, можно использовать методы, такие как важность признаков (feature importance) или отображение деревьев решений, чтобы понять, какие признаки наиболее сильно влияют на классификацию.

Что можно почитать еще?

1. SVM. Подробный разбор метода опорных векторов, реализация на python.
2. Задача классификации (Classification problem)
3. Линейные модели классификации и регрессии.

Используемая литература

1. Уравнение гиперплоскости в задачах бинарной классификации
2. Логистическая регрессия
3. Классический SVM