МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт искуственного интеллекта

Курсовая работа

по дисциплине

"Математическая логика и теория алгоритмов"

Тема курсовой работы

"Решение линейных задач классификации"

Выполнил:

студент группы ККСО-03-18

Самойленко А.С.

Научный руководитель:

Кузнецов Владимир Сергеевич

Москва — 2021

Оглавление

Введение

Обучение с учителем имеет два основных применения: классификация и регрессия. Классификация дает возможность предсказать по значениям признаков, к какому классу будет относиться новый объект.

Линейный классификатор - алгоритм классификации, основанный на построении линейной разделяющей поверхности. В случае двух классов разделяющей поверхностью является гиперплоскость, которая делит пространство признаков на два полупространства. В случае большего числа классов разделяющая поверхность кусочно-линейна.

Линейными называют классификаторы, в которых предсказанное значение вычисляется в виде $= \overrightarrow{w} \cdot \overrightarrow{d} + b$, где $\overrightarrow{d} = (d_1, d_2, \ldots, d_n)$ - нормализованный вектор из частот слов в документе, w – вектор линейных весов той же размерности, что и признаковое пространство, а b - некоторое скалярное значение. Естественной интерпретацией для c в дискретном случае будет разделяющая гиперплоскость между различными классами.

Регрессионные модели (такие, как метод наименьших квадратов) - более явные и традиционные. Тем не менее, они обычно используются в случаях, когда целевые переменные являются числовыми, а не номинальными.

Определения

Задача классификации — задача, в которой имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется **выборкой**. Классовая принадлежность остальных объектов неизвестна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Классифицировать объект - значит, указать номер (или наименование) класса, к которому относится данный объект.

Классификация объекта - номер или наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту.

О задаче

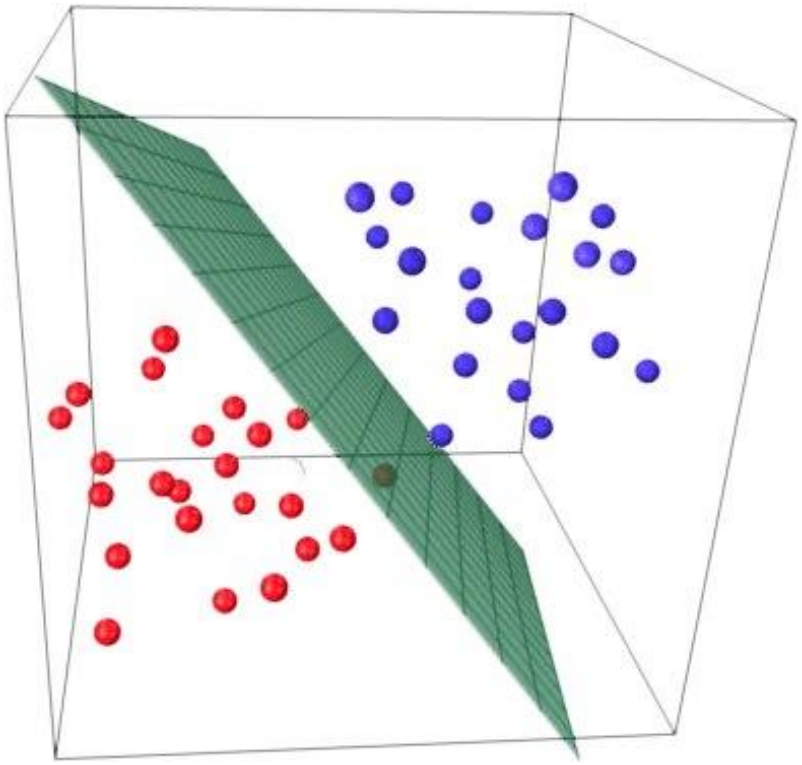
Будем решать задачу **бинарной классификации**. Задачей бинарной классификации является отнесение объекта к одному из двух классов. Точнее, выделить объекты одного класса из выборки, а все остальные объекты по умолчанию попадают в другой класс, который обозначается как -1 или 0. Например, ответ верный или нет.

Для решения поставленной задачи нужна обученная модель классификатора. Сначала алгоритм тренируется на объектах из обучающей выборки, для которых заранее известны метки классов. Далее уже обученный алгоритм предсказывает метку класса для каждого объекта из отложенной/тестовой выборки. Метки классов могут принимать значения $Y = \{-1, +1\}$. Объект — вектор с N признаками $x = (x_1, x_2, \ldots, x_n) \in R^n$.

При обучении алгоритм должен построить функцию $F(x) = y$, которая принимает в себя аргумент x — объект из пространства и выдает метку класса y.

Основная идея линейного классификатора заключается в том, что признаковое пространство может быть разделено гиперплоскостью на два полупространства, в каждом из которых прогнозируется одно из двух значений целевого класса.

Если это можно сделать без ошибок, то обучающая выборка называется **линейно разделимой**.



Рассмотрим задачу бинарной классификации, причем метки целевого класса обозначим $+1$ (положительные примеры) и -1 (отрицательные примеры).

Линейный классификатор

Рассмотрим математическую модель линейного классификатора. Суть ее практически не отличается от линейной модели регрессии, рассмотренной ранее, только в качестве ответа здесь идет уже не вещественное число, а 0 или 1. Поэтому полученный вещественный ответ нужно перевести в 0 или 1. Для этого используется либо знаковая функция, либо пороговая соответственно:

$$a(x) = sign(\sum_{j=1}^{d+1} w_j x_j) = sign\langle w, x\rangle$$

и

$$a(x) = \begin{cases} 1, \ \sum_{j=1}^{d+1} w_j x_j > t \\ 0, \ \sum_{j=1}^{d+1} w_j x_j \leq t \end{cases}$$

где t - порог.

Рассмотрим геометрический смысл классификатора. Если построение регрессионной модели сводилось к построению прямой, заменяющей набор точек выборки, то построение классификатора сводится к построению плоскости $\langle W, X\rangle = 0$, оптимально разделяющую объекты выборки на 2 класса:

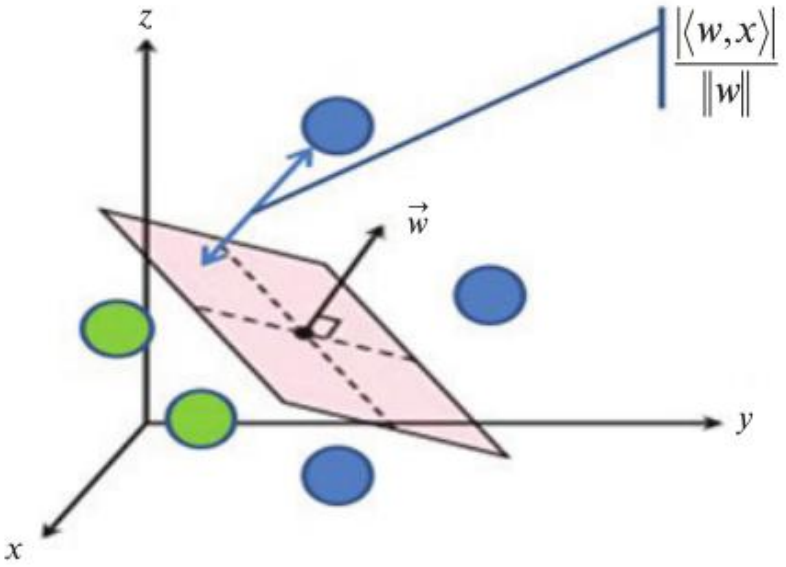


Рис. 1: Геометрический смысл бинарного классификатора

Причем скалярное произведение $\langle w, x\rangle$ с одной стороны плоскости будет положительным, а с другой — отрицательным.

Тогда расстояние r от объекта, имеющего признаковое опи- сание , до плоскости $\langle w, x\rangle = 0$ рассчитывается по формуле:

$$r = \frac{|\langle w, x\rangle|}{||w||}$$

.

Введем понятие отступа :

$$M_i = y_i\langle w, x_i\rangle$$

Отступ является мерой правильности ответа алгоритма классификатора. Он служит мерой уверенности в правильном выборе класса для объекта: чем он больше, тем надежнее объект отнесен к этому классу. Если объект попал в свой класс, то отступ будет положительным, если не в свой класс, то отступ будет

отрицательным. Таким образом, количество положительных отступов показывает количество правильных ответов, количество отрицательных — количество ошибок алгоритма.

На основе отступа можно сформировать функционал ошибки бинарной классификации Q:

$$Q(a, X) = \frac{1}{n} \cdot \sum_{i=1}^{n} [M_i < 0].$$

В такой трактовке отступ становится функцией потерь, поскольку показывает, сколько неправильных ответов дает алгоритм a на выборке X.

Функция потерь (отступ) является пороговой функцией и **имеет разрыв в точке 0**

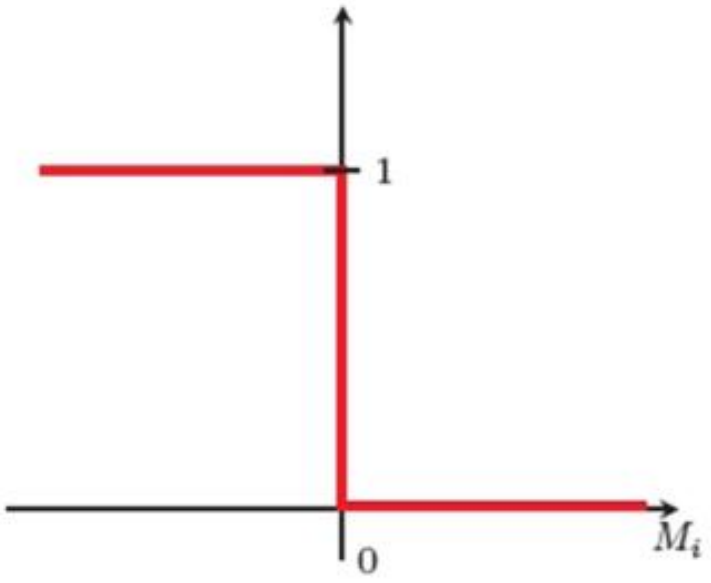


Рис. 2: График функции потерь

Для нахождения весов алгоритма необходимо дифференцировать функционал ошибки, а в случае бинарной классификации — функцию потерь. Поскольку данная функция не гладкая и имеет разрыв в 0, это сделать невозможно. Поэтому для нахождения минимума функционала ошибки в задаче классификации **дифференцируют оценку функции потерь**.

Для построения оценки функции потерь введем некую гладкую и неразрывную функцию $L(M_i)$, которая покрывает функцию потерь:

$$L(M_i) \geqslant [M_i < 0].$$

Тогда можно построить оценку функционала ошибки $\bar{Q}(a, X)$:

$$\bar{Q}(a, X) \geqslant Q(a, X),$$

$$\bar{Q}(a, X) = \frac{1}{n} \cdot \sum_{i=1}^{n} L(M_i)$$

Поскольку функция $L(M_i)$ гладкая и неразрывная, то оценку функционала ошибки уже можно дифференцировать и нужно будет минимизировать для нахождения значения весов бинарного классификатора.

В качестве гладкой функции $L(M_i)$ могут быть использованы разные функции. Рассмотрим 3 из них:

- **логистическая функция потерь** (используется в логистической регрессии)

$$L(M) = log_2(exp(-M))$$

- **экспоненциальная функция потерь** (выражается через экспоненту)

$$L(M) = (exp(-M))$$

- **кусочно-линейная функция потерь** (используется в методе опорных векторов)

$$L(M) = max(0, 1 - M)$$

На рисунке ниже приведен вид всех 3 оценок функционала ошибки. Синей линией изображена экспоненциальная оценка функции потерь, оранжевой линией — логистическая оценка функции потерь, серой — кусочно-линейная оценка функции потерь.

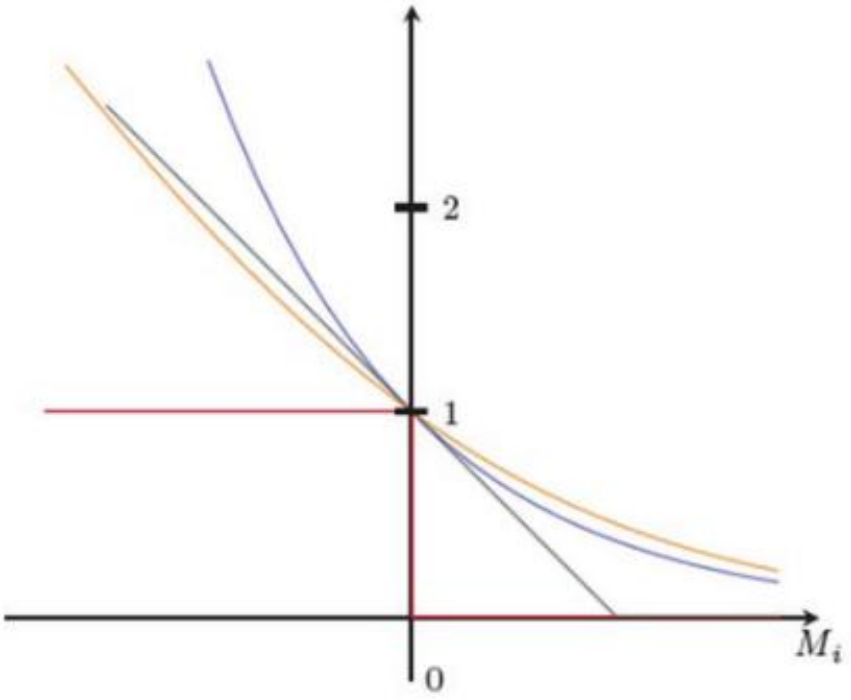


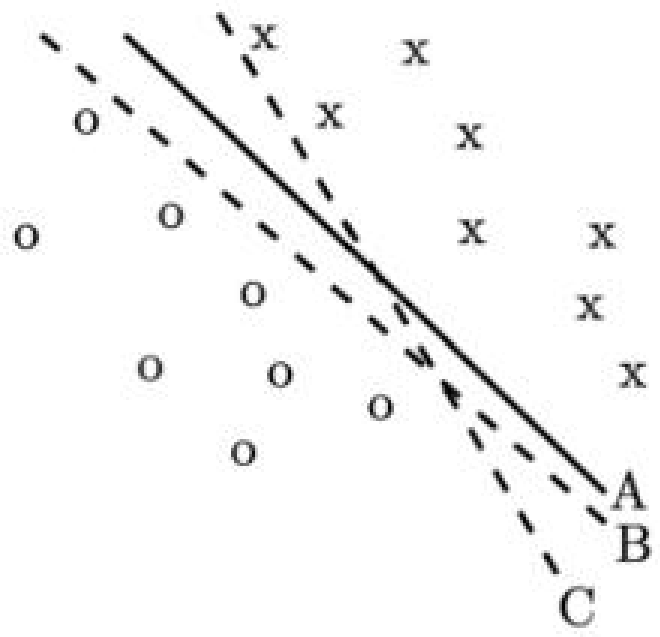
Рис. 3: График оценок функционала ошибки

Классификатор SVM

Метод Опорных Векторов или SVM (от англ. $SupportVectorMachines$) — это линейный алгоритм используемый в задачах классификации и регрессии. Данный алгоритм имеет широкое применение на практике и может решать как линейные так и нелинейные задачи. **Главным принципом** SVM является определение разделителя в искомом пространстве, который разделяет классы наилучшим образом. То есть находит такую границу (разделяющую гиперплоскость) между классами, которая была бы максимально удалена от «опорных» представителей каждого из классов и при этом была бы оптимальна.

Рассмотрим пример, показанный на рисунке ниже, на котором мы имеем два класса, обозначенных через "x"и "o". На рисунке также обозначены три гиперплоскости – A, B, и C. Очевидно, что гиперплоскость A производит наилучшее разделение классов, потому что расстояние от нее до любых точек максимально.

Иными словами, гиперплоскость A максимизирует зазор. Одно из преимуществ SVM-метода в том, что так как он пытается определить оптимальное направление разделения признакового пространства, рассматривая комбинации признаков, он достаточно устойчив к большим размерностям.



Заметим, что задача поиска лучшего разделителя является задачей оптимизации, которая может быть сведена к задаче квадратичного программирования. Так, большинство методов используют метод Ньютона для итерационной минимизации выпуклой функции. Он бывает достаточно медлителен, особенно для данных высокой размерности, таких, как текстовые документы.

Главная **цель** SVM **как классификатора** — найти уравнение разделяющей гиперплоскости

$$w_1x_1 + w_2x_2 + \cdots + w_nx_n + w_0 = 0$$

в пространстве R^n, которая бы разделила два класса неким оптимальным образом. Общий вид преобразования F объекта x в метку класса Y:

$$F(x) = sign(w^Tx - b)$$

Будем помнить, что мы обозначили $w = (w_1, w_2, \ldots, w_n),\ b = -w_0$. После настройки весов алгоритма w и b (обучения), все объекты, попадающие по одну сторону от построенной гиперплоскости, будут предсказываться как первый класс, а объекты, попадающие по другую сторону — второй класс.

Внутри функции $sign()$ стоит линейная комбинация признаков объекта с весами алгоритма, именно поэтому SVM относится к линейным алгоритмам. Разделяющую гиперплоскость можно построить разными способами, но в SVM веса w и b настраиваются таким образом, чтобы объекты классов лежали как можно дальше от разделяющей гиперплоскости.

Другими словами, **алгоритм максимизирует зазор** (англ. $margin$) между гиперплоскостью и объектами классов, которые расположены ближе всего к ней. Такие объекты и называют опорными векторами. Отсюда и название алгоритма.

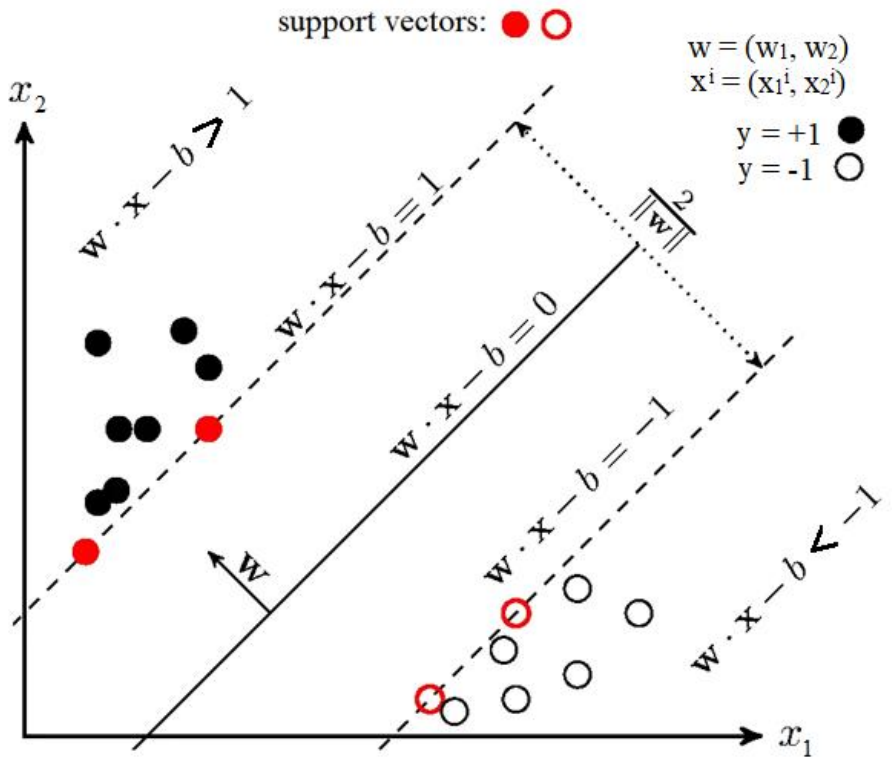

support vectors:
w = (w1, w2)
xi = (x1i, x2i)
y = +1
y = -1
x2
x1
w · x − b > 1
w · x − b = 1
w · x − b = 0
w · x − b = −1
w · x − b < −1
w
2/||w||

Рис. 4: SVM

Логистическая регрессия

Логистическая регрессия является частным случаем линейного классификатора, но она обладает хорошим свойством — прогнозировать вероятность p_+ отнесения примера x_i к классу " + ".

$$p_+ = P(y_i = 1|x_i, w)$$

Прогнозирование не просто ответа ($+1$ или -1), а именно вероятности отнесения к классу $+1$ во многих задачах является очень важным бизнес-требованием. Например, в задаче кредитного скоринга, где традиционно применяется логистическая регрессия, часто прогнозируют вероятность невозврата кредита (p_+).

Итак, мы хотим прогнозировать вероятность $(p_+) \in \{0, 1\}$. Каким образом преобразовать полученное значение в вероятность, пределы которой – $\{0, 1\}$? Для этого нужна некоторая функция $f : \mathbb{R} \to \{0, 1\}$. В модели логистической регрессии для этого берется функция:

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

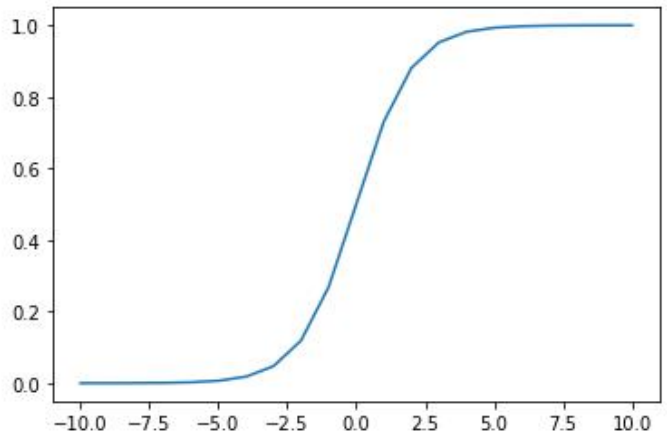
Ввод [1]:
```
import numpy as np
import matplotlib.pyplot as plt

def sigmoid(z):
    return 1 / (1 + np.e ** -z)

xrange = np.array(range(-10,11))
yrange = sigmoid(xrange)

plt.plot(xrange, yrange)
```

Out[1]: `[<matplotlib.lines.Line2D at 0x7faeafbaf190>]`



Обозначим $P(X)$ вероятностью происходящего события x. Тогда отношение вероятностей $OR(X)$ определяется из $\frac{P(X)}{1-P(X)}$, а это - отношение вероятностей того, произойдет ли событие или не произойдет. Вероятность и отношение шансов содержат одинаковую информацию. Но в то время как $P(X)$ находится в пределах от 0 до 1, $OR(X)$ находится в пределах от 0 до ∞.

Если вычислить логарифм $OR(X)$ (то есть так называемый логарифм шансов, или логарифм отношения вероятностей), то легко заметить, что $log(OR(X)) \in \mathbb{R}$. Его и будем прогнозировать с помощью МНК.

Посмотрим, как логистическая регрессия будет делать прогноз

$$p_+ = P(y_i = 1|x_i, w)$$

Пока будем считать, что веса w мы уже получили.

Шаг 1: Вычислить значение $w_0 + w_1x_1 + w_2x_2 + \cdots = w^Tx = 0$ задает гиперплоскость, разделяющую примеры на 2 класса;

Шаг 2: Вычислить логарифм отношения шансов: $log(OR_+) = w^Tx$

Шаг 3: Имея прогноз шансов на отнесение к классу +, то есть OR_+, вычислить p_+ с помощью простой зависимости:

$$p_+ = \frac{OR_+}{1 + OR_+} = \frac{e^{w^Tx}}{1 + e^{w^Tx}} = \frac{1}{1 + e^{-w^Tx}} = \sigma(w^Tx)$$

В правой части получили **сигмоид-функцию.**

Итак, логистическая регрессия прогнозирует вероятность отнесения примера к классу "+"(при условии, что мы знаем его признаки и веса модели) как сигма-преобразование линейной комбинации вектора весов модели и вектора признаков примера:

$$p_+(x_i) = P(y_i = 1|x_i, w) = \sigma(w^Tx)$$

Следующий вопрос: **как модель обучается?** Тут мы обращаемся к принципу максимального правдоподобия.

Теперь посмотрим, как из принципа максимального правдоподобия получается оптимизационная задача, которую решает логистическая регрессия, а именно, — минимизация логистической функции потерь. Только что мы увидели, что логистическая регрессия моделирует вероятность отнесения примера к классу "+"как:

$$p_+(x_i) = P(y_i = 1|x_i, w) = \sigma(w^T x)$$

$$p_-(x_i) = P(y_i = -1|x_i, w) = 1 - \sigma(w^T x) = \sigma(-w^T x)$$

Оба этих выражения можно объединить в одно:

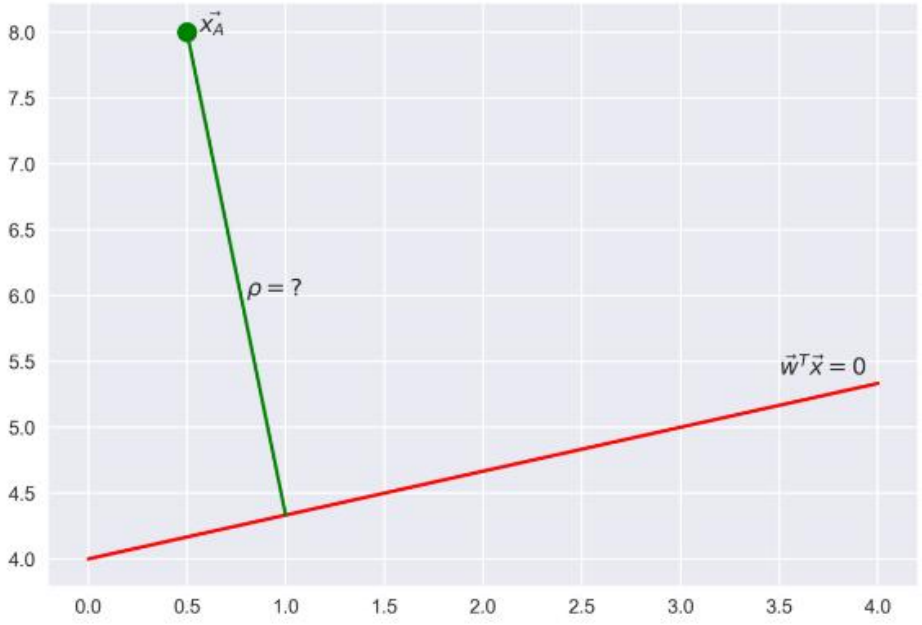
$$P(y = y_i|x_i, w) = \sigma(y_i w^T x)$$

Выражение $M(x_i) = y_i w^T x$ **называется отступом** $(margin)$ классификации на объекте x_i (не путать с зазором (тоже $margin$), про который чаще всего говорят в контексте SVM). Если он неотрицателен, модель не ошибается на объекте x_i , если же отрицателен – значит, класс для x_i спрогнозирован неправильно. Заметим, что отступ определен для объектов именно обучающей выборки, для которых известны реальные метки целевого класса y_i.

Чтобы понять, почему это мы сделали такие выводы, обратимся к геометрической интерпретации линейного классификатора.

Решим задачу линейной алгебры: найти расстояние от точки с радиус-вектором x_A до плоскости, которая задается уравнением $w^T x = 0$:

$$\rho(x_A, w^T x = 0) = \frac{w^T x_A}{||w||}$$



Чем больше по модулю выражение $w^T x_i$, тем дальше точка x_i находится от плоскости $w^T x_i = 0$.

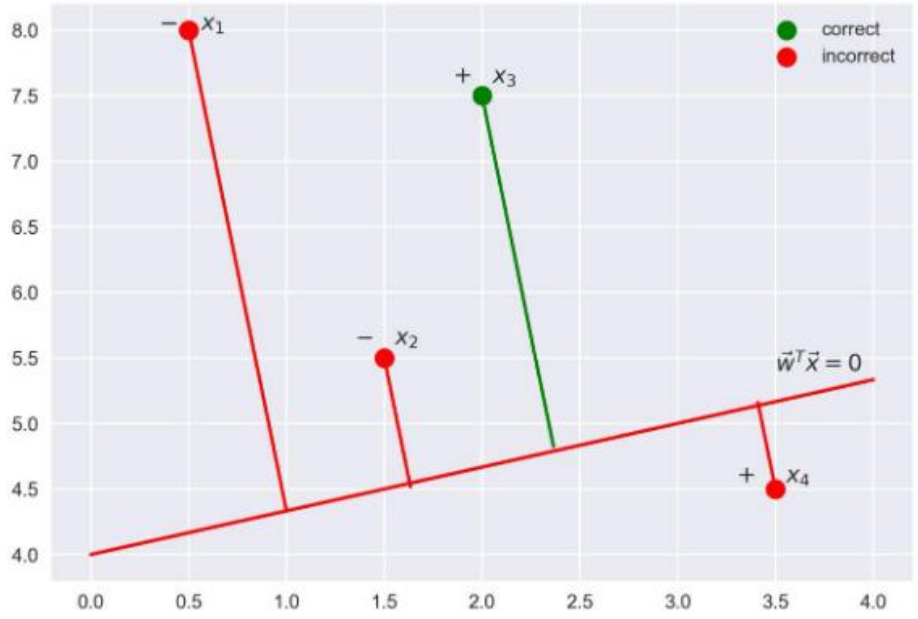
Значит, выражение $M(x_i) = y_i w^T x_i$ — это "уверенность"модели в классификации объекта x_i:

- если отступ большой (по модулю) и положительный, это значит, что метка класса поставлена правильно, а объект находится далеко от разделяющей гиперплоскости (такой объект классифицируется уверенно). **На рисунке** – x_3.

- если отступ большой (по модулю) и отрицательный, значит метка класса поставлена неправильно, а объект находится далеко от разделяющей гиперплоскости (скорее всего такой объект – аномалия, например, его метка в обучающей выборке поставлена неправильно). **На рисунке** – x_1.

- если отступ малый (по модулю), то объект находится близко к разделяющей гиперплоскости, а знак отступа определяет, правильно ли объект

классифицирован. **На рисунке** – x_2 **и** x_4**.**



Теперь распишем правдоподобие выборки, а именно, вероятность наблюдать данный вектор у выборки X. Делаем предположение: объекты приходят независимо, из одного распределения. Тогда:

$$P(y|X,w) = \prod_{i=1}^{l} P(y = y_i|x_i, w)$$

где l - длина выборки X.

Возьмём логарифм данного выражения:

$$logP(y|X,w) = log \prod_{i=1}^{l} P(y = y_i|x_i, w) = log \prod_{i=1}^{l} \sigma(y_i w^T x_i)$$

это в свою очередь равно:

$$\sum_{i=1}^{l} log\sigma(y_i w^T x_i) = -\sum_{i=1}^{l} log(1 + exp^{-y_i w^T x_i})$$

То есть в данном случае принцип максимизации правдоподобия приводит к минимизации выражения:

$$\mathcal{L}_{log}(X, y, w) = \sum_{i=1}^{l} log(1 + exp^{-y_i w^T x_i})$$

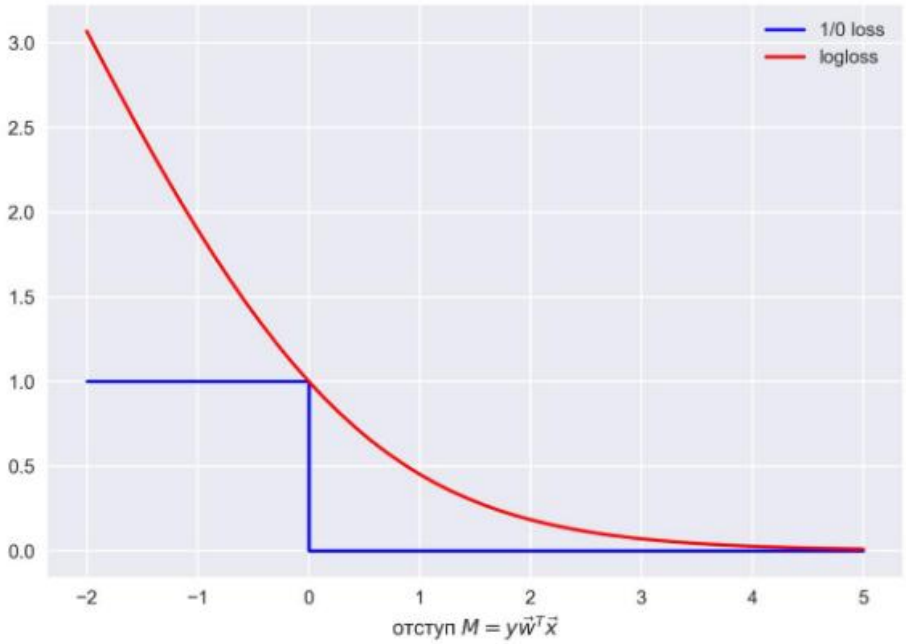
Это **логистическая функция потерь**, просуммированная по всем объектам обучающей выборки.

Посмотрим на новую функцию как на функцию от отступа:

$$L(m) = log(1 + exp^{-M})$$

Нарисуем ее график, а также график 1/0 функциий потерь ($zero - one\ loss$), которая просто штрафует модель на 1 за ошибку на каждом объекте (отступ отрицательный):

$$L_{1/0}(M) = [M < 0]$$



Картинка отражает общую идею, что в задаче классификации, не умея напрямую минимизировать число ошибок (по крайней мере, градиентными методами это не сделать – производная $1/0$ функций потерь в нуле обращается в бесконечность), мы минимизируем некоторую ее верхнюю оценку. В данном случае это логистическая функция потерь (где логарифм двоичный, но это не принципиально), и справедливо следующее:

$$\mathcal{L}_{1/0}(X, y, w) = \sum_{i=1}^{l}[M(x_i) < 0] \leqslant \sum_{i=1}^{l} log(1 + exp^{-y_i w^T x_i}) = \mathcal{L}_{log}(X, y, w)$$

где $\mathcal{L}_{1/0}(X, y, w)$ - число ошибок логистической регрессии с весами W на выборке (X, y).

То есть, уменьшая верхнюю оценку $\mathcal{L}_{log}$ на число ошибок классификации, мы таким образом надеемся уменьшить и само число ошибок.

Отличия SVM и логистической регрессии

Методы обучения линейных классификаторов **различаются, главным образом, выбором функции** $\mathcal{L}(M)$, способом регуляризации и выбором численного метода оптимизации.

Метод опорных векторов соответствует кусочно-линейной аппроксимации:

$$[M < 0] \leqslant (1 - M)_+$$

При этом обязательно применяется регуляризация с квадратичной нормой. Задача обучения решается специализированными методами квадратичного программирования.

Логистическая регрессия соответствует логарифмической аппроксимации:

$$[M < 0] \leqslant log(1 + exp^{-M})$$

Задача обучения решается градиентным методом второго порядка с методом наименьших квадратов

Логистическая регрессия является частным случаем обобщённой линейной модели регрессии.

Практическая часть

С помощью программы из **Приложения 1** сгенерируем случайные искусственные данные:

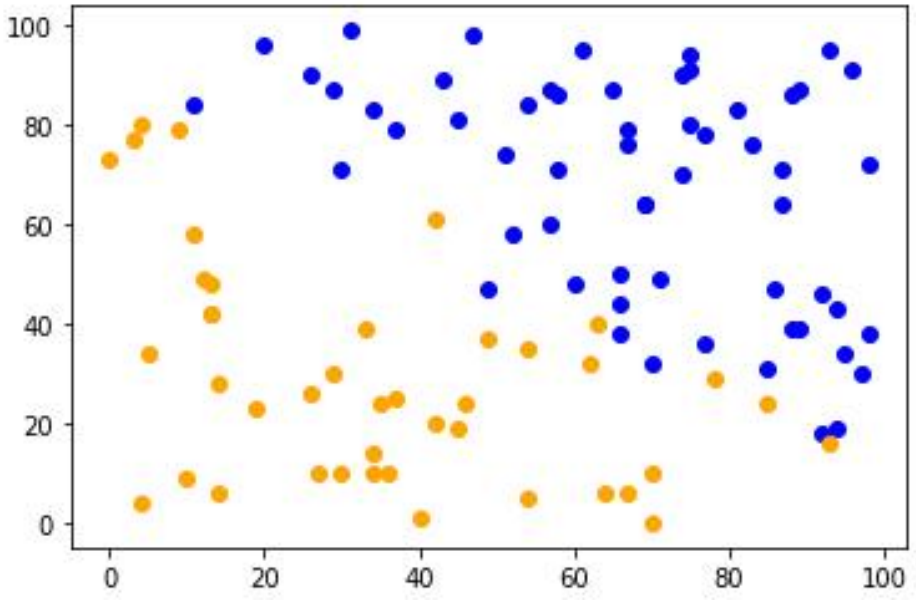


Рис. 5: Сгенерированные данные

Граница была задана функцией $y = 100 - x$. В данные внесены шумы: если точка близка к границе, то выбор ее принадлежности к классу осуществлялся случайно. Как видим, идеальной разделяющей поверхности нет.

Решим задачу с помощью логистической регрессии. Запустив код из **Приложения 2**, мы получим такой вывод:

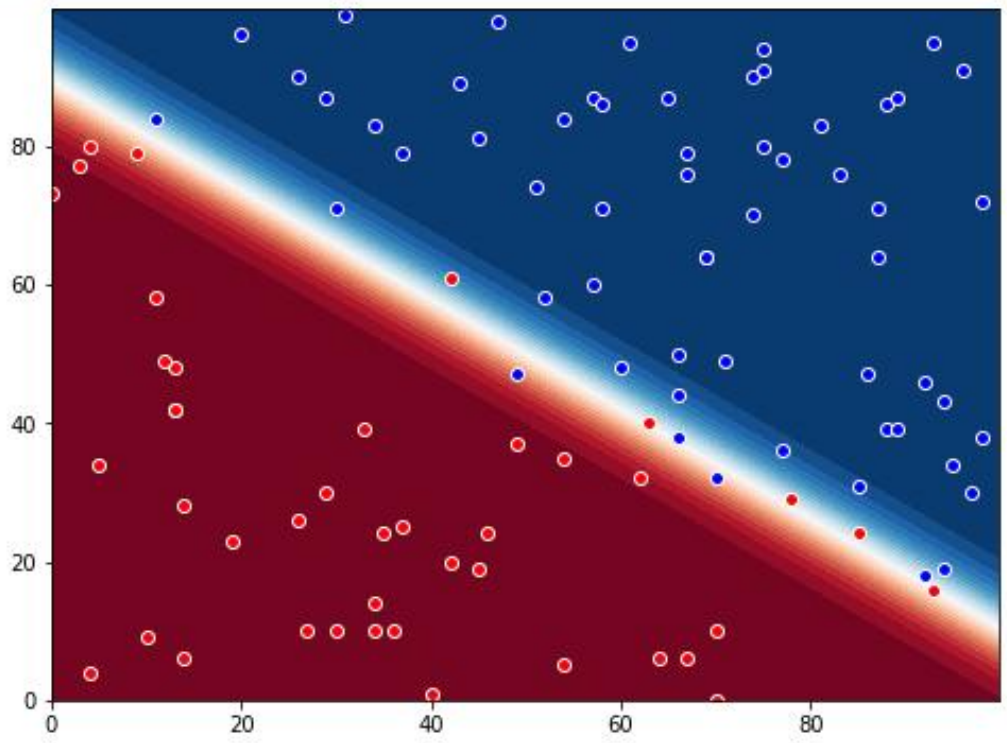


Рис. 6: Классификация с помощью логистической регрессии

На рисунке четко видна разделяющая плоскость (в случае двумерного пространства — прямая), причем насыщенность цвета обозначает вероятность отнесения точки к классу. То есть чем насыщеннее цвет — тем больше "уверенность"классификатора.

Если мы будем решать задачу с помощью классификатора SVM из **приложения 3** с режимом ядра "*linear*", то получим такую картину:

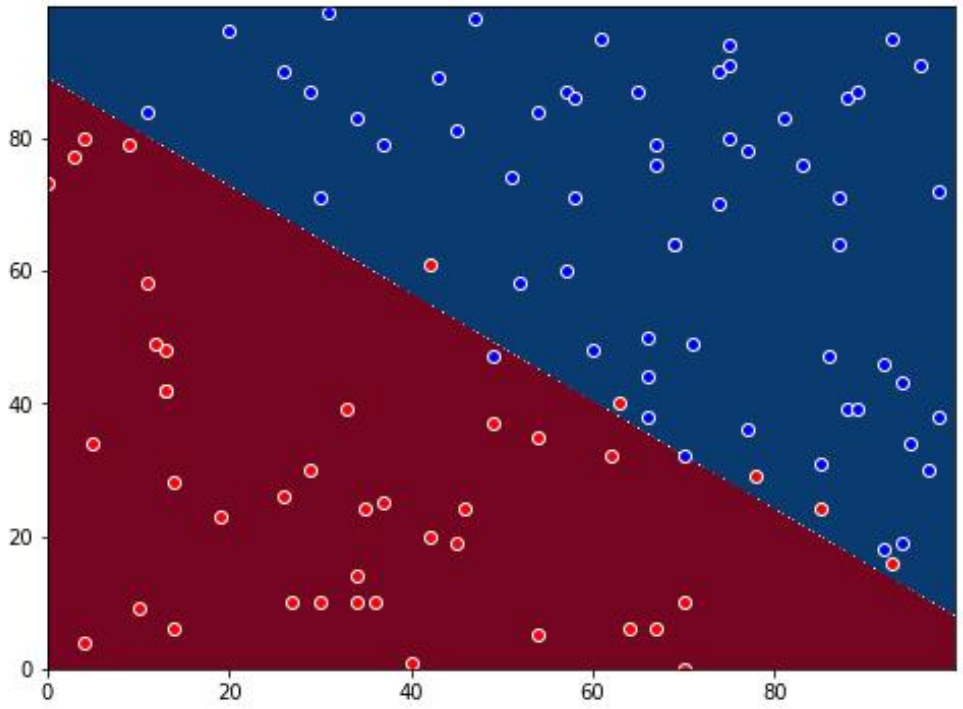


Рис. 7: Классификация с помощью SVM, режим ядра "*linear*"

Разделяющая поверхность очень похожа на решение логистической регрессии, однако в данном случае у нас нет вероятности отнесения точки к какому-либо классу: у нас есть лишь ответ $+1$ или -1.

Данное решение содержит ошибки, но мы можем постараться убрать их, сделав ядро SVM нелинейным (rbf):

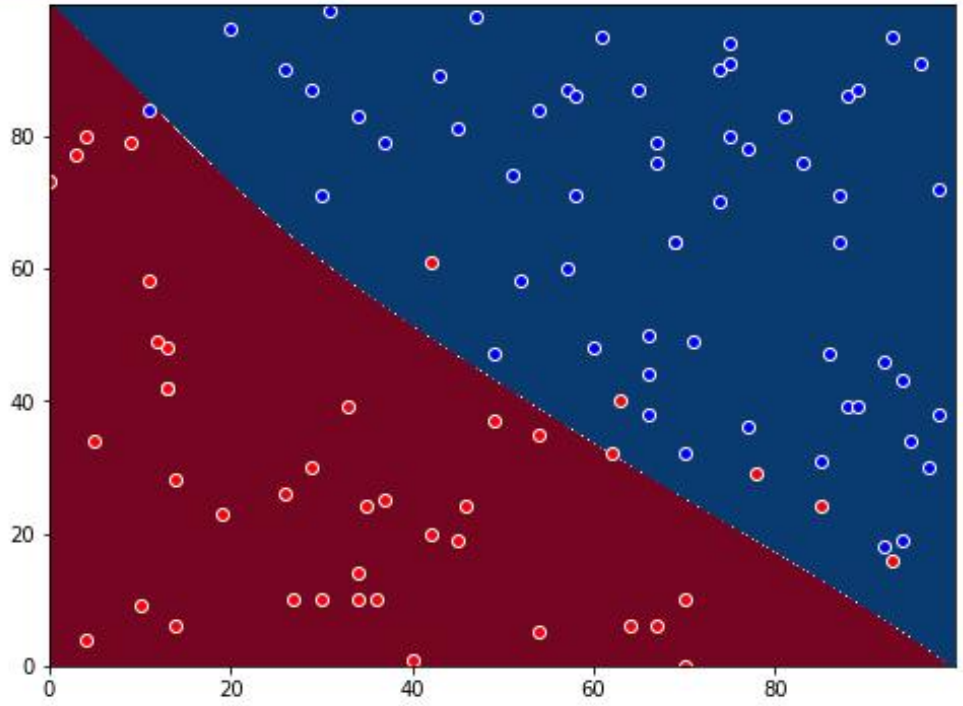


Рис. 8: Классификация с помощью SVM, режим ядра "rbf"

Заключение

Задача классификации — классическая задача машинного обучения. На самом деле, задачи классификации могут решаться вышеизложенными методами не только для двух классов, но и для многоклассовой классификации. Для этого используются другие подходы, как, например, $1 - vs - all$.

SVM и логистическая регрессия часто используются в современном машинном обучении. Но SVM может предсказывать лишь класс объекта из выборки, а логистическая регрессия может предсказывать вероятность отнесения данного объекта к классу — такое преимущество логистическая регрессия имеет из-за своей функции потерь

Литература

[1] Открытый курс машинного обучения [электронный ресурс]: https://habr.com/ru/company/ods/blog/323890/ (10.11.2021)

[2] Открытый курс машинного обучения. Тема 4. [электронный ресурс]: https://habr.com/ru/company/ods/blog/323890/ (15.11.2021)

[3] Логистическая регрессия [электронный ресурс]: https://edu.vsu.ru/pluginfile.php/1348523/mod$_r$*esource*/*content*/1(17.11.2021)

[4] А.Н.Васильев, Д.А.Тархов. Нейростевое моделирование. Принципы. Алгоритмы. Приложения. СПб.: Изд-во Политехн. Ун-та, 2009. ISBN 978-5-7422-2272-9

[5] О.Жерон. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. Концепции, инструменты и техники для создания интеллектуальных систем. Вильямс, 2018, 688 с. ISBN 978-5-950-02962-2.

Приложение 1

```
import itertools
import matplotlib.pyplot as plt
import json


x = np.random.random(100) * 100
x = x.astype(np.int)

y = np.random.random(100) * 100
y = y.astype(np.int)


def get_label(x: int, y: int) -> int:
    expected_y = 100 - x
    margin = y - expected_y

    if abs(margin) < 10:
        label = np.random.randint(0, 2)
    elif margin > 0:
        label = 1
    else:
        label = 0

    return label
```

```
labels = np.array([get_label(x_point, y_point) for x_point, y_point

data = {
    "x": [],
    "y": [],
    "class": [],
}

for x_point, y_point, label in zip(x, y, labels):
    data['x'].append(int(x_point))
    data['y'].append(int(y_point))
    data['class'].append(int(label))

with open("data.json", 'w') as f:
    json.dump(data, f, indent=1)

upper_x = list(itertools.compress(x, labels))
upper_y = list(itertools.compress(y, labels))

labels = (labels + 1) \% 2 # invert labels 1 -> 0 and 0 -> 1

lower_x = list(itertools.compress(x, labels))
lower_y = list(itertools.compress(y, labels))

plt.scatter(upper_x, upper_y, color='blue')
plt.scatter(lower_x, lower_y, color='orange')
```

```
plt.show()
```

Приложение 2

```
import pandas as pd
from sklearn.svm import SVC
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_json("./data.json")

X = df.drop("class", axis=1)
y = df['class']

clf = SVC(kernel='rbf')
clf.fit(X, y)

x_grid, y_grid = np.mgrid[0:100:.1, 0:100:.1]
grid = np.c_[x_grid.ravel(), y_grid.ravel()]
preds = clf.predict(grid).reshape(x_grid.shape)

f, ax = plt.subplots(figsize=(8, 6))
contour = ax.contourf(x_grid, y_grid, preds, 25, cmap="RdBu", vmin=
plt.scatter(df[df['class'] == 1].x, df[df['class'] == 1].y,
color='blue', edgecolor="white")
plt.scatter(df[df['class'] == 0].x, df[df['class'] == 0].y,
color='red', edgecolor="white")
```

```
plt.show()
```

Приложение 3

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
import numpy as np
import matplotlib.pyplot as plt


df = pd.read_json("./data.json")

X = df.drop("class", axis=1)
y = df['class']

clf = LogisticRegression()
clf.fit(X, y)

x_grid, y_grid = np.mgrid[0:100:.1, 0:100:.1]
grid = np.c_[x_grid.ravel(), y_grid.ravel()]
probs = clf.predict_proba(grid)[:, 1].reshape(x_grid.shape)

f, ax = plt.subplots(figsize=(8, 6))
contour = ax.contourf(x_grid, y_grid, probs, 25, cmap="RdBu", vmin=

plt.scatter(df[df['class'] == 1].x, df[df['class'] == 1].y,
 color='blue', edgecolor="white")
```

```
plt.scatter(df[df['class'] == 0].x, df[df['class'] == 0].y,
 color='red', edgecolor="white")
plt.show()
```