Методы машинного обучения

Лекция 6

Классификация - Продолжение

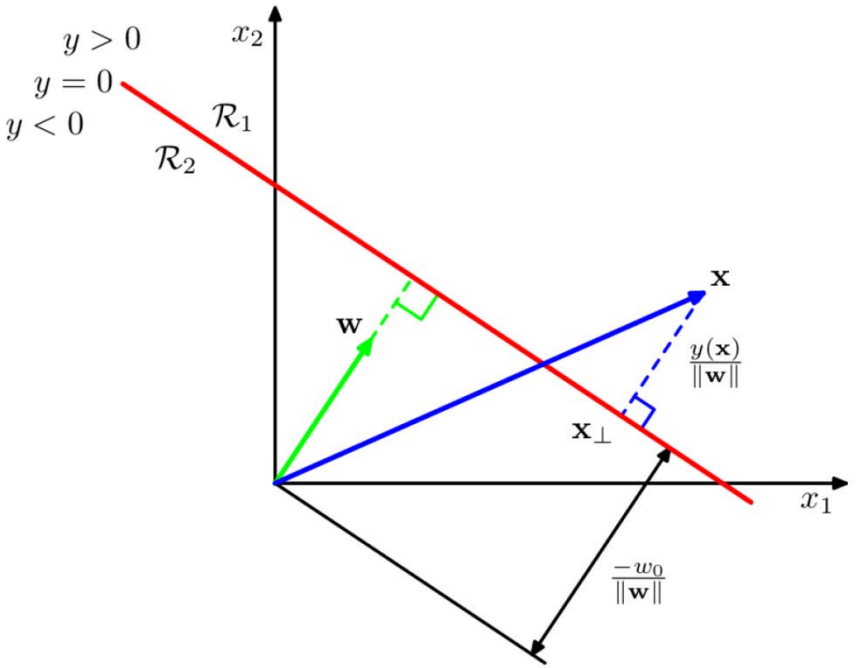
Задача классификации и разделяющая гиперплоскость

- Рассмотрим линейную дискриминантную функцию:

$$y(x) = w^T x + w_0$$

является гиперплоскостью и w – нормаль к ней.

- Расстояние от начала координат до гиперплоскости: $\frac{-w_0}{\|w\|}$
- $y(x)$ связано с расстоянием до гиперплоскости: $d = \frac{y(x)}{\|w\|}$

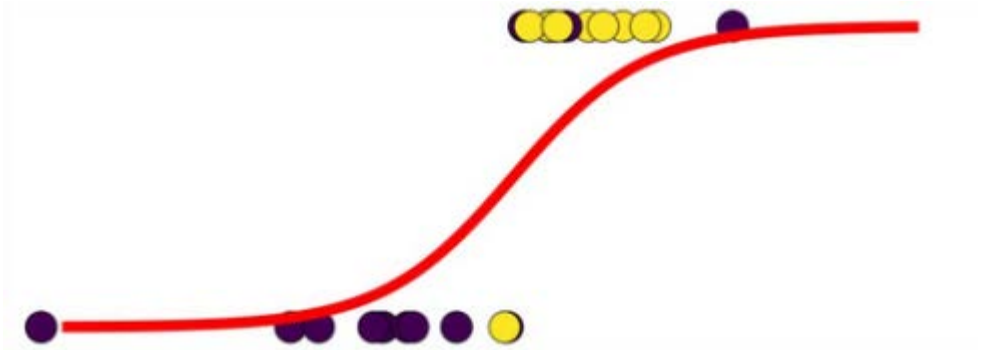


Логистическая регрессия: определение

Логистическая регрессия или **логит-модель** (*logit model*) — статистическая модель, используемая для прогнозирования вероятности возникновения некоторого события путём его сравнения с логистической кривой. Эта регрессия выдаёт ответ в виде вероятности бинарного события (1 или 0).

Применяется для прогнозирования вероятности возникновения некоторого события по значениям множества признаков.

- вводится так называемая зависимая переменная $y \in \{0, 1\}$ (0 - событие не произошло и 1 - событие произошло);
- множество независимых переменных (также называемых признаками, предикторами или регрессорами) — $x_1, x_2, \ldots, x_n \in \mathbb{R}$, на основе значений которых требуется вычислить вероятность принятия того или иного значения зависимой переменной;
- для простоты записи вводится фиктивный признак $x_0 = 1$.



Логистическая регрессия: описание модели

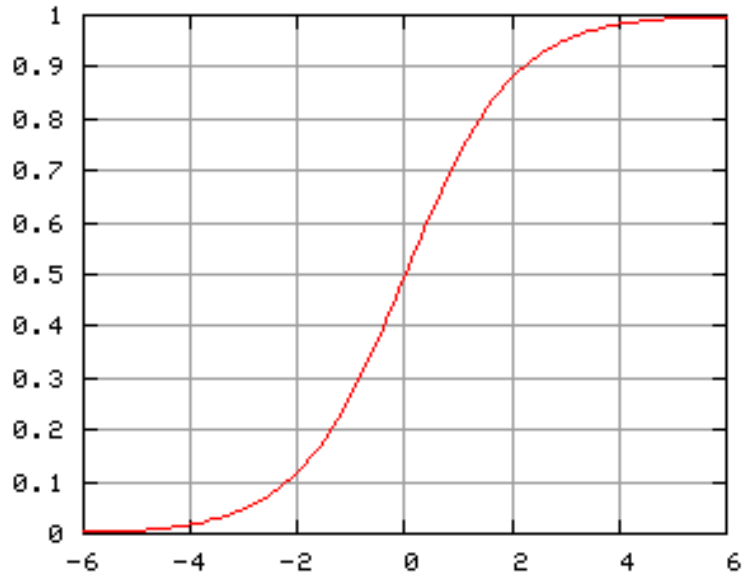
Делается предположение о том, что вероятность наступления события $y = 1$ равна:

$$P\{y = 1|x\} = f(z), \qquad z = \theta^T x = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

где

- *x* – вектор-столбец значений независимых переменных 1, x_1, x_2, …, x_n
- θ – вектор-столбец параметров (коэффициентов регрессии) – вещественные числа $\theta_0, \ldots, \theta_n$
- *f*(*z*) – логистическая функция (сигмоида или логит-функция):

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



Т.к. *y* принимает значения 0 и 1, то вероятность принять значение 0 равна:

$$P\{y = 0|x\} = 1 - f(z) = 1 - f(\theta^T x)$$

Функция распределения *y* при заданном *x*:

$$P\{y|x\} = f(\theta^T x)^y (1 - f(\theta^T x))^{1-y}, \qquad y \in \{0, 1\}$$

Логистическая регрессия: подбор параметров

Для подбора параметров θ_0, …, θ_n необходимо составить обучающую выборку, состоящую из наборов значений независимых переменных и соответствующих им значений зависимой переменной *y*.

- Формально, это множество пар $(x^{(1)}, y^{(1)})$, …, $(x^{(m)}, y^{(m)})$, где $x^{(i)} \in \mathbb{R}^n$ и $y^{(i)} \in \{0, 1\}$.
- Обычно используется метод максимального правдоподобия, согласно которому подбираются параметры θ, максимизирующие значение функции правдоподобия на обучающей выборке:

$$\hat{\theta} = arg\max_{\theta} L(\theta) = arg\max_{\theta} \prod_{i=1}^{m} P\{y = y^{(i)} | x = x^{(i)}\}$$

Определяются оценки максимального правдоподобия (maximum likelihood estimates), для которых значения параметров являются наиболее «правдоподобными» по отношению к наблюдаемым данным.

Подбор параметров: метод Ньютона

Максимизация функции правдоподобия эквивалентна максимизации её логарифма:

$$\ln L(\theta) = \sum_{i=1}^{m} \ln\ P\left\{y = y^{(i)} \middle| x = x^{(i)}\right\} = \sum_{i \in I_1} \ln P_i(\theta) + \sum_{i \in I_0} \ln(1 - P_i(\theta))$$

$$= \sum_{i=1}^{m} \left[y^{(i)} \ln f\left(\theta^T x^{(i)}\right) + \left(1 - y^{(i)}\right) \ln(1 - f\left(\theta^T x^{(i)}\right)),\right]$$

где $\theta^T x^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_n x_n^{(i)}$, $f(z) = \frac{1}{1+e^{-z}}$ I_0, I_1 - множества наблюдений, для которых $y^{\{i\}} = 0$ и $y^{\{i\}} = 1$ соответственно.

Можно показать, что градиент: $grad = \sum_i \left(y^{(i)} - f(\theta^T x)\right) x^{(i)}$

Гессиан *H* функции правдоподобия равен: $H = -\sum_i P_i(1 - P_i) X_i^T X_i \leq 0$

Гессиан функции— симметрическая квадратичная форма, описывающая поведение функции во втором порядке. Для функции $f(\theta)$ дважды дифференцируемой в точке θ.

Гессиан всюду отрицательно определенный, поэтому логарифмическая функция правдоподобия всюду вогнута. Для поиска максимума можно использовать метод Ньютона, который в этом случае будет всегда сходиться

$$\theta_{t+1} = \theta_t - \left(H(\theta_t)\right)^{-1} \cdot grad_t(\theta_t) = \theta_t - \Delta\theta_t$$

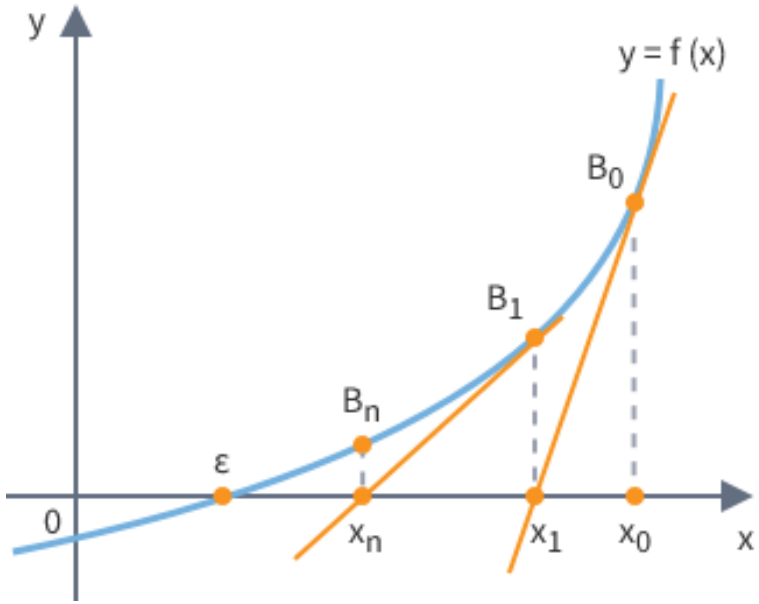
Метод Ньютона (метод касательных)

Итерационный численный метод нахождения корня (нуля) или поиска экстремума заданной функции многих переменных. Поиск решения осуществляется путём построения последовательных приближений.

Основная идея: задаётся начальное приближение вблизи предположительного корня, строится касательная к графику исследуемой функции в точке приближения, для которой находится пересечение с осью абсцисс. Эта точка берётся в качестве следующего приближения и т.д.

Алгоритм нахождения численного решения уравнения $f(x) = 0$ сводится к итерационной процедуре вычисления:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



Метод Ньютона применительно к задачам оптимизации

В случае решения задач оптимизации предполагается, что функция $f(x)$ дважды непрерывно дифференцируема. Поиск минимума функции $f(x)$ производится при помощи отыскания стационарной точки, т.е. точки удовлетворяющей уравнению:

$$f'(x) = 0.$$

Если x_k – точка, полученная на *k*-м шаге, то функция $f'(x)$ аппроксимируется своим уравнением касательной:

$$y = f'(x_k) + (x - x_k) \cdot f''(x_k),$$

а точка x^{k+1} выбирается как пересечение этой прямой с осью Ox, т.е.

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}.$$

В общем виде: Пусть необходимо найти минимум функции многих переменных $f(\vec{x})$: $\mathbb{R}^n \to \mathbb{R}$. Эта задача равносильна задаче нахождения нуля градиента $\nabla f(\vec{x})$.

$$\vec{x}_{k+1} = \vec{x}_k - H^{-1}(\vec{x}_k)\,\nabla f(\vec{x}_k)$$

где $H(\vec{x})$ — гессиан функции $f(\vec{x})$.

Гессиан функции — симметрическая квадратичная форма, описывающая поведение функции во втором порядке. Для функции $f(\vec{x})$ дважды дифференцируемой в точке x.

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Подбор параметров: градиентный спуск

Максимизация функции правдоподобия эквивалентна максимизации её логарифма:

$$\ln L(\theta) = \sum_{i=1}^{m} log\ \ P\{y = y^{(i)} | x = x^{(i)}\}$$

$$= \sum_{i=1}^{m} \left[y^{(i)} \ln f\left(\theta^T x^{(i)}\right) + \left(1 - y^{(i)}\right) \ln(1 - f\left(\theta^T x^{(i)}\right))\right],$$

где $\theta^T x^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_n x_n^{(i)}$, $f(z) = \frac{1}{1+e^{-z}}$,

а градиент $\nabla = \sum_i \left(y^{(i)} - f(\theta^T x)\right) x^{(i)}$.

Для максимизации этой функции может быть применён, например, метод градиентного спуска. Он заключается в выполнении следующих итераций, начиная с некоторого начального значения параметров θ:

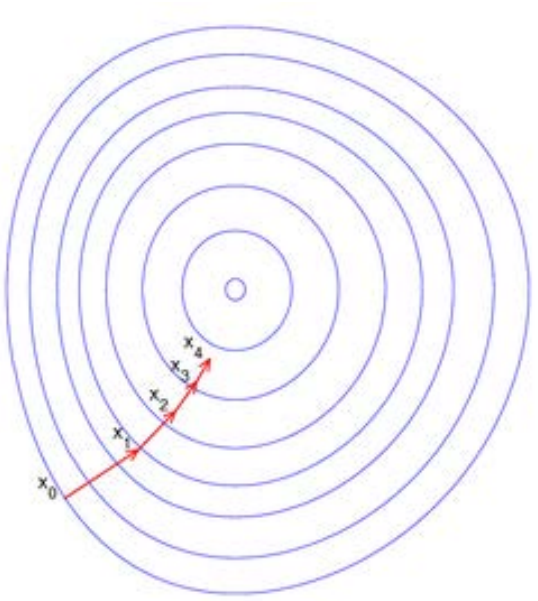
$$\theta_{t+1} = \theta_t + \lambda \nabla \ln L(\theta) = \theta_t + \lambda \sum_{i=1}^{m} \left(y^{(i)} - f(\theta^T x^{(i)})\right) x^{(i)}, \lambda > 0.$$

Метод градиентного спуска

Численный метод нахождения локального минимума или максимума функции с помощью движения вдоль градиента.

Пусть целевая функция имеет вид: $F(\vec{x})$: $\mathbb{X} \to \mathbb{R}$.

Задача оптимизации сформулирована следующим образом: $F(\vec{x}) \to \min\limits_{\vec{x} \in \mathbb{X}}$ (найти минимум).



Основная идея метода заключается в том, чтобы идти в направлении наискорейшего спуска, а это направление задаётся антиградиентом –∇*F*:

$$\vec{x}^{[j+1]} = \vec{x}^{[j]} - \lambda^{[j]} \nabla F\left(\vec{x}^{[j]}\right),$$

где

- градиентом $grad\ F = \nabla F$ называется n-мерный вектор $\left(\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \dots, \frac{\partial F}{\partial x_n}\right)$, компоненты которого равны частным производным *F* по всем ее аргументам.
- $\lambda^{[j]}$ задает скорость градиентного спуска и может быть выбрана:
  - постоянной (в этом случае метод может не сходиться);
  - убывающей в процессе градиентного спуска;
  - гарантирующей наискорейший спуск, тогда для поиска минимума $F(\vec{x})$ получаем:

$$\lambda^{[j]} = \underset{\lambda}{\operatorname{argmin}} F(\vec{x}^{[j+1]}) = \underset{\lambda}{\operatorname{argmin}} F(\vec{x}^{[j]} - \lambda^{[j]} \nabla F(\vec{x}^{[j]}))$$

Метод градиентного спуска: Алгоритм

1. Задать начальное приближение и точность расчета: $\vec{x}^{[0]}$, ε.
2. Рассчитать $\vec{x}^{[j+1]} = \vec{x}^{[j]} - \lambda^{[j]} \nabla F\left(\vec{x}^{[j]}\right)$,

   где $\lambda^{[j]} = \underset{\lambda}{\operatorname{argmin}} F(\vec{x}^{[j]} - \lambda^{[j]} \nabla F(\vec{x}^{[j]}))$.
3. Проверить условие остановки:
   - Если $\left|\vec{x}^{[j+1]} - \vec{x}^{[j]}\right| > \varepsilon$, $\left|F\left(\vec{x}^{[j+1]}\right) - F\left(\vec{x}^{[j]}\right)\right| > \varepsilon$ или $\left\|\nabla F\left(\vec{x}^{[j+1]}\right)\right\| > \varepsilon$ (выбирается одно из условий),

     то $j = j + 1$ перейти к шагу 2.
   - Иначе $\vec{x} = \vec{x}^{[j+1]}$ и останов.

Метод градиентного спуска: Особенности

- Рис.1. Геометрическая интерпретация метода градиентного спуска с постоянным шагом. На каждом шаге мы сдвигаемся по вектору антиградиента, "уменьшенному в λ раз".
- Рис.2. Ситуация, когда метод градиентного спуска сходится медленно.
- Рис.3. Геометрическая интерпретация метода наискорейшего спуска. На каждом шаге $\lambda^{[j]}$ выбирается так, чтобы следующая итерация была точкой минимума функции на луче L.
- Рис.4. Схождение к разным локальным минимумам в зависимости от начальной точки старта.

Рис.1.

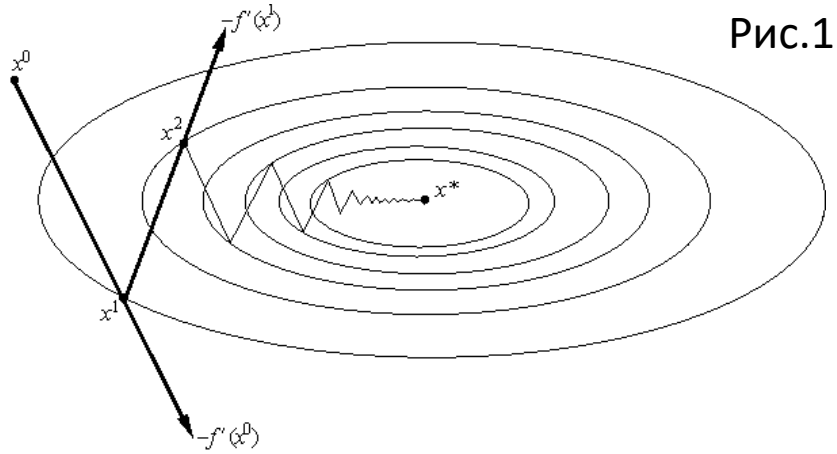


Рис.2.

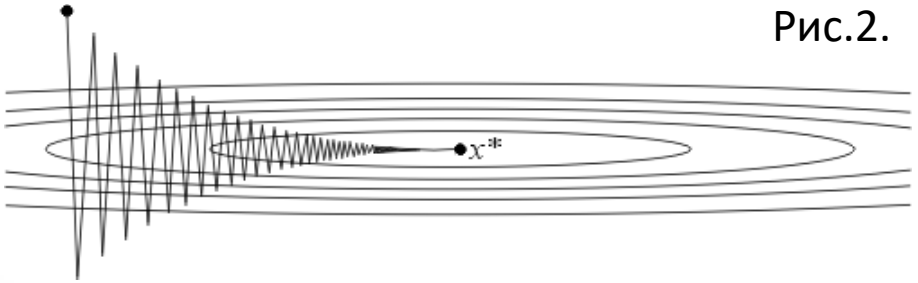


Рис.3.

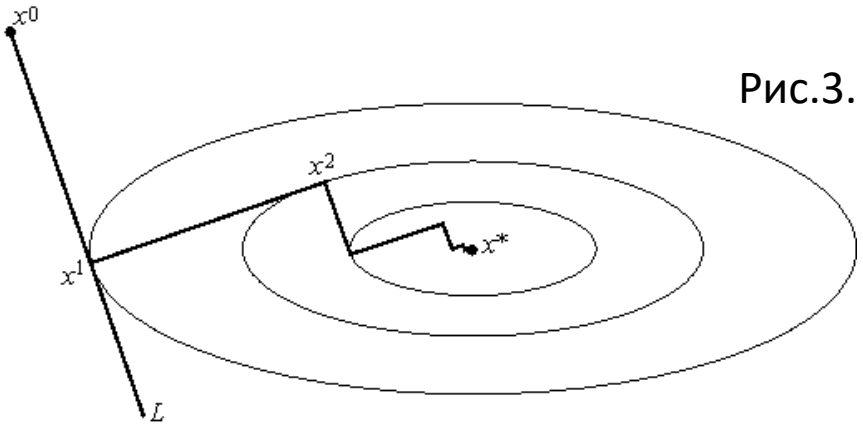
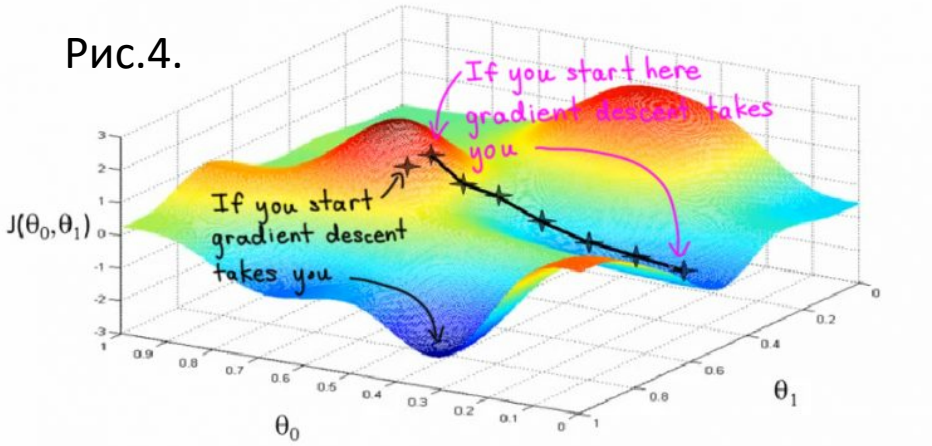


Рис.4.

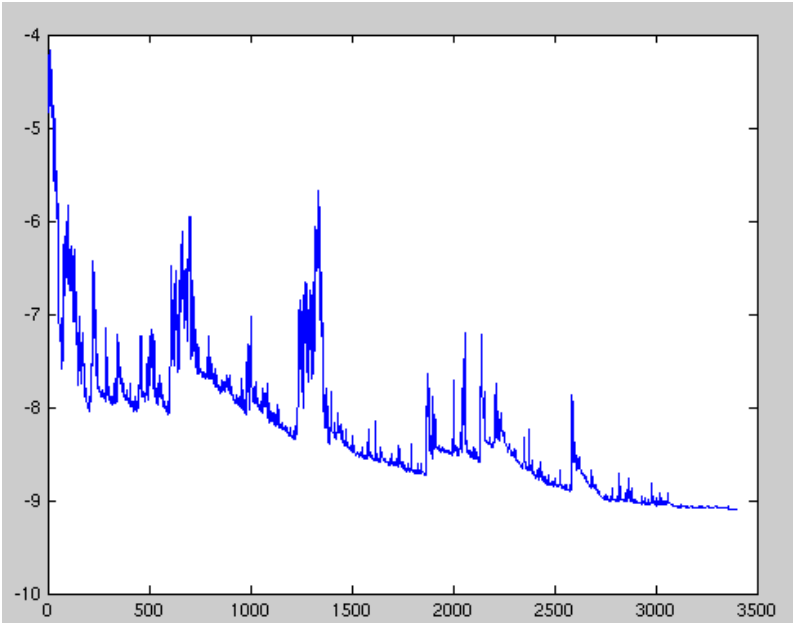


Стохастический градиентный спуск (Stochastic gradient descent, SGD)

Итерационный метод для оптимизации целевой функции с подходящими свойствами гладкости (например, дифференцируемость или субдифференцируемость). Его можно расценивать как стохастическую аппроксимацию оптимизации методом градиентного спуска, поскольку он заменяет реальный градиент, вычисленный из полного набора данных, оценкой, вычисленной из случайно выбранного подмножества данных.

Это сокращает задействованные вычислительные ресурсы и помогает достичь более высокой скорости итераций в обмен на более низкую скорость сходимости.

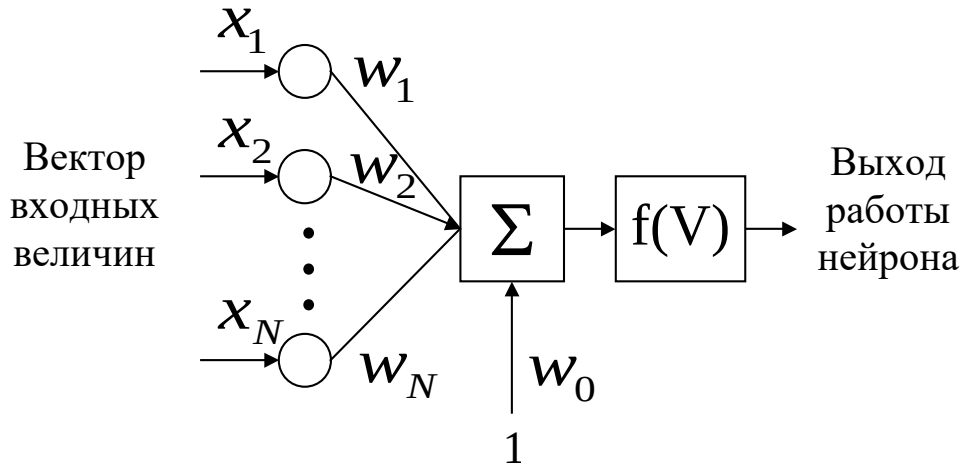
Особенно большой эффект достигается в приложениях, связанных с обработкой больших данных.



Представление в виде однослойной нейронной сети

Логистическую регрессию можно представить в виде однослойной нейронной сети с сигмоидальной функцией активации, веса которой есть коэффициенты логистической регрессии, а вес поляризации — константа регрессионного уравнения.

Структурная схема нейрона:



$$V = \sum_{i=0}^{N} w_i \cdot x_i$$

$$f(V) = \frac{1}{1 + \exp(-bV)}$$

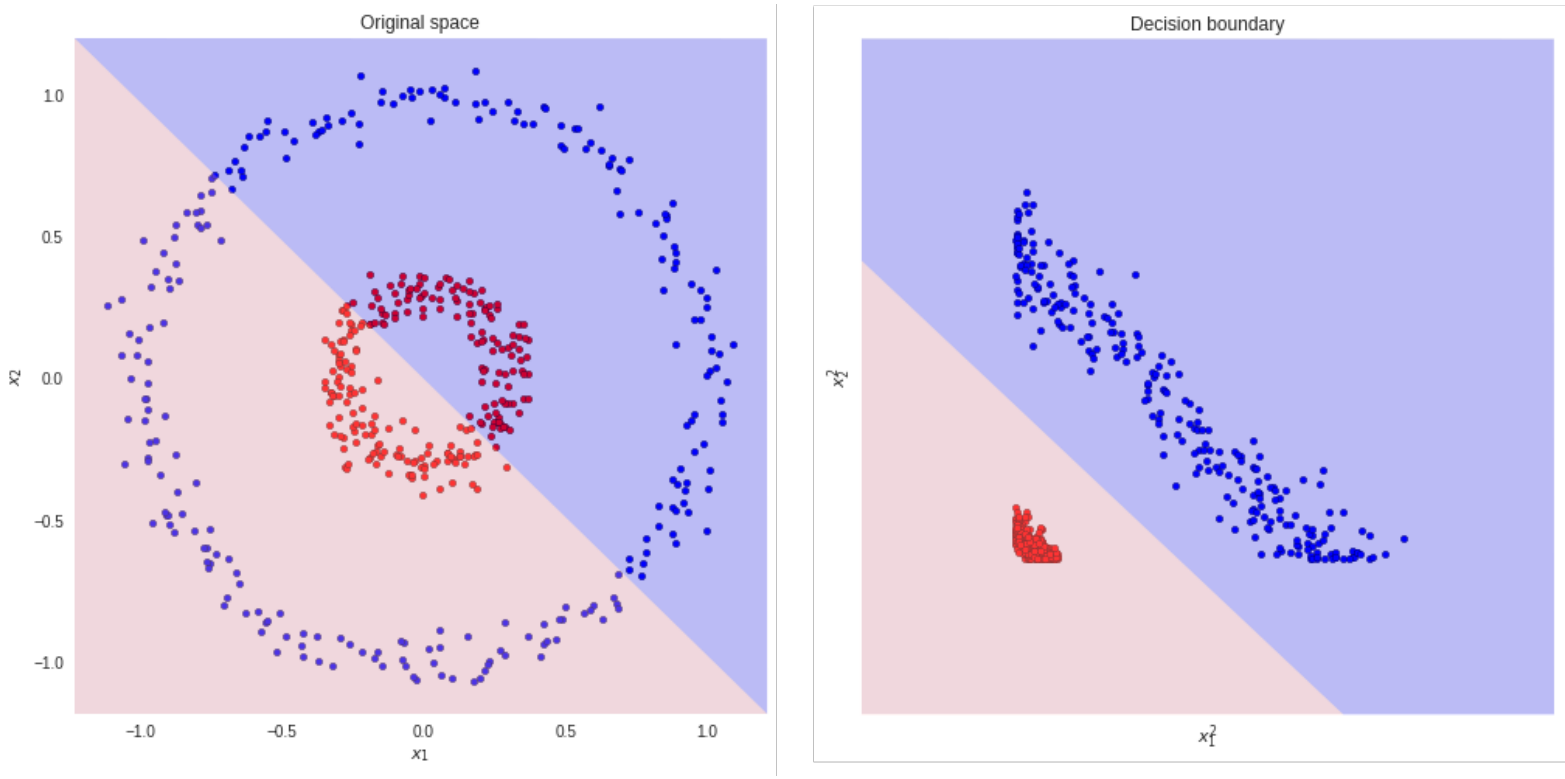
Однослойная нейронная сеть может успешно решить лишь задачу линейной сепарации. Поэтому возможности по моделированию нелинейных зависимостей у логистической регрессии отсутствуют.

Другой взгляд на классификацию

- В линейном случае мы хотим спроецировать точки в размерность 1 (на нормаль разделяющей гиперплоскости) так, чтобы в этой одномерной размерности они хорошо разделялись
- В этом смысле классификация – это метод радикального сокращения размерности.
- Рассмотрим классификацию с этих позиций и в каком-то смысле попробуем добиться оптимальности.

Проецирование исходных данных в новое пространство

Для решения задач классификации с двумя или более классами большинство алгоритмов машинного обучения применяют какое-то преобразование к входным данным с эффектом уменьшения исходных входных измерений до меньшего числа. Цель состоит в том, чтобы спроецировать данные в новое пространство. Затем, после проецирования, алгоритм пытается классифицировать точки путем нахождения линейного разделения.



Линейный дискриминант Фишера

Линейный дискриминантный анализ (ЛДА), а также связанный с ним линейный дискриминант Фишера — методы статистики и машинного обучения, применяемые для нахождения линейных комбинаций признаков, наилучшим образом разделяющих два или более класса объектов или событий. Полученная комбинация может быть использована в качестве линейного классификатора или для сокращения размерности пространства признаков перед последующей классификацией.

Линейный дискриминант Фишера в первоначальном значении - метод, определяющий расстояние между распределениями двух разных классов объектов или событий. Он может использоваться в задачах машинного обучения при статистическом (байесовском) подходе к решению задач классификации.

Предположим, что обучающая выборка удовлетворяет помимо базовых гипотез байесовского классификатора также следующим гипотезам:

- Классы распределены по нормальному закону
- Матрицы ковариаций классов равны

Тогда статистический подход приводит к линейному дискриминанту, и именно этот алгоритм классификации в настоящее время часто понимается под термином *линейный дискриминант Фишера*

Разделение на два класса на плоскости

Рассмотрим случай задачи классификации двух классов (**К = 2**). Синие и красные точки в $\mathbb{R}^2$. В общем виде, можно взять любой входной D-мерный вектор и спроецировать его на D’ измерение. В этом случае **D** представляет размерность исходного входного пространства, в то время как D’ - это новая размерность пространства.

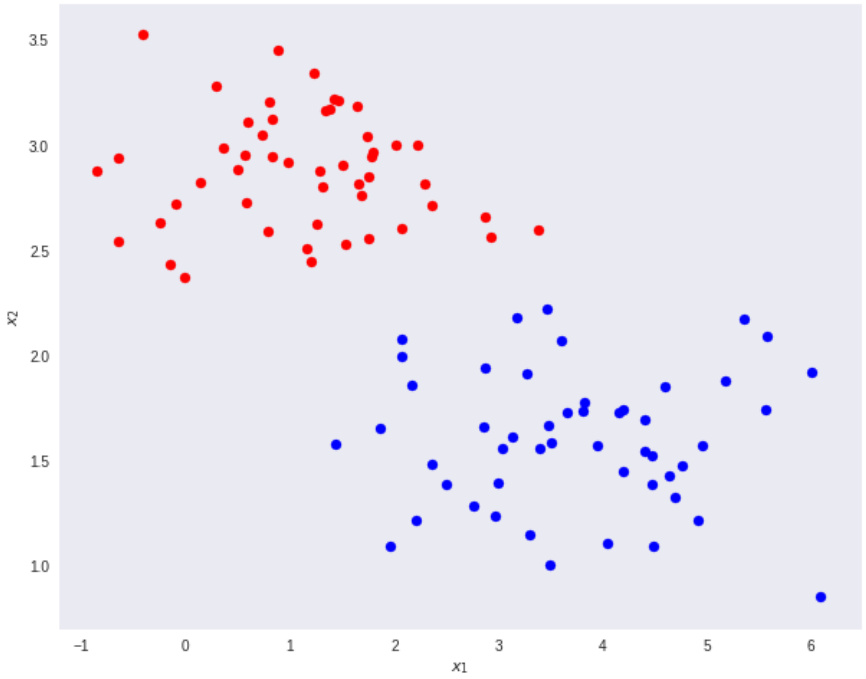
В случае проецирования на одномерное измерение (числовую линию), т.е. **D’=1** можно выбрать порог **T** для разделения классов в новом пространстве. Учитывая входной вектор **X**:

- если прогнозируемое значение $Y \geq T$ тогда **X** принадлежит к классу C1 (класс 1).
- в противном случае он классифицируется как C2 (класс 2).

Вектор **Y** (прогнозы) равен линейной комбинации входов **X** и весов **W.**

$$Y = W^T X$$

Хотим уменьшить исходные размеры данных от **D=2** в **D’=1**. Другими словами, получить преобразование, которое отображает векторы $\mathbb{R}^2 \rightarrow \mathbb{R}^1$.



Первая идея

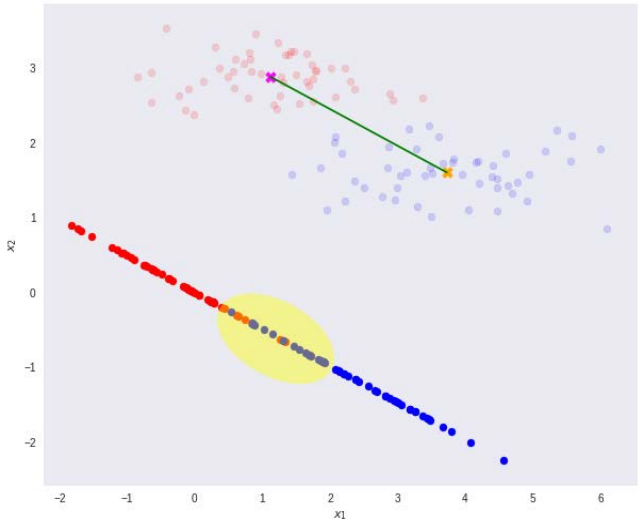
Рассмотрим два класса C_1 и C_2 с N_1 и N_2 точками.

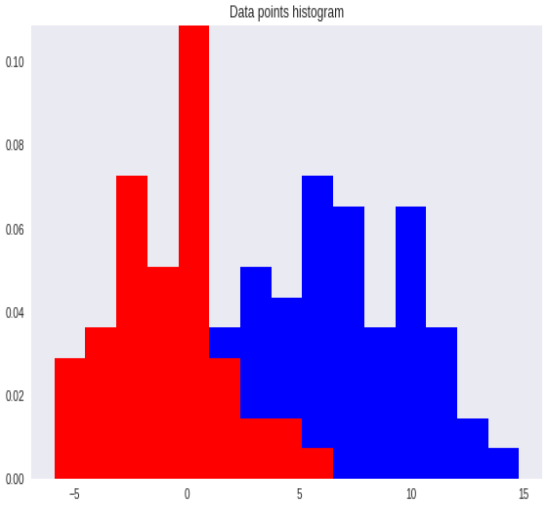
Первая идея – найти серединный перпендикуляр между центрами кластеров. Вычислим средние векторы m_1 и m_2.

$$m_1 = \frac{1}{N_1}\sum_{n\in C_1} x_n\,, \qquad m_2 = \frac{1}{N_2}\sum_{n\in C_2} x_n$$

Другими словами, хотим проецировать данные на вектор W, соединяющий центры кластеров.

Важно отметить, что любой вид проекции на меньшее измерение может повлечь некоторую потерю информации. В этом сценарии обратите внимание, что эти два класса четко разделимы (линией) в их исходном пространстве. После проецирования данные демонстрируют некоторое перекрытие классов - это показано желтым эллипсом на графике и гистограммой справа.





Идея, предложенная Фишером

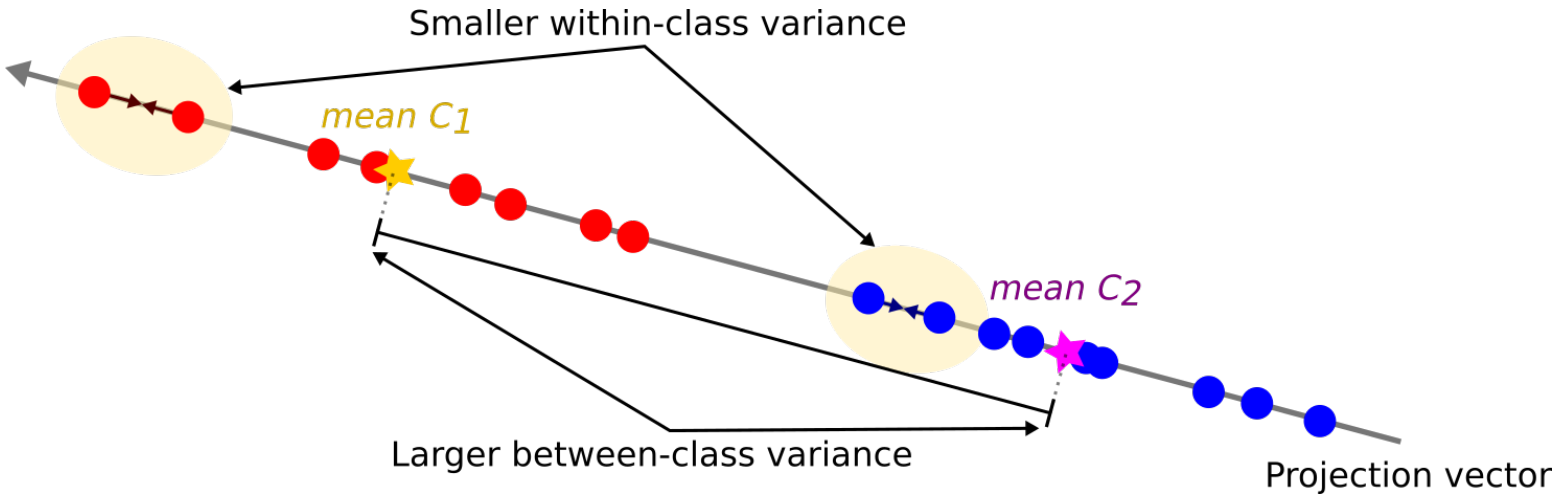
Максимизировать функцию, которая обеспечит наибольшее разделение между спроецированными центрами классов, а также даст наименьшую дисперсию внутри каждого класса, тем самым минимизируя перекрытие классов.

Другими словами, линейный дискриминант Фишера выбирает проекцию, которая максимизирует разделение классов. Для этого он максимизирует соотношение между дисперсией между классами и дисперсией внутри класса.

Для проецирования данных в меньшее измерение и во избежание перекрытия классов метод поддерживает 2 свойства.

- Большая разница между классами наборов данных.
- Небольшая дисперсия в каждом из классов наборов данных.

Обратите внимание, что большая разница между классами означает, что прогнозируемые средние классы должны быть как можно дальше друг от друга. Напротив, небольшая внутриклассовая дисперсия позволяет удерживать проецируемые точки данных ближе друг к другу в рамках одного класса.



Поиск проекции с указанными свойствами

Выборочная дисперсия в проекции $y_n = W^T x_n$

$$S_1^2 = \sum_{n \in C_1} (y_n - m_1)^2,\ S_2^2 = \sum_{n \in C_2} (y_n - m_2)^2$$

Чтобы найти проекцию с указанными свойствами, Линейный дискриминант Фишера вычисляет весовой вектор W по следующему критерию (критерий Фишера):

$$J(W) = \frac{(m_2 - m_1)^2}{S_1^2 + S_2^2} = \frac{\boldsymbol{W^T S_B W}}{\boldsymbol{W^T S_W W}}$$

Числитель – between-class variance (межклассовая дисперсия).

Знаменатель - within-class variance (внутриклассовая дисперсия).

$$S_B = (m_2 - m_1)(m_2 - m_1)^T$$

$$S_W = \sum_{n \in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n \in C_2} (x_n - m_2)(x_n - m_2)^T$$

После дифференцирования по W получим, что $J(W)$ максимален при:

$$(W^T S_B W) S_W W = (W^T S_W W) S_B W$$

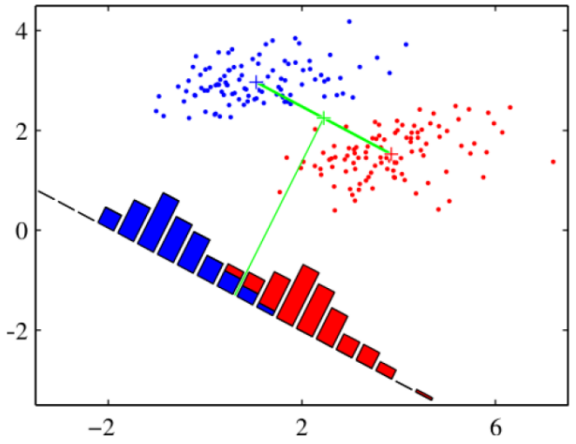
Причем $S_B W$ будет в направлении $m_2 - m_1$, а длина W нас не интересует.

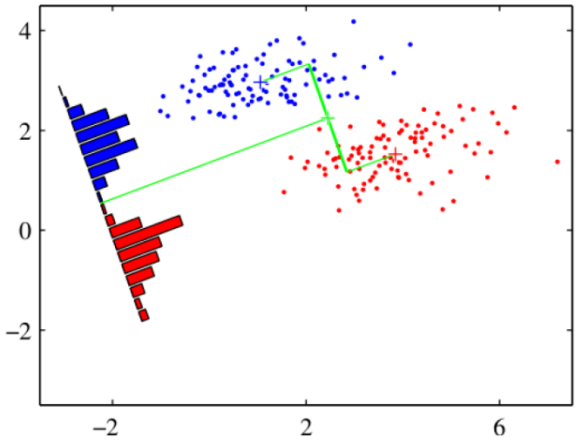
Получаем:

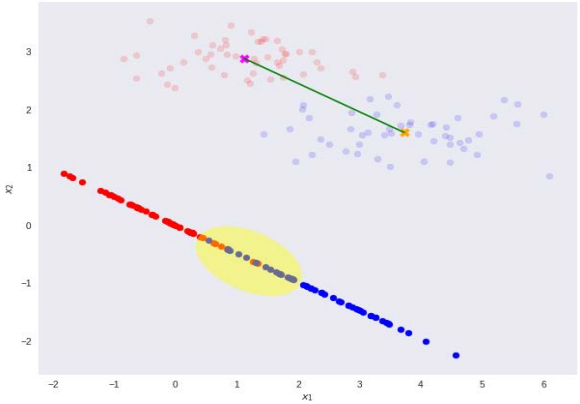
$$W \propto S_W^{-1}(m_2 - m_1)$$

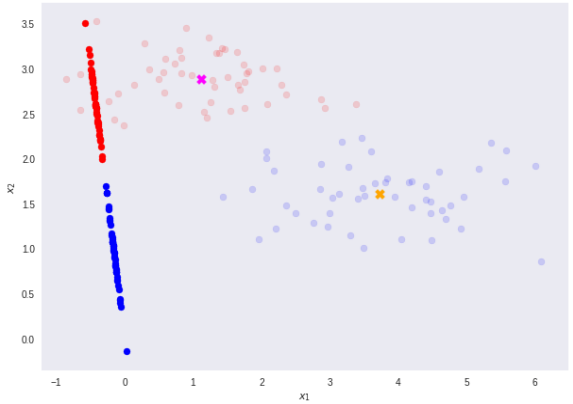
То есть, W (наше желаемое преобразование) прямо пропорционально обратной внутриклассовой ковариации. Матрица умножает на разность классов.

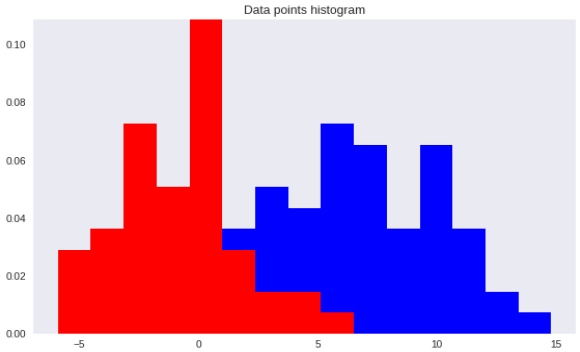
Геометрический смысл

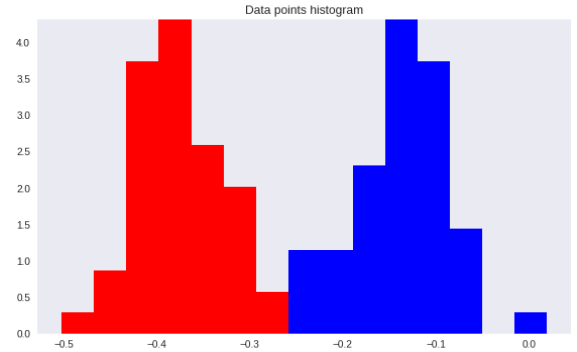




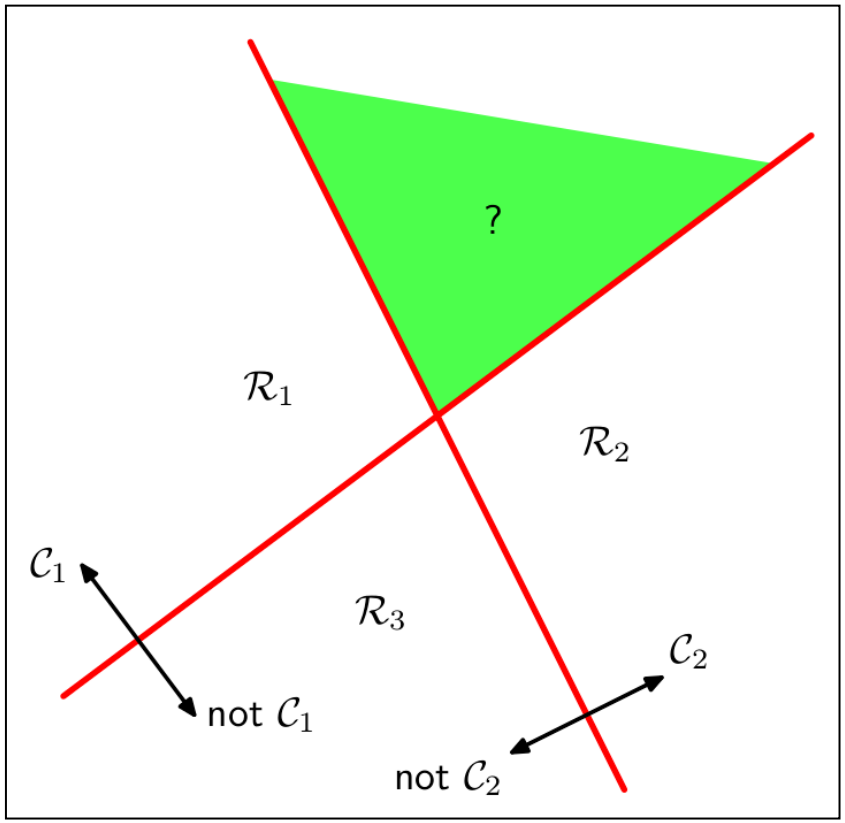


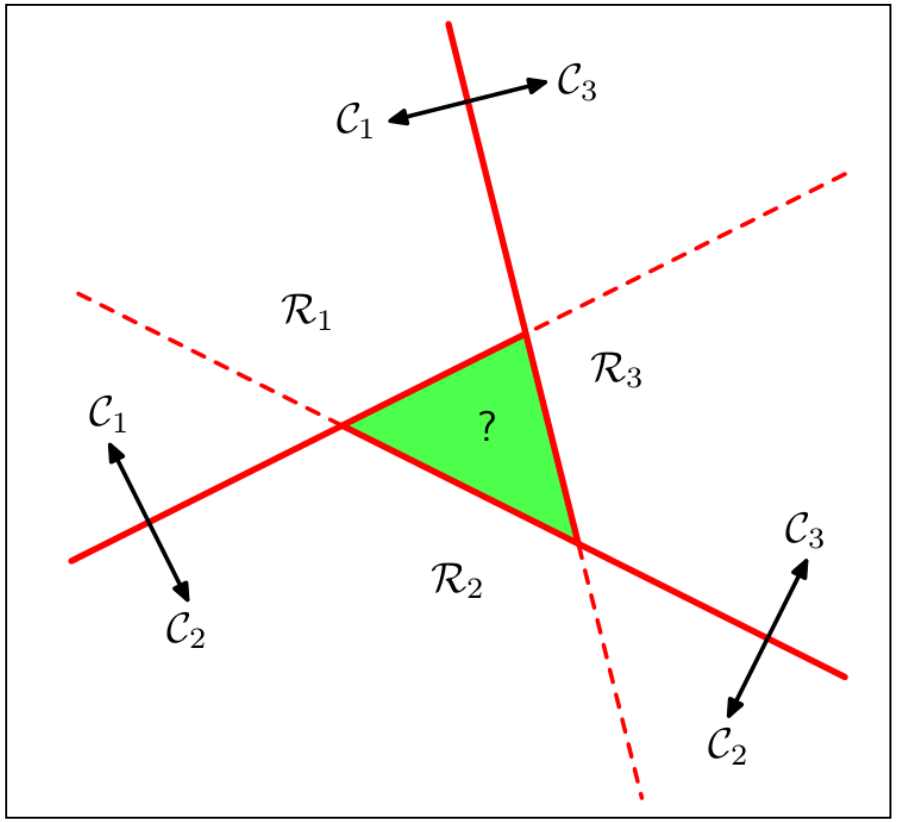






Линейное разделение и несколько классов





Метрики качества классификации
Матрица ошибок (Confusion Matrix)

Основной инструмент для сравнения моделей, показывает, сколько объектов каждого класса было верно и неверно классифицировано, позволяет определить, в какой степени алгоритм делает ошибки при классификации каждого класса.

В случае бинарной кластеризации, матрица ошибок состоит из четырех основных элементов:

- True Positives (TP): Количество объектов, которые были правильно классифицированы как положительные (верное предсказание положительного класса).
- False Positives (FP): Количество объектов, которые были неправильно классифицированы как положительные (ложное предсказание положительного класса), а ошибка классификации называется ошибкой I рода (**ложная тревога**).
- True Negatives (TN): Количество объектов, которые были правильно классифицированы как отрицательные (верное предсказание отрицательного класса).
- False Negatives (FN): Количество объектов, которые были неправильно классифицированы как отрицательные (ложное предсказание отрицательного класса), а ошибка классификации называется ошибкой II рода (**пропуск цели**).

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

Метрики качества классификации Точность, полнота, F-мера и т.д.

Точность (Precision): показывает, как много из объектов, предсказанных как положительные, действительно принадлежат положительному классу.

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

Полнота (Recall): показывает, как много из всех фактически положительных объектов было правильно обнаружено моделью.

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

Специфичность (Specificity): показывает, как много из объектов отрицательного класса было правильно классифицировано.

$$Specificity = \frac{TN}{TN + FP}$$

F-мера (F1-score): гармоническое среднее точности и полноты. Учитывает оба аспекта, что делает ее полезной в случаях, когда оба показателя важны.

$$\text{F1−score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Среднегармоническая F-мера (F_β): Для нивелирования перекоса в балансе классов используется коэффициент β.

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

Точность точности (accuracy): сравнение по доле правильно классифицированных экземпляров из общего числа экземпляров в выборке.

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

Метрики качества классификации ROC-AUC

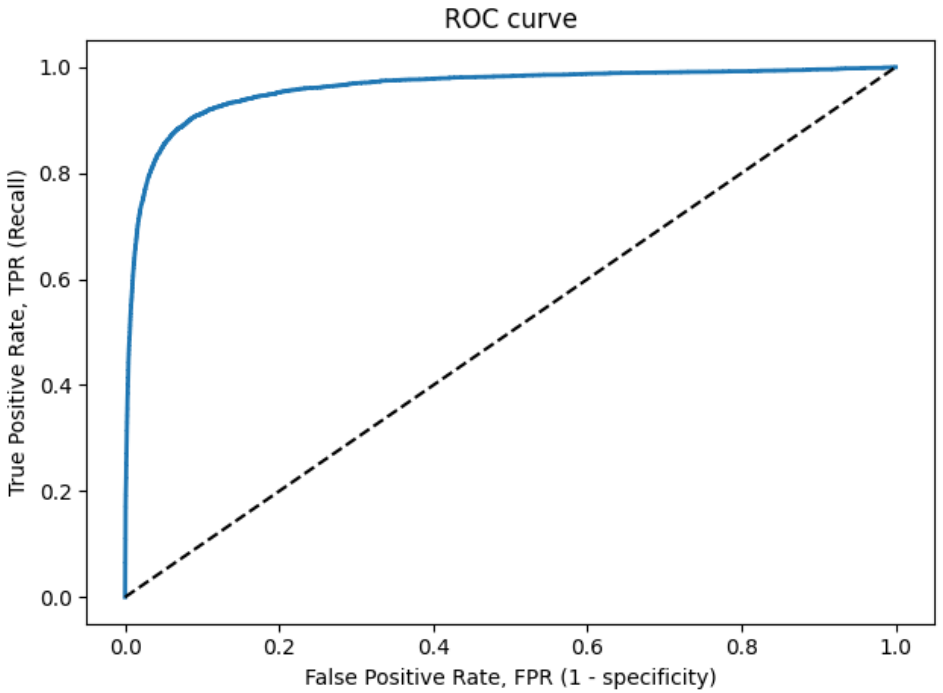
ROC — Receiver Operating Characteristic curve / AUC — Area Under Curve

ROC-кривая показывает зависимость между долей ложноположительных и долей истинно положительных классификаций. Данная оценка работы алгоритма классификации рассчитывается как площадь под ROC кривой, которая строится в координатах TPR (True Positive Rate) и FPR (False Positive Rate).

$$TPR = \frac{TP}{TP + FN}, \qquad FPR = \frac{FP}{FP + TN}.$$

Алгоритм построения кривой:

1. Запустить классификатор на тестовой выборке;
2. Отсортировать результаты по уверенности классификатора в принадлежности объекта к классу;
3. Пока не кончились элементы:
   - Взять объект с максимальной уверенностью;
   - Сравнить метку с реальной;
   - Пересчитать TPR и FPR на взятых объектах;
   - Поставить точку;
4. Построить кривую по точкам.



Метрики качества классификации
Precison-recall (PR) кривая

Задача: Необходимо выбрать 100 релевантных из 1 миллиона документов.

Алгоритм 1 возвращает 100 документов, 90 из которых релевантны. Таким образом,

$$TPR = \frac{TP}{TP + FN} = \frac{90}{90 + 10} = 0.9 \quad FPR = \frac{FP}{FP + TN} = \frac{10}{10 + 999890} = 0.00001$$

Алгоритм 2 возвращает 2000 документов, 90 из которых релевантны.

$$TPR = \frac{TP}{TP + FN} = \frac{90}{90 + 10} = 0.9 \quad FPR = \frac{FP}{FP + TN} = \frac{1910}{1910 + 997990} = 0.00191$$

Если рассматривать полноту и точность :

Алгоритм 1

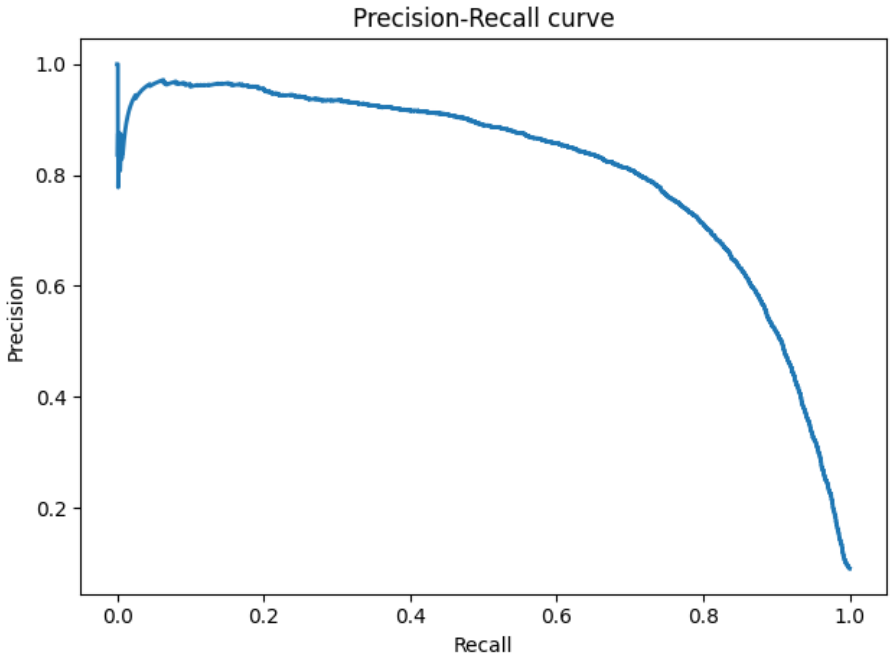
$$precision = \frac{TP}{TP + FP} = 90/(90 + 10) = 0.9$$

$$recall = \frac{TP}{TP + FN} = 90/(90 + 10) = 0.9$$

Алгоритм 2

$$precision = \frac{TP}{TP + FP} = \frac{90}{90 + 1910} = 0.045$$

$$recall = \frac{TP}{TP + FN} = \frac{90}{90 + 10} = 0.9$$



Спасибо за внимание