Машинное обучение в экономике

Логистическая регрессия и метод опорных векторов

Потанин Богдан Станиславович

доцент, научный сотрудник, кандидат экономических наук

2023–2024

- Методы классификации:
  - Логистическая регрессия.
  - Метод опорных векторов.
  - Мультиномиальная логистическая регрессия.
- Базовые понятия:
  - Численная оптимизация: методы лкоальной и глобальной оптимизации, градиентный спуск, скорость обучения, условия остановки, линейный поиск с возвратом.
  - Модели и функция потерь.
  - Градиентный бустинг.
  - Опорные вектора и отступ.

- Предположим, что условные вероятности могут быть оценены как линейные комбинации признаков (регрессоров):

$$P(Y_i = 1|X_i = x_i) = g\left(X_i\beta\right)$$

- Коэффициенты β часто называют **весами**, а функция $g()$ принимает значения от 0 до 1, поскольку отражает условные вероятности.
- В качестве функции $g()$ удобно взять функцию распределения некоторого распределения с носителем на R. Наиболее популярным в машинном обучении является логистическое распределение, при котором мы получаем **логит модель**:

$$g(t) = \frac{1}{1 + e^{-t}}$$

- В эконометрике не менее популярной является функция распределения стандартного нормального распределения $g(t) = \Phi(t)$, при которой мы получаем **пробит модель**.

- Для оценивания условных вероятностей достаточно оценить параметры β методом максимального правдоподобия:

$$L(\beta; X, Y) = \prod_{i:y_i=1} P(Y_i = 1|X_i) \prod_{i:y_i=0} P(Y_i = 0|X_i) = \prod_{i:y_i=1} g(X_i\beta) \prod_{i:y_i=0} 1 - g(X_i\beta) =$$

$$\prod_{i:y_i=1} \frac{1}{1 + e^{-X_i\beta}} \prod_{i:y_i=0} 1 - \frac{1}{1 + e^{-X_i\beta}}$$

$$\ln L(\beta; X, Y) = \sum_{i=1}^{n} -\ln(1 + e^{-X_i\beta}) - (1 - Y_i)X_i\beta$$

- Можно показать, что логарифм функции правдоподобия является вогнутой функцией по β при любых x_i, а значит ее максимум является единственным.
- В отличие от линейного МНК, в данном случае не существует аналитического выражения для $\hat{\beta}$, что мотивирует **максимизацию численными методами**.

Численная оптимизация позволяет находить приблизительный максимум или минимум функции без необходимости искать аналитическое решение.

- Методы **локальной** оптимизации (BFGS, градиентный спуск) как правило работают достаточно быстро, но позволяют находить лишь локальные экстремумы. Методы **глобальной** оптимизации (генетический алгоритм, метод отжига – SA) позволяют найти несколько экстремумов, один из которых может оказаться глобальным. Однако, глобальная оптимизация обычно крайне затратна по времени.
- Методы локальной оптимизации часто опираются на Градиент (градиентный спуск, ADAM) или Гессиан функции (BFGS, BHHH). В последнем случае число итераций алгоритма, как правило, оказывается меньше, но время каждой итерации – больше, особенно, при большом числе оцениваемых параметров.

Поскольку число оцениваемых параметров в эконометрических моделях, как правило, относительно невелико (в сравнении с моделями машинного обучения), то чаще используются алгоритмы, использующие информацию о Гессиане (BFGS, BHHH).

Алгоритм **градиентного спуска** является одним из простейших численных методов нахождения минимума функции.

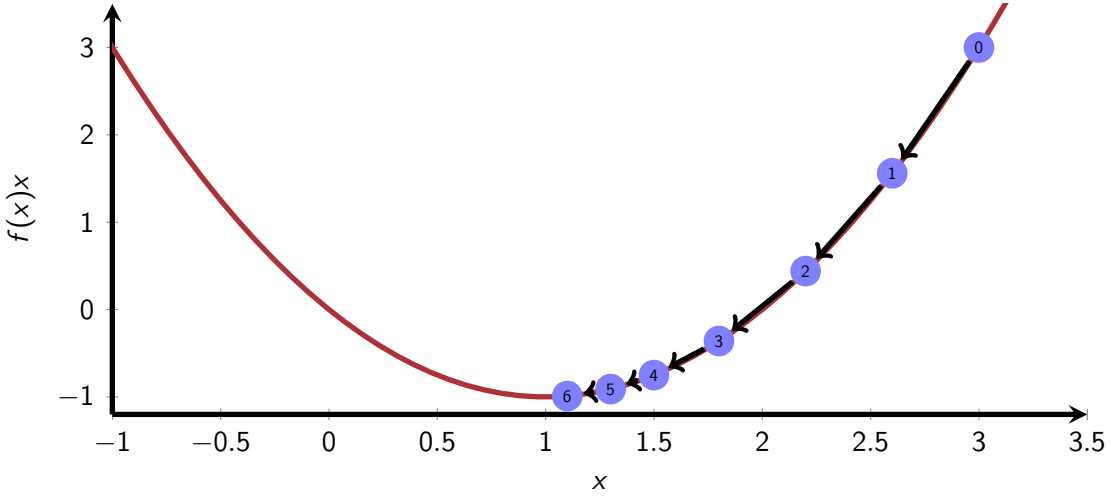
- Выбираем произвольную начальную точку x_0.
- Считаем градиент функции в этой точке $\nabla f(x_0)$.
- Переходим в новую точку $x_1 = x_0 - \alpha \nabla f(x_0)$, где α – малая положительная константа, регулирующая **скорость обучения** (learning rate).
- Повторяем процедуру до тех пор, пока не будут соблюдены **условия остановки** (termination conditions), например, о том, что $|\nabla f(x_0)| < \varepsilon$, где ε – маленькое положительное число.

Нетрудно показать аналитически, что функция $f(x) = x^2 - 2x$ достигает минимума в точке $x^* = 1$. В качестве альтернативы аналитическому решению попробуем приблизиться к минимуму с помощью 10 итераций описанного алгоритма, произвольным образом полагая $x_0 = 3$ и $\alpha = 0.2$.

i	0	1	2	3	4	5	6	7	8	9	10
x_i	3	2.20	1.72	1.43	1.26	1.16	1.09	1.06	1.03	1.02	**1.01**
$\nabla f(x_i)$	4	2.40	1.44	0.86	0.52	0.31	0.19	0.11	0.07	0.04	0.02
$f(x_i)$	3	0.44	-0.48	-0.81	-0.93	-0.98	-0.99	-1.00	-1.00	-1.00	-1.00

Численная оптимизация

Графическая иллюстрация одномерной локальной численной оптимизации



- Рассмотрим функцию $f(x)$ в точке x^*.
- Алгоритм градиентного спуска возможет благодаря тому, что при соблюдении некоторых условий регулярности, включая $\nabla f(x^*) \neq 0$, существует такая константа $\tau > 0$ (для удобства предположим что существует наибольшая из таких констант и будем рассматривать ее), что для любого τ^*, такого, что $0 < \tau^* < \tau$, соблюдается:

$$f(x^* - \tau \nabla f(x^*)) < f(x^*)$$

- **Важная интуиция** – обычно, чем ближе x^* к точке минимума, тем меньше τ.
- **Проблема** – по мере приближения к минимуму возрастает риск использования слишком большой скорости обучения $\alpha > \tau$. Если в алгоритме градиентного спуска скорость обучения является постоянной, то в какой-то момент алгоритм может не уменьшить, а увеличить значение функции.
- **Популярное решение** - воспользоваться **линейным поиском с возвратом**, основная идея которого заключается в том, чтобы каждую итерацию алгоритма уменьшать скорость обучения α в $\beta > 1$ раз, до тех пор, пока не будут соблюдены некоторые условия, в том числе гарантиерующие, что в новой точке значение функции окажется меньше, чем в изначальной.

- **Проблема** – на первый взгляд линейная форма условных вероятностей $X_i\beta$ снижает гибкость модели.
- **Решение** – можно добавить нелинейность в модель, например, взяв не только сами признаки, но и некоторые нелинейные функции от них.
- Например, вместо возраста age_i можно взять его полином третьей степени просто добавив в число признаков age_i^2 и age_i^3.
- Другой пример: чтобы учесть взаимодействие между возрастом age_i и доходом income_i можно добавить в число признаков их произведение $\text{age}_i \times \text{income}_i$.
- Оптимальная спецификация логистической регрессии может быть подобрана, например, с помощью кросс-валидации.

Мультиномиальная логистическая регрессия

Определение

- Мультиномиальная логистическая регрессия используется в случае, когда необходимо предсказать значение одной из K взаимоисключающих альтернатив.
- Например, необходимо спрогнозировать, какое мороженое предпочтет индивид: шоколадное, ванильное или эскимо. В данном случае $K = 3$.
- Влияние признаков x_i на полезность (склонность к выбору) k-й альтернативы описывается как:

$$u(k, x_i) = X_i\beta_k + \varepsilon_{ki}, \qquad \varepsilon_{ki} \sim \text{extreme error value distribution, i.i.d.}$$

- Индивид выбирает альтернативу, приносящую ему наибольшую полезность.
- Можно показать, что условные вероятности выбора k-й альтернативы записываются при помощи **softmax** функции:

$$P(Y_i = k|X_i = x_i) = e^{-x_i\beta_k} / \sum_{j=1}^{K} e^{-x_j\beta_j} \qquad k \in \{1, ..., K\}, \beta_1 = (0, ...0).$$

- Параметры β_j оцениваются с помощью метода максимального правдоподобия.

Функция потерь

Основная идея

- Одним из общих подходов к спецификации моделей в машинном обучении является использование **функций потерь**, обозначаемых по аналогии с функцией правдоподобия (важно их не путать):

$$\mathsf{L}(Y, F(X)) = \frac{1}{n}\sum_{i=1}^{n}\mathsf{L}(Y_i, F(X_i))$$

- Функции $F(X_i)$ обычно именуют **моделями** и они отражают либо прогноз Y_i, либо прогноз его некоторой характерстики, например, вероятности $P(Y_i = 1|X_i)$.
- Исследователь стремится найти модель $F(X)$ в классе K_F (например, все случайные леса), **минимизирующую функцию потерь**.
- Часто для удобства модели обозначают как $F(X_i; \theta)$, где θ отражает вектор параметров. В таком случае класс определяется всеми допустимыми значениями θ.
- Рассмотрим **линейную модель** $F(X; \theta) = X\theta$, непрерывную целевую переменную Y_i и **квадратичную функцию потерь**:

$$\mathsf{L}(Y, F(X; \theta)) = \sum_{i=1}^{n}(Y_i - F(X_i; \theta))^2 = \sum_{i=1}^{n}(Y_i - X\theta)^2$$

- Очевидно, что данная функция потерь минимизируется при $\theta = \hat{\theta}$, где $\hat{\theta}$ – МНК оценка, то есть $\hat{\theta} = \left(XX^T\right)^{-1}XY$. Таким образом, оптимальной будет модель $F(X_i; \hat{\theta}) = X\left(XX^T\right)^{-1}XY$.

- Рассмотрим модель $F(X_i;\theta) = 1/\left(1+e^{-X_i\theta}\right)$, бинарную целевую переменную Y_i и **логистическую функцию потерь**:

$$\mathrm{L}\left(Y, F(X_i;\theta)\right) = \sum_{i=1}^{n} -Y_i \ln\left(F(X_i;\theta)\right) - (1-Y_i)\ln\left(1-F(X_i;\theta)\right) =$$

$$= \sum_{i=1}^{n} -Y_i \ln\left(1/\left(1+e^{-X_i\theta}\right)\right) - (1-Y_i)\ln\left(1-1/\left(1+e^{-X_i\theta}\right)\right) =$$

$$= \sum_{i:y_i=1} \ln(1+e^{-X_i\theta}) + \sum_{i:y_i=0} X_i\theta + \ln(1+e^{-X_i\theta}) = \sum_{i=1}^{n} \ln(1+e^{-X_i\beta}) + (1-Y_i)X_i\beta$$

- Обратим внимание, что мы получили логистическую регрессию, поскольку логистическая функция потерь при данной модели совпадает, с противоположным знаком, с функцией правдоподобия логистической регрессии.
- Логистическую регрессию можно получить и с помощью иных комбинаций моделей и функции потерь. В частности (проверьте самостоятельно), с помощью нередко встречающегося в литературе альтернативного определения логистической функции потерь:

$$\mathrm{L}\left(Y, F(X_i;\theta)\right) = \sum_{i=1}^{n} \ln\left(1+e^{(2Y_i-1)F(X_i;\theta)}\right) \quad F(X_i;\theta) = \ln\left(\frac{P\left(Y_i=1|X_i\right)}{1-P\left(Y_i=1|X_i\right)}\right) \quad P\left(Y_i=1|X_i\right) = \frac{1}{1+e^{-X_i\theta}}$$

Функция потерь

Подбор гиперпараметров

- Ранее мы обсуждали, что для подбора гиперпараметров могут применяться различные метрики качества прогнозов, такие как AUC, $F1$-метрика и прибыль от прогнозов.
- В качестве альтернативы можно подбирать гиперпараметр исходя из минимизации функции потерь. Кроме того, иногда удобно рассматривать преобразованную метрику, например, умноженную на минус единицу прибыль от прогноза, как функцию потерь.
- Оптимальная функция потерь подбирается исходя из двух **ключевых соображений**:
  - Удобство численной оптимизации.
  - Связь с показателем, непосредственно интересующим исследователя или бизнес.
- Например, если в качестве функции потерь использовать умноженную на минус единицу прибыль, то содержательный смысл ее минимизации очевиден. Однако оптимизационная задача может оказаться весьма сложной и ее решение потребует больших вычислительных мощностей, что может мотивировать исследователя перейти к менее содержательно обоснованной, но более удобной с точки зрения оптимизации логистической функции потерь.
- Иногда модель обучается с использованием одной функции потерь, а гиперпараметры подбираются с помощью другой. Например, в задаче классификации поиск весов (коэффициентов) признаков часто удобно осуществлять с помощью логистической функции потерь. Однако, подбор гиперпараметров, таких как оптимальная спецификация регрессоров или порог прогнозирования, можно осуществить ориентируясь на максимизацию прибыли.
- Иногда различные группы гиперпараметров может быть удобно подбирать с помощью разных функций потерь.

- Представим, что у нас уже есть модель $F_M(X)$ и мы хотим ее **улучшить** (boost) за счет добавления модели $h_{M+1}(X)$ из класса H_{M+1} (например, все регрессионные деревья) с весом γ_{M+1}. В результате получаем:

$$F_{M+1}(X) = F_M(X) + \gamma_{M+1} h_{M+1}(X)$$

- В **идеальном случае** мы хотели бы найти функцию и вес, минимизирующие функцию потерь:

$$(h_{M+1}(X), \gamma_{M+1}) = \underset{(h^*_{M+1}(X) \in H_{M+1}, \gamma^*_{M+1} \in R)}{\operatorname{argmin}} L(Y, F_M(X) + \gamma^*_{M+1} h^*_{M+1}(X))$$

- **Проблема** – решение данной оптимизационной задачи крайне затруднительно на практике.
- **Решение** – вместо того, чтобы полностью решать данную задачу, можно, по аналогии с градиентным спуском, попытаться найти $h_{M+1}(X)$ и γ_{M+1}, которые пусть и не минимизируют, но по крайней мере уменьшают функцию потерь.

- Предполагая некоторые условия регулярности по поводу функции потерь (по аналогии с градиентным спуском) можно гарантировать существование такой γ_{M+1}, что:

$$L\left(Y, F_M(X) - \gamma_{M+1}\frac{dL(Y, F_M(X))}{dF_M(X)}\right) < L(Y, F_M(X))$$

- **Идея** – чтобы улучшить качество модели $F_M(X)$, можно взять любую γ_{M+1}, при которой соблюдается соответствующее неравенство, и в качестве функции $h_{M+1}(X)$ положить $-\frac{dL(Y,F_M(X))}{dF_M(X)}$. Однако, воплощению этой идеи препятствует серьезная проблема.
- **Проблема** – отрицательный градиент не является моделью, принадлежащей к классу H_{M+1}, поскольку зависит не только от X, но и от Y. Также, из-за этого модель $h_{M+1}(X)$, а значит и модель $F_{M+1}(X)$, нельзя применять для прогнозирования на новых данных, в которых значения целевой переменной Y неизвестны.
- **Решение** – получить $h_{M+1}(X)$, обучив модель из класса H_{M+1} прогнозировать отрицательный градиент $-\frac{dL(Y,F_M(X))}{dF_M(X)}$ с помощью X. Тогда значения Y обучающей выборки будут использованы лишь для оценивания функции $h_{M+1}(X)$.

Градиентный бустинг

Описание алгоритма

- **Первый шаг** – выбирается базовая функция, обычно в форме константы:

$$F_0(X) = \underset{c \in R}{\operatorname{argmin}} \; L(Y, c)$$

  Обычно решением этой задачи является среднее $c = \overline{Y}$.
- *Примечание* – в самом простом случае при любых M, где M – номер итерации алгоритма, параметры γ_M предполагаются равными одной и той же (обычно маленькой) константе $\gamma_M = \gamma$. Тогда γ можно рассматривать как гиперпараметр, который можно подобрать с помощью кросс-валидации.
- **Второй шаг** – Рассчитывается отрицательный градиент $r_1 = -\frac{dL(Y, F_M(X))}{dF_M(X)}$, именуемый **остатком**.
- **Третий шаг** – определяется класс рассматриваемых моделей H_1. Обычно используется один и тот же класс для всех итераций алгоритма $H_M = H$. В качестве модели $h_1(X)$ выбирается модель из класса H, обученная прогнозировать остатки r_1 с помощью признаков X.
- *Примечание* – в качестве функций $h_M(X)$ обычно рассматриваются регрессионные деревья (даже в классификационной задаче, поскольку они аппроксимируют остатки, являющиеся непрерывными переменными) небольшой глубины, как правило не превышающей 10.
- **Четвертый шаг** – формируется новая модель:

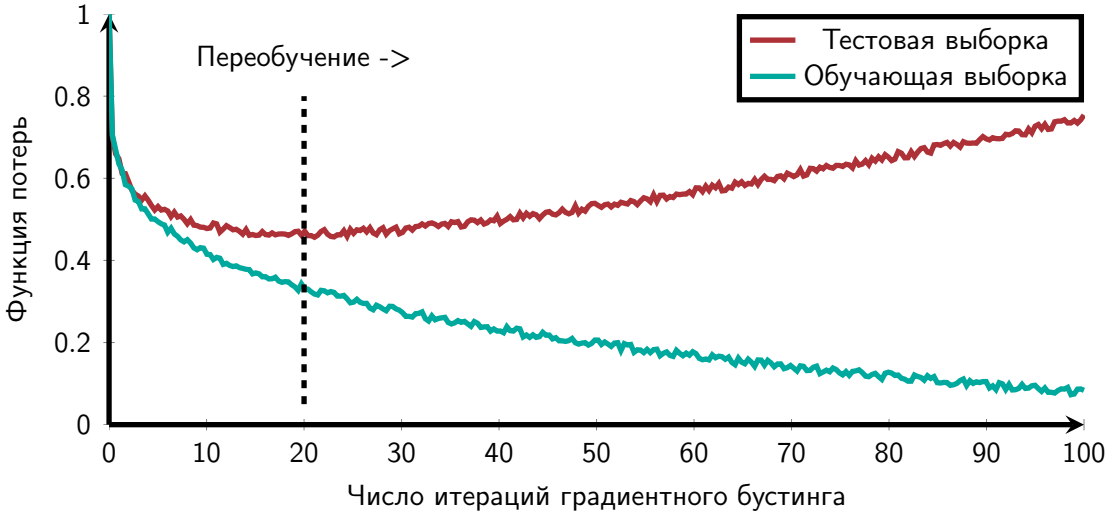
$$F_1(X) = F_0(X) + \gamma h_1(X)$$

- **Пятый шаг** – алгоритм повторяется (совершает очередную итерацию со второго шага) до соблюдения критерия остановки, например, фиксированное число раз или до тех пор, пока при очередной итерация функции потерь не возрастет. Последнее обычно происходит вследствие неподходящего значения γ, недостаточно точной аппроксимации остатков r_M функцией $h_M(X)$ или достижения локального максимума.

- В качестве базовой функции можно взять и достаточно сложную модель, которую вы намереваетесь улучшить. Базовая модель $F_0(X)$ не обязательно должна быть быть похожа на модели для аппроксимации градиента $h_M(X)$
- При слишком большом числе итераций алгоритма модель склонна к переобучению. Поэтому желательно проверять, например, графически, нельзя ли повысить точность модели на тестовой выборке за счет уменьшения числа итераций алгоритма. На практике можно взять большое число итераций алгоритма и затем оценить качество прогнозов на тестовой выборке поочередно каждый раз исключая последнюю из оцененных моделей.

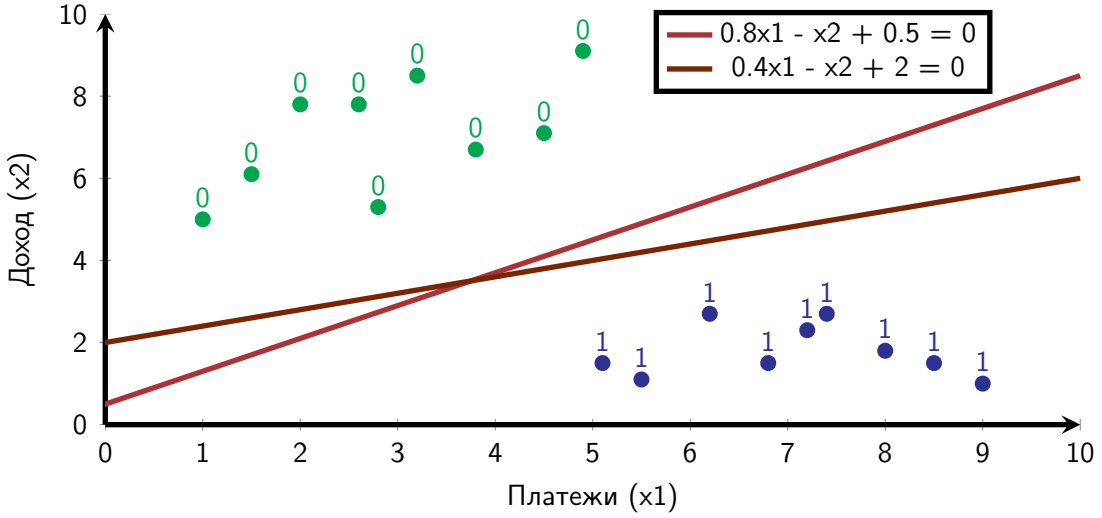
Градиентный бустинг

Графическая иллюстрация проблемы переобучения



Метод опорных векторов

Графическая иллюстрация идеи на примере дефолта



Метод опорных векторов SVM

Случай с наличием разделяющей гиперплоскости

- **Предположение** – существуют линии, которые можгут полностью разграничить два класса в зависимости от значений признаков.
- **Проблема** – какую из линий выбрать, если их бесконечно много?
- **Решение** – выбираем линию таким образом, чтобы максимизировать **перпендикулярное** расстояние от нее до ближайшего наблюдения. Этой расстояние именуется **отступом** (margin).
- После того, как мы выбрали линию, наблюдения, имеющие наименьшее перпендикулярное расстояние до этой линии, именуются **опорными векторами**.
- Разделяющая линия задается уравнением $\beta_0 + x\beta = 0$, где β и β_0 подбираются из соображений минимизации отступа.
- Следуя традиции и для удобства класс 0 будем обозначать как -1.

- Рассмотрим β_0 и β такие, что $\beta_0 + X_i\beta = 0$ задает разделяющую линию с максимальным отступом.
- Определим классификатор следующим образом:

$$\hat{y}(x) = \begin{cases} 1, \text{ если } \beta_0 + X_i\beta \geq c_1 \text{ (точки над отступом разделяющей линии)} \\ -1, \text{ если } \beta_0 + X_i\beta \leq -(c_{-1}) \text{ (точки под отступом разделяющей линии)} \end{cases}$$

- Если $c_1 > c_{-1}$, то геометрически очевидно, что отступ не является максимальным, поскольку сдвинув линию вниз мы сможем его увеличить. По аналогии невозможен случай $c_1 < c_{-1}$, а значит $c_1 = c_{-1}$.
- Поскольку умножении равенства $\beta_0 + X_i\beta = 0$ на константу его не изменяет, то без потери общности можно положить $c_1 = c_{-1} = 1$, откуда получаем классификатор:

$$\hat{y}(x) = \begin{cases} 1, \text{ если } \beta_0 + X_i\beta \geq 1 \\ -1, \text{ если } \beta_0 + X_i\beta \leq -1 \end{cases}$$

- Определим перпендикулярное расстояние от наблюдения X_i до линии $\beta_0 + X_i\beta = 0$:

$$d(X_i; \beta_0, \beta) = \frac{|\beta_0 + X_i\beta|}{\sqrt{\beta_1^2 + ... + \beta_m^2}} = \frac{|\beta_0 + X_i\beta|}{||\beta||}$$

- Обозначим через q произвольный опорный вектор, то есть наблюдение, находящееся на расстоянии отступа от разграничивающей линии $\beta_0 + X_i\beta = 0$.
- Из введенного ранее определения классификатора следует, что $|\beta_0 + \beta q| = 1$, а значит $d(q, \beta_0, \beta_1) = 1/||\beta||$.
- Поскольку отступ определяется через опорный вектор q, то задача максимизации отступа может быть сведена к задаче максимизации $d(q, \beta_0, \beta_1)$, что эквивалентно минимизации $||\beta||$.
- При решении этой задачи важно гарантировать, что найденные β_0 и β соответствуют линии, являющейся разграничивающей линией, то есть наблюдения различных классов должны лежать по разные стороны от нее:

$$\begin{cases} \beta_0 + X_i\beta \geq 1, \text{ если } Y_i = 1 \\ \beta_0 + X_i\beta \leq -1, \text{ если } Y_i = -1 \end{cases} \iff Y_i(\beta_0 + X_i\beta) \geq 1$$

- Таким образом, для удобства избавляясь от квадратного корня (строго возрастающая функция) задачу максимизации отступа можно сформулировать как следующую задачу условной минимизации (**квадратичное программировнаие**):

$$(\hat{\beta}_0, \hat{\beta}) = \underset{(\beta_0,\beta)}{\operatorname{argmin}} \ \beta_1^2 + ... + \beta_m^2 \quad \text{при ограничении} \quad Y_i \left(\beta_0 + X_i\beta\right) \geq 1$$

- Эта оптимизационная задача не имеет аналитического решения, однако минимум может быть найден численными методами.
- Напомним, что при этом классификатор определяется следующим образом:

$$\hat{y}(x) = \begin{cases} 1, \text{ если } \beta_0 + X_i\beta \geq 1 \\ -1, \text{ если } \beta_0 + X_i\beta \leq -1 \end{cases}$$

- Очевидно, что на практике разделяющая линия существует очень редко, а значит имеется такое наблюдение X_i, что при любых β_0 и β_1 неравенство $Y_i(\beta_0 + X_i\beta) \geq 1$ не соблюдается.
- В таком случае необхоидмо допустить возможность нарушения абсолютного разграничения, то есть наблюдения из класса 1 могут попадать в область наблюдений -1 и наоборот.
- Тогда оптимизационную задачу можно привести к виду:

$$\underset{(\beta_0,\beta,\xi_1,\ldots,\xi_n)}{\operatorname{argmin}} \sum_{j=1}^{m} \beta_j^2 + C\sum_{i=1}^{n} \xi_i, \text{ при ограничении } Y_i(\beta_0 + X_i\beta) \geq 1 - \xi_i$$

- Константа C является гиперпараметром, который определяет вес штрафов ξ_i в оптимизационной задаче и подбирается с помощью кросс-валидации.