Машинное обучение в экономике

Логистическая регрессия и метод опорных векторов

Потанин Богдан Станиславович

доцент, кандидат экономических наук

2024–2025

- Методы классификации:
  - Логистическая регрессия.
  - Метод опорных векторов.
- Базовые понятия:
  - Численная оптимизация: методы локальной и глобальной оптимизации, градиентный спуск, скорость обучения, условия остановки, линейный поиск с возвратом, квази-Ньютоновские методы.
  - Функция потерь и модель.
  - Градиентный бустинг.
  - Опорные векторы и отступ.

- Предположим, что условные вероятности могут быть оценены как функции от линейных комбинаций признаков:

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

- Коэффициенты β часто называют **весами**, а функция $g()$ принимает значения от 0 до 1, то есть трансформирует **линейный предиктор** $X_i\beta$ в 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 (почти) 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 значительном числе оцениваемых параметров.

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

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

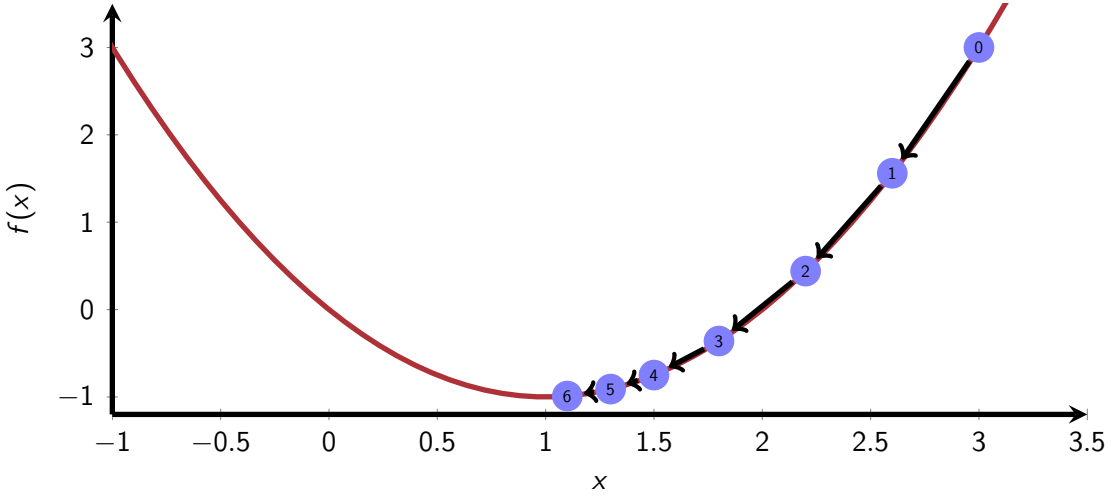
- Выбираем произвольную начальную точку x_0.
- Считаем градиент функции в этой точке $\nabla f(x_0)$.
- Переходим в новую точку $x_1 = x_0 - \alpha \nabla f(x_0)$, где α – малая положительная константа, регулирующая **скорость обучения** (learning rate).
- Повторяем процедуру до тех пор, пока не будут 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

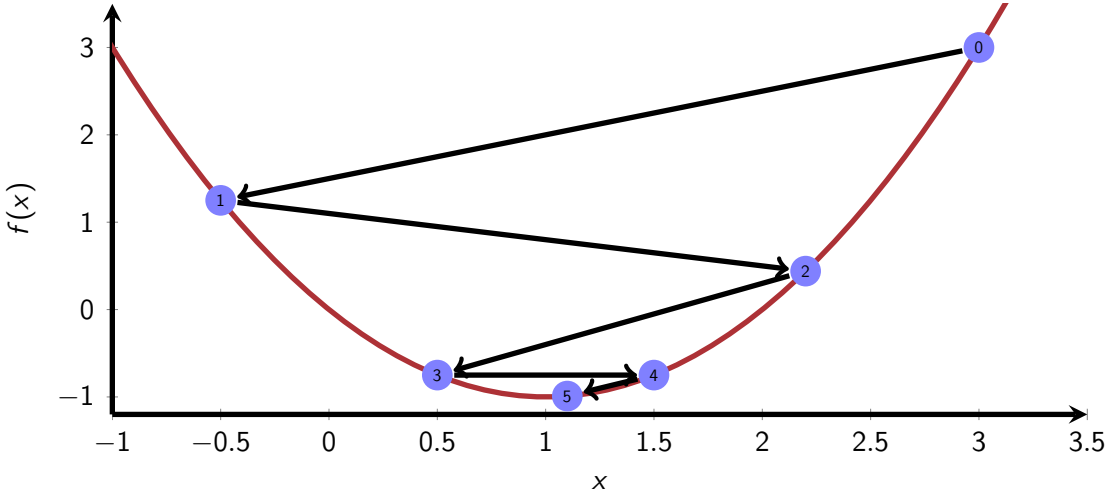
Численная оптимизация

Графическая иллюстрация одномерной локальной численной оптимизации



Численная оптимизация

Графическая иллюстрация градиентного спуска с большой скоростью обучения



- Рассмотрим функцию $f(x)$ в точке x^*.
- Алгоритм градиентного спуска работает 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 любой положительной константы τ^*, такой, что $0 < \tau^* \leq \tau$, соблюдается:

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

- **Проблема** – величина τ зависит от точки x^*, поэтому, при фиксированной скорости обучения α если при некотором x^* окажется, что $\alpha > \tau$, то вследствие очередной итерации градиенного спуска значение функции может не уменьшиться, а увеличиться.
- **Популярное решение** - воспользоваться **линейным поиском с возвратом**, при котором каждую итерацию градиентного спуска:
  1. Полагается $\alpha = \alpha_0$, где $\alpha_0 > 0$ отвечает за начальную скорость обучения.
  2. До тех пор, пока не соблюдены некоторые условия, в том числе гарантирующие $\alpha_j \leq \tau$, скорость обучения снижается в некоторое число раз $\alpha_j = \gamma \alpha_{j-1} = \gamma^j \alpha_0$, где $\gamma \in (0, 1)$.
  3. Полученная скорость обучения используется для нахождения очередной точки $x_{n+1} = x_n - \alpha_j \nabla f(x_n)$.

- Оценки $\hat{\beta}$ находятся с помощью максимизации функции правдоподобия по β одним из методов численной оптимизации.
- **Важно** - для перехода от численной минимизации к максимизации достаточно умножить функцию на -1.
- Условные вероятности оцениваются за счет подстановки $\hat{\beta}$ в формулу:

$$\hat{p}_x = \hat{P}(Y_i = 1 | X_i = x) = g\left(x\hat{\beta}\right)$$

- В случае логистической регрессии получаем:

$$\hat{p}_x = \frac{1}{1 + e^{-x\hat{\beta}}}$$

- Прогнозирование осуществляется классическим образом:

$$\hat{y}(x) = \begin{cases} 1, \text{ если } \hat{p}_x \geq c \\ 0, \text{ в противном случае} \end{cases}$$

- **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.
- **Пример 1** – вместо возраста age_i можно взять его полином третьей степени просто добавив в число признаков age_i^2 и age_i^3.
- **Пример 2** – 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.

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

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

- Функции $F(x)$ обычно именуют **моделями** и они отражают либо прогноз целевой переменной Y_i, либо оценку ее некоторой характерстики, например, условной вероятности $P(Y_i = 1|X_i = x)$.
- Исследователь стремится найти модель $F(x)$ в классе K_F (например, все случайные леса), **минимизирующую функцию потерь**.
- Часто для удобства модели обозначают как $F(x;\theta)$, 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**:

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

- Очевидно, что в данном случае мы получаем линейную регрессию и функция потерь минимизируется при $\theta = \hat{\theta}$, где $\hat{\theta}$ – МНК-оценка, то есть:

$$\hat{\theta} = \left(X^T X\right)^{-1} X^T Y$$

- Таким образом, функция потерь минимизируется при использовании модели:

$$F(x;\hat{\theta}) = x\hat{\theta} = x\underbrace{\left(X^T X\right)^{-1} X^T Y}_{\hat{\theta}}$$

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

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

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

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

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

- **Вывод** – мы получили **логистическую регрессию**, поскольку логистическая функция потерь при данной модели совпадает, с противоположным знаком, с логарифмом функции правдоподобия логистической регрессии.

- Одну и ту же модель машинного обучения можно получить при различных спецификациях функции потерь $\mathrm{L}(Y, F(x))$ и модели $F(x)$.
- Например, логистическую регрессию можно получить и с помощью нередко встречающегося в литературе альтернативного определения логистической функции потерь (проверьте самостоятельно):

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

$$F(x;\theta) = \ln\left(\frac{P\left(Y_i = 1|X_i = x\right)}{1 - P\left(Y_i = 1|X_i = x\right)}\right)$$

$$P\left(Y_i = 1|X_i = x\right) = \frac{1}{1 + e^{-x\theta}}$$

Функция потерь

Подбор гиперпараметров

- Обычно параметры модели оцениваются с использованием дифференцируемой по параметрам функции потерь.
- Гиперпараметры подбираются с помощью этой же или другой функции потерь, либо исходя из некоторых критериев качества, таких как, например, AUC или F1-метрика.
- При умножении на -1 различные критерии качества, включая AUC или F1-метрику, можно рассматривать как функции потерь, однако, поскольку они не дифференцируемы по параметрам, их неудобно использовать для обучения основных параметров (не гиперпараметров) модели.
- Оптимальная функция потерь подбирается исходя из двух **ключевых соображений**:
  - Удобство численной оптимизации: дифференцируемость, выпуклость и т.д.
  - Связь с показателем, непосредственно интересующим исследователя или бизнес.
- 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 или F1-метрику.

- Обратим внимание, что $F(X)$ обозначает вектор значений модели:

$$F(X) = \begin{bmatrix} F(X_1) & F(X_2) & \dots & F(X_m) \end{bmatrix}^T$$

- Рассмотрим градиент функции потерь (скаляр) по значениям модели $F(X)$:

$$\frac{d\mathrm{L}(Y,t)}{dt}\Big|_{t=F(X)} = \begin{bmatrix} \frac{d\mathrm{L}(Y_{\mathbf{1}},t_{\mathbf{1}})}{dt_{\mathbf{1}}}\Big|_{t_{\mathbf{1}}=F(X_{\mathbf{1}})} & \frac{d\mathrm{L}(Y_{\mathbf{2}},t_{\mathbf{2}})}{dt_{\mathbf{2}}}\Big|_{t_{\mathbf{2}}=F(X_{\mathbf{2}})} & \dots & \frac{d\mathrm{L}(Y_m,t_m)}{dt_m}\Big|_{t_m=F(X_m)} \end{bmatrix}^T$$

- Для краткости обозначим его как:

$$\frac{d\mathrm{L}(Y,F(X))}{dF(X)} = \begin{bmatrix} \frac{d\mathrm{L}(Y_{\mathbf{1}},F(X_{\mathbf{1}}))}{dF(X_{\mathbf{1}})} & \frac{d\mathrm{L}(Y_{\mathbf{2}},F(X_{\mathbf{2}}))}{dF(X_{\mathbf{2}})} & \dots & \frac{d\mathrm{L}(Y_m,F(X_m))}{dF(X_m)} \end{bmatrix}^T$$

- Обозначение X будет использоваться для матрицы признаков обучающей выборки, а x – для вектора признаков произвольного наблюдения из обучающей или тестовой выборки.

- 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:

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

- **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:

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

- **Идея** – чтобы улучшить качество модели $F_M(X)$, можно взять любую γ_{M+1}, при которой соблюдается соответствующее неравенство, и в качестве модели $h_{M+1}(X)$ положить $-\frac{d\mathsf{L}(Y,F_M(X))}{dF_M(X)}$. Однако, воплощению этой идеи препятствует серьезная проблема.
- **Проблема** – отрицательный градиент не является моделью, принадлежащей к классу H_{M+1}, и 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 $-\frac{d\mathsf{L}(Y,F_M(X))}{dF_M(X)}$ с помощью X. Тогда значения Y обучающей выборки будут использованы лишь для оценивания функции $h_{M+1}(X)$.

- Выбирается базовая функция, обычно в форме константы:

$$F_0(x) = \underset{c \in R}{\text{argmin}} \ \mathrm{L}(Y, c)$$

  Как правило решением этой задачи является выборочное среднее $c = \overline{Y}$.
- Для каждого наблюдения рассчитывается **остаток** $r_{1i} = -\frac{d\mathrm{L}(Y_i, F_0(X_i))}{dF_0(X_i)}$.
- Определяется класс рассматриваемых моделей H_1. В качестве модели $h_1(X)$ выбирается **регрессионная** модель из класса H_1 (обычно регрессионные деревья небольшой глубины), обученная прогнозировать остатки r_{1i} с помощью признаков X_i.
- Формируется новая модель:

$$F_1(x) = F_0(x) + \gamma_1 h_1(x)$$

- Алгоритм повторяется до выполнения критерия остановки, например, фиксированное число раз.
- Модели H_M обычно одинаковы на всех итерациях, а γ_M часто различаются и каждую итерацию обучения могут находиться линейным поиском.

- В качестве базовой функции можно взять и достаточно сложную модель, которую вы намереваетесь улучшить.
- Базовая модель $F_0(X)$ не обязательно должна 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 валидационной выборке за счет уменьшения числа итераций (гиперпараметр) алгоритма. На практике можно прекратить добавлять новые модели, если, например, функция потерь на валидационной выборке начинает расти.
- В качестве основы градиентного бустинга может выступать не только алгоритм градиентного спуска, но и другие методы численной оптимизации, в том числе использующие информацию о Гессиане.

Градиентный бустинг

Пример первого шага алгоритма

X_i	1	2	5	7	8
Y_i	6	9	12	15	21
$F_0(x)$	9	9	12	16	$(12+15+21)/3 = 16$
$-\nabla \mathrm{L}$	-6	0	0	-2	$-2 \times (16-21) = 10$
$h_1(x)$	-3	-3	-1	4	$(-2+10)/2 = 4$
$F_1(x)$	7.5	7.5	11.5	18	$16 + 0.5 \times 4 = 18$

- Предполагается квадратичная функция потерь:

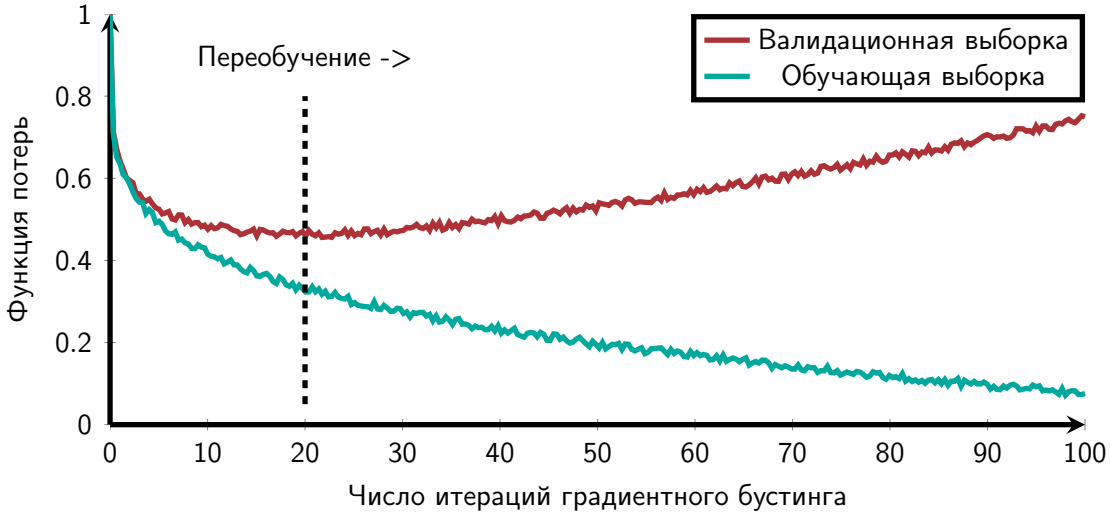
$$\mathrm{L}(y, F(x)) = (F(x) - y)^2 \qquad \nabla \mathrm{L}(y, F(x)) = 2(F(x) - y)$$

- В качестве базовой модели $F_0(x)$, прогнозирующей Y_i с помощью X_i, используется метод ближайших соседей с 3-мя соседями и расстоянием Манхэттен.
- В качестве модели $h_1(x)$, прогнозирующей остаток $r_1 = -\nabla \mathrm{L}(Y_i, F_0(X_i))$ с помощью X_i, используется метод ближайших соседей с 2-мя соседями и Евклидовым расстоянием.
- Посчитаем прогноз в точке $x = 6$ при $\gamma_1 = 0.5$:

$$F_1(6) = F_0(6) + \gamma_1 h_1(6) = \frac{12+15+21}{3} + 0.5 \times \frac{0-2}{2} = 15.5$$

Градиентный бустинг

Графическая иллюстрация проблемы переобучения



- **Идея** – для того, чтобы быстрее найти экстремум функции, можно использовать информацию не только о градиенте, но и о Гессиане.
- **Метод Ньютона** аналогичен градиентному спуску, но совершает очередной шаг с использованием информации не только о градиенте, но и о Гессиане:

$$x_{n+1} = x_n - \alpha \nabla f(x_n) \left(\mathrm{H}f(x_n)\right)^{-1}$$

Где $\mathrm{H}f(t)$ обозначает Гессиан функции $f()$ в точке t, то есть:

$$\left(\mathrm{H}f(t)\right)_{ij} = \frac{\partial^2 f(t)}{\partial t_i \partial t_j}$$

- **Проблема** – если функция $f(t)$ является многомерной, то есть $t \in R^m$, где m достаточно велико, то расчет Гессиана и нахождение обратной ему матрицы оказываются достаточно затруднительными с вычислительной точки зрения задачей.
- **Решение** – воспользоваться квази-Ньютоновскими методами, аппроксимирующими Гессиан с помощью градиента.

- Одним из наиболее популярных Квази-Ньютоновских методов является алгоритм Бройдена—Флетчера—Гольдфарба—Шанно, кратко именуемый **BFGS**.
- Обозначим через B_n аппроксимацию Гессиана $\mathrm{H}f(x_n)$.
- **Идея** – для того, чтобы избежать (ресурсозатратного) взятия обратной матрицы, аппкросимируется не сам Гессиан, а сразу обратная ему матрица:

$$B_{n+1}^{-1} = B_n^{-1} + g(\nabla f(x_n), \nabla f(x_{n+1}), B_n^{-1}, \alpha)$$

  Где $g()$ некоторая функция, опускаемая для краткости.
- В качестве начальной аппроскимации B_0 обычно применяют диагональную матрицу.
- На ранних итерациях аппроксимация Гессиана может быть достаточно неточной, однако, начиная с некоторого n аппроксимации B_n оказываются весьма хорошими.
- Шаг BFGS аналогичен шагу метода Ньютона, за тем исключением, что вместо Гессиана используется его аппроксимация:

$$x_{n+1} = x_n - \alpha \nabla f(x_n) B_n^{-1}$$

- Для наглядности рассмотрим одномерную функцию $f(x)$, где $x \in R$.
- Мы хотим из точки x_n прийти в точку $x_{n+1} = x_n + \triangle x$, которая приблизит нас к экстремуму функции.
- Аппкросимируем значение функции в гипотетической точке экстремума x_{n+1} с помощью разложения ряда Тейлора второго порядка с использованием начальной точки x_n:

$$f(x_{n+1}) = f(x_n + \Delta x) \approx f(x_n) + f'(x_n)(x_n + \Delta x - x_n) + 0.5f''(x_n)(x_n + \Delta x - x_n)^2 =$$
$$= f(x_n) + f'(x_n)\Delta x + 0.5f''(x_n)\Delta x^2$$

- Если x_{n+1} является точкой экстремума, то $f'(x_{n+1}) = 0$, а значит при некоторых условиях производная аппроксимации, полученной разложением в ряд Тейлора, также будет близка к 0:

$$\frac{df(x_{n+1})}{dx_{n+1}} = \frac{df(x_n + \Delta x)}{d\Delta x} \approx f'(x_n) + f''(x_n)\Delta x = 0 \implies \Delta x = -\frac{f'(x_n)}{f''(x_n)}$$

- Данная аппроксимация мотивирует шаг $x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$.
- **Вывод** – мы ищем экстремум функции $f()$, аппроксимируя ее квадратичным полиномом в точке x_n и новую точку x_{n+1} находим как экстремум этой аппроксимации.

- Напомним, что в методе Ньютона шаг имеет вид:

$$\textbf{Одномерный случай: } x_{n+1} = x_n - \alpha \frac{f'(x_n)}{f''(x_n)}$$

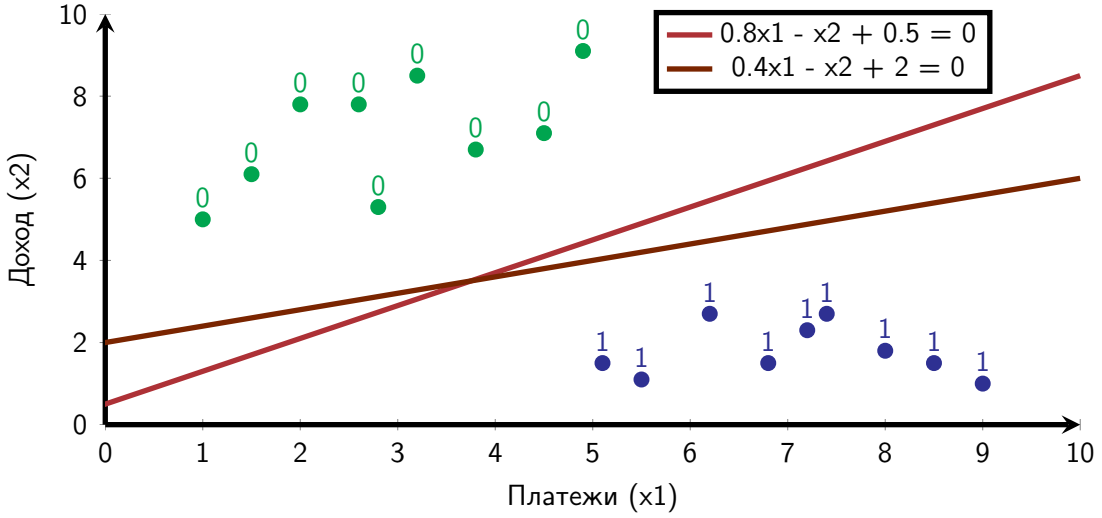
$$\textbf{Многомерный случай: } x_{n+1} = x_n - \alpha \nabla f(x_n) \left(\mathrm{H}f(x_n)\right)^{-1}$$

$$\textbf{Квази-Ньютоновский метод: } = x_{n+1} = x_n - \alpha \nabla f(x_n) B_n^{-1}$$

- **Проблема** – в одномерном случае если $f''(x_n) < 0$, то мы приближамся к максимуму, а при $f''(x_n) > 0$ – к минимуму. Это создает проблемы с максимизацией (минимизацией) функций, имеющих выпуклые (вогнутые) участки, поскольку на этих участках вторые производные положительны (отрицательны).
- В многомерном случае мы приближаемся к максимуму, если Гессиан $\left(\mathrm{H}f(x_n)\right)^{-1}$ отрицательно определен и к минимуму – если Гессиан определен положительно.
- **Популярное решение** – в некоторых квази-Ньютоновских методах, включая BFGS, знакоопределенность аппроксимаций Гессиана B_n всегда совпадает со знакоопределенностью начальной матрицы B_0. Поэтому, несмотря на то, что, например, на выпуклых участиках аппроксимация Гессиана максимизируемой функции будет неточной, алгоритм все равно, при некоторых условиях, продолжит движение в сторону точки максимума.

Метод опорных векторов

Графическая иллюстрация идеи на примере дефолта



- **Предположение** – существуют линии, которые могут 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. Это расстояние именуется **отступом** (margin).
- После того, как мы выбрали линию, наблюдения, имеющие наибольшее 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 максимизации 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 умножение 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 необходимо 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.