

# Обучение с учителем. Классификация. Байесовский подход. Решающие деревья и ансамбли.

Санкт-Петербургский государственный университет  
Кафедра статистического моделирования

**Машинное обучение** — это раздел искусственного интеллекта, в котором разрабатываются методы и алгоритмы, позволяющие компьютерам обнаруживать закономерности в данных и делать прогнозы без явных инструкций.

## Пример задач:

- Регрессия: предсказание стоимости недвижимости, количества продаж некоторого товара, погоды.
- Классификация: предсказание ценовой категории товара, типа изображения, болеет ли человек или нет.

**Дано:**

- Пространство объектов  $\mathcal{X}$  — множество описаний объектов (например, фотографии, тексты, таблицы с признаками).
- Пространство ответов  $\mathcal{Y}$  — множество меток или значений, которые нужно предсказывать (например, классы «кот»/«собака» или числовые значения цен).

**Формализация задачи:** должна достигаться минимальная ошибка предсказания.

Функционал ошибки  $Q(\mathbf{X}, \mathbf{Y})$ :

Регрессия	Классификация
$\mathbb{E} \ A(\xi) - \eta\ _2^2$	$\mathbb{E} \ [(A(\xi)) \neq \eta]\ _1$

Для решения задачи необходимо определить:

- **Модель.** Пример: линейная регрессия (регрессия), SVM (классификация).
- **Функцию потерь и функционал ошибки.** Пример: MSE (регрессия), кросс-энтропия (классификация).
- **Метод оптимизации.** Пример: стохастический

В задаче классификации множество ответов  $y_i$  дискретно. Рассмотрим линейный классификатор:

$$A(x_i, w) = \text{sign}\left(\sum_{j=1}^k w_j x_j + w_0\right). \quad (1)$$

Множество ответов:  $y_i \in \{+1, -1\}$ . Линейный классификатор разделяет пространство на два класса. Будем считать, что среди  $k$  признаков есть единичный. Функционал ошибки:

$$Q(\mathbf{X}, \mathbf{Y}) = \frac{1}{n} \sum_{i=1}^n [y_i \langle x_i, w \rangle < 0] = \frac{1}{n} \sum_{i=1}^n [M(x_i, y_i) < 0] \rightarrow \min_w \quad (2)$$

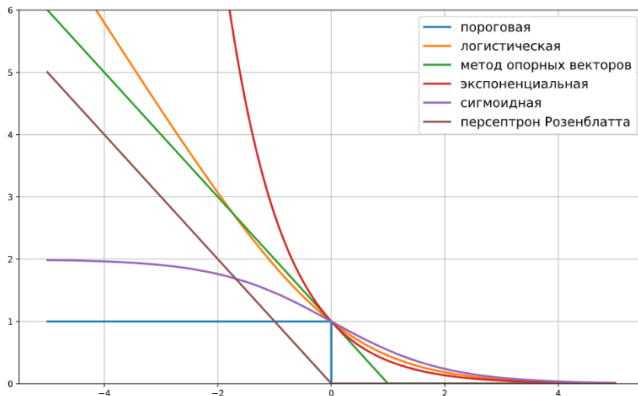
$M(x_i, y_i)$  — **величина отступа** (margin), абсолютная величина отступа говорит о степени уверенности классификатора.

# Классификация

**Проблема:** такой функционал нельзя минимизировать через градиентные методы.

**Решение:** можно минимизировать мажорирующий функционал.

Ступенчатая функция потерь (ошибка на одном объекте) —  $[M < 0]$ .



**Рис.:** Верхние оценки для пороговой функции потерь.

Различные функции потерь:

- 1 Логистическая:  $L(M) = \ln(1 + e^{-M})$
- 2 Кусочно-линейная (SVM):  $L(M) = \max(0, 1 - M)$
- 3 Кусочно-линейная (NN):  $L(M) = \max(0, -M)$
- 4 Экспоненциальная:  $L(M) = e^{-M}$
- 5 Сигмоидная:  $L(M) = \frac{1}{1+e^M}$  (ANN - моделирование нейронных связей)

# Классификация: связь с модельным подходом

Рассмотрим логистическую регрессию:  $L(M) = \ln(1 + e^{-M})$ .

Логит:

$$z_i = \langle x_i, w \rangle = \ln \frac{p}{1-p}, p = \frac{1}{1 + e^{-\langle x_i, w \rangle}} = \sigma(\langle x_i, w \rangle) \quad (3)$$

— апостериорная вероятность.

Функционал ошибки:

$$\begin{aligned} Q(X, Y) &= \sum_{i=1}^n \ln(1 + e^{-y_i \langle x_i, w \rangle}) = \\ &= - \sum_{i=1}^n ([y_i = 1] \ln \frac{1}{1 + e^{-z_i}} + [y_i = -1] \ln \frac{1}{1 + e^{z_i}}) = \\ &= - \sum_{i=1}^n ([y_i = 1] \sigma(z_i) + [y_i = -1] (1 - \sigma(z_i))) \end{aligned}$$

Применим модельный подход с распределением Бернулли (ММП). Метки классов  $\{0, 1\}$ .

$$p(y|X, w) = \prod_{i=1}^n p_i^{y_i} (1 - p_i)^{1-y_i}. \quad (4)$$

Прологарифмируем.

$$\sum_{i=1}^n y_i \ln(\sigma(\langle x_i, w \rangle)) + (1 - y_i) \ln(1 - \sigma(\langle x_i, w \rangle)) \quad (5)$$

Таким образом, ММП в данной задаче эквивалентен минимизации функционала с логистической функцией потерь.



- 1 У нас есть выборка из случайной величины (например, сторона монетки).
- 2 Мы хотим узнать распределение параметра  $\theta$  (например, вероятность выпадения орла). Мы предполагаем, что он имеет некоторое **априорное** распределение.
- 3 Если применить полученные знания о выборке, то мы перейдем к **апостериорному** распределению.

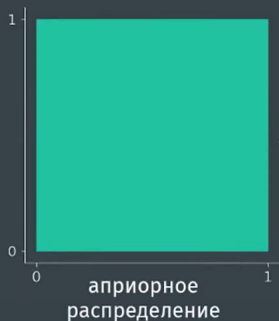
$p(\theta)$  — **априорное** распределение параметра,  $p(x|\theta)$  — правдоподобие. Мы хотим узнать  $p(\theta|x)$ .

**Апостериорное** распределение:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \quad (6)$$

## Априорное и апостериорное распределения

$$\theta = [0,1], p(\theta) \equiv 1 \quad \times \quad p(d|\theta) = \theta^6(1-\theta)^4 \quad \propto \quad p(\theta|d) = \frac{\theta^6(1-\theta)^4}{B(7,5)}$$



$$+ d = [0,1,1,0,1,0,1,1,1,0] =$$

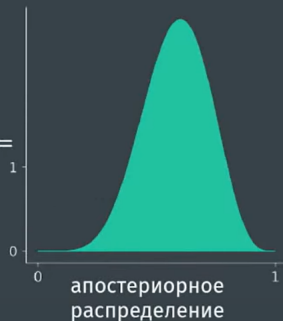


Рис.: Переход к апостериорному распределению.

# Байесовский подход: общие понятия

- Частотный подход: "объективная неопределенность"
- Байесовский подход: "субъективное незнание"

Почему мы не можем всегда пользоваться частотным подходом (ММП)?  $n$  должно быть большим.

**Результатом** решения задачи в байесовском подходе является апостериорное распределение.

Что дает байесовский подход?

- 1 Регуляризация.** Мы получаем распределение, а не точечную оценку.
- 2 Композитность:** возможность объединить знания:

$$p(y|x, z) = \frac{p(z|y)p(y|x)}{\int p(z|y)p(y|z)dz}$$

- 3 Инкрементальное получение результата.**

# Байесовский подход: общие понятия

Формально: модель задается как параметрическое семейство распределений  $\{\mathbb{P}_\theta | \theta \in \Theta \in \mathbb{R}^k\}$  с заданным на  $\Theta$  априорным распределением  $p(\theta)$ .

Каждый раз при получении новых данных мы обновляем наше апостериорное распределение.

Пусть мы  $m$  раз обновляли апостериорное распределение, получая данные  $x_i$ :

$$p(\theta) \rightarrow p(\theta|x_1) \rightarrow \rightarrow p(\theta|x_1, x_2) \rightarrow \dots \rightarrow p(\theta|x_1, \dots, x_m)$$

Понятно, что  $m$ -ый шаг равносителен одному шагу с данными  $x_1, \dots, x_m$ .

# Байесовский подход: предел апостериорного распределения

Что будет происходить с апостериорным распределением, если будет поступать неограниченное количество данных?

- При выполнении *условий регулярности*, апостериорное распределение  $p(\theta|x_m)$  при  $m \rightarrow \infty$  приближается к нормальному распределению  $\mathcal{N}(\theta_0, (mI(\theta_0))^{-1})$ .
- $\theta_0$  — параметр истинного распределения, если распределение лежит в нашей модели.
- Если распределение не лежит в нашей модели, то  $\theta_0$  минимизирует KL-расстояние от истинного распределения до нашей модели.
- $I(\theta_0)$  — информация Фишера.

Т.е.  $p(\theta|x_m)$  стремится к вырожденному распределению в точке  $\theta_0$ .

Что если  $\theta$  — многомерный параметр?

Пусть есть апостериорное распределение  $p(\theta|x)$ . По нему мы можем посчитать **маргинальное распределение** конкретной компоненты  $\theta$  или группы компонент:

$$p(\theta_i|x) = \int p(\theta|x) d\theta_{-i}$$

Такие интегралы часто приходится считать численно.

# Байесовский подход: как описать апостериорное распределение?

Мы по понятным причинам не можем полностью описать апостериорное распределение.

Что тогда можно взять?

- 1 Точечные характеристики: среднее  $\int \phi(\theta)p(\theta|x)d\theta$ , максимум  $\operatorname{argmax}_{\phi(\theta)}p(\theta|x)$ .
- 2 **Достоверный интервал** — множество, в котором  $\phi(\theta)$  лежит с опр. вероятностью.

Если данных много, то искать апостериорное распределение может быть затруднительно:  $\int p(x|\theta)p(\theta)d\theta$  не считается аналитически для любых  $p(x|\theta)$  и  $p(\theta)$ .

Что делать в таком случае?

**Сопряженное апостериорное распределение** — распределение из параметрического семейства, для которого апостериорное распределение также в этом семействе.

Таким образом, часто априорное распределение выбирают исходя из наличия сопряженного апостериорного.



# Байесовский подход: сопряженные распределения

Примеры сопряженных распределений:

- Бернулли - Гамма
- Биномиальное - Гамма
- Нормальное с изв.  $\sigma^2$  - Нормальное (параметр  $\mu$ )
- Нормальное с изв.  $\mu$  - Гамма (параметр  $\sigma^2$ )

Пример:

- $X \sim B(\theta) \Rightarrow p(x_n|\theta) = \theta^{\sum x_i} (1 - \theta)^{(n - \sum x_i)}$ .
- Априорное распределение:  $\theta \sim \text{Beta}(\alpha, \beta)$ ,  
 $p(\theta) = \theta^{\alpha-1} (1 - \theta)^{\beta-1} / B(\alpha, \beta)$ .
- $p(\theta|x_n) \propto \theta^{\sum x_i + \alpha - 1} (1 - \theta)^{n - \sum x_i + \beta - 1}$
- Получаем, что  $\theta|x_n \sim \text{Beta}(\alpha + \sum x_i, \beta + n - \sum x_i)$ .
- **Трюк** с сопряженными распределениями: если будут новые данные, поместим их в параметры *Beta*!

- **Неинформативные:** не несут дополнительной информации о значениях параметров (равномерные на  $\theta$ ). Но: информация о том, что мы не можем предпочесть конкретное значение любому другому.
- **Информативные:** несут существенную дополнительную информацию о значениях параметра. Например: накопленные знания в некоторой предметной области. Если данных мало, то априорное распределение может ухудшить понимание.
- **Слабо информативные:** несут частичную информацию. Избавляемся от экстремальных значений. Что-то похожее на регуляризацию.

## Виды априорных распределений

Пусть  $d = [1,1,1,1,1,1,1,1,1,1]$  — десять бросков монетки.

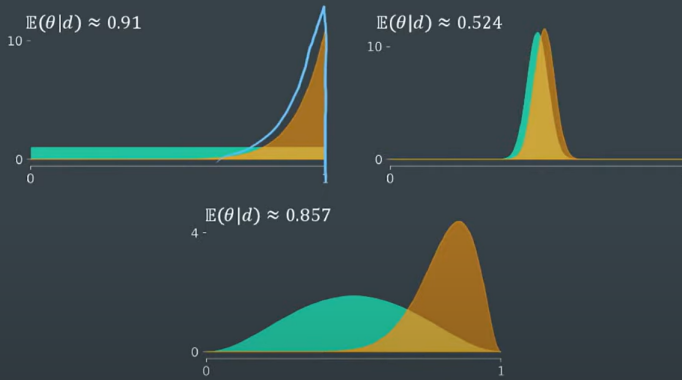


Рис.: Априорные и апостериорные распределения.

Рассмотрим модель линейной регрессии:

$$Y|\beta \sim \mathcal{N}(x\beta, \sigma^2 I_n), \beta \sim \mathcal{N}(0, \sigma_0^2 I_d).$$

Тогда

$$p(\beta|y) \propto \exp\left\{-\frac{1}{2\sigma^2} \sum (y_i - x_i\beta)^2\right\} \exp\left\{-\frac{1}{2\sigma_0^2} \beta^T \beta\right\}.$$

Максимизация апостериорной плотности равносильна минимизации

$$\sum (y_i - x_i\beta)^2 + \frac{\sigma^2}{\sigma_0^2} \sum \beta_j^2.$$

Это же L2-регрессия с  $\lambda = \frac{\sigma^2}{\sigma_0^2}$ !

- Пусть мы умеем считать  $p(\theta|X)$ .
- Тогда мы можем посчитать  $p(X_{new}|\theta)$  — **предсказательное распределение**.

$$\begin{aligned} p(X_{new}|\theta) &= \int_{\theta} p(X_{new}, \theta|X) d\theta = \int_{\theta} p(X_{new}|\theta, X) p(\theta|X) d\theta \\ &= \int_{\theta} p(X_{new}|\theta) p(\theta|X) d\theta \end{aligned}$$

Проще сгенерировать данные:

- 1 Генерируем  $\theta'$  из апостериорного распределения.
- 2 Генерируем новое наблюдение  $X_{new}$  из  $p(*|\theta')$ .

В байесовском подходе вопрос "верна ли модель" не корректен — модель не может быть полностью верна.

Корректный вопрос: *оказывают ли недостатки модели существенное влияние на выводы?*

- Если модель хорошая, то данные из предсказательного распределения должны быть похожи на реальные данные.
- Похожесть можно определять с помощью тестовых статистик  $T$ .

$P(T(X_{new}) > T(X)|X)$  — апостериорный предсказательный p-value.

- Если апостериорный предсказательный p-value близок к 0 или 1, то качество модели низкое.

Как считать апостериорный предсказательный p-value?

$$p = \int \int [T(X_{new} \geq T(X))] p(X_{new}|\theta) p(\theta|X) d(X_{new}) d\theta$$

Как считают на самом деле:

- 1 Генерируют  $K$  параметров  $\theta'$  из апостериорного распределения.
- 2 Для каждого из них генерируют  $X_{new} \sim p(*|\theta')$ .
- 3 Для каждого  $X_{new}$  считают  $T(X_{new})$ .
- 4 Значение  $p$  примерно равно доле  $T(X_{new})$  больших  $T(X)$ .

# Байесовский подход: принятие решений

В байесовском подходе мы отходим от отвержения/не отвержения гипотезы в пользу **принятия решений**.

- Если не нужно принимать решение — лучше оставить неопределенность в виде апостериорного распределения.
- Если принять решение все же нужно, то принимаем решение, учитывая риски.

Пусть есть набор решений  $\delta_1, \dots, \delta_k$ . **Риск**  $R_i$  — с.в., отражающая неопределенность последствий принятия решения  $\delta_i$ .

После получения данных  $X$  посчитаем **апостериорные риски**  $R_i|X$  и выберем решение, минимизирующее **средний апостериорный риск**  $E(R_i|X)$ .



Пример: классификация.

- Вероятностное пр-во  $X \times Y$ :  $X$  — объекты,  $Y$  — метки классов; известна плотность  $p(x|y)p(y)$ .
- Априорные вероятности  $p(y)$  известны.
- Функции правдоподобия классов  $p(x|y)$  известны.
- Решение — выбор класса.
- Для каждой пары  $(y, y')$  определен штраф  $\lambda_{y,y'}$ .

# Байесовский подход: классификация

Риск для объекта  $x$ :

$$R_{y'} = \sum_{y \in Y} \lambda_{y,y'} [Y|x = y].$$

Средний апостериорный риск:

$$ER_{y'} = \sum_{y \in Y} \lambda_{y,y'} p(y|x) \propto \sum_{y \in Y} \lambda_{y,y'} p(x|y)p(y)$$

Т.е. Классификатор выглядит как:

$$a(x) = \operatorname{argmin}_{y'} \sum_{y \in Y} \lambda_{y,y'} p(x|y) =$$

$$\operatorname{argmin}_{y'} \sum_{y \in Y} \lambda_{y,y'} p(x|y)p(y)$$

Решающее дерево — это бинарное дерево, которое строится следующим образом:

- 1 В корне дерева: выбирается признак и некоторый порог  $t$ , проверяется условие для каждого объекта  $x_{ij} < t$ . Если условие не выполняется, объект помещается в левый лист, иначе в правый (в случае категориального признака: проверка условия  $x_{ij} = c$ ). Далее процедура повторяется в листах и т.д.
- 2 Построение дерева *прекращается* при достижении некоторых условий, например, некоторой глубины.
- 3 *Предсказание* decision tree — некоторое отображение от выборки в листе, куда должен попасть новый объект, например, среднее значение или самый частый класс.

Решающие деревья позволяют восстанавливать нелинейные зависимости произвольной сложности, применяются в задачах регрессии и классификации.



Рис.: Пример дерева решений.

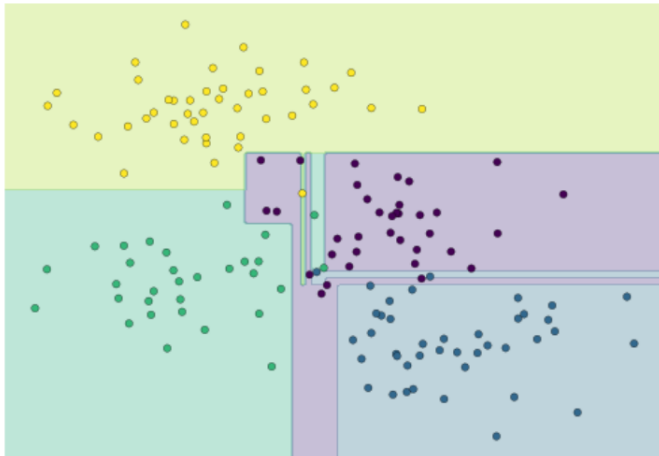


Рис.: Пример классификации с decision tree.

# Решающие деревья: построение дерева

Как построить дерево, которое будет хорошо решать задачу регрессии или классификации?

- Можно построить дерево, которое имеет нулевую ошибку на любой выборке, но оно будет **переобученным**.
- Построение минимального решающего дерева с нулевой ошибкой — NP полная задача.

Пусть имеется некоторый функционал ошибки  $Q(X, j, t)$  для выборки в вершине,  $j$  — номер признака,  $t$  — некоторый порог; и некоторое условие останова.

**Жадный алгоритм:**

- 1 На каждой итерации выполняется поиск  $j, t$ , которые минимизируют  $Q(X, j, t)$ .
- 2 Построение прекращается при выполнении условия останова.

После построения дерева можно произвести его стрижку (**prunning**) — удаление некоторых вершин с целью повышения обобщательной способности.

Какой функционал использовать для разбиения выборки в вершинах?

$$Q(R_m, j, s) = \frac{|R_l|}{|R_m|} H(R_l) + \frac{|R_r|}{|R_m|} H(R_r), \quad (7)$$

где  $H(R)$  — **критерий информативности**, который оценивает качество распределения целевой переменной среди объектов  $R$ . Чем меньше разнообразие целевой переменной — тем меньше значение критерия информативности.

# Решающие деревья: критерии информативности

Критерии информативности:

- Регрессия:  $H(R) = \min_{c \in \mathbb{Y}} \frac{1}{|R|} \sum_{(x_i, y_i) \in R} (y_i - c)^2$

$$H(R) = \frac{1}{|R|} \sum_{(x_i, y_i) \in R} \left( y_i - \frac{1}{|R|} \sum_{(x_j, y_j) \in R} y_j \right)^2$$

т.е. дисперсия в листе.

- Классификация: пусть  $p_k = \frac{1}{|R|} \sum_{(x_i, y_i) \in R} [y_i = k]$ .  
 $\hat{k} = \operatorname{argmax}_k p_k$ .

$$H(R) = \min_{c \in \mathbb{Y}} \frac{1}{|R|} \sum_{(x_i, y_i) \in R} [y_i \neq c].$$

т.е. оптимальное предсказание —  $\hat{k}$ . Следовательно,  
 $H(R) = 1 - p_{\hat{k}}$ .



Критерии информативности:

- **Критерий Джини** (Jini impurity): пусть при классификации ответ в листе — набор вероятностей. Качество такого набора можно измерить по **Brier score**:

$$H(R) = \min_{\sum_k c_k = 1} \frac{1}{|R|} \sum_{(x_i, y_i) \in R} \sum_{k=1}^K (c_k - [y_i = k])^2.$$

Оптимальный набор —  $\hat{c} = (p_1, \dots, p_K)$ . Подставим этот набор и получим критерий Джини:

$$H(R) = \sum_{k=1}^K p_k(1 - p_k)$$

Критерии информативности:

- **Энтропийный критерий:** рассматривается  
кросс-энтропия

$$H(R) = \min_{\sum_k c_k = 1} \left( -\frac{1}{|R|} \sum_{(x_i, y_i) \in R} \sum_{k=1}^K [y_i = k] \ln c_k \right).$$

После некоторых вычислений получаем, что оптимальный набор соотв.  $p_i$ . Энтропийный критерий:

$$H(R) = - \sum_{k=1}^K p_k \ln p_k$$

# Решающие деревья: критерии информативности

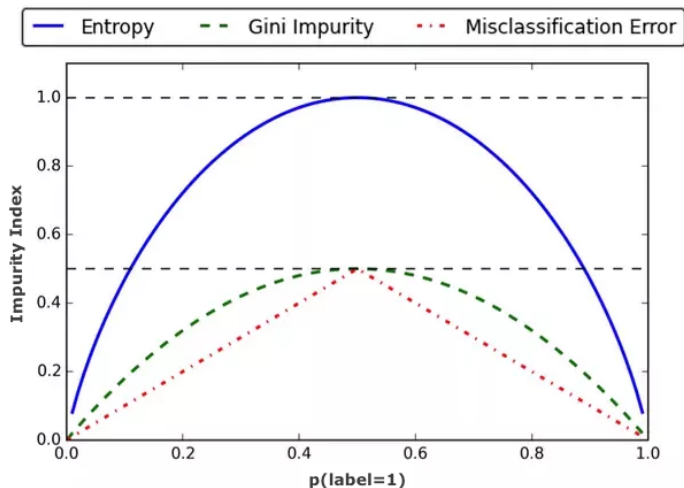


Рис.: Критерии информативности.

# Решающие деревья: критерии останова

- Ограничение максимальной глубины
- Ограничение минимального числа объектов в листе
- Ограничение максимального числа листьев
- Однородная выборка
- Минимальное улучшение функционала

Преимущества decision tree:

- Интерпретируемость
- Простая обработка категориальных признаков

Недостатки:

- Сильно примитивнее других моделей.

Обратимся к разложению ошибки на смещение и разброс (bias-variance tradeoff).

**Bagging** — метод, который позволяет уменьшить разброс модели. Этот метод обучает некоторое число алгоритмов так, что каждый алгоритм обучается на отдельных выборках, которые получены из исходной посредством бутстрапа. Разложение MSE в задаче регрессии:

$$MSE = (\eta - \mathbb{E}\hat{\eta})^2 + \mathbb{E}(\mathbb{E}\hat{\eta} - \hat{\eta})^2$$

Разброс для ансамбля с усреднением,  $\hat{\eta} = \frac{1}{M} \sum_{i=1}^M \hat{\eta}_i$ :

$$Var(\hat{\eta}) = \frac{1}{M^2} \sum_{i=1}^M Var(\hat{\eta}_i) + \frac{1}{M^2} \sum_{i \neq j} Cov(\hat{\eta}_i, \hat{\eta}_j)$$

Набор алгоритмов называется **ансамблем**, предсказание ансамбля в бэггинге:

$$a_M(x) = \frac{1}{M} \sum_{m=1}^M b_m(x)$$

Смещение бэггинга совпадает со смещением базового алгоритма. При этом если базовые алгоритмы **некоррелированы**, то разброс уменьшается в  $M$  раз.

Модель **Random Forest** — улучшение бэггинга решающих деревьев. Деревья становятся менее коррелированными благодаря тому, что при построении дерева в каждой вершине признак выбирается из случайного подмножества заданного размера  $p$ . Например,  $p = \sqrt{d}$ , где  $d$  — число признаков, для регрессии.

Ограничения в построении дерева выбираются так, чтобы деревья получались глубокими — такие деревья имеют низкое смещение.

**Бустинг** — метод ансамблирования, в котором каждый добавляемый в композицию алгоритм обучается на ошибки ансамбля на предыдущем шаге. **Пример:** Рассмотрим задачу регрессии

$$\frac{1}{2} \sum_{i=1}^N (a(x_i) - y_i)^2 \rightarrow \min_a. \quad (8)$$

Пусть предсказание алгоритма — сумма предсказаний некоторых других базовых моделей:

$$a_M(x) = \sum_{m=1}^M b_{\theta_m, m}(x).$$

Построим первый базовый алгоритм:

$$b_{\theta_1, 1}(x) = \operatorname{argmin}_{\theta} \frac{1}{2} \sum_{i=1}^N (b_{\theta}(x_i) - y_i)^2,$$



Остатки:

$$s_i^{(1)} = y_i - b_1(x_i).$$

Следующий алгоритм:

$$b_{\theta_2,2}(x) = \operatorname{argmin}_{\theta} \frac{1}{2} \sum_{i=1}^N (b_{\theta}(x_i) - s_i^{(1)})^2.$$

И так далее. Заметим, что мы обучаемся на антиградиент:

$$s_i^{(M)} = y_i - a_{M-1}(x_i) = -\frac{d}{dz} \frac{1}{2} (z - y_i)^2 \Big|_{z=a_{M-1}(x_i)}$$

Это базовая реализация бустинга.

# Gradient Boosting

Будем строить предсказание ансамбля как взвешенную сумму предсказаний базовых алгоритмов. Пусть имеется дифференцируемый функционал  $L(y, z)$ .

$$a_M(x) = \sum_{m=0}^M \gamma_m b_{\theta_m, m}(x).$$

Как инициализировать значения?

- $b_0(x) = 0$
- Классификация:  $b_0(x) = \operatorname{argmax}_{y \in \mathbb{Y}} \sum_{i=1}^N [y_i = y]$
- Регрессия:  $b_0(x) = \frac{1}{N} \sum_{i=1}^N y_i$ .

Добавление нового алгоритма в ансамбль:

$$\sum_{i=1}^N L(y_i, a_{M-1}(x_i) + \gamma_M b_{\theta_M, M}(x_i)) \rightarrow \min_{\theta_M, \gamma_M}.$$

# Gradient Boosting

**Идея:** предсказание следующего алгоритма должно быть противоположно производной  $L(y, z)$  в точке  $z = a_{M-1}(x_i)$ .

Тогда вектор сдвигов совпадает с антиградиентом  $L$ .

Таким образом, добавляя новый алгоритм, мы делаем шаг градиентного спуска.

Рассмотрим задачу регрессии:

$$b_{\theta_M, M}(x) = \operatorname{argmin}_{\theta} \sum_{i=1}^N (b_{\theta}(x_i) - y_i)^2.$$

После построения нового алгоритма, выбирается новый шаг:

$$\gamma_M = \operatorname{argmin}_{\gamma \in \mathbb{R}} \sum_{i=1}^N L(y_i, a_{M-1}(x_i) + \gamma b_{\theta_M, M}(x_i)).$$

Описанный метод называется градиентным бустингом.

- Примитивные базовые алгоритмы плохо приближают антиградиент.
- Сложные базовые алгоритмы приведут к переобучению.

Решение этих проблем — **сокращение шага**: вместо перехода в оптимальную точку делается укороченный шаг:

$$a_M(x) = a_{M-1}(x) + \eta \gamma_M b_{\theta_{M,M}}(x), \eta \in (0; 1]$$

$\eta$  — шаг обучения.

Другое решение: стохастический градиентный бустинг.

# Gradient Boosting: Классификация

Функция потерь для классификации:

$$L(y, z) = \ln(1 + e^{-yz}).$$

Тогда построение базового алгоритма:

$$b_{\theta_M, M} = \operatorname{argmin}_{\theta} \sum_{i=1}^N \left( b_{\theta}(x_i) - \frac{y_i}{1 + e^{y_i a_{M-1}(x_i)}} \right)^2.$$

Рассмотрим логистическую функцию потерь:

$$Q(a_M) = \sum_{i=1}^N \ln(1 + e^{-y_i a_{M-1}(x_i)}) e^{-y_i \gamma_M b_{\theta_M, M}(x_i)}.$$

Объекты с большим margin можно исключить.

# Gradient Boosting: Решающие деревья

Градиентный бустинг деревьев — один из самых сильных методов машинного обучения.

Представим предсказание решающего дерева в виде

$$b_m(x) = \sum_{j=1}^{J_m} b_{mj}[x \in R_j],$$

где  $j$  — индекс листа,  $R_j$  — область разбиения,  $b_{mj}$  — ответ в листе.

Предсказание ансамбля:

$$a_M(x) = a_{M-1}(x) + \gamma_M \sum_{j=1}^{J_M} b_{Mj}[x \in R_j] =$$

$$a_{M-1}(x) + \sum_{j=1}^{J_M} \gamma_M b_{Mj}[x \in R_j].$$

Таким образом, добавление в ансамбль решающего дерева с  $J_M$  листьями эквивалентно добавлению  $J_M$  предикатов  $x \in R_j$ .  
Градиентный бустинг может лишь снизить смещение базовых моделей, а разброс бустинга не меньше разброса базового алгоритма.  
Поэтому в бустинге используются неглубокие решающие деревья, которые не склонны к переобучению.

# Gradient Boosting: AdaBoost

**AdaBoost:** используется экспоненциальная функция потерь

$$L(y, z) = e^{-yz}.$$

Метод сводит задачу поиска базового алгоритма к минимизации доли неверных ответов с весами при объектах. При использовании экспоненциальной функции потерь:

$$s_i = y_i e^{-y_i \sum_{m=1}^{M-1} \gamma_m b_m(x_i)} = y_i w_i.$$

Что будет при наличии шумовых объектов? Отступ на них будет большой отрицательный, классификатор будет настраиваться исключительно на них.

Теперь рассмотрим логистическую функцию потерь и получим:

$$s_i = y_i \frac{1}{1 + \exp(y_i a_{M-1}(x_i))}.$$

Шумовой объект будет иметь вес всего лишь в два раза больше.



Преимущества градиентного бустинга:

- Точнее многих архитектур.
- Быстро обучается.
- Хорошо работает с категориальными признаками.
- Множество хороших реализаций: XGBoost, CatBoost, LightGBM, с поддержкой Map Reduce.

Недостатки:

- Легко переобучается (Но: настройка patience, L1, L2 и т.д.)