

Простое голосование классификаторов

Обучающая выборка: $X^\ell = (x_i, y_i)_{i=1}^\ell$, $x_i \in X$, $y_i \in \{-1, +1\}$

Базовые классификаторы: $b_1(x), \dots, b_T(x)$, $b_t: X \rightarrow \{-1, +1\}$

Простое голосование базовых классификаторов:

$$a(x) = \operatorname{sign} \sum_{t=1}^T b_t(x)$$

Композиция $a(x)$ может быть лучше базовых $b_1(x), \dots, b_T(x)$,
если они лучше случайного гадания и достаточно различны.

Способы повышения различности базовых классификаторов:

- обучение по случайным подвыборкам,
- обучение по выборке со случайными весами объектов,
- обучение по случайным подмножествам признаков,
- использование различных моделей классификации,
- использование различных начальных приближений,
- использование рандомизации при обучении b_1, \dots, b_T .

Бэггинг (bagging, bootstrap aggregating) [Breiman, 1996]:
 $b_t(x)$ обучаются независимо по случайным подвыборкам
длины ℓ с повторениями (как в методе bootstrap),
доля объектов, попадающих в выборку: $(1 - \frac{1}{e}) \approx 0.632$

Метод случайных подпространств
(RSM, random subspace method) [Ho, 1998]:
 $b_t(x)$ обучаются по случайным подмножествам n' признаков.

Совместим обе идеи в одном алгоритме.

$\mathcal{F} = \{f_1, \dots, f_n\}$ — признаки,
 $\mu(\mathcal{G}, U)$ — метод обучения алгоритма по подвыборке $U \subseteq X^\ell$,
использующий только признаки из $\mathcal{G} \subseteq \mathcal{F}$.

Breiman L. Bagging predictors // Machine Learning, 1996.

Ho T. K. The random subspace method for constructing decision forests // IEEE Trans. on Pattern Analysis and Machine Intelligence, 1998.

Вход: обучающая выборка X^ℓ ; параметры: T

ℓ' — длина обучающих подвыборок;

n' — длина признакового подописания;

ε_1 — порог качества базовых алгоритмов на обучении;

ε_2 — порог качества базовых алгоритмов на контроле;

Выход: базовые алгоритмы b_t , $t = 1, \dots, T$;

1: **для всех** $t = 1, \dots, T$

2: $U_t :=$ случайное подмножество X^ℓ длины ℓ' ;

3: $\mathcal{G}_t :=$ случайное подмножество \mathcal{F} длины n' ;

4: $b_t := \mu(\mathcal{G}_t, U_t)$;

5: **если** $Q(b_t, U_t) > \varepsilon_1$ или $Q(b_t, X^\ell \setminus U_t) > \varepsilon_2$ **то**

6: не включать b_t в композицию;

Композиция — простое голосование: $a(x) = \text{sign} \sum_{t=1}^T b_t(x)$.

Обучение случайного леса:

- бэггинг над решающими деревьями
- усечение дерева (pruning) не производится
- признак в каждой вершине дерева выбирается из случайного подмножества k из n признаков
- для регрессии рекомендуется $k = \lfloor n/3 \rfloor$
- для классификации рекомендуется $k = \lfloor \sqrt{n} \rfloor$

Подбор числа деревьев T по критерию *out-of-bag*:
число ошибок на объектах x_i , если не учитывать голоса
деревьев, для которых x_i был обучающим:

$$\text{out-of-bag}(a) = \sum_{i=1}^{\ell} \left[\text{sign} \left(\sum_{t=1}^T [x_i \notin U_t] b_t(x_i) \right) \neq y_i \right] \rightarrow \min$$

Это несмешённая оценка обобщающей способности.

- RF — один из самых сильных методов машинного обучения.
- RF обычно лишь немного уступает градиентному бустингу.
- Бэггинг позволяет вычислять оценки out-of-bag:
 - для оптимизации числа базовых алгоритмов T ,
 - для оценивания важности признаков,
 - для выбора параметра k в RF.
- RSM хорош, когда много неинформативных признаков.
- Бэггинг, RF, RSM эффективно распараллеливаются.

Задача восстановления зависимости $y: X \rightarrow Y$ по точкам обучающей выборки (x_i, y_i) , $y_i = y(x_i)$, $i = 1, \dots, \ell$.

Определение

Линейной композицией базовых алгоритмов $a_t(x) = C(b_t(x))$, $t = 1, \dots, T$, называется суперпозиция функций

$$a(x) = C\left(\sum_{t=1}^T \alpha_t b_t(x)\right),$$

где $C: \mathbb{R} \rightarrow Y$ — решающее правило, $\alpha_t \geq 0$.

- **Пример 1:** классификация на 2 класса, $Y = \{-1, +1\}$;
 $C(b) = \text{sign}(b)$, $a(x) = \text{sign}(b(x))$,
 $b: X \rightarrow \mathbb{R}$ — дискриминантная функция.
- **Пример 2:** регрессия, $Y = \mathbb{R}$,
 $C(b) = b$, $a(x) = b(x)$, решающее правило не используется.

Линейная композиция базовых алгоритмов:

$$b(x) = \sum_{t=1}^T \alpha_t b_t(x), \quad x \in X, \quad \alpha_t \in \mathbb{R}_+.$$

Функционал качества с произвольной функцией потерь $\mathcal{L}(b, y)$:

$$Q(\alpha, b) = \sum_{i=1}^{\ell} \mathcal{L}\left(\underbrace{\sum_{t=1}^{T-1} \alpha_t b_t(x_i)}_{u_{T-1,i}} + \alpha b(x_i), y_i\right) \rightarrow \min_{\alpha, b}.$$

$u_{T,i}$

Ищем вектор $u = (b(x_i))_{i=1}^{\ell}$ из R^ℓ , минимизирующий $Q(\alpha, b)$.

$u_{T-1} = (u_{T-1,i})_{i=1}^{\ell}$ — текущее приближение вектора u

$u_T = (u_{T,i})_{i=1}^{\ell}$ — следующее приближение вектора u

- Градиентный метод минимизации $Q(u) \rightarrow \min, u \in \mathbb{R}^\ell$:

$u_0 :=$ начальное приближение;

$$u_{T,i} := \textcolor{red}{u_{T-1,i}} - \alpha g_i, \quad i = 1, \dots, \ell;$$

$g_i = \mathcal{L}'(u_{T-1,i}, y_i)$ — компоненты вектора градиента,
 α — градиентный шаг.

- Добавление базового алгоритма b_T :

$$u_{T,i} := \textcolor{red}{u_{T-1,i}} + \alpha b_T(x_i), \quad i = 1, \dots, \ell$$

Будем искать такой базовый алгоритм b_T , чтобы
вектор $(b_T(x_i))_{i=1}^\ell$ приближал вектор антиградиента $(-g_i)_{i=1}^\ell$:

$$b_T := \arg \max_b \sum_{i=1}^{\ell} (b(x_i) + g_i)^2$$

Алгоритм градиентного бустинга (Gradient Boosting)

Вход: обучающая выборка X^ℓ ; **параметр T** ;

Выход: базовые алгоритмы и их веса $\alpha_t b_t$, $t = 1, \dots, T$;

1: инициализация: $u_i := 0$, $i = 1, \dots, \ell$;

2: **для всех** $t = 1, \dots, T$

3: найти базовый алгоритм, приближающий градиент:

$$b_t := \arg \min_b \sum_{i=1}^{\ell} (b(x_i) + \mathcal{L}'(u_i, y_i))^2;$$

4: решить задачу одномерной минимизации:

$$\alpha_t := \arg \min_{\alpha > 0} \sum_{i=1}^{\ell} \mathcal{L}(u_i + \alpha b_t(x_i), y_i);$$

5: обновить значения композиции на объектах выборки:

$$u_i := u_i + \alpha_t b_t(x_i); \quad i = 1, \dots, \ell;$$

Известно, что рандомизации могут повышать качество композиции за счёт повышения различности базовых алгоритмов (на этом основаны bagging, RF, RSM)

Идея:

на шагах 3–5 использовать не всю выборку X^ℓ ,
а случайную подвыборку с повторениями, как в бэггинге.

Преимущества:

- улучшается качество
- улучшается сходимость
- уменьшается время обучения

Исторически первый вариант бустинга (1995).

Задача классификации на два класса, $Y = \{-1, +1\}$,

$\mathcal{L}(b(x_i), y_i) = e^{-b(x_i)y_i}$ — экспоненциальная функция потерь,
убывающая функция отступа $M_i = b(x_i)y_i$

Преимущества:

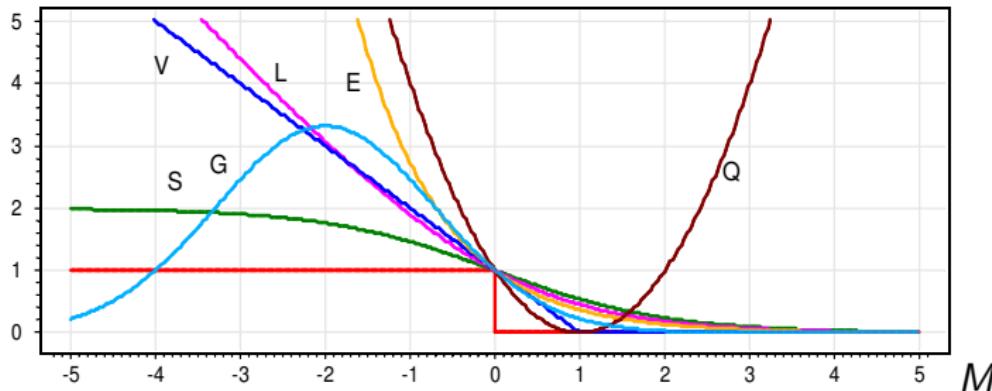
- для обучения b_t на каждом шаге t решается стандартная задача минимизации взвешенного эмпирического риска
- задача оптимизации α_t решается аналитически

Недостаток:

- AdaBoost слишком чувствителен к выбросам из-за экспоненциального роста функции потерь при $M_i < 0$

Частные случаи при различных функциях потерь \mathcal{L}

Функции потерь $\mathcal{L}(M)$ в задачах классификации на два класса



$E(M) = e^{-M}$ — экспоненциальная (AdaBoost);

$L(M) = \log_2(1 + e^{-M})$ — логарифмическая (LogitBoost);

$G(M) = \exp(-cM(M + s))$ — гауссовская (BrownBoost);

$Q(M) = (1 - M)^2$ — квадратичная;

$S(M) = 2(1 + e^M)^{-1}$ — сигмоидная;

$V(M) = (1 - M)_+$ — кусочно-линейная (SVM);

Решающее дерево — это кусочно-постоянная функция:

$$b(x) = \sum_{t=1}^T \alpha_t [x \in \Omega_t],$$

где T — число листьев,

Ω_t — область t -го листа,

α_t — прогноз в t -м листе.

Идея: каждый лист — базовый алгоритм $b_t(x) = [x \in \Omega_t]$;
градиентным шагом определяется прогноз α_t в t -м листе:

$$\alpha_t = \arg \min_{\alpha > 0} \underbrace{\sum_{x_i \in \Omega_t} \mathcal{L}(u_{t-1,i} + \textcolor{red}{\alpha}, y_i)}_{\text{суммарная потеря в } t\text{-м листе}}.$$

После определения всех α_t можно добавить в композицию
следующее дерево, оптимизировав его структуру по MSE.

Оптимизация прогнозов в листьях:

$$\alpha_t = \arg \min_{\alpha > 0} \sum_{x_i \in \Omega_t} \mathcal{L}(u_{t-1,i} + \alpha, y_i).$$

Для некоторых функций потерь решение находится аналитически:

- средний квадрат ошибок, MSE, $\mathcal{L}(b, y) = (b - y)^2$:

$$\alpha_t = \frac{1}{|\Omega_t|} \sum_{x_i \in \Omega_t} (y_i - u_{t-1,i}).$$

- средняя абсолютная ошибка, MAE, $\mathcal{L}(b, y) = |b - y|$:

$$\alpha_t = \operatorname{median}_{x_i \in \Omega_t} \{y_i - u_{t-1,i}\}.$$

В общем случае аналитического решения нет.

- Градиентный бустинг — наиболее общий из всех бустингов:
 - произвольная функция потерь
 - произвольное пространство оценок R
 - подходит для регрессии, классификации, ранжирования
- Важное открытие середины 90-х: обобщающая способность бустинга не ухудшается с ростом сложности T
- Стохастический вариант SGB — лучше и быстрее
- Градиентный бустинг над решающими деревьями часто работает лучше, чем случайный лес
- Технология Yandex.MatrixNet — это градиентный бустинг над «небрежными» решающими деревьями ODT (ODT — oblivious decision tree)