Продвинутые методы ансамблирования

К. В. Воронцов
vokov@forecsys.ru

Этот курс доступен на странице вики-ресурса
http://www.MachineLearning.ru/wiki
«Машинное обучение (курс лекций, К.В.Воронцов)»

МФТИ • 22 апреля 2021

Содержание

Напоминание. Определение ансамбля

$X^\ell = (x_i, y_i)_{i=1}^\ell \subset X \times Y$ — обучающая выборка, $y_i = y^*(x_i)$

$a_t\colon X \to Y,\ \ t = 1, \dots, T$ — обучаемые *базовые алгоритмы*

Идея ансамбля: возможно ли из множества плохих алгоритмов a_t построить один хороший?

Декомпозиция базовых алгоритмов $a_t(x) = C(b_t(x))$
$a_t\colon X \xrightarrow{b_t} R \xrightarrow{C} Y$, где R — более удобное *пространство оценок*,
C — *решающее правило*, как правило, весьма простого вида

Ансамбль базовых алгоритмов $b_1, \dots, b_T$:

$$a(x) = C\bigl(F(b_1(x), \dots, b_T(x), x)\bigr),$$

$F\colon R^T \times X \to R$ — *агрегирующая функция* или *мета-алгоритм*

Напоминание. Агрегирующие (корректирующие) функции

Общие требования к агрегирующей функции:

- $F(b_1, \ldots, b_T, x) \in \left[\min_t b_t, \max_t b_t\right]$ — среднее по Коши $\forall x$
- $F(b_1, \ldots, b_T, x)$ монотонно не убывает по всем b_t

Примеры агрегирующих функций:

- простое голосование (simple voting):

  $$F(b_1, \ldots, b_T) = \frac{1}{T} \sum_{t=1}^{T} b_t$$

- взвешенное голосование (weighted voting):

  $$F(b_1, \ldots, b_T) = \sum_{t=1}^{T} \alpha_t b_t, \quad \sum_{t=1}^{T} \alpha_t = 1, \quad \alpha_t \geqslant 0$$

- смесь алгоритмов (mixture of experts)
  с функциями компетентности (gating function) $g_t \colon X \to \mathbb{R}$

  $$F(b_1, \ldots, b_T, x) = \sum_{t=1}^{T} g_t(x) b_t(x)$$

Анализ смещения–разброса (bias–variance)

Задача регрессии: $Y = \mathbb{R}$
Квадратичная функция потерь: $\mathscr{L}(a, y) = \bigl(a(x) - y\bigr)^2$
Вероятностная постановка: $X^\ell = (x_i, y_i)_{i=1}^\ell \sim p(x, y)$
Метод обучения: $\mu\colon 2^X \to A$, т.е. выборка $\mapsto$ алгоритм

Задача минимизации среднеквадратичного риска:

$$R(a) = \mathsf{E}_{x,y}(a(x) - y)^2 = \int_X \int_Y \bigl(a(x) - y\bigr)^2 p(x, y)\, dx\, dy \to \min_a$$

Идеальный минимизатор среднеквадратичного риска:

$$a^*(x) = \mathsf{E}(y|x) = \int_Y y\, p(y|x)\, dy$$

Основная мера качества метода обучения μ:

$$Q(\mu) = \mathsf{E}_{X^\ell} \mathsf{E}_{x,y} \bigl(\mu(X^\ell)(x) - y\bigr)^2$$

Разложение ошибки на шум, смещение и разброс

Теорема

В случае квадратичной функции потерь для любого μ

$$Q(\mu) = \underbrace{\mathsf{E}_{x,y}\bigl(a^*(x) - y\bigr)^2}_{\text{шум (noise)}} +$$

$$+ \underbrace{\mathsf{E}_{x,y}\bigl(\bar{a}(x) - a^*(x)\bigr)^2}_{\text{смещение (bias)}} +$$

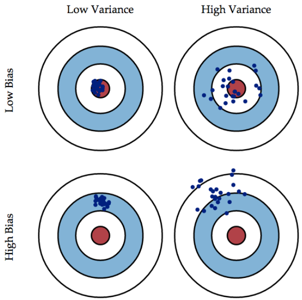
$$+ \underbrace{\mathsf{E}_{x,y}\mathsf{E}_{X^\ell}\bigl(\mu(X^\ell)(x) - \bar{a}(x)\bigr)^2}_{\text{разброс (variance)}}$$

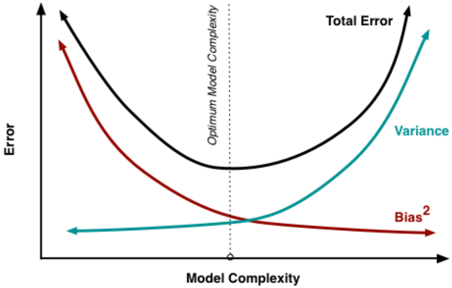
$\bar{a}(x) = \mathsf{E}_{X^\ell}\bigl(\mu(X^\ell)(x)\bigr)$ — средний ответ обученного алгоритма

Разложение ошибки на шум, смещение и разброс

Качественное понимание: по мере роста сложности модели

- смещение (bias) уменьшается
- разброс (variance) увеличивается





Анализ смещения–разброса для простого голосования

Обучение базовых алгоритмов по случайным подвыборкам:
$b_t = \mu(X_t^k),\;\; X_t^k \sim X^\ell,\;\; t = 1, \dots, T$

Ансамбль — простое голосование: $a_T(x) = \frac{1}{T}\sum_{t=1}^{T} b_t(x)$

Смещение ансамбля совпадает со смещением отдельного базового алгоритма:

$$\text{bias} = \mathsf{E}_{x,y}\bigl(a^*(x) - \mathsf{E}_{X^\ell} b_t(x)\bigr)^2$$

Разброс состоит из дисперсии и различности (ковариации):

$$\text{variance} = \tfrac{1}{T}\mathsf{E}_{x,y}\mathsf{E}_{X^\ell}\bigl(b_t(x) - \mathsf{E}_{X^\ell} b_t(x)\bigr)^2 +$$
$$+ \tfrac{T-1}{T}\mathsf{E}_{x,y}\mathsf{E}_{X^\ell}\bigl(b_t(x) - \mathsf{E}_{X^\ell} b_t(x)\bigr)\bigl(b_s(x) - \mathsf{E}_{X^\ell} b_s(x)\bigr)$$

Почему сложные ансамбли не переобучаются?

С позиций анализа отступов:

- ансамблирование не увеличивает сложность модели
- но с каждой итерацией увеличивает зазор между классами

С позиций анализа смещения–разброса:

- разнообразие базовых алгоритмов уменьшает разброс
- бэггинг уменьшает только разброс
- бустинг уменьшает и смещение, и разброс

Практическое сравнение: boosting / bagging / RSM

- бустинг лучше для классов с границами сложной формы
- бэггинг и RSM лучше для коротких обучающих выборок
- RSM лучше, когда много неинформативных признаков
- бэггинг параллельно обучает базовые алгоритмы b_t
- бустинг обучает каждый b_t параллельно по частям выборки

Градиентный бустинг для произвольной функции потерь

Линейный ансамбль базовых алгоритмов b_t из семейства $\mathscr{B}$:

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

Эвристика: обучаем α_T, b_T при фиксированных предыдущих.
Критерий качества с заданной гладкой функцией потерь $\mathscr{L}(b, y)$:

$$Q(\alpha, b; X^\ell) = \sum_{i=1}^{\ell} \mathscr{L}\Bigl(\underbrace{\underbrace{\sum_{t=1}^{T-1} \alpha_t b_t(x_i)}_{f_{T-1,i}} + \alpha b(x_i)}_{f_{T,i}}, y_i\Bigr) \to \min_{\alpha, b}.$$

$f_{T-1} = \bigl(f_{T-1,i}\bigr)_{i=1}^{\ell}$ — вектор текущего приближения
$f_T = \bigl(f_{T,i}\bigr)_{i=1}^{\ell}$ — вектор следующего приближения

G.Friedman. Greedy function approximation: a gradient boosting machine. 1999.

Параметрическая аппроксимация градиентного шага

Градиентный метод минимизации $Q(f) \to \min$, $f \in \mathbb{R}^\ell$:

$$f_0 := \text{начальное приближение;}$$
$$f_{T,i} := f_{T-1,i} - \alpha g_i, \quad i = 1, \dots, \ell;$$

$g_i = \mathscr{L}'_f\bigl(f_{T-1,i},\, y_i\bigr)$ — компоненты вектора градиента,
α — градиентный шаг.

Это очень похоже на добавление одного базового алгоритма:

$$f_{T,i} := f_{T-1,i} + \alpha b(x_i), \quad i = 1, \dots, \ell$$

Идея: будем искать такой базовый алгоритм $b_T \in \mathscr{B}$, чтобы вектор $\bigl(b_T(x_i)\bigr)_{i=1}^{\ell}$ приближал вектор антиградиента $\bigl(-g_i\bigr)_{i=1}^{\ell}$:

$$b_T := \arg\min_{b \in \mathscr{B}} \sum_{i=1}^{\ell} \bigl(b(x_i) + g_i\bigr)^2$$

Алгоритм градиентного бустинга (Gradient Boosting)

Вход: обучающая выборка X^ℓ; параметр T;

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

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

для всех $t = 1, \dots, T$

базовый алгоритм, приближающий антиградиент:

$$b_t := \arg\min_{b \in \mathscr{B}} \sum_{i=1}^{\ell} \bigl(b(x_i) + \mathscr{L}'(f_i, y_i)\bigr)^2;$$

задача одномерной минимизации:

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

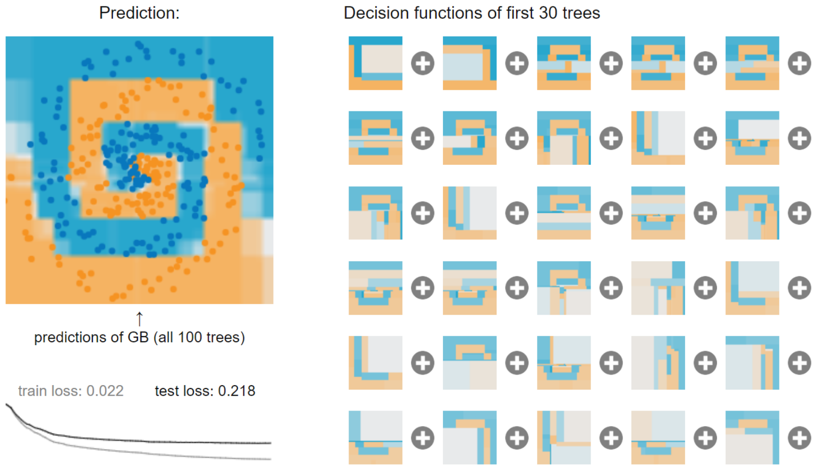
обновление вектора значений на объектах выборки:

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

Каждый следующий базовый алгоритм обучается так, чтобы по возможности исправить ошибки предыдущих алгоритмов.

Пример. Классификация синтетической выборки

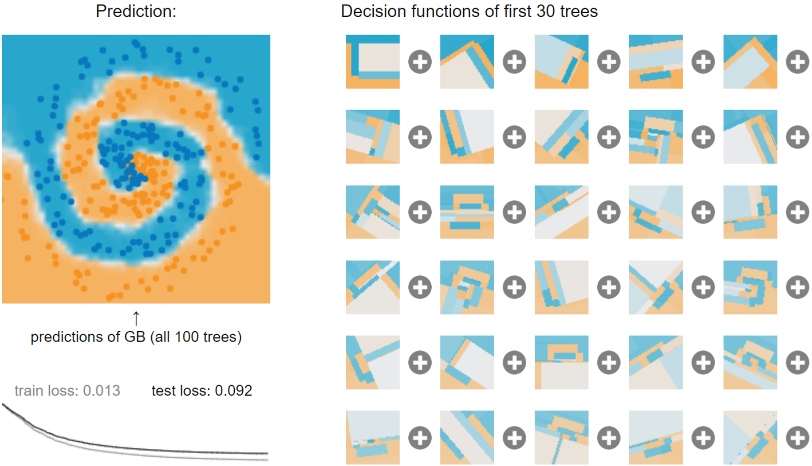
100 деревьев глубины 5



http://arogozhnikov.github.io/2016/07/05/gradient_boosting_playground.html

Пример. Классификация синтетической выборки

100 деревьев глубины 5, с подбором вращения каждого дерева



http://arogozhnikov.github.io/2016/07/05/gradient_boosting_playground.html

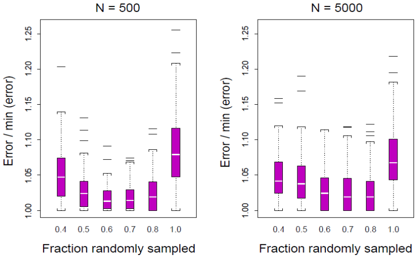
Стохастический градиентный бустинг (SGB)

Идея: при оптимизации b_t и α_t использовать не всю выборку X^ℓ, а случайную подвыборку, по аналогии с бэггингом

Преимущества:

- улучшается сходимость, уменьшается время обучения
- улучшается обобщающая способность ансамбля
- можно использовать несмещённые оценки out-of-bag

Эксперименты:
относительная ошибка при различном объёме выборки N



Вывод:
оптимально сэмплировать около 60–80% выборки

Friedman G. Stochastic Gradient Boosting. 1999.

Частные случаи GB: регрессия, AdaBoost и другие

Регрессия: $\mathscr{L}(b, y) = (b - y)^2$

- $b_T(x)$ обучается на разностях $y_i - \sum\limits_{t=1}^{T-1} \alpha_t b_t(x_i)$
- если регрессии b_t линейные, то α_t можно не обучать.

Классификация: $\mathscr{L}(b, y) = e^{-by}$, $b_t \in \{-1, 0, +1\}$

- GB в точности совпадает с AdaBoost [Freund, 1995]

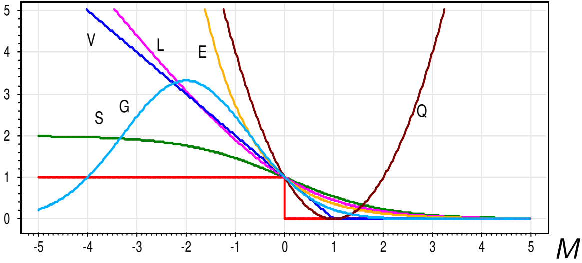
Классификация: $\mathscr{L}(b, y) = \mathcal{L}(-by)$, $b_t \in \mathbb{R}$

- GB совпадает с AnyBoost [Mason, 2000]

Y.Freund, R.Schapire. A decision-theoretic generalization of online learning and an application to boosting. 1995.
L.Mason et al. Boosting algorithms as gradient descent. 2000.

Варианты бустинга для двухклассовой классификации

Гладкие аппроксимации пороговой функции потерь $[M < 0]$:



$E(M) = e^{-M}$ — экспоненциальная (AdaBoost);
$L(M) = \log_2(1 + e^{-M})$ — логарифмическая (LogitBoost);
$Q(M) = (1 - M)^2$ — квадратичная (GentleBoost);
$G(M) = \exp\bigl(-cM(M + s)\bigr)$ — гауссовская (BrownBoost);
$S(M) = 2(1 + e^{M})^{-1}$ — сигмоидная;
$V(M) = (1 - M)_+$ — кусочно-линейная (из SVM);

XGBoost: популярная и быстрая реализация GB над деревьями

Деревья регрессии и классификации (CART):

$$b(x, w) = \sum_{k \in K} w_k B_k(x)$$

где $B_k(x)$ — бинарный индикатор $[x$ попадает в лист $k]$,
w_k — значение в листе k, K — множество листьев дерева.
Для любого x одно и только одно слагаемое не равно нулю.

Критерий качества с суммой L_0 и L_2 регуляризаторов:

$$Q(w) = \sum_{i=1}^{\ell} \mathscr{L}\bigl(a(x_i) + b(x_i, w), y_i\bigr) + \gamma|K| + \frac{\lambda}{2} \sum_{k \in K} w_k^2 \;\to\; \min_w,$$

где $a(x_i) = \sum\limits_{t=1}^{T-1} \alpha_t b_t(x_i)$ — ранее построенная часть ансамбля.
В некоторых случаях задача имеет аналитическое решение.

XGBoost: приближённое аналитическое решение для w_j

Приблизим $\mathscr{L}(a+b,y) \approx \mathscr{L}(a,y) + b\mathscr{L}'(a,y) + \frac{b^2}{2}\mathscr{L}''(a,y)$:

$$\Phi(w) = \sum_{i=1}^{\ell}\Bigl(g_i b_i + \frac{1}{2}h_i b_i^2\Bigr) + \gamma|K| + \frac{\lambda}{2}\sum_{k\in K} w_k^2 \;\to\; \min_w,$$

где $b_i = \sum_k w_k B_k(x_i)$, $g_i = \mathscr{L}'\bigl(a(x_i),y_i\bigr)$, $h_i = \mathscr{L}''\bigl(a(x_i),y_i\bigr)$.

Из условий $\frac{\partial\Phi(w)}{\partial w_k} = 0$ находим оптимальное значение листа k:

$$w_k = -\frac{\sum_i g_i B_k(x_i)}{\lambda + \sum_i h_i B_k(x_i)}$$

Подставляя w_k обратно в $\Phi(w)$, выводим критерий ветвления:

$$\Phi(B_1,\dots,B_k) = -\frac{1}{2}\sum_{k\in K}\frac{\bigl(\sum_i g_i B_k(x_i)\bigr)^2}{\lambda + \sum_i h_i B_k(x_i)} + \gamma|K| \;\to\; \min$$

XGBoost и другие варианты GB

Преимущества XGBoost (eXtreme Gradient Boosting):

- L_2 регуляризация сокращает переобучение
- L_0 регуляризация упрощает деревья (pruning)
- как и общий GB, допускает произвольные функции потерь
- очень быстрая реализация за счёт аналитических формул
- имеет механизм обработки пропущенных значений

Что ещё бывает:

- Light GBM — для обучения на сверхбольших данных
- Яндекс.MatrixNet — GB над Oblivious Decision Tree
- Яндекс.CatBoost — для категориальных признаков

Основные мотивации CatBoost

Две проблемы:

- Переобучение (смещённость, target leakage) в градиентах: $g_i = \mathscr{L}'\bigl(a_{t-1}(x_i), y_i\bigr)$ вычисляются в тех же точках x_i, по которым ансамбль $a_{t-1}(x)$ обучался аппроксимировать y_i
- Надо обрабатывать категориальные признаки с большим числом редких значений (пользователь, регион, город, реклама, рекламодатель, товар, документ, автор, и т.д.)

Приём, похожий на Out-Of-Bag и на онлайновые методы:

- для получения несмещённых оценок на объекте x_i хранить и достраивать ансамбль на выборке без этого объекта
- Как сделать, чтобы этих выборок было не слишком много?
- Как сделать, чтобы они не сильно перекрывались?

L.Prokhorenkova et al. CatBoost: unbiased boosting with categorical features. 2019.

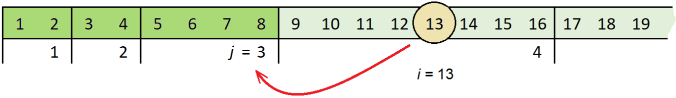
Упорядоченный бустинг (ordered boosting)

Идеи:

- вычислять g_i по модели a_{t-1}, которая не обучалась на x_i
- строить обучающие подвыборки удваивающейся длины
- построить много таких случайно перемешанных выборок

Обозначения:

$\sigma_1, \ldots, \sigma_s$ — случайные перестановки выборки X^ℓ
X^{rj} — подвыборка первых 2^j объектов из $\sigma_r(X^\ell)$
a_t^{rj} — модель (ансамбль-полуфабрикат), обученный по X^{rj}
$g_{ti} = \mathscr{L}'\bigl(a_{t-1}^{rj}(x_i), y_i\bigr)$ — градиент в точке (x_i, y_i) для модели, которая по ней не обучалась; для этого $j = \lfloor \log_2(i-1) \rfloor$



L.Prokhorenkova et al. CatBoost: unbiased boosting with categorical features. 2019.

Модификация градиентного бустинга

сгенерировать случайные перестановки $\sigma_0, \sigma_1, \dots, \sigma_s$;

для всех $t = 1, \dots, T$:

- выбрать перестановку σ_r случайно из $\sigma_1, \dots, \sigma_s$;
- вычислить несмещённый вектор градиента g_{ti}, $i = 1, \dots, \ell$;
- $b_t := \arg\min\limits_b \sum\limits_{i=1}^{\ell} \bigl(b(x_i) + g_{ti}\bigr)^2$, где в критерии ветвления слагаемые объектов x_i вычисляются по X^{rj};
- **для всех** деревьев b_t^{rj}:
  - скопировать общую для них структуру дерева из b_t;
  - вычислить значения в листьях по X^{rj};
- вычислить значения в листьях для b_t по X^{0j};
- вычислить α_t и обновить $f_i := f_i + \alpha_t b_t(x_i)$;

Способы обработки категориальных признаков

Пусть V — множество (словарь) значений признака $f(x)$

Стандартные методы либо громоздкие, либо переобучаются:

- бинаризация (one-hot encoding): $b_v(x) = [f(x) = v]$
- группирование (кластеризация) значений (LightGBM)
- статистика по целевому признаку (target statistics, TS):
  $$\tilde{f}(x) = \frac{\sum_{i=1}^{\ell}[f(x_i) = f(x)]y_i + \gamma p}{\sum_{i=1}^{\ell}[f(x_i) = f(x)] + \gamma}$$

CatBoost:

- статистика TS также вычисляется по перестановкам X^{rj}:
  $$\tilde{f}(x) = \frac{\sum_{x_i \in X^{rj}}[f(x_i) = f(x)]y_i + \gamma p}{\sum_{x_i \in X^{rj}}[f(x_i) = f(x)] + \gamma}$$
- конъюнкции категориальных признаков создаются «налету» в процессе построения деревьев

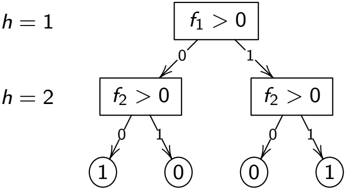
Небрежные решающие деревья (Oblivious Decision Tree, ODT)

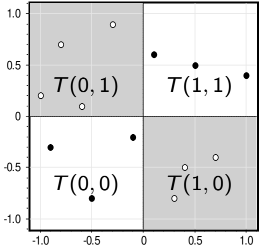
Решающая таблица: дерево глубины H, $D_v = \{0,1\}$;
для всех узлов уровня h условие ветвления $f_h(x)$ *одинаково*;
на уровне h ровно 2^{h-1} вершин; X делится на 2^H ячеек.

Классификатор задаётся *таблицей решений* $T\colon \{0,1\}^H \to Y$:

$$a(x) = T\bigl(f_1(x), \dots, f_H(x)\bigr).$$

Пример: задача XOR, $H = 2$.





R.Kohavi, C.-H.Li. Oblivious decision trees, graphs, and top-down pruning. 1995.

Алгоритм обучения ODT

Вход: выборка X^ℓ; множество признаков F; глубина дерева H;
Выход: признаки f_h, $h = 1, \dots, H$; таблица $T\colon \{0,1\}^H \to Y$;

для всех $h = 1, \dots, H$
   предикат с максимальным выигрышем определённости:
   $f_h := \arg\max\limits_{f \in \mathrm{bin}\{F\}} \mathrm{Gain}\,(f_1, \dots, f_{h-1}, \beta)$;

классификация по мажоритарному правилу:
$T(\beta) := \mathrm{Major}\,(U_{H\beta})$;

Выигрыш от ветвления на уровне h по всей выборке X^ℓ:

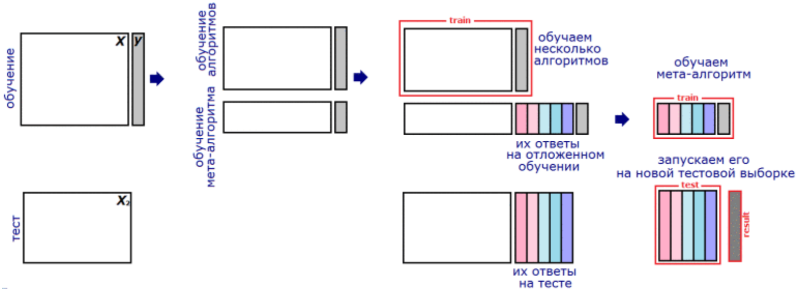
$$\mathrm{Gain}\,(f_1, \dots, f_h) = \Phi(X^\ell) - \sum_{\beta \in \{0,1\}^h} \frac{|U_{h\beta}|}{\ell}\, \Phi(U_{h\beta}),$$

$$U_{h\beta} = \bigl\{x_i \in X^\ell\colon f_s(x_i) = \beta_s,\ s = 1..h\bigr\},\ \ \beta = (\beta_1, \dots, \beta_h) \in \{0,1\}^h.$$

Блендинг (Blending) — смешивание базовых алгоритмов

Идея: базовые алгоритмы $b_t(x)$ как (мета)признаки подаём на вход любому ML алгоритму, не обязательно линейному.

Проблема: этот (мета)алгоритм нельзя обучать на тех же данных, что и базовые $b_t(x)$, будет переобучение!

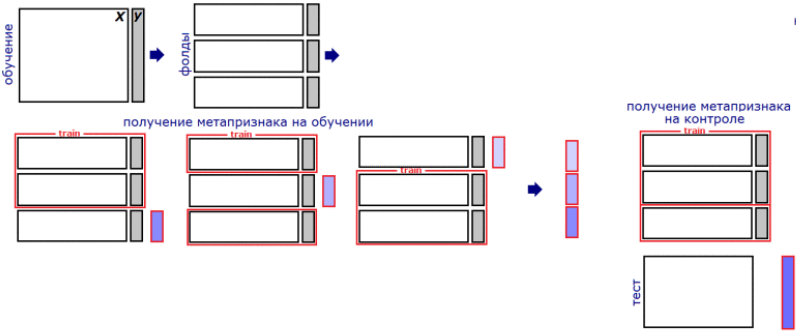


Новая проблема: для обучения используется не вся выборка.

https://dyakonov.org/2017/03/10/стекинг-stacking-и-блендинг-blending

Классический стэкинг (Stacking)

Решение проблемы: разбиение выборки на k блоков (k-fold)



Новая проблема: вместо одного метапризнака $b_t(x)$ имеем k похожих, но разных $b_{tj}(x)$, $j = 1, \dots, k$.

Вариант решения: усреднение метапризнаков $b_t(x) = \frac{1}{k}\sum\limits_{j=1}^{k} b_{tj}(x)$

Линейный взвешенный стэкинг (Feature-Weighted Linear Stacking)

$b(x) = \sum\limits_{t=1}^{T} \alpha_t b_t(x)$ — линейный стэкинг (ридж-регрессия)

$\alpha_t(x) = \sum\limits_{j=1}^{L} v_{tj} f_j(x)$ — теперь веса w_t зависят от x через $f_j(x)$

Критерий оптимизации — ридж-регрессия:

$$Q(v) = \sum_{i=1}^{\ell} \Bigl(\sum_{t=1}^{T} \sum_{j=1}^{L} v_{tj} f_j(x_i) b_t(x_i) - y_i \Bigr)^2 + \frac{\lambda}{2} \sum_{t=1}^{T} \sum_{j=1}^{L} v_{tj}^2 \;\to\; \min_v$$

Метапризнаки f_j могут быть как фиксированными, так и обучаемыми (задача симметрична относительно b_t и f_j)

FWLS использовался командой #2 в конкурсе NetflixPrize

Joseph Sill et al. Feature-Weighted Linear Stacking. 2009.

Квазилинейный ансамбль (смесь алгоритмов)

Смесь алгоритмов (Mixture of Experts)

$$b(x) = \sum_{t=1}^{T} g_t(x) b_t(x),$$

$b_t\colon X \to \mathbb{R}$ — базовый алгоритм,
$g_t\colon X \to \mathbb{R}$ — функция компетентности, шлюз (gate).
Чем больше $g_t(x)$, тем выше доверие к ответу $b_t(x)$.

Условие нормировки: $\sum\limits_{t=1}^{T} g_t(x) = 1$ для любого $x \in X$.

Нормировка «мягкого максимума» SoftMax: $\mathbb{R}^T \to \mathbb{R}^T$:

$$\tilde{g}_t(x) = \mathrm{SoftMax}_t\bigl(g_1(x), \dots, g_T(x); \gamma\bigr) = \frac{e^{\gamma g_t(x)}}{e^{\gamma g_1(x)} + \dots + e^{\gamma g_T(x)}}.$$

При $\gamma \to \infty$ SoftMax выделяет максимальную из T величин.

Вид функций компетентности

Функции компетентности выбираются из содержательных соображений и могут определяться:

- признаком $f(x)$:
$$g(x;\alpha,\beta) = \sigma(\alpha f(x) + \beta), \quad \alpha,\beta \in \mathbb{R};$$
- неизвестным направлением $\alpha \in \mathbb{R}^n$:
$$g(x;\alpha,\beta) = \sigma(x^{\mathsf{T}}\alpha + \beta), \quad \alpha \in \mathbb{R}^n,\ \beta \in \mathbb{R};$$
- расстоянием до неизвестной точки $\alpha \in \mathbb{R}^n$:
$$g(x;\alpha,\beta) = \exp(-\beta\|x-\alpha\|^2), \quad \alpha \in \mathbb{R}^n,\ \beta \in \mathbb{R};$$

где $\alpha,\beta \in \mathbb{R}$ — параметры, *частично* обучаемые по выборке, $\sigma(z) = \frac{1}{1+e^{-z}}$ — сигмоидная функция.

Выпуклые функции потерь

Функция потерь $\mathscr{L}(b, y)$ называется *выпуклой* по b, если $\forall\, y \in Y,\ \ \forall\, b_1, b_2 \in R,\ \ \forall\, g_1, g_2 \geqslant 0\colon\ \ g_1 + g_2 = 1$, выполняется

$$\mathscr{L}(g_1 b_1 + g_2 b_2, y) \leqslant g_1 \mathscr{L}(b_1, y) + g_2 \mathscr{L}(b_2, y).$$

Интерпретация: потери растут не медленнее, чем величина отклонения от правильного ответа y.

Примеры выпуклых функций потерь:

$$\mathscr{L}(b, y) = \begin{cases} (b-y)^2 & \text{— квадратичная (МНК-регрессия);} \\ e^{-by} & \text{— экспоненциальная (AdaBoost);} \\ \log_2(1 + e^{-by}) & \text{— логарифмическая (LR);} \\ (1 - by)_+ & \text{— кусочно-линейная (SVM).} \end{cases}$$

Пример невыпуклой функции потерь: $\mathscr{L}(b, y) = [by < 0]$.

Основная идея применения выпуклых функций потерь

Пусть $\forall x \ \sum\limits_{t=1}^{T} g_t(x) = 1$ и функция потерь $\mathscr{L}$ выпукла.

Тогда $Q(a)$ распадается на T независимых критериев Q_t:

$$Q(a) = \sum_{i=1}^{\ell} \mathscr{L}\biggl(\sum_{t=1}^{T} g_t(x_i) b_t(x_i), y_i\biggr) \leqslant \sum_{t=1}^{T} \underbrace{\sum_{i=1}^{\ell} g_t(x_i) \mathscr{L}\bigl(b_t(x_i), y_i\bigr)}_{Q_t(g_t, b_t)}$$

Итерационный процесс, аналогичный EM-алгоритму:

начальное приближение функций компетентности g_t;
повторять
  M-шаг: $b_t := \arg\min_b Q_t(g_t, b)$ при фиксированных g_t;
  E-шаг: оценить все g_t при фиксированных b_t;
пока значения компетентностей $g_t(x_i)$ не стабилизируются;

Алгоритм ME (Mixture of Experts): обучение смеси алгоритмов

Итерационный процесс, аналогичный EM-алгоритму:

Вход: выборка X^ℓ, начальные $(g_t)_{t=1}^T$, параметры T, δ, γ;
Выход: $g_t(x), b_t(x), \quad t = 1, \dots, T$;
повторять

$\bigl(\tilde{g}_1(x_i), \dots, \tilde{g}_T(x_i)\bigr) := \mathrm{SoftMax}\bigl(g_1(x_i), \dots, g_T(x_i); \gamma\bigr)$;

$\tilde{g}_t^0 := \tilde{g}_t$ для всех $t = 1, \dots, T$;

M-шаг: при фиксированных g_t обучить все b_t:

$b_t := \arg\min\limits_b \sum_{i=1}^\ell \tilde{g}_t(x_i) \mathscr{L}\bigl(b(x_i), y_i\bigr), \quad t = 1, \dots, T$;

E-шаг: при фиксированных b_t оценить все g_t:

$g_t := \arg\min\limits_{g_t} \sum_{i=1}^\ell \mathscr{L}\Bigl(\sum_{s=1}^T \tilde{g}_s(x_i) b_s(x_i), y_i\Bigr), \quad t = 1, \dots, T$;

пока $\max\limits_{t,i} \bigl|\tilde{g}_t(x_i) - \tilde{g}_t^0(x_i)\bigr| > \delta$;

Обучение смеси с автоматическим определением числа T

Вход: выборка X^ℓ, параметры ℓ_0, $\mathscr{L}_0$, δ, γ;

Выход: T, $g_t(x), b_t(x)$, $\; t = 1, \dots, T$;

начальное приближение:

$$b_1 := \arg\min_b \sum_{i=1}^{\ell} \mathscr{L}\bigl(b(x_i), y_i\bigr), \qquad g_1(x_i) := 1, \;\; i = 1, \dots, \ell;$$

для всех $t = 2, \dots, T$

- множество «трудных» объектов:
  $X_t := \bigl\{x_i \colon \mathscr{L}(a_{t-1}(x_i), y_i) > \mathscr{L}_0\bigr\}$;
- **если** $|X_t| \leqslant \ell_0$ **то выход**;
- $b_t := \arg\min_b \sum_{x_i \in X_t} \mathscr{L}\bigl(b(x_i), y_i\bigr)$;
- $g_t := \arg\min_{g_t} \sum_{i=1}^{\ell} \mathscr{L}\Bigl(\sum_{s=1}^{t} g_s(x_i) b_s(x_i), y_i\Bigr)$;
- $(g_s, b_s)_{s=1}^t := \mathrm{ME}\bigl(X^\ell, (g_s)_{s=1}^t, t, \delta, \gamma\bigr)$;

Резюме

- Ансамбли позволяют решать сложные задачи, которые плохо решаются отдельными базовыми алгоритмами.
- Обучать ансамбль целиком слишком сложно.
  Поэтому обучаем базовые алгоритмы по одному.
- Важное открытие середины 90-х: обобщающая способность бустинга не ухудшается с ростом сложности T.
- Градиентный бустинг — наиболее общий из всех бустингов:
  — произвольная функция потерь
  — произвольное пространство оценок R
  — подходит для регрессии, классификации, ранжирования
- Чаще всего GB применяется к решающим деревьям
- RF и SGB — универсальные модели машинного обучения
- FWLS и ME — квазилинейные ансамбли, $\alpha_t(x)$
- Смеси алгоритмов нужна хорошая модель компетентности