

Обучение ансамблей

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 • 21 ноября 2020

1 Взвешенное голосование

- Ещё одно теоретическое обоснование ансамблей
- Градиентный бустинг
- Варианты градиентного бустинга

2 Алгоритм CatBoost

- Упорядоченный бустинг
- Категориальные признаки
- Небрежные решающие деревья

3 Нелинейное ансамблирование

- Стэкинг
- Линейный стэкинг, взвешенный по признакам
- Смеси с функциями компетентности

Напоминание. Определение ансамбля

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

$a_t: X \rightarrow Y$, $t = 1, \dots, T$ — обучаемые базовые алгоритмы

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

Декомпозиция базовых алгоритмов $a_t(x) = C(b_t(x))$

$a_t: X \xrightarrow{b_t} R \xrightarrow{C} Y$, где R — более удобное пространство оценок, C — решающее правило, как правило, весьма простого вида

Ансамбль базовых алгоритмов b_1, \dots, b_T :

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

$F: R^T \times X \rightarrow R$ — агрегирующая функция или мета-алгоритм

Напоминание. Агрегирующие (корректирующие) функции

Общие требования к агрегирующей функции:

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

Примеры агрегирующих функций:

- простое голосование (simple voting):

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

- взвешенное голосование (weighted voting):

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

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

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

Анализ смещения–разброса (bias–variance)

Задача регрессии: $Y = \mathbb{R}$

Квадратичная функция потерь: $\mathcal{L}(a, y) = (a(x) - y)^2$

Вероятностная постановка: $X^\ell = (x_i, y_i)_{i=1}^\ell \sim p(x, y)$

Метод обучения: $\mu: 2^X \rightarrow A$, т.е. выборка \mapsto алгоритм

Задача минимизации среднеквадратичного риска:

$$R(a) = E_{x,y}(a(x) - y)^2 = \int_X \int_Y (a(x) - y)^2 p(x, y) dx dy \rightarrow \min_a$$

Идеальный минимизатор среднеквадратичного риска:

$$a^*(x) = E(y|x) = \int_Y y p(y|x) dy$$

Основная мера качества метода обучения μ :

$$Q(\mu) = E_{X^\ell} E_{x,y}(\mu(X^\ell)(x) - y)^2$$

Разложение ошибки на шум, смещение и разброс

Теорема

В случае квадратичной функции потерь для любого μ

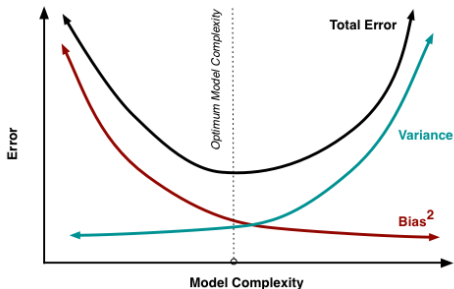
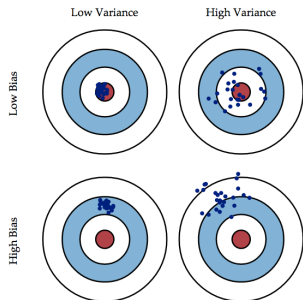
$$\begin{aligned}
 Q(\mu) = & \underbrace{E_{x,y}(a^*(x) - y)^2}_{\text{шум (noise)}} + \\
 & + \underbrace{E_{x,y}(\bar{a}(x) - a^*(x))^2}_{\text{смещение (bias)}} + \\
 & + \underbrace{E_{x,y}E_{X^\ell}(\mu(X^\ell)(x) - \bar{a}(x))^2}_{\text{разброс (variance)}}
 \end{aligned}$$

$\bar{a}(x) = E_{X^\ell}(\mu(X^\ell)(x))$ — 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:

$$b_t = \mu(X_t^k), \quad X_t^k \sim X^\ell, \quad t = 1, \dots, T$$

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

Смещение ансамбля совпадает со смещением отдельного базового алгоритма:

$$\text{bias} = E_{x,y} (a^*(x) - E_{X^\ell} b_t(x))^2$$

Разброс состоит из дисперсии и различности (ковариации):

$$\begin{aligned} \text{variance} = & \frac{1}{T} E_{x,y} E_{X^\ell} (b_t(x) - E_{X^\ell} b_t(x))^2 + \\ & + \frac{T-1}{T} E_{x,y} E_{X^\ell} (b_t(x) - E_{X^\ell} b_t(x)) (b_s(x) - E_{X^\ell} b_s(x)) \end{aligned}$$

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 и RSM распараллеливаются легче, чем бустинг

Градиентный бустинг для произвольной функции потерь

Линейная (выпуклая) комбинация базовых алгоритмов:

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

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

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

$\underbrace{\hspace{10em}}_{f_{T,i}}$

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

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

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

Параметрическая аппроксимация градиентного шага

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

f_0 := начальное приближение;

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

$g_i = \mathcal{L}'(f_{T-1,i}, y_i)$ — 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 , чтобы вектор $(b_T(x_i))_{i=1}^\ell$ приближал вектор антиградиента $(-g_i)_{i=1}^\ell$:

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

Алгоритм градиентного бустинга (Gradient Boosting)

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

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

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

для всех $t = 1, \dots, T$

базовый алгоритм, приближающий антиградиент:

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

задача одномерной минимизации:

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

обновление вектора значений на объектах выборки:

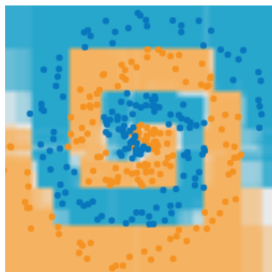
$$f_i := f_i + \alpha_t b_t(x_i); \quad i = 1, \dots, \ell;$$

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

Пример. Классификация синтетической выборки

100 деревьев глубины 5

Prediction:

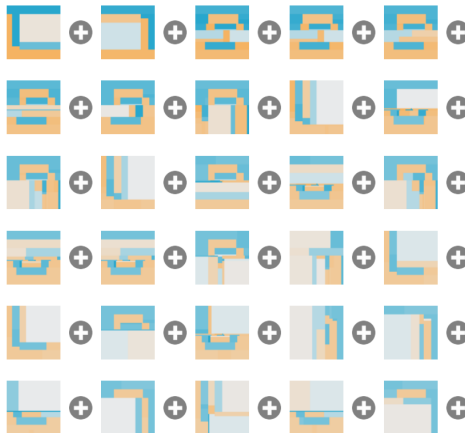


↑
predictions of GB (all 100 trees)

train loss: 0.022 test loss: 0.218



Decision functions of first 30 trees

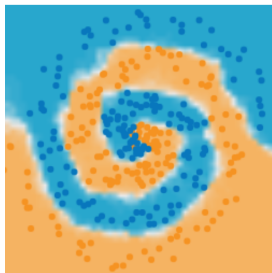


http://arogozhnikov.github.io/2016/07/05/gradient_boosting_playground.html

Пример. Классификация синтетической выборки

100 деревьев глубины 5, с подбором вращения каждого дерева

Prediction:

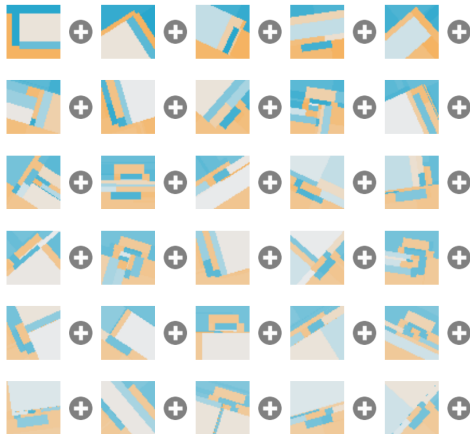


↑
predictions of GB (all 100 trees)

train loss: 0.013 test loss: 0.092



Decision functions of first 30 trees



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

Стохастический градиентный бустинг (SGB)

Идея: на шагах 3–5 использовать не всю выборку X^ℓ , а случайную подвыборку, по аналогии с бэггингом

Преимущества:

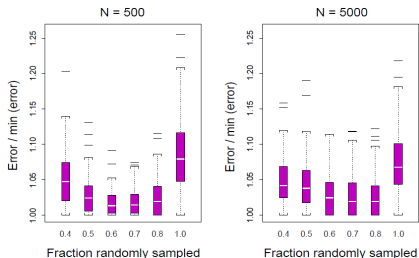
- улучшается сходимость, уменьшается время обучения
- улучшается обобщающая способность ансамбля
- можно использовать несмещённые оценки out-of-bag

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

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

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

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

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

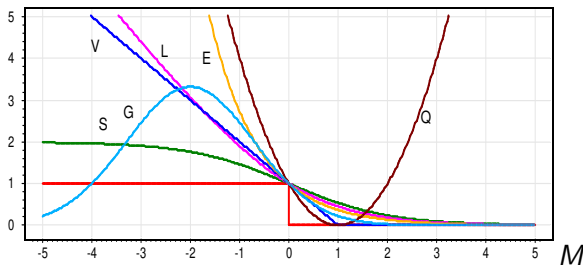
- GB совпадает с AnyBoost [Mason, 2000]

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);

$G(M) = \exp(-cM(M + s))$ — 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:

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

где $a(x_i) = \sum_{t=1}^{T-1} \alpha_t b_t(x_i)$ — ранее построенная часть ансамбля.

В некоторых случаях задача имеет аналитическое решение.

XGBoost: приближённое аналитическое решение для w_j

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

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

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

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:

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

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: $g_i = \mathcal{L}'(a_{t-1}(x_i), y_i)$ 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 онлайнные 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

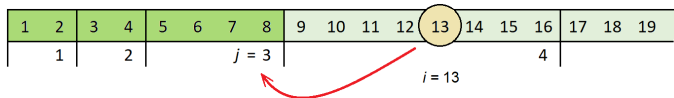
Обозначения:

$\sigma_1, \dots, \sigma_s$ — 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}

$g_{ti} = \mathcal{L}'(a_{t-1}^{rj}(x_i), y_i)$ — градиент в точке (x_i, y_i) для модели, которая по ней не обучалась; для этого $j = \lfloor \log_2(i-1) \rfloor$



Модификация градиентного бустинга

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

для всех $t = 1 \dots T$:

выбрать перестановку σ_r ;

вычислить несмещённый вектор градиента g_{ti} , $i = 1 \dots \ell$;

$b_t := \arg \min_b \sum_{i=1}^{\ell} (b(x_i) + g_{ti})^2$, 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_j ;

Способы обработки категориальных признаков

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

Стандартные методы либо громоздки, 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

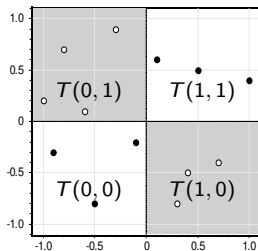
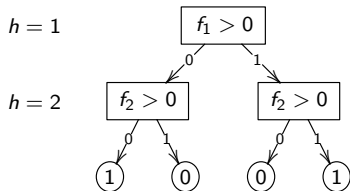
Небрежные решающие деревья (Oblivious Decision Tree, ODT)

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* $T: \{0, 1\}^H \rightarrow Y$:

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

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 $T: \{0, 1\}^H \rightarrow Y$;

для всех $h = 1, \dots, H$

 предикат с максимальным выигрышем определённости:

$$f_h := \arg \max_{f \in \text{bin}\{F\}} \text{Gain}(f_1, \dots, f_{h-1}, \beta);$$

 классификация по мажоритарному правилу:

$$T(\beta) := \text{Major}(U_{H\beta});$$

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

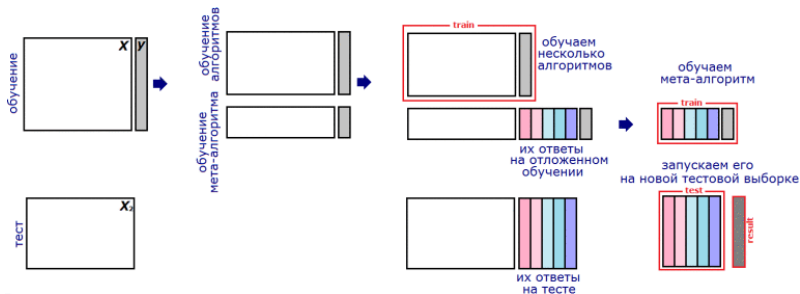
$$\text{Gain}(f_1, \dots, f_h) = \Phi(X^\ell) - \sum_{\beta \in \{0,1\}^h} \frac{|U_{h\beta}|}{\ell} \Phi(U_{h\beta}),$$

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

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!

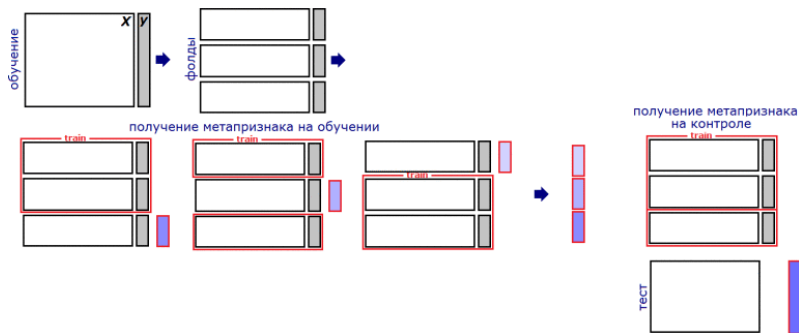


Новая проблема: для обучения используется не вся выборка.

<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 $b_t(x) = \frac{1}{k} \sum_{j=1}^k b_{tj}(x)$

Линейный взвешенный стэкинг (Feature-Weighted Linear Stacking)

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

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

Критерий оптимизации — ридж-регрессия:

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

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)

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

$b_t: X \rightarrow \mathbb{R}$ — базовый алгоритм,

$g_t: X \rightarrow \mathbb{R}$ — функция компетентности, шлюз (gate).

Чем больше $g_t(x)$, тем выше доверие к ответу $b_t(x)$.

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

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

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

При $\gamma \rightarrow \infty$ 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 $\alpha \in \mathbb{R}^n$:

$$g(x; \alpha, \beta) = \sigma(x^\top \alpha + \beta), \quad \alpha \in \mathbb{R}^n, \beta \in \mathbb{R};$$

- 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 $\mathcal{L}(b, y)$ называется *выпуклой* по b , если $\forall y \in Y, \forall b_1, b_2 \in R, \forall g_1, g_2 \geq 0: g_1 + g_2 = 1$, выполняется

$$\mathcal{L}(g_1 b_1 + g_2 b_2, y) \leq g_1 \mathcal{L}(b_1, y) + g_2 \mathcal{L}(b_2, y).$$

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

Примеры выпуклых функций потерь:

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

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

Основная идея применения выпуклых функций потерь

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

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

$$Q(a) = \sum_{i=1}^{\ell} \mathcal{L} \left(\sum_{t=1}^T g_t(x_i) b_t(x_i), y_i \right) \leq \sum_{t=1}^T \underbrace{\sum_{i=1}^{\ell} g_t(x_i) \mathcal{L}(b_t(x_i), y_i)}_{Q_t(g_t, b_t)}$$

Итерационный процесс, аналогичный EM-алгоритму:

начальное приближение функций компетентности g_t ;

повторять

M-шаг: при фиксированных g_t обучить все b_t ;

E-шаг: при фиксированных b_t оценить все g_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: $g_t(x), b_t(x), t = 1, \dots, T$;

повторять

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

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

M-шаг: при фиксированных g_t обучить все b_t :

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

E-шаг: при фиксированных b_t оценить все g_t :

$$g_t := \arg \min_{g_t} \sum_{i=1}^{\ell} \mathcal{L} \left(\sum_{s=1}^T \tilde{g}_s(x_i) b_s(x_i), y_i \right), \quad t = 1, \dots, T;$$

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

Обучение смеси с автоматическим определением числа T

Вход: выборка X^ℓ , параметры $l_0, \mathcal{L}_0, \delta, \gamma$;

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

начальное приближение:

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

для всех $t = 2, \dots, T$

множество трудных объектов:

$$X_t := \{x_i : \mathcal{L}(a_{t-1}(x_i), y_i) > \mathcal{L}_0\};$$

если $|X_t| \leq l_0$ **то выход;**

$$b_t := \arg \min_b \sum_{x_i \in X_t} \mathcal{L}(b(x_i), y_i);$$

$$g_t := \arg \min_{g_t} \sum_{i=1}^{\ell} \mathcal{L} \left(\sum_{s=1}^t g_s(x_i) b_s(x_i), y_i \right);$$

$$(g_s, b_s)_{s=1}^t := \text{ME} (X^\ell, (g_s)_{s=1}^t, t, \delta, \gamma);$$

- 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 GB — универсальные модели машинного обучения
- Смеси алгоритмов нужна хорошая модель компетентности