Методы машинного обучения. Продвинутые методы ансамблирования

Воронцов Константин Вячеславович
www.MachineLearning.ru/wiki?title=User:Vokov
вопросы к лектору: k.vorontsov@iai.msu.ru

материалы курса:
github.com/MSU-ML-COURSE/ML-COURSE-24-25
орг.вопросы по курсу: ml.cmc@mail.ru

ВМК МГУ • 10 декабря 2024

Содержание

Напоминание. Линейные ансамбли: бэггинг и бустинг

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

$a_t(x) = C(b_t(x))$ — базовые алгоритмы, $t = 1, \dots, T$

$C(b)$ — решающее правило, $C(b) = \mathrm{sign}(b)$ при $Y = \{-1, +1\}$

Взвешенное голосование:

$$a(x) = C\bigl(b(x)\bigr), \quad b(x) = \left(\sum_{t=1}^T \alpha_t b_t(x)\right), \quad x \in X, \quad \alpha_t \geqslant 0.$$

	бэггинг (bagging)	**бустинг** (boosting)
голосование	простое, $\alpha_t = \frac{1}{T}$	взвешенное, $\alpha_t \to \mathrm{opt}$
построение b_t	параллельное	последовательное
различность b_t	случайные подвыборки	взвешивание объектов
оптимизация	$\sum_i \mathscr{L}(b_t(x_i), y_i) \to \min$	$\sum_i \mathscr{L}(a(x_i), y_i) \to \min$

S.González, S.García, J.Del Ser, L.Rokach, F.Herrera. A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities. 2020

Обоснование бустинга (случай классификации на 2 класса)

Усиленная *частота ошибок* классификатора $a(x) = \operatorname{sign} b(x)$:

$$\nu_\theta(b, X^\ell) = \frac{1}{\ell}\sum_{i=1}^{\ell}\bigl[b(x_i)y_i \leqslant \theta\bigr], \quad y_i \in \{-1, +1\}, \quad \theta > 0$$

Обычная частота ошибок $\nu_0(b, X^\ell) \leqslant \nu_\theta(b, X^\ell)$ при $\theta > 0$.

Теорема (Freund, Schapire, Lee, Bartlett, 1998)

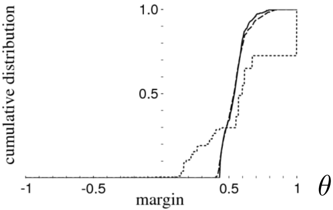
Если $|\mathscr{B}| < \infty$, то $\forall\theta > 0$, $\forall\eta \in (0,1)$ с вероятностью $1-\eta$

$$\mathsf{P}[a(x)y < 0] \leqslant \nu_\theta(b, X^\ell) + C\sqrt{\frac{\ln|\mathscr{B}|\ln\ell}{\ell\theta^2} + \frac{1}{\ell}\ln\frac{1}{\eta}}$$

Основной вывод: оценка зависит от $|\mathscr{B}|$, но не от T.
Голосование не увеличивает сложность семейства базовых алгоритмов, а лишь усредняет их ответы.

Обоснование бустинга: что же всё-таки происходит?

Распределение отступов: доля объектов, имеющих отступ меньше заданного θ после 5, 100, 1000 итераций (Задача UCI:vehicle)

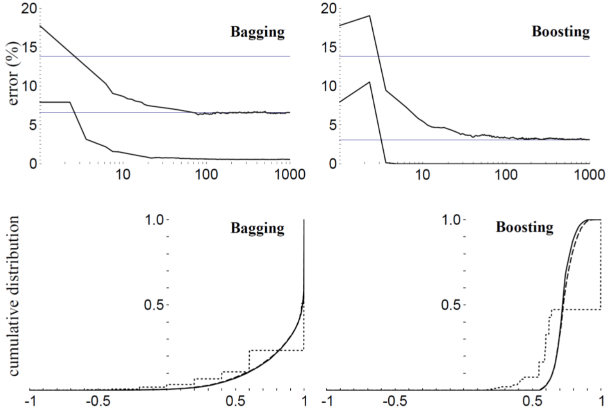


- С ростом T распределение отступов сдвигается вправо, увеличивается зазор между классами в пространстве векторов растущей размерности $\bigl(b_1(x), \ldots, b_T(x)\bigr)$
- Значит, в оценке можно уменьшать второй член, увеличивая θ при неизменной $\nu_\theta(b, X^\ell) = \nu_0(b, X^\ell)$.
- Можно уменьшить второй член, если уменьшить $|\mathscr{B}|$, то есть взять простое семейство базовых алгоритмов.

R.E.Schapire, Y.Freund, Wee Sun Lee, P.Bartlett. Boosting the margin: a new explanation for the effectiveness of voting methods. Annals of Statistics, 1998.

Бэггинг не столь успешно раздвигает классы

Ошибки на обучении и тесте. Снизу распределение отступов.



R.E.Schapire, Y.Freund, Wee Sun Lee, P.Bartlett. Boosting the margin: a new explanation for the effectiveness of voting methods. Annals of Statistics, 1998.

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:

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

Основная мера качества метода обучения μ:

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

Идеальный минимизатор среднеквадратичного риска:

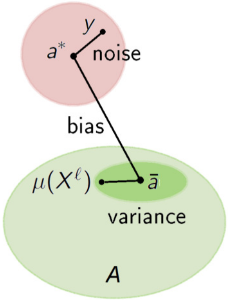
$$a^*(x) = \mathsf{E}(y|x) = \int_Y y\, p(y|x)\, dy$$

Разложение ошибки на шум, смещение и разброс

$a^*(x) = \mathsf{E}(y|x)$ — неизвестная идеальная зависимость y от x
$y(x) \sim p(y|x)$ — наблюдаемый ответ на объекте x
$a = \mu(X^\ell)$ — аппроксимация, выбранная по X^ℓ из семейства A
$\bar{a}(x) = \mathsf{E}_{X^\ell}\bigl(a(x)\bigr)$ — средний ответ обученного алгоритма

Теорема. При квадратичной функции потерь для любого μ

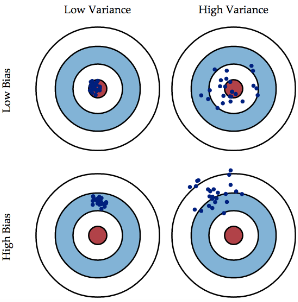
$$Q(\mu) = \underbrace{\mathsf{E}_{x,y}\bigl(a^*(x) - y\bigr)^2}_{\text{шум (noise)}} + \underbrace{\mathsf{E}_{x,y}\bigl(\bar{a}(x) - a^*(x)\bigr)^2}_{\text{смещение (bias)}} + \underbrace{\mathsf{E}_{x,y}\mathsf{E}_{X^\ell}\bigl(\mu(X^\ell)(x) - \bar{a}(x)\bigr)^2}_{\text{разброс (variance)}}$$

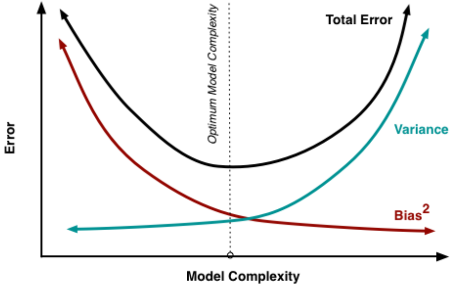


Разложение ошибки на шум, смещение и разброс

Качественное понимание: по мере роста сложности модели

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





Pedro Domingos. A Unified Bias-Variance Decomposition and its Applications. 2000
Brady Neal. On the Bias-Variance Tradeoff: Textbooks Need an Update. 2019

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):

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

Почему сложные ансамбли не переобучаются?

С позиций анализа отступов:

- ансамблирование не увеличивает сложность модели
- но с каждой итерацией увеличивает зазор между классами
- бустинг увеличивает зазор эффективнее, чем бэггинг

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 легче параллелится (по базовым алгоритмам b_t)
- в бустинге параллелится обучение b_t по частям выборки

Недостатки бэггинга и бустинга

- задача минимизировать число T вообще не ставится
- композиция из сотен алгоритмов не интерпретируема
- не удаётся строить короткие композиции из «сильных» алгоритмов типа SVM (только длинные из «слабых»)

Несколько эмпирических наблюдений:

- веса алгоритмов не важны для оптимизации отступов
- веса объектов не важны для обеспечения различности

Предлагается:

- обучать базовые алгоритмы последовательно (как бустинг),
- обучать их на подвыборках, но не случайных (как бэггинг),
- оптимизировать распределение отступов композиции,
- использовать простое голосование (комитет большинства)

Оптимизация распределения отступов на каждом шаге

Идея: явно управлять распределением отступов, максимизируя различность базовых алгоритмов и минимизируя их число.

Возьмём $b(x) = \frac{1}{T}\sum_{t=1}^{T} b_t(x)$, $a(x) = \operatorname{sign}(b(x))$, $Y = \{\pm 1\}$.

Критерий качества ансамбля — число ошибок на обучении:

$$Q(a, X^\ell) = \sum_{i=1}^{\ell} \bigl[y_i a(x_i) < 0\bigr] = \sum_{i=1}^{\ell} \Bigl[\underbrace{y_i b_1(x_i) + \cdots + y_i b_T(x_i)}_{M_{iT}} < 0\Bigr],$$

$M_{it} = y_i b_1(x_i) + \cdots + y_i b_t(x_i)$ — *отступ* (margin) объекта x_i.

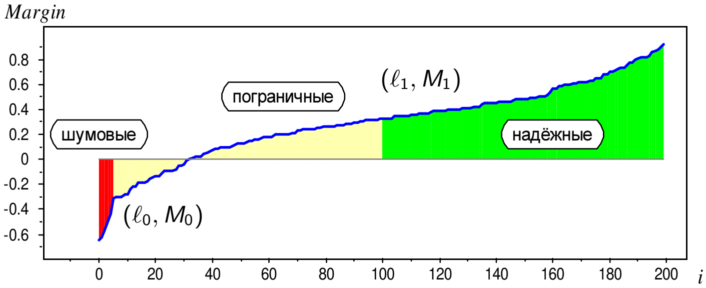
Эвристика: обучение b_t компенсирует ошибки ансамбля:

$$Q(b_t, U_t) = \sum_{x_i \in U_t} \bigl[y_i b_t(x_i) < 0\bigr] \to \min_{b_t},$$

$U_t = \bigl\{x_i \colon M_0 < M_{i,t-1} \leqslant M_1\bigr\}$, M_0, M_1 — параметры метода

Формирование выборки для обучения базового алгоритма

Упорядочим объекты по возрастанию отступов $M_{i,t-1}$:



Принцип выравнивания распределения отступов

два случая, когда b_t на объекте x_i обучать не надо:
$M_{i,t-1} < M_0$, $i < \ell_0$ — объект x_i шумовой
$M_{i,t-1} > M_1$, $i > \ell_1$ — объект x_i надёжный

Алгоритм ComBoost (Committee Boosting)

Вход: выборки X^ℓ, X^k; параметры $T, \ell_0, \ell_1, \ell_2, \Delta\ell$;
Выход: $b_1, \dots, b_T$;
$b_1 := \arg\min\limits_b Q(b, X^\ell)$; отступы $M_i = y_i b_1(x_i)$, $i = 1, \dots, \ell$;
для всех $t = 2, \dots, T$:
  упорядочить выборку X^ℓ по возрастанию отступов M_i;
  для всех $\ell' = \ell_1, \dots, \ell_2$ с шагом $\Delta\ell$:
    $U_t = \{x_i \in X^\ell \colon \ell_0 \leqslant i \leqslant \ell'\}$;
    $b_{t\ell'} := \arg\min\limits_b Q(b, U_t)$ — инкрементное обучение;
  выбрать наилучший $b_t \in \{b_{t\ell'}\}$ по критерию $Q(a, X^k)$;
  обновить отступы: $M_i := M_i + y_i b_t(x_i)$, $i = 1, \dots, \ell$;
пока Q существенно улучшается.

Маценов А. А. Комитетный бустинг: минимизация числа базовых алгоритмов при простом голосовании. ММРО-13, 2007.

Результаты эксперимента на 4 задачах из репозитория UCI

По 50 случайным разбиениям «обучение : контроль» = 4 : 1

	ionoshere	pima	bupa	votes
`SVM`	12,9	24,2	42,0	4,6
`ComBoost`$_0$ `[SVM]` (T)	12,6 (**4**)	23,1 (**2**)	34,2 (**5**)	4,0 (**2**)
`ComBoost [SVM]` (T)	**12,3** (5)	**22,5** (**2**)	30,9 (**5**)	**3,8** (3)
`AdaBoost [SVM]` (T)	15,0 (65)	22,7 (18)	**30,6** (15)	4,0 (8)
`Parzen`	6,3	25,1	41,6	6,9
`ComBoost`$_0$ `[Parzen]`	6,1	25,0	38,1	6,8
`ComBoost [Parzen]`	**5,8**	**24,7**	30,6	**6,2**
`AdaBoost [Parzen]`	6,0	24,8	**30,5**	6,5

`ComBoost`$_0$ — без подбора длины подвыборки U_t в цикле $\ell' = \ell_1, \dots, \ell_2$
`Parzen` — метод окна Парзена с подбором ширины окна по leave-one-out

Результат: ComBoost способен строить короткие ансамбли из сильных и устойчивых базовых алгоритмов

Маценов А. А. Комитетный бустинг: минимизация числа базовых алгоритмов при простом голосовании. ММРО-13, 2007.

Обобщение для задач с произвольным числом классов

$Y = \{1, \dots, M\}$, ансамбль — простое голосование, причём каждый базовый алгоритм b_{yt} голосует только за свой класс y:

$$a(x) = \arg\max_{y \in Y} \Gamma_y(x); \qquad \Gamma_y(x) = \frac{1}{|T_y|} \sum_{t \in T_y} b_{yt}(x).$$

В алгоритме ComBoost три небольших изменения:

- обобщённое определение отступа M_i:

$$M_i = \Gamma_{y_i}(x_i) - \max_{y \in Y \setminus \{y_i\}} \Gamma_y(x_i).$$

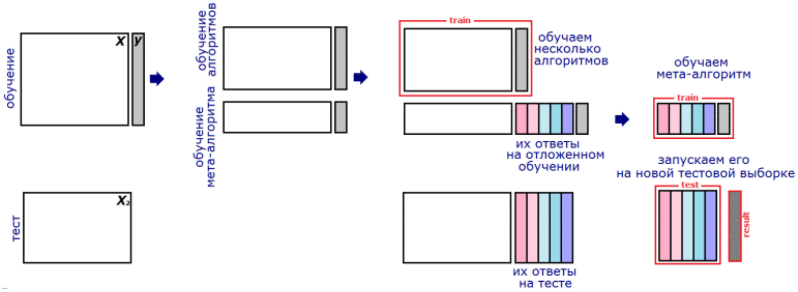
- придётся решать, для какого класса строить очередной b_{yt} (например, для того y, на котором доля ошибок больше)
- изменится пересчёт отступов в конце итерации

Allwein E. L., Schapire R. E., Singer Y. Reducing multiclass to binary: A unifying approach for margin classifiers. 2000

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!

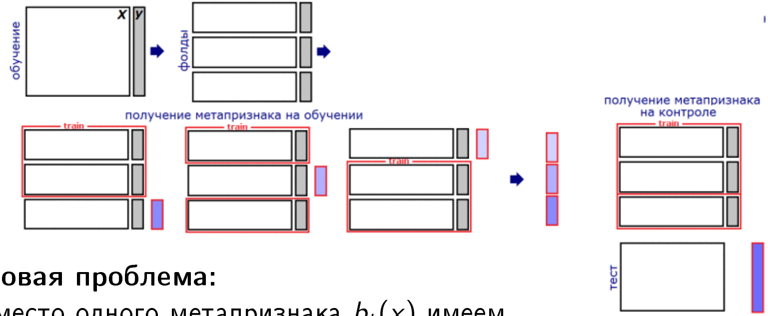


Новая проблема: для обучения используется не вся выборка.

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 $b_{tj}(x)$, $j = 1, \dots, k$.

Варианты решения: обучить $b_t(x)$ заново на всей выборке, либо усреднить метапризнаки: $b_t(x) = \frac{1}{k} \sum_{j=1}^{k} b_{tj}(x)$.

David H. Wolpert. Stacked generalization // Neural networks. 1992

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 α_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.

Смесь алгоритмов (Mixture of Experts)

$b_t\colon X \to \mathbb{R}$ — базовые алгоритмы, $t = 1, \dots, T$
$g_t\colon X \to \mathbb{R}$ — функция компетентности (шлюз, gate) для $b_t(x)$

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

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.

Растригин Л. А., Эренштейн Р. Х. Коллективные правила распознавания. 1981.
Hien D. Nguyen, Faicel Chamroukhi. Practical and theoretical aspects of mixture-of-experts modeling: An overview. 2018

Вид функций компетентности

Функции компетентности определяются из практических соображений, в зависимости от особенностей задачи, например:

- по признаку $f(x)$:
$$g(x;\alpha,\beta) = \sigma(\alpha f(x) + \beta), \quad \alpha,\beta \in \mathbb{R};$$
- по неизвестному направлению $\alpha \in \mathbb{R}^n$:
$$g(x;\alpha,\beta) = \sigma(x^{\mathsf{T}}\alpha + \beta), \quad \alpha \in \mathbb{R}^n,\ \beta \in \mathbb{R};$$
- по расстоянию 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 параметры α,β фиксируются или обучаются 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 $\forall\, y \in Y$, $\forall\, b_1, b_2 \in R$, $\forall\, g_1, g_2 \geqslant 0$: $g_1 + g_2 = 1$, 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:

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

Итерационный процесс, два шага на каждой итерации:

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

Алгоритм ME (Mixture of Experts): обучение смеси алгоритмов

Вход: выборка X^ℓ, функции $\bigl(g_t(x)\bigr)_{t=1}^{T}$, параметры T, δ, γ;

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

повторять

нормировать функции компетентности:

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

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

обучить базовые алгоритмы при фиксированных $\tilde{g}_t$:

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

обучить функции компетентности при фиксированных b_t:

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

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

Очередную компоненту обучаем на наиболее трудных объектах:

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:
$b_1 := \arg\min\limits_b \sum_{i=1}^{\ell} \mathscr{L}\bigl(b(x_i), y_i\bigr)$, $\quad g_1(x_i) := 1$, $i = 1, \dots, \ell$;
для всех $t = 2, \dots$
- множество «наиболее 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**;
- обучить $b_t := \arg\min\limits_b \sum_{x_i \in X_t} \mathscr{L}\bigl(b(x_i), y_i\bigr)$;
- обучить $g_t := \arg\min\limits_{g_t} \sum_{i=1}^{\ell} \mathscr{L}\Bigl(\sum_{s=1}^{t} g_s(x_i) b_s(x_i), y_i\Bigr)$;
- $(g_s, b_s)_{s=1}^t := \mathrm{ME}\bigl(X^\ell, (g_s)_{s=1}^t, t, \delta, \gamma\bigr)$;

Резюме (общее по трём лекциям)

- Ансамбли позволяют решать сложные задачи, которые плохо решаются отдельными базовыми алгоритмами.
- 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
- CatBoost — для категориальных признаков, без переобучения
- FWLS и ME — квазилинейные ансамбли, $\alpha_t(x)$
- Смеси алгоритмов нужна хорошая модель компетентности