Композиции классификаторов (часть 2)

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)»

ШАД Яндекс • 1 октября 2019

Содержание

Определение композиции (напоминание)

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

$a(x) = C(b(x))$ — алгоритм, где

$b\colon X \to R$ — *базовый алгоритм* (алгоритмический оператор),

$C\colon R \to Y$ — *решающее правило*,

R — *пространство оценок*;

Определение

Композиция базовых алгоритмов $b_1, \ldots, b_T$

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

где $F\colon R^T \to R$ — *корректирующая операция*.

Зачем вводится R?

В задачах классификации множество отображений $\{F\colon R^T \to R\}$ существенно шире, чем $\{F\colon Y^T \to Y\}$.

Примеры корректирующих операций (напоминание)

- **Пример 1:** Простое голосование (Simple Voting):

$$F\big(b_1(x),\ldots,b_T(x)\big) = \frac{1}{T}\sum_{t=1}^{T} b_t(x), \quad x \in X.$$

- **Пример 2:** Взвешенное голосование (Weighted Voting):

$$F\big(b_1(x),\ldots,b_T(x)\big) = \sum_{t=1}^{T} \alpha_t b_t(x), \quad x \in X, \quad \alpha_t \in \mathbb{R}.$$

- **Пример 3:** Смесь алгоритмов (Mixture of Experts)

$$F\big(b_1(x),\ldots,b_T(x)\big) = \sum_{t=1}^{T} g_t(x) b_t(x), \quad x \in X, \quad g_t \colon X \to \mathbb{R}.$$

Стохастические методы построения композиций

Чтобы алгоритмы в композиции были различными

- их обучают по (случайным) подвыборкам,
- либо по (случайным) подмножествам признаков.

Первую идею реализует bagging (bootstrap aggregation) [Breiman, 1996]: подвыборки длины ℓ с повторениями, доля объектов, попадающих в выборку: $(1 - \frac{1}{e}) \approx 0.632$

Вторую идею реализует RSM (random subspace method) [Ho, 1998].

Совместим обе идеи в одном алгоритме.

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

Бэггинг и метод случайных подпространств

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, $t = 1, \ldots, T$;

для всех $t = 1, \ldots, T$
  $U_t :=$ случайное подмножество X^ℓ длины ℓ';
  $\mathscr{G}_t :=$ случайное подмножество $\mathscr{F}$ длины n';
  $b_t := \mu(\mathscr{G}_t, U_t)$;
  если $Q(b_t, U_t) > \varepsilon_1$ или $Q(b_t, X^\ell \setminus U_t) > \varepsilon_2$ **то**
    не включать b_t в композицию;

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

Сравнение: boosting — bagging — RSM

- Бустинг лучше для больших обучающих выборок и для классов с границами сложной формы
- Бэггинг и RSM лучше для коротких обучающих выборок
- RSM лучше в тех случаях, когда признаков больше, чем объектов, или когда много неинформативных признаков
- Бэггинг и RSM эффективно распараллеливаются, бустинг выполняется строго последовательно

И ещё несколько эмпирических наблюдений:

- Веса алгоритмов не столь важны для выравнивания отступов
- Веса объектов не столь важны для обеспечения различности
- Короткие композиции из «сильных» алгоритмов типа SVM строить труднее, чем длинные из слабых

Оптимизация распределения отступов на каждом шаге

Возьмём $Y = \{\pm 1\}$, $F(b_1, \ldots, b_T) = \frac{1}{T}\sum_{t=1}^{T} b_t$, $C(b) = \text{sign}(b)$.

Функционал качества композиции — число ошибок на обучении:

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

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

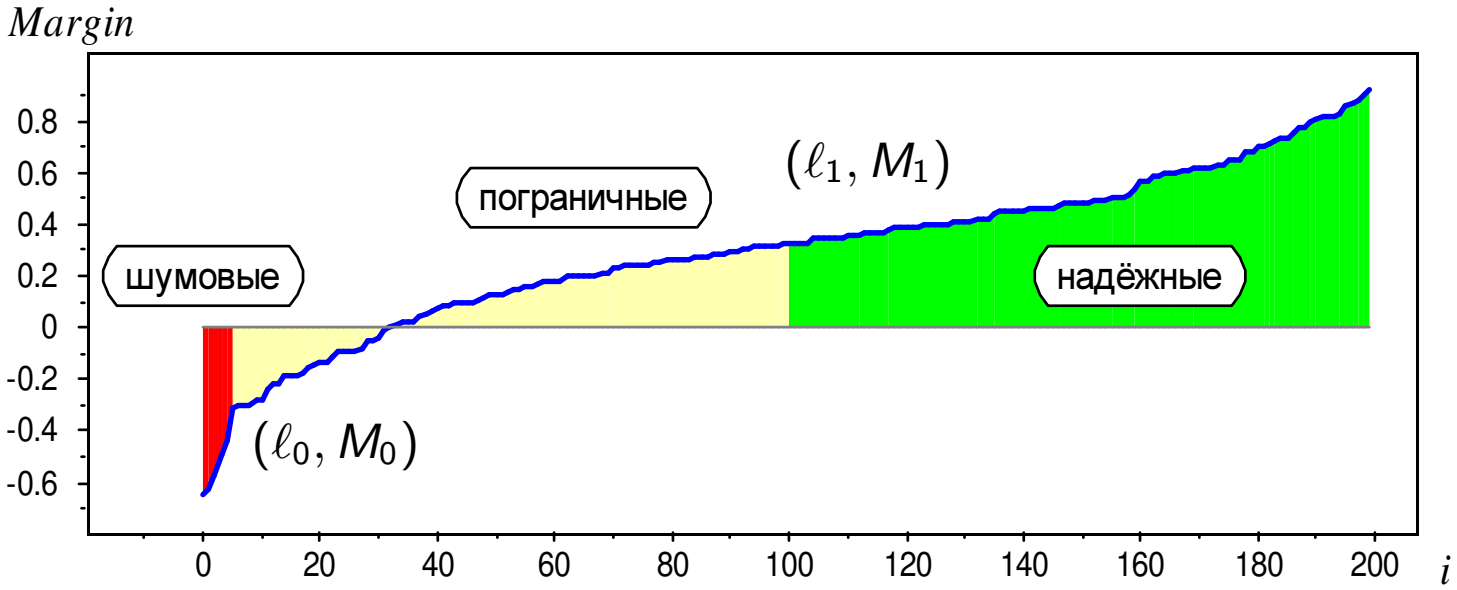
Эвристика: чтобы b_t компенсировал ошибки композиции,

$$Q(b, U_t) = \sum_{x_i \in U_t} \big[y_i b(x_i) < 0\big] \to \min_b,$$

где $U_t = \big\{x_i : M_0 < M_{i,t-1} \leqslant M_1\big\}$,
M_0, M_1 — параметры метода обучения.

Подбор параметров M_0 и M_1

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



Принцип максимизации и выравнивания 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 надёжно классифицируется.

Алгоритм ComBoost (Committee Boosting)

Вход: обучающая выборка X^ℓ; параметры $T, \ell_0, \ell_1, \ell_2, \Delta\ell$;
Выход: $b_1, \ldots, b_T$

$b_1 := \arg\min_b Q(b, X^\ell)$;
упорядочить X^ℓ по возрастанию $M_i = y_i b_t(x_i)$, $i = 1, \ldots, \ell$;
для всех $t = 1, \ldots, T$
  для всех $k = \ell_1, \ldots, \ell_2$ с шагом $\Delta\ell$
    $U_t = \{x_i \in X^\ell : \ell_0 \leqslant i \leqslant k\}$;
    $b_{tk} := \arg\min_b Q(b, U_t)$;
  выбрать наилучший $b_t \in \{b_{tk}\}$ по критерию Q;
  обновить отступы: $M_i := M_i + y_i b_t(x_i)$, $i = 1, \ldots, \ell$;
  упорядочить выборку X^ℓ по возрастанию отступов M_i;
  опция: скорректировать значения параметров $\ell_0, \ell_1, \Delta\ell$;
пока Q существенно улучшается.

Результаты эксперимента на 4 задачах из репозитория UCI

Средняя частота ошибок на контроле по 50 случайным разбиениям в отношении «обучение : контроль» = 4 : 1.

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

ComBoost$_0$ — с подбором ℓ_0 и $\ell_1 = \ell_2$ по скользящему контролю;
ComBoost — с подбором длины подвыборки U_t;
Parzen — окно Парзена с евклидовой метрикой и подбором ширины окна скользящим контролем LOO.

Результаты эксперимента на 4 задачах из репозитория UCI

Мощность композиций:

Число базовых алгоритмов	ionoshere	pima	bupa	votes
$\texttt{ComBoost}_0$ над SVM	**4**	**2**	**5**	**2**
ComBoost над SVM	5	**2**	5	3
AdaBoost над SVM	65	18	15	8

Критерий останова: отсутствие существенного улучшения качества классификации обучающей выборки.

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

Обобщение для задач с произвольным числом классов

Пусть теперь $Y = \{1, \ldots, M\}$.

Композиция — 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 только два изменения:
— изменится определение отступа M_i:

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

— в алгоритме ComBoost на шаге 3 придётся решать,
за какой класс строить очередной базовый алгоритм,
кроме того, немного изменится шаг 7 (пересчёт отступов).

Преобразование простого голосования во взвешенное

Линейный классификатор над признаками $b_t(x)$:

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

1. Метод обучения: SVM, логистическая регрессия, и т.п.:

$$Q(\alpha, X^\ell) = \sum_{i=1}^{\ell} \mathscr{L}\left(y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)\right) \to \min_{\alpha}.$$

2. Регуляризация: $\alpha_t \geqslant 0$ либо LASSO: $\sum_{t=1}^{T} |\alpha_t| \leqslant \varkappa$.

3. Наивный байесовский классификатор
приводит к простому аналитическому решению:

$$\alpha_t = \ln \frac{1 - p_t}{p_t}, \quad t = 1, \ldots, T,$$

где p_t — оценка вероятности ошибки базового алгоритма b_t.

Случайный лес (Random Forest)

Обучение случайного леса:

- бэггинг над решающими деревьями, без pruning
- признак в каждой вершине дерева выбирается из случайного подмножества k из n признаков
- регрессия: $k = \lfloor n/3 \rfloor$; классификация: $k = \lfloor \sqrt{n} \rfloor$

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:

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

Это несмещённая оценка обобщающей способности.

Breiman L. Random Forests. Machine Learning, 2001.

Задача минимизации среднеквадратичного риска

Задача регрессии: $Y = \mathbb{R}$
Квадратичная функция потерь: $L(y, a) = \big(a(x) - y\big)^2$
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

Среднеквадратичный риск:

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

Идеальный минимизатор среднеквадратичного риска:

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

Основная мера качества метода обучения μ:

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

Разложение ошибки на шум, вариацию и смещение

Теорема

В случае квадратичной функции потерь для любого μ

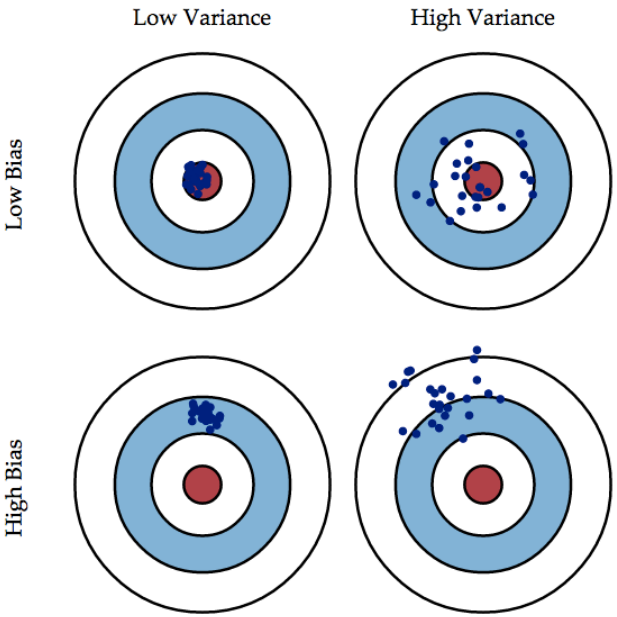
$$Q(\mu) = \underbrace{\mathsf{E}_{x,y}\big(a^*(x) - y\big)^2}_{\text{шум (noise)}} + \underbrace{\mathsf{E}_{x,y}\big(\bar{a}(x) - a^*(x)\big)^2}_{\text{смещение (bias)}} + \underbrace{\mathsf{E}_{x,y}\mathsf{E}_{X^\ell}\big(\mu(X^\ell)(x) - \bar{a}(x)\big)^2}_{\text{разброс (variance)}},$$

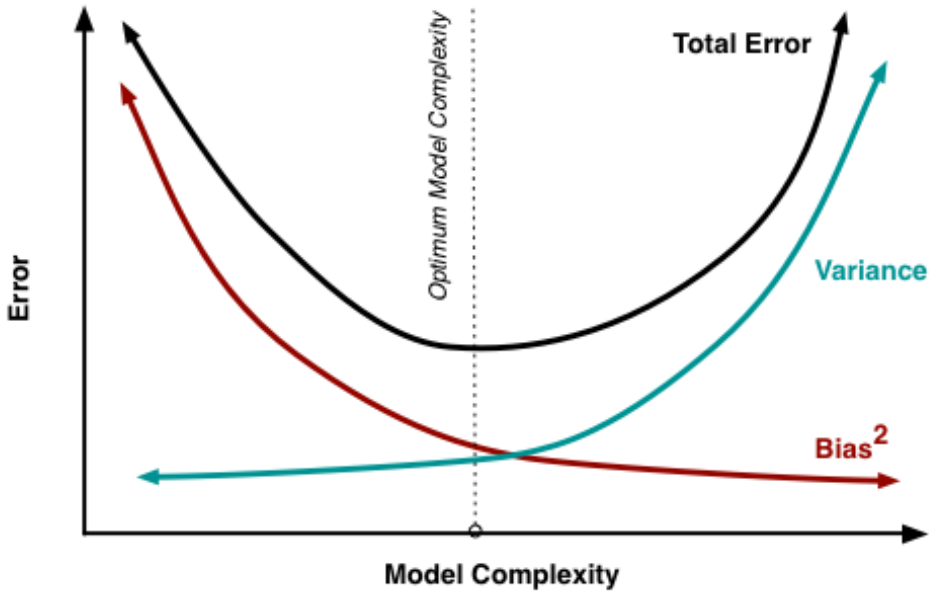
$\bar{a}(x) = \mathsf{E}_{X^\ell}\big(\mu(X^\ell)(x)\big)$ — средний ответ обученного алгоритма

Разложение ошибки на шум, вариацию и смещение

Качественное понимание: по мере роста сложности модели

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





Упражнение. Метод k ближайших соседей

Вероятностная модель данных: $p(y|x) = f(x) + \mathcal{N}(0, \sigma^2)$
Метод k ближайших соседей:

$$a(x) = \frac{1}{k} \sum_{j=1}^{k} y(x^{(j)}),$$

где $x^{(j)}$ — j-й сосед объекта x
$a^*(x) = f(x)$ — истинная зависимость
$\bar{a}(x) = \mathsf{E}_{X^\ell} \frac{1}{k} \sum_{j=1}^{k} f(x^{(j)})$ — средний ответ

Разложение bias-variance (чем меньше k, тем сложнее модель):

$$Q(\mu) = \underbrace{\sigma^2}_{\text{шум}} + \underbrace{\mathsf{E}_{x,y}\Big(\bar{a}(x) - f(x)\Big)^2}_{\text{смещение}} + \underbrace{\tfrac{1}{k}\sigma^2}_{\text{разброс}}$$

Простое голосование

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

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

Смещение композиции совпадает со смещением отдельного базового алгоритма:

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

Разброс состоит из дисперсии и ковариации:

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

Резюме. Почему бустинг и бэггинг работают?

- бэггинг уменьшает разброс
- бустинг уменьшает и смещение, и разброс
- корреляция базовых алгоритмов увеличивает разброс

Случайный лес (Random Forest) — разновидность бэггинга:

- RF уменьшает корреляции базовых алгоритмов
- 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.

Квазилинейная композиция (смесь алгоритмов)

Смесь алгоритмов (Mixture of Experts)

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

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

$g_t : X \to \mathbb{R}$ — 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: $\mathbb{R}^T \to \mathbb{R}^T$:

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

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 $\forall\, y \in Y,\ \forall\, b_1, b_2 \in R,\ \forall\, g_1, g_2 \geqslant 0$: $g_1 + g_2 = 1$, 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 $\forall x \sum_{t=1}^{T} g_t(x) = 1$ и функция потерь $\mathscr{L}$ выпукла.

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

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

Итерационный процесс, аналогичный EM-алгоритму:

1: начальное приближение функций компетентности g_t;
2: **повторять**
3: **M-шаг:** при фиксированных g_t обучить все b_t;
4: **E-шаг:** при фиксированных b_t оценить все g_t;
5: **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: $g_t(x), b_t(x), \quad t = 1, \ldots, T$;

1: **повторять**

2: $\big(\tilde{g}_1(x_i), \ldots, \tilde{g}_T(x_i)\big) := \mathrm{SoftMax}\big(g_1(x_i), \ldots, g_T(x_i); \gamma\big)$;

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

4: **M-шаг:** при фиксированных g_t обучить все b_t:

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

5: **E-шаг:** при фиксированных b_t оценить все g_t:

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

6: **пока** $\max\limits_{t,i} \big|\tilde{g}_t(x_i) - \tilde{g}_t^0(x_i)\big| > \delta$.

Обучение смеси с автоматическим определением числа T

Вход: выборка X^ℓ, параметры ℓ_0, $\mathscr{L}_0$, δ, γ;
Выход: T, $g_t(x), b_t(x)$, $t = 1, \ldots, T$;

1: начальное приближение:
$b_1 := \arg\min_b \sum_{i=1}^{\ell} \mathscr{L}\big(b(x_i), y_i\big), \quad g_1(x_i) := 1, \ i = 1, \ldots, \ell;$
2: **для всех** $t = 2, \ldots, T$
3: множество трудных объектов:
$X_t := \big\{x_i : \mathscr{L}(a_{t-1}(x_i), y_i) > \mathscr{L}_0\big\};$
4: **если** $|X_t| \leqslant \ell_0$ **то выход**;
5: $b_t := \arg\min_b \sum_{x_i \in X_t} \mathscr{L}\big(b(x_i), y_i\big);$
6: $g_t := \arg\min_{g_t} \sum_{i=1}^{\ell} \mathscr{L}\left(\sum_{s=1}^{t} g_s(x_i) b_s(x_i), y_i\right);$
7: $(g_s, b_s)_{s=1}^t := \mathrm{ME}\left(X^\ell, (g_s)_{s=1}^t, t, \delta, \gamma\right);$

Резюме

- Простое голосование может работать не хуже бустинга
- Причины эффективности композиций: уменьшается разброс, корреляция и смещение базовых алгоритмов
- Смеси алгоритмов эффективны в тех задачах, где есть содержательные модели областей компетентности.