

Линейные ансамбли

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 • 6 ноября 2021

- 1 Простое голосование**
 - Общее определение ансамбля
 - Бэггинг и случайные подпространства
 - Случайные леса
- 2 Взвешенное голосование**
 - Адаптивный бустинг AdaBoost
 - Основная теорема AdaBoost
 - Алгоритм AdaBoost
- 3 Обоснования и варианты бустинга**
 - Эксперименты с бустингом
 - Теория обобщающей способности
 - Алгоритм ComBoost

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

Пространства оценок и решающие правила

- **Пример 1:** классификация, Y — конечное множество, $R = Y$, $C(b) \equiv b$ — решающее правило не используется.

- **Пример 2:** классификация на 2 класса, $Y = \{-1, +1\}$,

$$a(x) = \text{sign}(b(x)),$$

где $R = \mathbb{R}$, $b: X \rightarrow \mathbb{R}$, $C(b) \equiv \text{sign}(b)$.

- **Пример 3:** классификация на M классов $Y = \{1, \dots, M\}$,

$$a(x) = \arg \max_{y \in Y} b_y(x),$$

где $R = \mathbb{R}^M$, $b: X \rightarrow \mathbb{R}^M$, $C(b_1, \dots, b_M) \equiv \arg \max_{y \in Y} b_y$.

- **Пример 4:** регрессия, $Y = R = \mathbb{R}$,
 $C(b) \equiv b$ — решающее правило не нужно.

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) $g_t: X \rightarrow \mathbb{R}$

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

Проблема разнообразия (diversity) базовых алгоритмов

Измерение с.в. ξ по независимым наблюдениям $\{\xi_t\}$:

- $E \frac{1}{T}(\xi_1 + \dots + \xi_T) = E\xi$ — матожидание среднего
- $D \frac{1}{T}(\xi_1 + \dots + \xi_T) = \frac{1}{T}D\xi$ — дисперсия $\rightarrow 0$ при $T \rightarrow \infty$

Но базовые алгоритмы не являются независимыми с.в.:

- решают одну и ту же задачу
- настраиваются на один целевой вектор (y_i)
- обычно выбираются из одной и той же модели

Способы повышения *разнообразия* базовых алгоритмов:

- обучение по различным (случайным) подвыборкам
- обучение по различным (случайным) наборам признаков
- обучение из разных параметрических моделей
- обучение с использованием рандомизации
- (иногда даже) обучение по зашумлённым данным

Методы стохастического ансамблирования

Способы повышения разнообразия с помощью рандомизации:

- bagging (bootstrap aggregating) — подвыборки обучающей выборки «с возвращением», в каждую выборку попадает $(1 - \frac{1}{e}) \approx 63.2\%$ объектов
- pasting — случайные обучающие подвыборки
- random subspaces — случайные подмножества признаков
- random patches — случ. подмножества признаков и объектов
- cross-validated committees — выборка разбивается на k блоков (k -fold) и делается k обучений без одного блока

Пусть $\mu: (G, U) \mapsto b$ — метод обучения по подвыборке $U \subseteq X^\ell$, использующий только признаки из $G \subseteq F^n = \{f_1, \dots, f_n\}$

Tin Kam Ho. The random subspace method for constructing decision forests. 1998.
Leo Breiman. Bagging predictors // Machine Learning. 1996.

Методы стохастического ансамблирования в одном псевдо-коде

Вход: обучающая выборка X^ℓ ; параметры: T ,
 ℓ' — объём обучающих подвыборок,
 n' — размерность признаков подпространств,
 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: $U_t :=$ случайная подвыборка объёма ℓ' из X^ℓ ;
- 3: $G_t :=$ случайное подмножество мощности n' из F^n ;
- 4: $b_t := \mu(G_t, U_t)$;
- 5: если $Q(b_t, U_t) > \varepsilon_1$ то не включать b_t в ансамбль;
- 6: если $Q(b_t, X^\ell \setminus U_t) > \varepsilon_2$ то не включать b_t в ансамбль;

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

Несмещённая оценка ошибок

Out-of-bag — несмещённая оценка ансамбля на объекте:

$$\text{OOB}(x_i) = \frac{1}{|T_i|} \sum_{t \in T_i} b_t(x_i), \quad T_i = \{t: x_i \notin U_t\}$$

Несмещённая оценка ошибки ансамбля на обучающей выборке:

$$\text{OOB}(X^\ell) = \sum_{i=1}^{\ell} \mathcal{L}(\text{OOB}(x_i), y_i),$$

где $\mathcal{L}(b(x_i), y_i)$ — значение функции потерь на объекте x_i .

Оценивание важности признаков f_j , $j = 1, \dots, n$:

$$\text{importance}_j = \frac{\text{OOB}^j(X^\ell) - \text{OOB}(X^\ell)}{\text{OOB}(X^\ell)} \cdot 100\%,$$

где при вычислении $b_t(x_i)$ для OOB^j значения признака f_j случайным образом перемешиваются на всех объектах $x_i \neq U_t$.

Преобразование простого голосования во взвешенное

- Линейная модель над готовыми признаками $b_t(x)$:

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

- Обучение: МНК для регрессии, LR для классификации:

$$Q(\alpha, X^\ell) = \sum_{i=1}^{\ell} \mathcal{L}(b(x_i), y_i) \rightarrow \min_{\alpha}.$$

Регуляризация: $\alpha_t \geq 0$ либо LASSO: $\sum_{t=1}^T |\alpha_t| \leq \kappa$.

- Наивный байесовский классификатор предполагает независимость с.в. $b_t(x)$ и даёт аналитическое решение:

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

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. По умолчанию $k = \lfloor n/3 \rfloor$ для регрессии, $k = \lfloor \sqrt{n} \rfloor$ для классификации

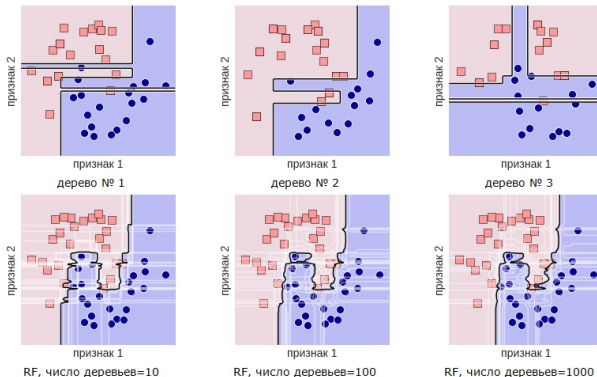
Параметры, которые можно настраивать (в частности, по OOB):

- число T деревьев
- число k случайно выбираемых признаков
- максимальная глубина деревьев
- минимальное число объектов в расщепляемой подвыборке
- минимальное число объектов в листьях
- критерий расщепления: MSE для регрессии, энтропийный или Джини для классификации

Breiman L. Random Forests. Machine Learning, 2001.

Постепенное сглаживание разделяющей поверхности

Пример разделения выборки с помощью отдельных деревьев (показаны соответствующие бутстреп-подвыборки) и случайного леса с числом деревьев 10, 100, 1000:



<https://dyakonov.org/2019/04/19/ансамбли-в-машинном-обучении>

Разновидности решающих лесов

- Случайный лес (Random Forest)
- Использование большого числа простых решающих деревьев в качестве признаков, в любом классификаторе.
- Oblique Random Forest, Rotation Forest
 $f_V(x)$ — линейные комбинации признаков, выбираемые по энтропийному критерию информативности.
- Решающий список из решающих деревьев:
 - при образовании статистически ненадёжного листа этот лист заменяется переходом к следующему дереву;
 - следующее дерево строится по объединению подвыборок, прошедших через ненадёжные листы предыдущего дерева.

https://en.wikipedia.org/wiki/Random_forest

Преимущества и ограничения стохастического ансамблирования

Преимущества:

- метод-обёртка (envelop) над базовым методом обучения
- подходит для классификации, регрессии и других задач
- простая реализация и простое распараллеливание
- возможность получения несмещённых оценок ООВ
- возможность оценивания важности признаков
- RF — один из лучших универсальных методов в ML

Ограничения:

- требуется ооооочень много базовых алгоритмов
- трудно агрегировать устойчивые базовые методы обучения

<https://dyakonov.org/2016/11/14/случайный-лес-random-forest>

Бустинг для задачи классификации с двумя классами

Возьмём $Y = \{\pm 1\}$, $b_t: X \rightarrow \{-1, 0, +1\}$, $C(b) = \text{sign}(b)$.
 $b_t(x) = 0$ — отказ (лучше промолчать, чем соврать).

Взвешенное голосование:

$$a(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t b_t(x)\right), \quad x \in X.$$

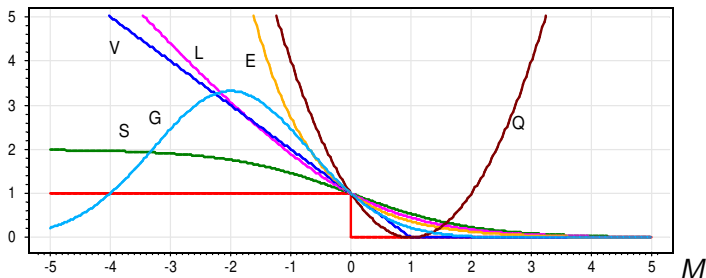
Функционал качества композиции — число ошибок на X^ℓ :

$$Q_T = \sum_{i=1}^{\ell} \left[y_i \sum_{t=1}^T \alpha_t b_t(x_i) < 0 \right].$$

Две основные эвристики бустинга:

- фиксация $\alpha_1 b_1(x), \dots, \alpha_{t-1} b_{t-1}(x)$ при добавлении $\alpha_t b_t(x)$;
- гладкая аппроксимация пороговой функции потерь [$M \leq 0$].

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);

Экспоненциальная аппроксимация пороговой функции потерь

Оценка функционала качества Q_T сверху:

$$Q_T \leq \tilde{Q}_T = \sum_{i=1}^{\ell} \underbrace{\exp\left(-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)\right)}_{w_i} \exp(-y_i \alpha_T b_T(x_i))$$

Нормированные веса: $\tilde{W}^\ell = (\tilde{w}_1, \dots, \tilde{w}_\ell)$, $\tilde{w}_i = w_i / \sum_{j=1}^{\ell} w_j$.

Взвешенное число ошибочных (negative) и правильных (positive) классификаций при векторе весов $U^\ell = (u_1, \dots, u_\ell)$:

$$N(b, U^\ell) = \sum_{i=1}^{\ell} u_i [b(x_i) = -y_i]; \quad P(b, U^\ell) = \sum_{i=1}^{\ell} u_i [b(x_i) = y_i].$$

$1 - N - P$ — взвешенное число отказов от классификации.

Основная теорема бустинга (для AdaBoost)

Пусть \mathcal{B} — достаточно богатое семейство базовых алгоритмов.

Теорема (Freund, Schapire, 1996)

Пусть для любого нормированного вектора весов U^ℓ существует алгоритм $b \in \mathcal{B}$, классифицирующий выборку хотя бы немного лучше, чем наугад: $P(b; U^\ell) > N(b; U^\ell)$.

Тогда минимум функционала \tilde{Q}_T достигается при

$$b_T = \arg \max_{b \in \mathcal{B}} \sqrt{P(b; \tilde{W}^\ell)} - \sqrt{N(b; \tilde{W}^\ell)}.$$
$$\alpha_T = \frac{1}{2} \ln \frac{P(b_T; \tilde{W}^\ell)}{N(b_T; \tilde{W}^\ell)}.$$

Доказательство (шаг 1 из 2)

Воспользуемся тождеством $\forall \alpha \in \mathbb{R}, \forall b \in \{-1, 0, +1\}$:
 $e^{-\alpha b} = e^{-\alpha} [b=1] + e^{\alpha} [b=-1] + [b=0]$.

Положим для краткости $\alpha = \alpha_T$ и $b_i = b_T(x_i)$. Тогда

$$\begin{aligned} \tilde{Q}_T &= \left(\underbrace{e^{-\alpha} \sum_{i=1}^{\ell} \tilde{w}_i [b_i = y_i]}_P + \underbrace{e^{\alpha} \sum_{i=1}^{\ell} \tilde{w}_i [b_i = -y_i]}_N + \underbrace{\sum_{i=1}^{\ell} \tilde{w}_i [b_i = 0]}_{1-P-N} \right) \underbrace{\sum_{i=1}^{\ell} w_i}_{\tilde{Q}_{T-1}} \\ &= (e^{-\alpha} P + e^{\alpha} N + (1 - P - N)) \tilde{Q}_{T-1} \rightarrow \min_{\alpha, b}. \end{aligned}$$

$$\frac{\partial}{\partial \alpha} \tilde{Q}_T = (-e^{-\alpha} P + e^{\alpha} N) \tilde{Q}_{T-1} = 0 \Rightarrow e^{-\alpha} P = e^{\alpha} N \Rightarrow e^{2\alpha} = \frac{P}{N}.$$

Получили требуемое: $\boxed{\alpha_T = \frac{1}{2} \ln \frac{P}{N}}$.

Доказательство (шаг 2 из 2)

Подставим оптимальное значение $\alpha = \frac{1}{2} \ln \frac{P}{N}$ обратно в \tilde{Q}_T :

$$\begin{aligned}\tilde{Q}_T &= (e^{-\alpha}P + e^{\alpha}N + (1 - P - N))\tilde{Q}_{T-1} = \\ &= (1 + \sqrt{\frac{N}{P}}P + \sqrt{\frac{P}{N}}N - P - N)\tilde{Q}_{T-1} = \\ &= \left(1 - (\sqrt{P} - \sqrt{N})^2\right)\tilde{Q}_{T-1} \rightarrow \min_b.\end{aligned}$$

Поскольку \tilde{Q}_{T-1} не зависит от α_T и b_T , минимизация \tilde{Q}_T эквивалентна либо максимизации $\sqrt{P} - \sqrt{N}$ при $P > N$, либо максимизации $\sqrt{N} - \sqrt{P}$ при $P < N$, однако второй случай исключён условием теоремы.

Получили $b_T = \arg \max_b \sqrt{P} - \sqrt{N}$. Теорема доказана.

Следствие 1. Исходный (классический) вариант AdaBoost

Пусть отказов нет, $b_t: X \rightarrow \{\pm 1\}$. Тогда $P = 1 - N$.

Теорема (Freund, Schapire, 1995)

Пусть для любого нормированного вектора весов U^ℓ существует алгоритм $b \in \mathcal{B}$, классифицирующий выборку хотя бы немного лучше, чем наугад: $N(b; U^\ell) < \frac{1}{2}$.

Тогда минимум функционала \tilde{Q}_T достигается при

$$b_T = \arg \min_{b \in \mathcal{B}} N(b; \tilde{W}^\ell).$$

$$\alpha_T = \frac{1}{2} \ln \frac{1 - N(b_T; \tilde{W}^\ell)}{N(b_T; \tilde{W}^\ell)}.$$

Следствие 2. Сходимость

Теорема

Если на каждом шаге семейство \mathcal{B} и метод обучения обеспечивают построение базового алгоритма b_t такого, что

$$\sqrt{P(b_t; \widetilde{W}^\ell)} - \sqrt{N(b_t; \widetilde{W}^\ell)} = \gamma_t > \gamma$$

при некотором $\gamma > 0$, то за конечное число шагов будет построен корректный алгоритм $a(x)$.

Доказательство. Q_T сходится к нулю со скоростью геометрической прогрессии:

$$Q_{T+1} \leq \widetilde{Q}_{T+1} = \widetilde{Q}_T(1 - \gamma^2) \leq \dots \leq \widetilde{Q}_1(1 - \gamma^2)^T.$$

Наступит момент, когда $\widetilde{Q}_T < 1$.

Но тогда $Q_T = 0$, поскольку $Q_T \in \{0, 1, \dots, \ell\}$.

Алгоритм AdaBoost (исходный вариант)

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

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

1: инициализировать веса объектов:

$$w_i := 1/\ell, \quad i = 1, \dots, \ell;$$

2: **для всех** $t = 1, \dots, T$

3: обучить базовый алгоритм:

$$b_t := \arg \min_b N(b; W^\ell);$$

4: $\alpha_t := \frac{1}{2} \ln \frac{1 - N(b_t; W^\ell)}{N(b_t; W^\ell)}$;

5: обновить веса объектов:

$$w_i := w_i \exp(-\alpha_t y_i b_t(x_i)), \quad i = 1, \dots, \ell;$$

6: нормировать веса объектов:

$$w_0 := \sum_{j=1}^{\ell} w_j;$$

$$w_i := w_i / w_0, \quad i = 1, \dots, \ell;$$

Эвристики и рекомендации

- **Базовые классификаторы (weak classifiers):**
 - решающие деревья — используются чаще всего;
 - пороговые правила, т.н. «решающие пни» (data stumps)
$$\mathcal{B} = \left\{ b(x) = [f_j(x) \leq \theta] \mid j = 1, \dots, n, \theta \in \mathbb{R} \right\};$$
 - для SVM бустинг не эффективен.

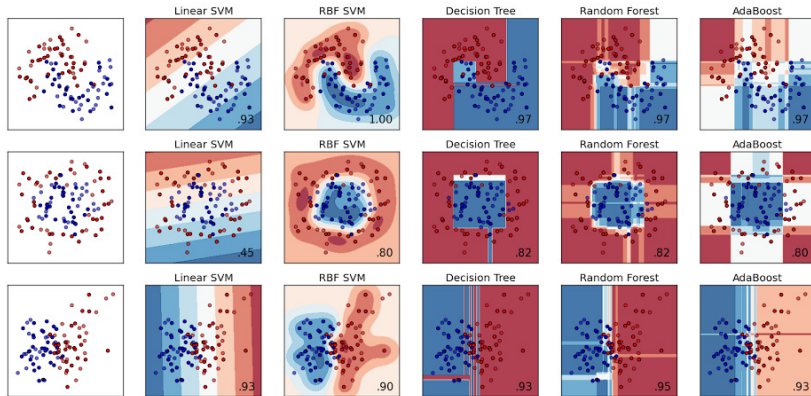
- **Отсев шума:** отбросить объекты с наибольшими w_i .
- **Модификация формулы для α_t на случай $N = 0$:**

$$\alpha_t := \frac{1}{2} \ln \frac{1 - N(b_t; W^\ell) + \frac{1}{\ell}}{N(b_t; W^\ell) + \frac{1}{\ell}};$$

- **Дополнительный критерий остановки:**
увеличение частоты ошибок на контрольной выборке.

Случайный лес и бустинг в сравнении с другими методами

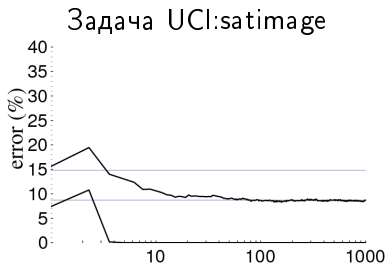
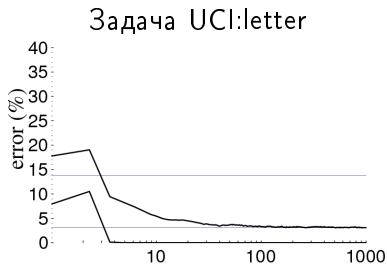
Эксперименты на трёх двумерных модельных выборках:



Решения могут выглядеть странно... тем не менее, RF и бустинг — одни из самых сильных универсальных методов в ML

Эксперименты с алгоритмом классификации AdaBoost

Удивительное отсутствие переобучения вплоть до $T = 1000$
(нижняя кривая — обучение, верхняя — тест):

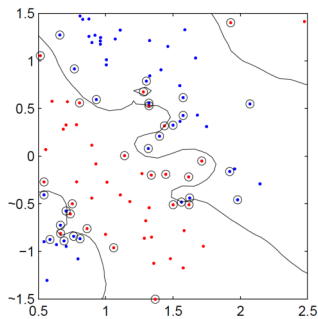
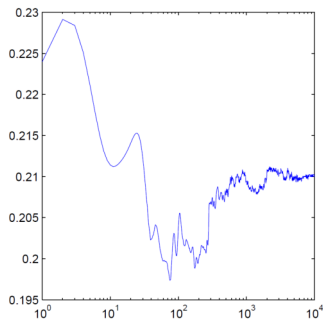


До этих экспериментов считалось, что увеличение числа параметров неизбежно приводит к переобучению

Schapire, Freund, Lee, Bartlett. Boosting the margin: a new explanation for the effectiveness of voting methods // Annals of Statistics, 1998.

Иногда AdaBoost всё же переобучается...

... но не сильно, и на тысячах базовых классификаторах.
Слева: зависимость ошибки на тестовой выборке от $|T|$.
Справа: разделяющая поверхность при переобучении.



G.Rätsch, T.Onoda, K.R.Müller. An improvement of AdaBoost to avoid overfitting. 1998.

Обоснование бустинга (случай классификации на 2 класса)

Усиленная частота ошибок классификатора $\text{sign } b(x)$, $b \in \mathcal{B}$:

$$\nu_\theta(b, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [b(x_i)y_i \leq \theta], \quad \theta > 0.$$

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

Теорема (Freund, Schapire, Bartlett, 1998)

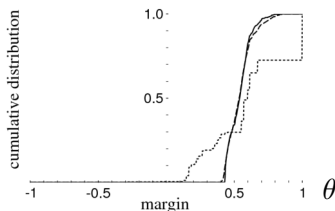
Если $|\mathcal{B}| < \infty$, то $\forall \theta > 0$, $\forall \eta \in (0, 1)$ с вероятностью $1 - \eta$

$$P[ya(x) < 0] \leq \nu_\theta(a, X^\ell) + C \sqrt{\frac{\ln |\mathcal{B}| \ln \ell}{\ell \theta^2} + \frac{1}{\ell} \ln \frac{1}{\eta}}$$

Основной вывод: оценка зависит от $|\mathcal{B}|$, 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)

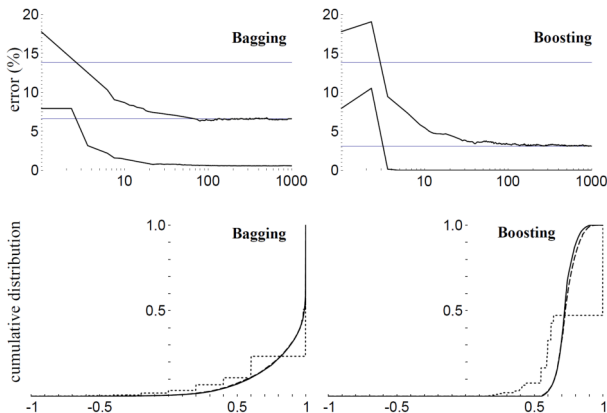


- С ростом T распределение отступов сдвигается вправо, то есть бустинг «раздвигает» классы в пространстве векторов растущей размерности $(b_1(x), \dots, b_T(x))$
- Значит, в оценке можно уменьшать второй член, увеличивая θ при неизменной $\nu_\theta(a, X^\ell) = \nu_0(a, X^\ell)$.
- Можно уменьшить второй член, если уменьшить $|\mathcal{B}|$, то есть взять простое семейство базовых алгоритмов.

Schapire R., Freund Y., Lee W. S., Bartlett P. Boosting the margin: a new explanation for the effectiveness of voting methods. 1998.

Бэггинг не столь успешно раздвигает классы

Ошибки на обучении и тесте. Снизу распределение отступов.



Schapire R., Freund Y., Lee W. S., Bartlett P. Boosting the margin: a new explanation for the effectiveness of voting methods. 1998.

Недостатки AdaBoost

- Чрезмерная чувствительность к выбросам из-за e^{-M}
- Неинтерпретируемое нагромождение из сотен алгоритмов
- Не удаётся строить короткие композиции из «сильных» алгоритмов типа SVM (только длинные из «слабых»)
- Требуются достаточно большие обучающие выборки (бэггинг обходится более короткими)

Способы устранения:

- Отсев выбросов по критерию увеличения веса w_i
- Градиентный бустинг с произвольными функциями потерь
- Явная оптимизация распределения отступов

Несколько эмпирических наблюдений:

- Веса алгоритмов не столь важны для выравнивания отступов
- Веса объектов не столь важны для обеспечения различности

Оптимизация распределения отступов на каждом шаге

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

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

Критерий качества ансамбля — число ошибок на обучении:

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

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

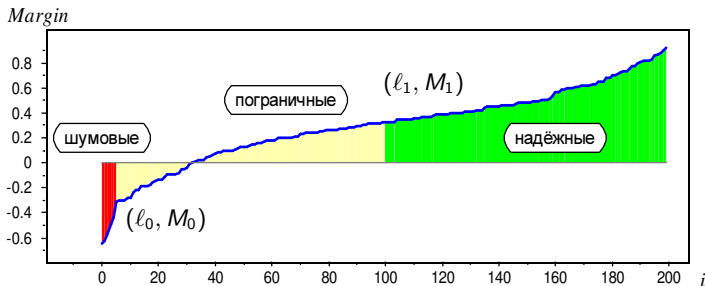
Эвристика: b_t компенсирует ошибки ансамбля,

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

$U_t = \{x_i: M_0 < M_{i,t-1} \leq M_1\}$, M_0, M_1 — параметры метода

Формирование выборки для обучения базового алгоритма

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:

$M_{i,t-1} < M_0$, $i < l_0$ — объект x_i шумовой;

$M_{i,t-1} > M_1$, $i > l_1$ — 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, \dots, b_T

- 1: $b_1 := \arg \min_b Q(b, X^\ell)$;
упорядочить X^ℓ по возрастанию $M_i = y_i b_1(x_i)$, $i = 1, \dots, \ell$;
- 2: **для всех** $t = 1, \dots, T$
- 3: **для всех** $k = \ell_1, \dots, \ell_2$ с шагом $\Delta\ell$
- 4: $U_t = \{x_i \in X^\ell : \ell_0 \leq i \leq k\}$;
- 5: $b_{tk} := \arg \min_b Q(b, U_t)$ — инкрементное обучение;
- 6: выбрать наилучший $b_t \in \{b_{tk}\}$ по критерию Q ;
- 7: обновить отступы: $M_i := M_i + y_i b_t(x_i)$, $i = 1, \dots, \ell$;
- 8: упорядочить выборку X^ℓ по возрастанию отступов M_i ;
- 9: опция: скорректировать значения параметров $\ell_0, \ell_1, \Delta\ell$;
- 10: **пока** Q существенно улучшается.

Результаты эксперимента на 4 задачах из репозитория UCI

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

	ionosphere	pima	bupa	votes
SVM	12,9	24,2	42	4,6
AdaBoost [SVM]	15	22,7	30,6	4
AdaBoost [SVM], $ T =$	(65)	(18)	(15)	(8)
ComBoost [SVM]	12,3	22,5	30,9	3,8
ComBoost [SVM], $ T =$	(5)	(2)	(5)	(3)

- При одинаковом критерии остановки $|T|$ существенно меньше у ComBoost по сравнению с AdaBoost
- ComBoost способен строить ансамбли из небольшого числа 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

Пусть теперь $Y = \{1, \dots, M\}$.

Композиция — простое голосование, причём каждый базовый алгоритм b_{yt} голосует только за свой класс y :

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

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).

- Ансамбли позволяют решать сложные задачи, которые плохо решаются отдельными базовыми алгоритмами
- Обычно ансамбль строится *алгоритмом-обёрткой* (envelop): базовые алгоритмы обучаются готовыми методами
- Базовые алгоритмы: компромисс качество/различность
- Две основные эвристики бустинга (и не только AdaBoost):
 - обучать базовые алгоритмы по одному
 - использовать гладкую замену пороговой функции потерь
- Важное открытие середины 90-х: обобщающая способность бустинга не ухудшается с ростом сложности T
- Практическое сравнение бустинга и бэггинга:
 - бустинг лучше для классов с границами сложной формы
 - бэггинг лучше для коротких обучающих выборок
 - бэггинг легче распараллеливается