

Композиции классификаторов

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)»

февраль 2011

Содержание

- 1 Композиции классификаторов**
 - Задачи обучения композиций
 - Классический алгоритм AdaBoost
 - Градиентный бустинг
- 2 Бэггинг и комитетные методы**
 - Бэггинг и метод случайных подпространств
 - Простое и взвешенное голосование
 - Голосование по старшинству
- 3 Смеси алгоритмов**
 - Идея областей компетентности
 - Итерационный метод обучения смеси
 - Последовательное наращивание смеси

Определение композиции

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

$a(x) = C(b(x))$ — алгоритм, где

$b: X \rightarrow R$ — базовый алгоритм (алгоритмический оператор),

$C: R \rightarrow Y$ — решающее правило,

R — пространство оценок;

Определение

Композиция базовых алгоритмов b_1, \dots, b_T

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

где $F: R^T \rightarrow R$ — корректирующая операция.

Зачем вводится R ?

В задачах классификации множество отображений

$\{F: R^T \rightarrow R\}$ существенно шире, чем $\{F: Y^T \rightarrow Y\}$.

Примеры пространств оценок и решающих правил

- **Пример 1:** 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 2:** 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 3:** регрессия, $Y = \mathbb{R}$:
 $C(b) \equiv b$ — решающее правило не нужно.

Примеры корректирующих операций

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

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

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

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

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

$$F(b_1(x), \dots, b_T(x)) = \sum_{t=1}^T g_t(x) b_t(x), \quad x \in X, \quad g_t: X \rightarrow \mathbb{R}.$$

Последовательное обучение композиции

Функционал качества базового алгоритма, \mathcal{L} — функция потерь:

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

Итерационный процесс:

$$b_1 = \arg \min_b Q(b, X^\ell); \quad (1)$$

$$b_2 = \arg \min_{b, F} Q(F(b_1, b), X^\ell);$$

...

$$b_t = \arg \min_{b, F} Q(F(b_1, \dots, b_{t-1}, b), X^\ell). \quad (2)$$

Идея: Свести задачу (2) к стандартной (1) с весами объектов и, возможно, с другой функцией потерь:

$$b_t = \arg \min_b \sum_{i=1}^{\ell} w_i \tilde{\mathcal{L}}(b(x_i), y_i).$$

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:

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

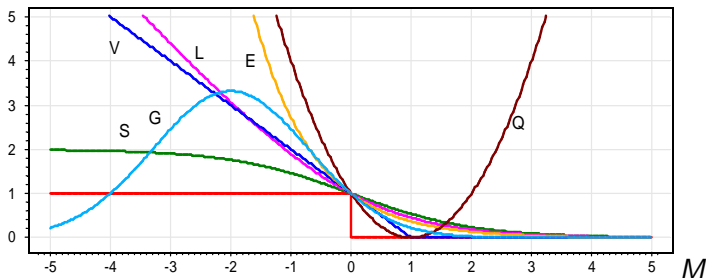
Функционал качества композиции — число ошибок на X^ℓ :

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

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);

Основная теорема бустинга, но прежде несколько обозначений

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 B — достаточно богатое семейство базовых алгоритмов.

Теорема (Freund, Schapire, 1996)

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

Тогда минимум функционала \tilde{Q}_T достигается при

$$b_T = \arg \max_{b \in B} \sqrt{P(b; \tilde{W}^\ell)} - \sqrt{N(b; \tilde{W}^\ell)}.$$

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

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

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. 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 $b \in B$, классифицирующий выборку хотя бы немного лучше, чем наугад: $N(b; U^\ell) < \frac{1}{2}$.

Тогда минимум функционала \tilde{Q}_T достигается при

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

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

Следствие 2. Сходимость

Теорема

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

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

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:

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

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

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

Алгоритм AdaBoost

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 (data stumps)

$$B = \left\{ b(x) = [f_j(x) \leq \theta] \mid j = 1, \dots, n, \theta \in \mathbb{R} \right\};$$

— для SVM бустинг не эффективен.

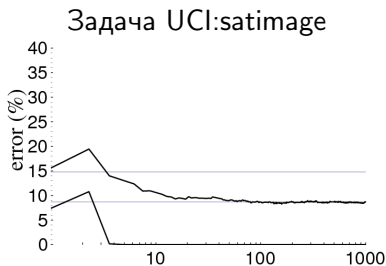
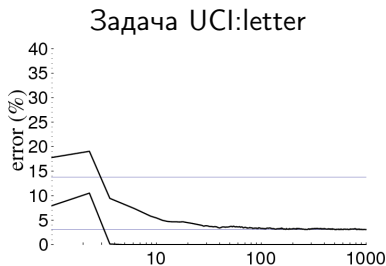
- **Отсев шума:** отбросить объекты с наибольшими w_i .
- **Модификация формулы для α_t на случай $N = 0$:**

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

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

Эксперименты с бустингом

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



Schapire, Freund, Lee, Bartlett (1998) Boosting the margin: a new explanation for the effectiveness of voting methods // Annals of Statistics Vol.26, No.5, Pp. 1651–1686.

Обоснование бустинга

Усиление понятия частоты ошибок алгоритма $a(x) = \text{sign } b(x)$:

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

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

Теорема (Freund, Schapire, Bartlett, 1998)

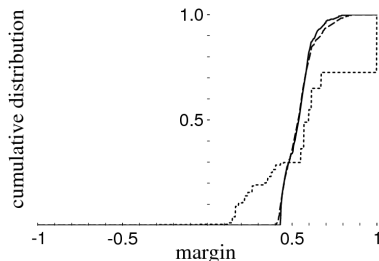
Если $|B| < \infty$, то $\forall \theta > 0$, $\forall \eta \in (0, 1)$ с вероятностью $1 - \eta$

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

Основной вывод: оценка не зависит от T явно.

Голосование не увеличивает сложность эффективно используемого множества алгоритмов.

Обоснование бустинга: что же всё-таки происходит?



- С ростом T распределение отступов сдвигается вправо. Значит, в оценке можно увеличить θ , не изменив $\nu_\theta(a, X^\ell)$ и уменьшив второй член.
- Ещё второй член можно уменьшить, если уменьшить $|B|$, то есть взять простое семейство базовых алгоритмов.

Недостатки AdaBoost

- Чрезмерная чувствительность к выбросам из-за e^{-M} .
- AdaBoost строит «чёрные ящики». Громоздкие композиции из сотен алгоритмов не интерпретируемы.
- Не удаётся строить короткие композиции из «сильных» алгоритмов типа SVM (только длинные из слабых).
- Требуются достаточно большие обучающие выборки (бэггинг обходится более короткими — см. далее).

Способы устранения:

- Другие аппроксимации пороговой функции потерь.
- Непрерывные вещественные базовые алгоритмы $b_t: X \rightarrow \mathbb{R}$.
- Явная оптимизация отступов (ComBoost может строить короткие композиции из хороших алгоритмов).

Обобщение бустинга

Возьмём $Y = \{\pm 1\}$, $b_t: X \rightarrow \mathbb{R}$, $C(b) = \text{sign}(b)$;

$\mathcal{L}(M)$ — функция потерь, гладкая функция отступа M ;

$M_T(x_i) = y_i \sum_{t=1}^T \alpha_t b_t(x_i)$ — отступ композиции на объекте x_i ;

Оценка сверху для числа ошибок композиции:

$$Q_T \leq \tilde{Q}_T = \sum_{i=1}^{\ell} \mathcal{L}(M_{T-1}(x_i) + \alpha y_i b(x_i)) \rightarrow \min_{\alpha, b}.$$

Линеаризация функции потерь по α в окрестности $\alpha = 0$:

$$\tilde{Q}_T \approx \sum_{i=1}^{\ell} \mathcal{L}(M_{T-1}(x_i)) - \alpha \sum_{i=1}^{\ell} \underbrace{-\mathcal{L}'(M_{T-1}(x_i))}_{w_i} y_i b(x_i) \rightarrow \min_b,$$

где w_i — веса объектов.

Принцип явной максимизации отступов

Минимизация линейризованного \tilde{Q}_T при фиксированном α

$$\tilde{Q}_T \approx \sum_{i=1}^{\ell} \mathcal{L}(M_{T-1}(x_i)) - \alpha \sum_{i=1}^{\ell} w_i y_i b(x_i) \rightarrow \min_b.$$

приводит к принципу *явной максимизации отступов* (direct optimization of margin, DOOM):

$$\sum_{i=1}^{\ell} w_i y_i b(x_i) \rightarrow \max_b.$$

Затем α определяется путём одномерной минимизации \tilde{Q}_T .

Итерации этих двух шагов приводят к алгоритму AnyBoost.

Замечание. AnyBoost переходит в AdaBoost в частном случае, при $b_t: X \rightarrow \{-1, 0, +1\}$ и $\mathcal{L}(M) = e^{-M}$.

Алгоритм AnyBoost

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

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

1: инициализировать отступы: $M_i := 0$, $i = 1, \dots, \ell$;

2: **для всех** $t = 1, \dots, T$

3: вычислить веса объектов:

$$w_i = -\mathcal{L}'(M_i), \quad i = 1, \dots, \ell;$$

4: обучить базовый алгоритм согласно принципу DOOM:

$$b_t := \arg \max_b \sum_{i=1}^{\ell} w_i y_i b(x_i);$$

5: решить задачу одномерной минимизации:

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

6: обновить значения отступов:

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

Резюме в конце лекции

- Композиции позволяют решать сложные задачи, которые плохо решаются отдельными алгоритмами.
- Обучать композицию целиком слишком сложно. Поэтому обучение композиции стараемся свести к вызову стандартных методов обучения базовых алгоритмов.
- Важное открытие середины 90-х: обобщающая способность бустинга не ухудшается с ростом сложности T .
- Существует очень много разновидностей бустинга. Бустингу можно придавать разные полезные свойства, выбирая базовое семейство и функцию потерь.
- Градиентный бустинг над решающими деревьями — один из лучших универсальных методов классификации.
- Бустинг хорошо строит длинные композиции из слабых классификаторов. А как строить короткие из сильных?

Стохастические методы построения композиций

- Алгоритмы в композиции должны быть различными;
- для этого их можно обучать по (случайным) подвыборкам,
- либо по (случайным) подмножествам признаков.

Первую идею реализует bagging (bootstrap aggregation) [Breiman, 1996], причём подвыборки берутся длины ℓ с возвращениями, как в методе bootstrap.

Вторую идею реализует RSM (random subspace method) [Duin, 2002].

Совместим обе идеи в одном алгоритме.

$\mathcal{F} = \{f_1, \dots, f_n\}$ — признаки,

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

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

- 1: **для всех** $t = 1, \dots, T$
- 2: $U :=$ случайное подмножество X^ℓ длины ℓ' ;
- 3: $\mathcal{G} :=$ случайное подмножество \mathcal{F} длины n' ;
- 4: $b_t := \mu(\mathcal{G}, U)$;
- 5: **если** $Q(b_t, U) > \varepsilon_1$ или $Q(b_t, X^\ell \setminus U) > \varepsilon_2$ **то**
- 6: не включать b_t в композицию;

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

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.
- Не удаётся строить короткие композиции из «сильных» алгоритмов типа SVM (только длинные из слабых).

Возможно ли строить композиции проще и аккуратнее?

Простое голосование в задаче классификации

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

Функционал качества композиции — число ошибок на обучении:

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

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

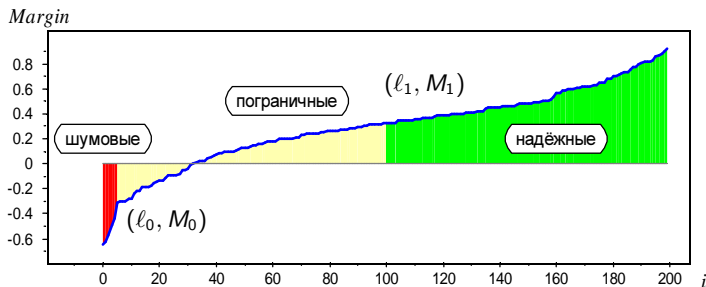
Эвристика: чтобы b_{t+1} компенсировал ошибки композиции,

$$Q(b, X^\ell, W^\ell) = \sum_{i=1}^{\ell} w_i [y_i b(x_i) < 0] \rightarrow \min_b,$$

положим $w_i = [M_0 < M_{it} \leq M_1]$ — бинарные веса объектов, где M_0, M_1 — параметры метода обучения.

Подбор параметров M_0 и M_1

Упорядочим объекты по возрастанию отступов M_{it} :



Два случая, когда b_{t+1} на объекте x_i обучать не надо:

$M_{it} < M_0$, $i < l_0$ — объект x_i шумовой;

$M_{it} > M_1$, $i > l_1$ — объект x_i уже надёжно классифицируется.

Алгоритм ComBoost (Committee Boosting)

Вход: обучающая выборка X^ℓ ; **параметры** T, ℓ_0, ℓ_1 ;

Выход: b_1, \dots, b_T

1: инициализировать веса и отступы:

$$w_i := 1; M_i := 0 \text{ для всех } i = 1, \dots, \ell;$$

2: **для всех** $t = 1, \dots, T$, пока не выполнен критерий останова

3: обучить базовый алгоритм:

$$b_t := \arg \min_b Q(b, X^\ell, W^\ell);$$

4: обновить значения отступов:

$$M_i := M_i + y_i b_t(x_i) \text{ для всех } i = 1, \dots, \ell;$$

5: упорядочить выборку X^ℓ по возрастанию отступов M_i ;

6: Опция: скорректировать значения параметров ℓ_0, ℓ_1 ;

7: $w_i := [\ell_0 < i \leq \ell_1]$ для всех $i = 1, \dots, \ell$;

Результаты эксперимента на 4 задачах из репозитория UCI

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

	ionosphere	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

ComBoost₀ — подбор l_1, l_2 по скользящему контролю;
ComBoost — оптимизация параметров l_1, l_2 на каждом шаге 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
ComBoost ₀ [SVM]	4	2	5	2
ComBoost [SVM]	5	2	5	3
AdaBoost [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 // Докл. всеросс. конф. ММРО-13, 2007, С.180–183.

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 :

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

В алгоритме только два изменения:

— изменится определение отступа M_i :

$$M_i = \frac{1}{|T_{y_i}|} \sum_{t \in T_{y_i}} b_{y_i t}(x_i) - \max_{y \in Y \setminus \{y_i\}} \frac{1}{|T_y|} \sum_{t \in T_y} b_{yt}(x_i).$$

— в алгоритме ComBoost на шаге 3 придётся решать, за какой класс строить очередной базовый алгоритм, кроме того, немного изменится шаг 4.

Преобразование простого голосования во взвешенное

Линейный классификатор над признаками $b_t(x)$:

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

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

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

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

Пример. Наивный байесовский классификатор приводит к простому аналитическому решению:

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

где p_t — оценка вероятности ошибки базового алгоритма b_t .

Голосование по старшинству

Возьмём $Y = \{1, \dots, M\}$, $R = \{0, 1\}$.

Голосование по старшинству $F: R^T \rightarrow Y$ задаётся алгоритмом:

- 1: **для всех** $t = 1, \dots, T$
- 2: **если** $b_t(x) = 1$ **то вернуть** c_t ;
- 3: **вернуть** c_0 .

b_1, \dots, b_T — упорядоченный список базовых алгоритмов;

$c_1, \dots, c_T \in Y$ — параметры корректирующей операции;

c_0 — отказ от классификации либо «наименее опасный класс».

$b_t(x) = 1 \iff$ правило b_t *покрывает* (или *выделяет*) объект x
(при этом объект x относится к классу c_t).

Жадный алгоритм обучения

Вход: выборка X^ℓ ; **параметры** T, ℓ_0 , **штраф за отказ** $\lambda \in [0, 1]$

Выход: $b_1, c_1, \dots, b_T, c_T, c_0$.

- 1: инициализировать веса: $w_i := 1$ для всех $i = 1, \dots, \ell$;
- 2: **для всех** $t = 1, \dots, T$, пока $\#\{i: w_i > 0\} > \ell_0$
- 3: выбрать класс c_t ;
- 4: сформировать бинарный вектор ответов для настройки b_t :
 $z_i := [y_i = c_t]$, для всех $i = 1, \dots, \ell$;
- 5: сформировать веса объектов:
$$w_i := \begin{cases} 0, & w_i = 0; \\ \lambda, & w_i \neq 0 \text{ и } y_i = c_t; \\ 1, & w_i \neq 0 \text{ и } y_i \neq c_t; \end{cases} \quad \text{для всех } i = 1, \dots, \ell$$
- 6: обучить базовый алгоритм различать объекты класса c_t :
 $b_t := \arg \min_b Q(b, (x_i, z_i)_{i=1}^\ell, (w_i)_{i=1}^\ell)$;
- 7: **если** $b_t(x_i) = 1$ **то** $w_i := 0$ для всех $i = 1, \dots, \ell$;

Эвристики и рекомендации

Подбор штрафа λ за не-выделение объекта своего класса:

- $\lambda \rightarrow 0$: штраф только за выделение чужого \rightarrow скорее всего, b_t будет выделять слишком мало своих \rightarrow переусложнение композиции \rightarrow переобучение;
- $\lambda \rightarrow 1$: одинаковый штраф за выделение чужого и не-выделение своего \rightarrow скорее всего, b_t будет делать слишком много ошибок \rightarrow недообучение;
- $\lambda \approx 0.2$ — с этого можно начинать, затем адаптировать.

Стратегии формирования последовательности c_1, \dots, c_T :

- Брать самый непокрытый класс.
- Брать тот класс, для которого построился лучший b_t .
- Задать приоритетный порядок классов.
- Построить список алгоритмов для каждого класса отдельно.

Резюме в конце лекции

- Простые эвристические методы обучения композиций могут строить более короткие композиции, работающие не хуже бустинга.
- Но в них появляются параметры, которые приходится подбирать индивидуально в каждой задаче.
- Голосование по большинству (простое или взвешенное), как правило, лучше голосования по старшинству.

Квазилинейная композиция (смесь алгоритмов)

Смесь алгоритмов (Mixture of Experts)

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

$b_t: X \rightarrow \mathbb{R}$ — 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 $\alpha \in \mathbb{R}^n$:

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

- 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 $\mathcal{L}(b, y)$ 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 $\sum_{t=1}^T g_t(x) = 1$ и функция потерь \mathcal{L} выпукла.

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

- 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: обучение смеси алгоритмов

Итерационный процесс, аналогичный EM-алгоритму:

Вход: выборка X^ℓ , нормированные $(g_t)_{t=1}^T$, параметры T, δ, γ ;

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

1: **повторять**

2: $g_t^0 := g_t$ для всех $t = 1, \dots, T$;

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

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

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

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

5: нормировать компетентности:

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

6: **пока** $\max_{t,i} |g_t(x_i) - g_t^0(x_i)| > \delta$.

Обучение смеси с автоматическим определением числа T

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

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

1: начальное приближение:

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

2: для всех $t = 2, \dots, T$

3: множество трудных объектов:

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

4: если $|X_t| \leq l_0$ то выход;

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

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

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

Резюме в конце лекции

- Обучение смесей алгоритмов основано на принципе «разделяй и властвуй».
- Смеси алгоритмов имеет смысл строить в тех задачах, где есть априорные соображения о виде областей компетентности.
- Кроме последовательного метода построения смесей алгоритмов, известен ещё иерархический.