Введение в искусственный интеллект. Машинное обучение

Лекция 10. Бустинг

МаТИС

26 апреля 2019г.

План лекции

1. AdaBoost
2. AnyBoost
3. GB / SGB

Типы бустинга

Обозначим взвешенную сумму выходов базовых классификаторов $b_t(x)$ как $a(x) = \sum_{t=1}^{T} \alpha_t b_t, \alpha_t \in \mathbb{R}$.

AdaBoost

- Базовые алгоритмы $b_t(x)$ принимают значения из дискретного множества (например, $\{-1, +1\}$),
- Функция потерь: $e^{-y_i a(x_i)}$

Типы бустинга

Обозначим взвешенную сумму выходов базовых классификаторов $b_t(x)$ как $a(x) = \sum_{t=1}^{T} \alpha_t b_t, \alpha_t \in \mathbb{R}$.

AdaBoost

- Базовые алгоритмы $b_t(x)$ принимают значения из дискретного множества (например, $\{-1, +1\}$),
- Функция потерь: $e^{-y_i a(x_i)}$

AnyBoost

- Базовые алгоритмы $b_t(x)$ принимают значения из $\mathbb{R}$,
- Функция потерь – гладкая функция от отступа $L(y_i a(x_i))$

Типы бустинга

Обозначим взвешенную сумму выходов базовых классификаторов $b_t(x)$ как $a(x) = \sum_{t=1}^{T} \alpha_t b_t, \alpha_t \in \mathbb{R}$.

AdaBoost

- Базовые алгоритмы $b_t(x)$ принимают значения из дискретного множества (например, $\{-1, +1\}$),
- Функция потерь: $e^{-y_i a(x_i)}$

AnyBoost

- Базовые алгоритмы $b_t(x)$ принимают значения из $\mathbb{R}$,
- Функция потерь – гладкая функция от отступа $L(y_i a(x_i))$

Gradient Boosting

- Базовые алгоритмы $b_t(x)$ принимают значения из $\mathbb{R}$,
- Функция потерь – гладкая функция от пары $L(y_i, a(x_i))$

Обозначения для AdaBoost

- Базовый алгоритм $b_t : X \to \{-1, +1\}$
- Вектор весов (взвешиваем объекты) $W^m = (w_1, \dots, w_m)$: $w_i = e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)}$
- Нормировка: $\widetilde{w}_i = \frac{w_i}{\sum_{j=1}^m w_j} \Rightarrow \sum_{i=1}^m \widetilde{w}_i = 1, 0 \le \widetilde{w}_i \le 1$
- Взвешенное число правильных классификаций алгоритма $b(x)$ по нормированному вектору U^m: $P(b; U^m) = \sum_{i=1}^m u_i[b(x) = y_i]$
- Взвешенное число ошибочных классификаций алгоритма $b(x)$ по нормированному вектору U^m: $N(b; U^m) = \sum_{i=1}^m u_i[b(x) = -y_i]$
- $P + N = 1.$

Классический AdaBoost – теорема

Теорема

Если для любого нормированного вектора U^m существует алгоритм $b \in A$, т.ч. $N(b; U^m) < \frac{1}{2}$, то минимум аппроксимированного Э.Р. $\widetilde{R_T}$ достигается на:

- $b_T = \arg\min_{b \in A} N(b; \widetilde{W^m})$
- $\alpha_T = \frac{1}{2} \ln \frac{1 - N(b; \widetilde{W^m})}{N(b; \widetilde{W^m})}$

Алгоритм AdaBoost

Алгоритм

- Инициализация весов: $w_i = \frac{1}{m}, i = 1, \ldots, m,$

Для $t = 1, \ldots, T$

- Обучение базового алгоритма $b_t = \arg\min_{b \in A} N(b; \widetilde{W^m}),$
- Вычисление нового веса $\alpha_t = \frac{1}{2} \ln \frac{1 - N(b_t; \widetilde{W^m})}{N(b_t; \widetilde{W^m})},$
- Обновление весов $w_i = w_i e^{-\alpha_t y_i b_t(x_i)}, i = 1, \ldots, m,$
- Перенормировка весов $w_i = \frac{w_i}{\sum_{j=1}^m w_j}, i = 1, \ldots, m.$

Об обновлении весов

Замечание относительно шага обновления весов $w_i = w_i e^{-\alpha_t y_i b_t(x_i)}$, $i = 1, \ldots, m$.

Об обновлении весов

Замечание относительно шага обновления весов $w_i = w_i e^{-\alpha_t y_i b_t(x_i)}$, $i = 1, \ldots, m$.

- b_t ошибается на объекте $x_i \Rightarrow y_i \neq b_t(x_i) \Rightarrow y_i b_t(x_i) = -1$

Об обновлении весов

Замечание относительно шага обновления весов $w_i = w_i e^{-\alpha_t y_i b_t(x_i)}$, $i = 1, \ldots, m$.

- b_t ошибается на объекте $x_i \Rightarrow y_i \neq b_t(x_i) \Rightarrow y_i b_t(x_i) = -1$
- b_t правильно классифицирует объект $x_i \Rightarrow y_i = b_t(x_i) \Rightarrow y_i b_t(x_i) = +1$

Об обновлении весов

Замечание относительно шага обновления весов $w_i = w_i e^{-\alpha_t y_i b_t(x_i)}, i = 1, \ldots, m$.

- b_t ошибается на объекте $x_i \Rightarrow y_i \neq b_t(x_i) \Rightarrow y_i b_t(x_i) = -1$
- b_t правильно классифицирует объект $x_i \Rightarrow y_i = b_t(x_i) \Rightarrow y_i b_t(x_i) = +1$
- Поскольку $N(b; U^m) < \frac{1}{2}$ для любого нормированного U^m, то
  $\alpha_t = \frac{1}{2} \ln \frac{1-N(b_t;\widetilde{W^m})}{N(b_t;\widetilde{W^m})} > \frac{1}{2} \ln \frac{\frac{1}{2}}{\frac{1}{2}} = \frac{1}{2} \ln 1 = 0$

Об обновлении весов

Замечание относительно шага обновления весов $w_i = w_i e^{-\alpha_t y_i b_t(x_i)}, i = 1, \dots, m$.

- b_t ошибается на объекте $x_i \Rightarrow y_i \neq b_t(x_i) \Rightarrow y_i b_t(x_i) = -1$
- b_t правильно классифицирует объект $x_i \Rightarrow y_i = b_t(x_i) \Rightarrow y_i b_t(x_i) = +1$
- Поскольку $N(b; U^m) < \frac{1}{2}$ для любого нормированного U^m, то $\alpha_t = \frac{1}{2} \ln \frac{1 - N(b_t; \widetilde{W^m})}{N(b_t; \widetilde{W^m})} > \frac{1}{2} \ln \frac{\frac{1}{2}}{\frac{1}{2}} = \frac{1}{2} \ln 1 = 0$
- Вес объекта x_i увеличивается в e^{α_t} раз, когда b_t допускает на нем ошибку,

Об обновлении весов

Замечание относительно шага обновления весов $w_i = w_i e^{-\alpha_t y_i b_t(x_i)}$, $i = 1, \ldots, m$.

- b_t ошибается на объекте $x_i \Rightarrow y_i \neq b_t(x_i) \Rightarrow y_i b_t(x_i) = -1$
- b_t правильно классифицирует объект $x_i \Rightarrow y_i = b_t(x_i) \Rightarrow y_i b_t(x_i) = +1$
- Поскольку $N(b; U^m) < \frac{1}{2}$ для любого нормированного U^m, то $\alpha_t = \frac{1}{2} \ln \frac{1 - N(b_t; \widetilde{W^m})}{N(b_t; \widetilde{W^m})} > \frac{1}{2} \ln \frac{\frac{1}{2}}{\frac{1}{2}} = \frac{1}{2} \ln 1 = 0$
- Вес объекта x_i увеличивается в e^{α_t} раз, когда b_t допускает на нем ошибку,
- Вес объекта x_i уменьшается в e^{α_t} раз, когда b_t правильно его классифицирует,

Об обновлении весов

Замечание относительно шага обновления весов $w_i = w_i e^{-\alpha_t y_i b_t(x_i)}$, $i = 1, \ldots, m$.

- b_t ошибается на объекте $x_i \Rightarrow y_i \neq b_t(x_i) \Rightarrow y_i b_t(x_i) = -1$
- b_t правильно классифицирует объект $x_i \Rightarrow y_i = b_t(x_i) \Rightarrow y_i b_t(x_i) = +1$
- Поскольку $N(b; U^m) < \frac{1}{2}$ для любого нормированного U^m, то $\alpha_t = \frac{1}{2} \ln \frac{1 - N(b_t; \widetilde{W^m})}{N(b_t; \widetilde{W^m})} > \frac{1}{2} \ln \frac{\frac{1}{2}}{\frac{1}{2}} = \frac{1}{2} \ln 1 = 0$
- Вес объекта x_i увеличивается в e^{α_t} раз, когда b_t допускает на нем ошибку,
- Вес объекта x_i уменьшается в e^{α_t} раз, когда b_t правильно его классифицирует,
- Т.о. наибольший вес будет у тех объектов, которые чаще неправильно классифицировались предыдущими алгоритмами (т.е. классификатору прежде всего нужно сосредоточиться именно на них!).

О фильтрации выбросов

- После построения некоторого количества базовых алгоритмов (например, $T = 10 \dots 30$) можно проанализировать распределение весов объектов:

О фильтрации выбросов

- После построения некоторого количества базовых алгоритмов (например, $T = 10 \dots 30$) можно проанализировать распределение весов объектов:
  - Объекты с максимальными весами $\widetilde{w}_i$, скорее всего, являются шумовыми выбросами

О фильтрации выбросов

- После построения некоторого количества базовых алгоритмов (например, $T = 10 \dots 30$) можно проанализировать распределение весов объектов:
  - Объекты с максимальными весами $\widetilde{w}_i$, скорее всего, являются шумовыми выбросами
  - Их нужно исключить из выборки

О фильтрации выбросов

- После построения некоторого количества базовых алгоритмов (например, $T = 10 \dots 30$) можно проанализировать распределение весов объектов:
  - Объекты с максимальными весами $\widetilde{w}_i$, скорее всего, являются шумовыми выбросами
  - Их нужно исключить из выборки
  - После чего начать построение композиции заново

О фильтрации выбросов

- После построения некоторого количества базовых алгоритмов (например, $T = 10\dots30$) можно проанализировать распределение весов объектов:
  - Объекты с максимальными весами $\widetilde{w}_i$, скорее всего, являются шумовыми выбросами
  - Их нужно исключить из выборки
  - После чего начать построение композиции заново
- Бустинг можно использовать как универсальный метод фильтрации выбросов перед применением любого другого метода классификации

Обобщающая способность бустинга: эмпирические замечания

- Во многих экспериментах тестовая ошибка практически постоянно уменьшалась по мере увеличения числа алгоритмов в композиции

Обобщающая способность бустинга: эмпирические замечания

- Во многих экспериментах тестовая ошибка практически постоянно уменьшалась по мере увеличения числа алгоритмов в композиции
  - Часто тестовая ошибка уменьшалась даже после достижения нулевой ошибки на обучающей выборке!

Обобщающая способность бустинга: эмпирические замечания

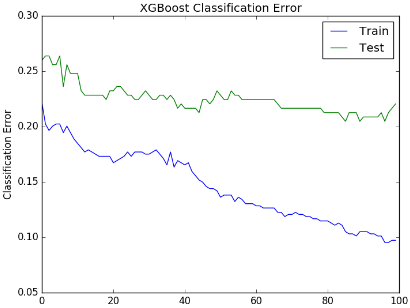
- Во многих экспериментах тестовая ошибка практически постоянно уменьшалась по мере увеличения числа алгоритмов в композиции
  - Часто тестовая ошибка уменьшалась даже после достижения нулевой ошибки на обучающей выборке!
- *Возможное теоретическое обоснование*: взвешенное голосование не увеличивает эффективную сложность алгоритма (т.о. не переобучаемся), а сглаживает ответы базовых алгоритмов

Обобщающая способность бустинга: эмпирические замечания

- Во многих экспериментах тестовая ошибка практически постоянно уменьшалась по мере увеличения числа алгоритмов в композиции
  - Часто тестовая ошибка уменьшалась даже после достижения нулевой ошибки на обучающей выборке!
- *Возможное теоретическое обоснование*: взвешенное голосование не увеличивает эффективную сложность алгоритма (т.о. не переобучаемся), а сглаживает ответы базовых алгоритмов
  - Т.к. стараемся увеличить отступы $y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)$

Обобщающая способность бустинга: эмпирические замечания

- Во многих экспериментах тестовая ошибка практически постоянно уменьшалась по мере увеличения числа алгоритмов в композиции
  - Часто тестовая ошибка уменьшалась даже после достижения нулевой ошибки на обучающей выборке!
- *Возможное теоретическое обоснование*: взвешенное голосование не увеличивает эффективную сложность алгоритма (т.о. не переобучаемся), а сглаживает ответы базовых алгоритмов
  - Т.к. стараемся увеличить отступы $y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)$
  - Тем не менее, бустинг не идеален: иногда получается его переобучить



- Что использовать в качестве базовых классификаторов:

AdaBoost: эмпирические замечания

- Что использовать в качестве базовых классификаторов:
    - Чаще всего используют решающие деревья

AdaBoost: эмпирические замечания

- Что использовать в качестве базовых классификаторов:
  - Чаще всего используют решающие деревья
  - Также используют совсем вырожденные случаи – т.н. “пни”: $b(x) = [f_j(x) \lesseqgtr r_j]$, где $x = (f_1(x), \ldots, f_n(x)) \in \mathbb{R}^n$

AdaBoost: эмпирические замечания

- Что использовать в качестве базовых классификаторов:
  - Чаще всего используют решающие деревья
  - Также используют совсем вырожденные случаи – т.н. “пни”: $b(x) = [f_j(x) \lessgtr r_j]$, где $x = (f_1(x), \ldots, f_n(x)) \in \mathbb{R}^n$
  - SVM используется редко (обучается достаточно долго, прироста большого не дает)

AdaBoost: эмпирические замечания

- Что использовать в качестве базовых классификаторов:
  - Чаще всего используют решающие деревья
  - Также используют совсем вырожденные случаи – т.н. “пни”: $b(x) = [f_j(x) \lessgtr r_j]$, где $x = (f_1(x), \ldots, f_n(x)) \in \mathbb{R}^n$
  - SVM используется редко (обучается достаточно долго, прироста большого не дает)
- Если вдруг при обучении получается нулевая ошибка ($N = 0$), то формула для выбора оптимального коэффициента приобретает вид $\alpha = \frac{1}{2} \ln \frac{1-N+\frac{1}{m}}{N+\frac{1}{m}} = \frac{1}{2} \ln(m+1)$

AdaBoost: эмпирические замечания

- Что использовать в качестве базовых классификаторов:
  - Чаще всего используют решающие деревья
  - Также используют совсем вырожденные случаи – т.н. “пни”: $b(x) = [f_j(x) \leqslant r_j]$, где $x = (f_1(x), \ldots, f_n(x)) \in \mathbb{R}^n$
  - SVM используется редко (обучается достаточно долго, прироста большого не дает)
- Если вдруг при обучении получается нулевая ошибка ($N = 0$), то формула для выбора оптимального коэффициента приобретает вид $\alpha = \frac{1}{2} \ln \frac{1 - N + \frac{1}{m}}{N + \frac{1}{m}} = \frac{1}{2} \ln(m + 1)$
- Нужно периодически производить фильтрацию выбросов в обучающей выборке

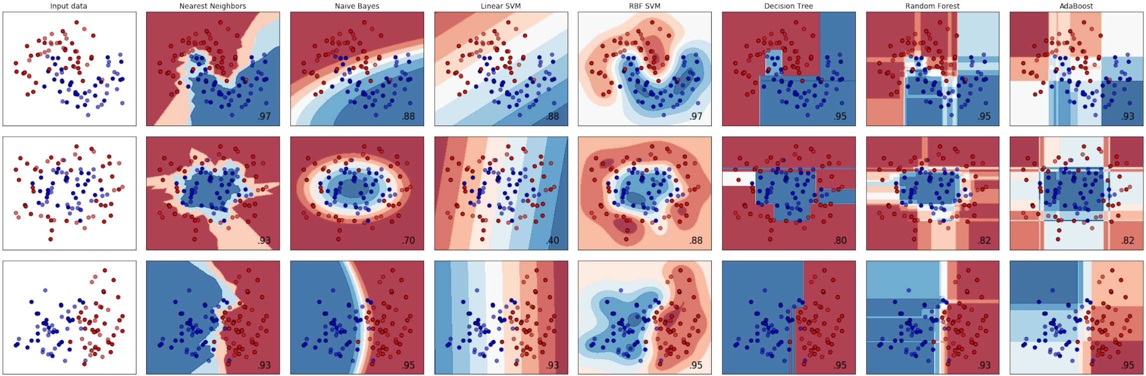
Визуализация работы основных методов классификации

Посмотрим результаты работы основных классификаторов на трех разных задачах[1].

[1] https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Визуализация работы основных методов классификации

Посмотрим результаты работы основных классификаторов на трех разных задачах[1].



[1] https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)

Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации

Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации
- Время обучения ансамбля (веса) на порядок меньше времени обучения базовых алгоритмов

Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации
- Время обучения ансамбля (веса) на порядок меньше времени обучения базовых алгоритмов
- Можно фильтровать выбросы

Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации
- Время обучения ансамбля (веса) на порядок меньше времени обучения базовых алгоритмов
- Можно фильтровать выбросы

Минусы

- Чувствителен к выбросам

Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации
- Время обучения ансамбля (веса) на порядок меньше времени обучения базовых алгоритмов
- Можно фильтровать выбросы

Минусы

- Чувствителен к выбросам
- Композиция совершенно неинтерпретируема

Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации
- Время обучения ансамбля (веса) на порядок меньше времени обучения базовых алгоритмов
- Можно фильтровать выбросы

Минусы

- Чувствителен к выбросам
- Композиция совершенно неинтерпретируема
- Базовые алгоритмы должны быть достаточно простыми, и их должно быть много (а лучше бы наоборот)

Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации
- Время обучения ансамбля (веса) на порядок меньше времени обучения базовых алгоритмов
- Можно фильтровать выбросы

Минусы

- Чувствителен к выбросам
- Композиция совершенно неинтерпретируема
- Базовые алгоритмы должны быть достаточно простыми, и их должно быть много (а лучше бы наоборот)
- Необходимость в достаточно большой обучающей выборке (т.к. нет процедуры бутстрэпа)

AnyBoost – обоснование

Перейдём к более общему случаю:

AnyBoost – обоснование

Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \to \mathbb{R}$

AnyBoost – обоснование

Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \to \mathbb{R}$
- Функции потерь $L(h_T)$, гладкой от отступа $h_T(x_i) = y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)$

AnyBoost – обоснование

Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \to \mathbb{R}$
- Функции потерь $L(h_T)$, гладкой от отступа $h_T(x_i) = y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)$

Принцип минимизации аппроксимированного Э.Р.:
$R_T \leq \widetilde{R_T} = \sum_{i=1}^{m} L(h_{T-1}(x_i) + \alpha_T y_i b_T(x_i)) \to \min_{\alpha_T, b_T}$.

AnyBoost – обоснование

Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \to \mathbb{R}$
- Функции потерь $L(h_T)$, гладкой от отступа $h_T(x_i) = y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)$

Принцип минимизации аппроксимированного Э.Р.:
$R_T \leq \widetilde{R_T} = \sum_{i=1}^{m} L(h_{T-1}(x_i) + \alpha_T y_i b_T(x_i)) \to \min_{\alpha_T, b_T}$.
Вспомним разложение Тейлора функции $f(x)$ в окрестности точки x_0:
$f(x) \approx f(x_0) + (x - x_0)f'(x_0)$.

AnyBoost – обоснование

Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \to \mathbb{R}$
- Функции потерь $L(h_T)$, гладкой от отступа $h_T(x_i) = y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)$

Принцип минимизации аппроксимированного Э.Р.:
$R_T \leq \widetilde{R_T} = \sum_{i=1}^{m} L(h_{T-1}(x_i) + \alpha_T y_i b_T(x_i)) \to \min_{\alpha_T, b_T}$.
Вспомним разложение Тейлора функции $f(x)$ в окрестности точки x_0:
$f(x) \approx f(x_0) + (x - x_0) f'(x_0)$.
Воспользуемся этим разложением для аппроксимированного Э.Р.:

- Пусть $x = h_{T-1}(x_i) + \alpha_T y_i b_T(x_i), x_0 = h_{T-1}(x_i)$

AnyBoost – обоснование

Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \to \mathbb{R}$
- Функции потерь $L(h_T)$, гладкой от отступа $h_T(x_i) = y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)$

Принцип минимизации аппроксимированного Э.Р.:
$R_T \leq \widetilde{R_T} = \sum_{i=1}^{m} L(h_{T-1}(x_i) + \alpha_T y_i b_T(x_i)) \to \min_{\alpha_T, b_T}$.
Вспомним разложение Тейлора функции $f(x)$ в окрестности точки x_0:
$f(x) \approx f(x_0) + (x - x_0) f'(x_0)$.
Воспользуемся этим разложением для аппроксимированного Э.Р.:

- Пусть $x = h_{T-1}(x_i) + \alpha_T y_i b_T(x_i), x_0 = h_{T-1}(x_i)$
- Тогда $\widetilde{R_T} \approx \sum_{i=1}^{m} L(h_{T-1}(x_i)) + \alpha_T \sum_{i=1}^{m} L'(h_{T-1}(x_i)) y_i b_T(x_i)$

AnyBoost – обоснование

Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \to \mathbb{R}$
- Функции потерь $L(h_T)$, гладкой от отступа $h_T(x_i) = y_i \sum_{t=1}^{T} \alpha_t b_t(x_i)$

Принцип минимизации аппроксимированного Э.Р.:
$R_T \leq \widetilde{R_T} = \sum_{i=1}^{m} L(h_{T-1}(x_i) + \alpha_T y_i b_T(x_i)) \to \min_{\alpha_T, b_T}$.
Вспомним разложение Тейлора функции $f(x)$ в окрестности точки x_0:
$f(x) \approx f(x_0) + (x - x_0)f'(x_0)$.
Воспользуемся этим разложением для аппроксимированного Э.Р.:

- Пусть $x = h_{T-1}(x_i) + \alpha_T y_i b_T(x_i), x_0 = h_{T-1}(x_i)$
- Тогда $\widetilde{R_T} \approx \sum_{i=1}^{m} L(h_{T-1}(x_i)) + \alpha_T \sum_{i=1}^{m} L'(h_{T-1}(x_i)) y_i b_T(x_i)$
- Обозначив за $w_i = -L'(h_{T-1}(x_i))$, получаем
  $\widetilde{R_T} \approx \sum_{i=1}^{m} L(h_{T-1}(x_i)) - \alpha_T \sum_{i=1}^{m} w_i y_i b_T(x_i)$

AnyBoost – обоснование

$$\widetilde{R_T} \approx \sum_{i=1}^{m} L(h_{T-1}(x_i)) - \alpha_T \sum_{i=1}^{m} w_i y_i b_T(x_i)$$

Зафиксировав α_T, переходим от задачи двумерной оптимизации $\widetilde{R_T} \to \min_{\alpha_T, b_T}$ к одномерной (по алгоритму):

AnyBoost – обоснование

$$\widetilde{R_T} \approx \sum_{i=1}^{m} L(h_{T-1}(x_i)) - \alpha_T \sum_{i=1}^{m} w_i y_i b_T(x_i)$$

Зафиксировав α_T, переходим от задачи двумерной оптимизации $\widetilde{R_T} \to \min_{\alpha_T, b_T}$ к одномерной (по алгоритму):

$$\sum_{i=1}^{m} w_i y_i b_T(x_i) \to \max_{b_T}$$

AnyBoost – обоснование

$$\widetilde{R_T} \approx \sum_{i=1}^{m} L(h_{T-1}(x_i)) - \alpha_T \sum_{i=1}^{m} w_i y_i b_T(x_i)$$

Зафиксировав α_T, переходим от задачи двумерной оптимизации $\widetilde{R_T} \to \min_{\alpha_T, b_T}$ к одномерной (по алгоритму):

$$\sum_{i=1}^{m} w_i y_i b_T(x_i) \to \max_{b_T}$$

Затем определяем α_T, подставив найденный b_T.

Алгоритм

- Инициализация отступов: $h^i = 0, i = 1, \ldots, m$,

Алгоритм AnyBoost

Алгоритм

- Инициализация отступов: $h^i = 0, i = 1, \dots, m$,

Для $t = 1, \dots, T$

- Вычисление весов $w_i = -L'(h^i)$,

Алгоритм AnyBoost

Алгоритм

- Инициализация отступов: $h^i = 0, i = 1, \ldots, m$,

Для $t = 1, \ldots, T$

- Вычисление весов $w_i = -L'(h^i)$,
- Обучение нового базового алгоритма $b_t = \arg\max_b \sum_{i=1}^{m} w_i y_i b(x_i)$,

Алгоритм AnyBoost

Алгоритм

- Инициализация отступов: $h^i = 0, i = 1, \ldots, m$,

Для $t = 1, \ldots, T$

- Вычисление весов $w_i = -L'(h^i)$,
- Обучение нового базового алгоритма $b_t = \arg\max_b \sum_{i=1}^m w_i y_i b(x_i)$,
- Вычисление нового веса $\alpha_t = \arg\min_\alpha \sum_{i=1}^m L(h^i + \alpha y_i b_t(x_i))$,

Алгоритм AnyBoost

Алгоритм

- Инициализация отступов: $h^i = 0, i = 1, \dots, m,$

Для $t = 1, \dots, T$

- Вычисление весов $w_i = -L'(h^i),$
- Обучение нового базового алгоритма $b_t = \arg\max_b \sum_{i=1}^m w_i y_i b(x_i),$
- Вычисление нового веса $\alpha_t = \arg\min_\alpha \sum_{i=1}^m L(h^i + \alpha y_i b_t(x_i)),$
- Обновление отступов $h^i = h^i + \alpha_t y_i b_t(x_i).$

Градиентный бустинг – обозначения

Рассмотрим самый общий случай – произвольную функцию потерь $L(a, y)$.
Функционал качества: $\widetilde{R_T} = \sum_{i=1}^{m} L(\sum_{t=1}^{T-1} \alpha_t b_t(x_i) + \alpha_T b_T(x_i), y_i) \to \min_{\alpha_T, b_T}$.

Градиентный бустинг – обозначения

Рассмотрим самый общий случай – произвольную функцию потерь $L(a, y)$.
Функционал качества: $\widetilde{R_T} = \sum_{i=1}^{m} L(\sum_{t=1}^{T-1} \alpha_t b_t(x_i) + \alpha_T b_T(x_i), y_i) \to \min_{\alpha_T, b_T}$.
Обозначения:

- Приближение для объекта x_i на шаге t: f_i^t,

Градиентный бустинг – обозначения

Рассмотрим самый общий случай – произвольную функцию потерь $L(a, y)$.
Функционал качества: $\widetilde{R_T} = \sum_{i=1}^{m} L(\sum_{t=1}^{T-1} \alpha_t b_t(x_i) + \alpha_T b_T(x_i), y_i) \to \min_{\alpha_T, b_T}$.
Обозначения:

- Приближение для объекта x_i на шаге t: f_i^t,
- Тогда функционал качества примет вид:
  $\widetilde{R_T} = \sum_{i=1}^{m} L(f_i^{T-1} + \alpha_T b_T(x_i), y_i) \to \min_{\alpha_T, b_T}$

Градиентный бустинг – обоснование

$$\widetilde{R_T} = \sum_{i=1}^{m} L(f_i^{T-1} + \alpha_T b_T(x_i), y_i) \to \min_{\alpha_T, b_T}$$

Градиентный бустинг – обоснование

$$\widetilde{R_T} = \sum_{i=1}^{m} L(f_i^{T-1} + \alpha_T b_T(x_i), y_i) \rightarrow \min_{\alpha_T, b_T}$$

Применение градиентного спуска для данной задачи:

Градиентный бустинг – обоснование

$$\widetilde{R_T} = \sum_{i=1}^{m} L(f_i^{T-1} + \alpha_T b_T(x_i), y_i) \to \min_{\alpha_T, b_T}$$

Применение градиентного спуска для данной задачи:

- $f_i^T = f_i^{T-1} - \eta g_i$, где $g_i = L'(f_i^{T-1}, y_i)$

Градиентный бустинг – обоснование

$$\widetilde{R_T} = \sum_{i=1}^{m} L(f_i^{T-1} + \alpha_T b_T(x_i), y_i) \to \min_{\alpha_T, b_T}$$

Применение градиентного спуска для данной задачи:

- $f_i^T = f_i^{T-1} - \eta g_i$, где $g_i = L'(f_i^{T-1}, y_i)$
- **Сравните**: итерация бустинга $f_i^T = f_i^{T-1} + \alpha_T b_T(x_i)$

Градиентный бустинг – обоснование

$$\widetilde{R_T} = \sum_{i=1}^{m} L(f_i^{T-1} + \alpha_T b_T(x_i), y_i) \to \min_{\alpha_T, b_T}$$

Применение градиентного спуска для данной задачи:

- $f_i^T = f_i^{T-1} - \eta g_i$, где $g_i = L'(f_i^{T-1}, y_i)$
- **Сравните**: итерация бустинга $f_i^T = f_i^{T-1} + \alpha_T b_T(x_i)$

Градиентный бустинг – обоснование

$$\widetilde{R_T} = \sum_{i=1}^{m} L(f_i^{T-1} + \alpha_T b_T(x_i), y_i) \to \min_{\alpha_T, b_T}$$

Применение градиентного спуска для данной задачи:

- $f_i^T = f_i^{T-1} - \eta g_i$, где $g_i = L'(f_i^{T-1}, y_i)$
- **Сравните**: итерация бустинга $f_i^T = f_i^{T-1} + \alpha_T b_T(x_i)$

Основная идея градиентного бустинга

Поиск нового базового алгоритма b_T для приближения антиградиента $(-L'(f_i^{T-1}, y_i))$, т.е. минимизация квадратичной ошибки: $b_T = \arg\min_b \sum_{i=1}^m \Big(b(x_i) - (-L'(f_i^{T-1}, y_i))\Big)^2$.

Алгоритм градиентного бустинга

Алгоритм

- Инициализация приближений: $f_i = 0, i = 1, \ldots, m,$

Алгоритм градиентного бустинга

Алгоритм

- Инициализация приближений: $f_i = 0, i = 1, \ldots, m$,

Для $t = 1, \ldots, T$

- Обучение нового базового алгоритма $b_t = \arg\min_b \sum_{i=1}^{m} (b(x_i) + L'(f_i, y_i))^2$,

Алгоритм градиентного бустинга

Алгоритм

- Инициализация приближений: $f_i = 0, i = 1, \ldots, m$,

Для $t = 1, \ldots, T$

- Обучение нового базового алгоритма $b_t = \arg\min_b \sum_{i=1}^{m} (b(x_i) + L'(f_i, y_i))^2$,
- Вычисление нового веса $\alpha_t = \arg\min_\alpha \sum_{i=1}^{m} L(f_i + \alpha b_t(x_i), y_i)$,

Алгоритм градиентного бустинга

Алгоритм

- Инициализация приближений: $f_i = 0, i = 1, \dots, m$,

Для $t = 1, \dots, T$

- Обучение нового базового алгоритма $b_t = \arg\min_b \sum_{i=1}^m (b(x_i) + L'(f_i, y_i))^2$,
- Вычисление нового веса $\alpha_t = \arg\min_\alpha \sum_{i=1}^m L(f_i + \alpha b_t(x_i), y_i)$,
- Обновление приближений $f_i = f_i + \alpha_t b_t(x_i), i = 1, \dots, m$.

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Алгоритм SGB

- Инициализация приближений: $f_i = 0, i = 1, \dots, m$,

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Алгоритм SGB

- Инициализация приближений: $f_i = 0, i = 1, \ldots, m,$

Для $t = 1, \ldots, T$

- Выбор случайного подмножества $I \subseteq \{1, \ldots, m\},$

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Алгоритм SGB

- Инициализация приближений: $f_i = 0, i = 1, \dots, m$,

Для $t = 1, \dots, T$

- Выбор случайного подмножества $I \subseteq \{1, \dots, m\}$,
- Обучение нового базового алгоритма $b_t = \arg\min_b \sum_{i \in I} (b(x_i) + L'(f_i, y_i))^2$,

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Алгоритм SGB

- Инициализация приближений: $f_i = 0, i = 1, \ldots, m$,

Для $t = 1, \ldots, T$

- Выбор случайного подмножества $I \subseteq \{1, \ldots, m\}$,
- Обучение нового базового алгоритма $b_t = \arg\min_b \sum_{i \in I} (b(x_i) + L'(f_i, y_i))^2$,
- Вычисление нового веса $\alpha_t = \arg\min_\alpha \sum_{i \in I} L(f_i + \alpha b_t(x_i), y_i)$,

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Алгоритм SGB

- Инициализация приближений: $f_i = 0, i = 1, \ldots, m$,

Для $t = 1, \ldots, T$

- Выбор случайного подмножества $I \subseteq \{1, \ldots, m\}$,
- Обучение нового базового алгоритма $b_t = \arg\min_b \sum_{i \in I} (b(x_i) + L'(f_i, y_i))^2$,
- Вычисление нового веса $\alpha_t = \arg\min_\alpha \sum_{i \in I} L(f_i + \alpha b_t(x_i), y_i)$,
- Обновление приближений $f_i = f_i + \alpha_t b_t(x_i), i \in I$.

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Алгоритм SGB

- Инициализация приближений: $f_i = 0, i = 1, \dots, m$,

Для $t = 1, \dots, T$

- Выбор случайного подмножества $I \subseteq \{1, \dots, m\}$,
- Обучение нового базового алгоритма $b_t = \arg\min_b \sum_{i \in I} \left(b(x_i) + L'(f_i, y_i)\right)^2$,
- Вычисление нового веса $\alpha_t = \arg\min_\alpha \sum_{i \in I} L(f_i + \alpha b_t(x_i), y_i)$,
- Обновление приближений $f_i = f_i + \alpha_t b_t(x_i), i \in I$.

Плюсы SGB

- Уменьшение времени обучения (меньше объектов на каждом шаге),

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Алгоритм SGB

- Инициализация приближений: $f_i = 0, i = 1, \dots, m$,

Для $t = 1, \dots, T$

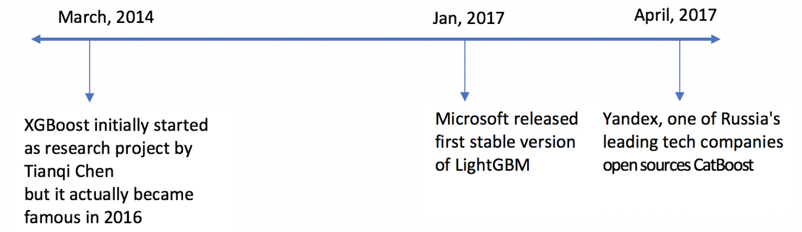
- Выбор случайного подмножества $I \subseteq \{1, \dots, m\}$,
- Обучение нового базового алгоритма $b_t = \arg\min_b \sum_{i \in I} (b(x_i) + L'(f_i, y_i))^2$,
- Вычисление нового веса $\alpha_t = \arg\min_\alpha \sum_{i \in I} L(f_i + \alpha b_t(x_i), y_i)$,
- Обновление приближений $f_i = f_i + \alpha_t b_t(x_i), i \in I$.

Плюсы SGB

- Уменьшение времени обучения (меньше объектов на каждом шаге),
- Ускорение сходимости (меньше шагов).

Алгоритмы XGBoost, LightGBM и CatBoost

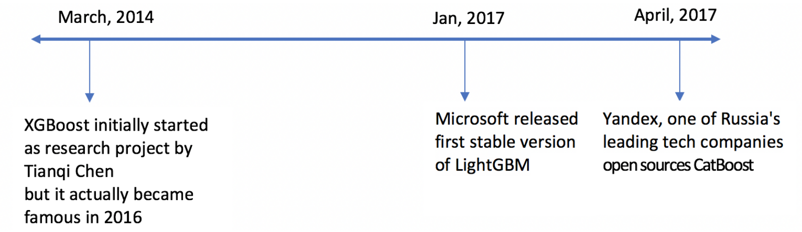
Наибольшую популярность на данный момент имеют реализации градиентного бустинга на решающих деревьях[2]:



[2] https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

Алгоритмы XGBoost, LightGBM и CatBoost

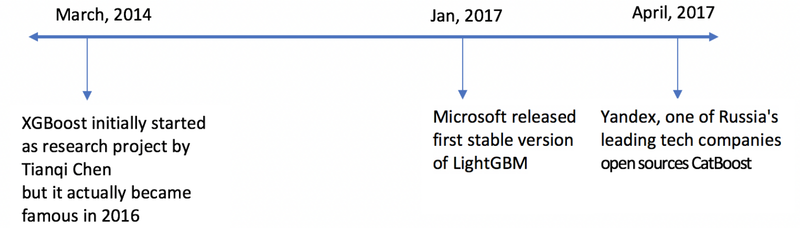
Наибольшую популярность на данный момент имеют реализации градиентного бустинга на решающих деревьях[2]:



[2] https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

Алгоритмы XGBoost, LightGBM и CatBoost

Наибольшую популярность на данный момент имеют реализации градиентного бустинга на решающих деревьях[2]:



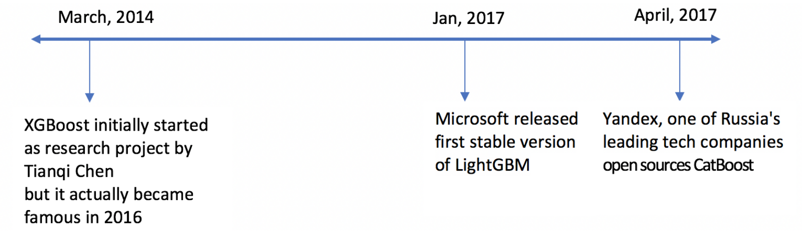
Эти реализации отличаются:

- Методом ветвления в узлах дерева при его обучении,

[2] https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

Алгоритмы XGBoost, LightGBM и CatBoost

Наибольшую популярность на данный момент имеют реализации градиентного бустинга на решающих деревьях[2]:



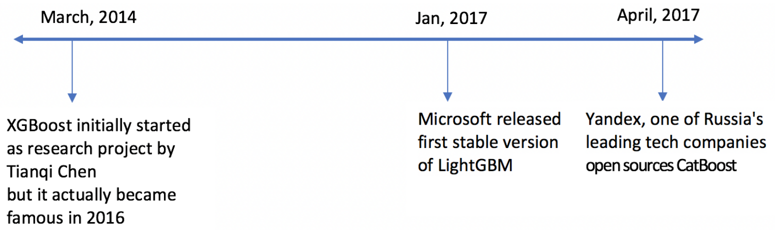
Эти реализации отличаются:

- Методом ветвления в узлах дерева при его обучении,
- Способом работы с категориальными признаками,

[2] https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

Алгоритмы XGBoost, LightGBM и CatBoost

Наибольшую популярность на данный момент имеют реализации градиентного бустинга на решающих деревьях[2]:



Эти реализации отличаются:

- Методом ветвления в узлах дерева при его обучении,
- Способом работы с категориальными признаками,
- Скоростью обучения / тестирования.

[2] https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

Сравнение алгоритмов композиции

- Бэггинг (благодаря процедуре бутстрэпа) может работать на небольших выборках,

Сравнение алгоритмов композиции

- Бэггинг (благодаря процедуре бутстрэпа) может работать на небольших выборках,
- Бустинг лучше работает на больших выборках (но и ошибка, скорее всего, будет меньше),

Сравнение алгоритмов композиции

- Бэггинг (благодаря процедуре бутстрэпа) может работать на небольших выборках,
- Бустинг лучше работает на больших выборках (но и ошибка, скорее всего, будет меньше),
- Стековое обобщение можно использовать как средство “выжимания” последних долей процента,

Сравнение алгоритмов композиции

- Бэггинг (благодаря процедуре бутстрэпа) может работать на небольших выборках,
- Бустинг лучше работает на больших выборках (но и ошибка, скорее всего, будет меньше),
- Стековое обобщение можно использовать как средство “выжимания” последних долей процента,
- Стековое обобщение можно (и нужно) использовать с алгоритмами разной природы,

Сравнение алгоритмов композиции

- Бэггинг (благодаря процедуре бутстрэпа) может работать на небольших выборках,
- Бустинг лучше работает на больших выборках (но и ошибка, скорее всего, будет меньше),
- Стековое обобщение можно использовать как средство "выжимания" последних долей процента,
- Стековое обобщение можно (и нужно) использовать с алгоритмами разной природы,
- Бэггинг лучше всего параллелится,

Сравнение алгоритмов композиции

- Бэггинг (благодаря процедуре бутстрэпа) может работать на небольших выборках,
- Бустинг лучше работает на больших выборках (но и ошибка, скорее всего, будет меньше),
- Стековое обобщение можно использовать как средство "выжимания" последних долей процента,
- Стековое обобщение можно (и нужно) использовать с алгоритмами разной природы,
- Бэггинг лучше всего параллелится,
- Бустинг позволяет фильтровать выбросы,

Сравнение алгоритмов композиции

- Бэггинг (благодаря процедуре бутстрэпа) может работать на небольших выборках,
- Бустинг лучше работает на больших выборках (но и ошибка, скорее всего, будет меньше),
- Стековое обобщение можно использовать как средство "выжимания" последних долей процента,
- Стековое обобщение можно (и нужно) использовать с алгоритмами разной природы,
- Бэггинг лучше всего параллелится,
- Бустинг позволяет фильтровать выбросы,
- Метод случайных подпространств (бутстрэп на признаках) необходим, когда у нас признаков очень много (или много шумовых).

Источники

На основе материалов сайта http://www.machinelearning.ru.