

# Введение в искусственный интеллект.

## Машинное обучение

Тема: Ансамблирование моделей. Три метода на букву “Б”

Бабин Д.Н., Иванов И.Е.

кафедра Математической Теории Интеллектуальных Систем



## ① Стековое обобщение

- ① Стековое обобщение
  - Блендинг

## ① Стековое обобщение

- Блендинг
- Стекинг

## ① Стековое обобщение

- Блендинг
- Стекинг

## ② Бутстрэп и пэстинг



## ① Стековое обобщение

- Блендинг
- Стекинг

## ② Бутстрэп и пэстинг

## ③ Бэггинг

- ① Стековое обобщение
  - Блендинг
  - Стекинг
- ② Бутстрэп и пэстинг
- ③ Бэггинг
- ④ Бустинг с дискретными базовыми алгоритмами

## Ансамбль методов

Это способ использования нескольких обучающих алгоритмов с целью получения лучшей эффективности предсказания (классификации или регрессии), чем могли бы получить от каждого обучающего алгоритма по отдельности



## Ансамбль методов

Это способ использования нескольких обучающих алгоритмов с целью получения лучшей эффективности предсказания (классификации или регрессии), чем могли бы получить от каждого обучающего алгоритма по отдельности

**Замечание.** Ансамбль методов не бесконечен: состоит из конкретного конечного множества альтернативных моделей.

## Ансамбль методов

Это способ использования нескольких обучающих алгоритмов с целью получения лучшей эффективности предсказания (классификации или регрессии), чем могли бы получить от каждого обучающего алгоритма по отдельности

**Замечание.** Ансамбль методов не бесконечен: состоит из конкретного конечного множества альтернативных моделей.

**Основные представители:**

- 1 Стековое обобщение (stacked generalization)
- 2 Бэггинг (bagging)
- 3 Бустинг (boosting)



# Зачем нужно ансамблирование

Рассмотрим часто применяемый на практике метод простого голосования:

$a(x) = \frac{1}{T} \sum_{t=1}^T b_t(x)$ , где  $T$  – число базовых алгоритмов регрессии,  $b_t(x)$  – сами базовые алгоритмы (для классификации с sign все рассматривается аналогично).



# Зачем нужно ансамблирование

Рассмотрим часто применяемый на практике метод простого голосования:

$a(x) = \frac{1}{T} \sum_{t=1}^T b_t(x)$ , где  $T$  – число базовых алгоритмов регрессии,  $b_t(x)$  – сами базовые алгоритмы (для классификации с sign все рассматривается аналогично).

Если  $y^*(x)$  – истинная функция ответа, то среднеквадратичная ошибка для базового алгоритма:  $E(b_t(x) - y^*(x))^2 = E\varepsilon_t^2(x)$ .



# Зачем нужно ансамблирование

Рассмотрим часто применяемый на практике метод простого голосования:

$a(x) = \frac{1}{T} \sum_{t=1}^T b_t(x)$ , где  $T$  – число базовых алгоритмов регрессии,  $b_t(x)$  – сами базовые алгоритмы (для классификации с sign все рассматривается аналогично).

Если  $y^*(x)$  – истинная функция ответа, то среднеквадратичная ошибка для базового алгоритма:  $E(b_t(x) - y^*(x))^2 = E\varepsilon_t^2(x)$ .

Средняя среднеквадратичная ошибка по всем базовым алгоритмам:

$$E_{avg} = \frac{1}{T} E \sum_{t=1}^T \varepsilon_t^2(x).$$



# Зачем нужно ансамблирование

Рассмотрим часто применяемый на практике метод простого голосования:

$a(x) = \frac{1}{T} \sum_{t=1}^T b_t(x)$ , где  $T$  – число базовых алгоритмов регрессии,  $b_t(x)$  – сами базовые алгоритмы (для классификации с sign все рассматривается аналогично).

Если  $y^*(x)$  – истинная функция ответа, то среднеквадратичная ошибка для базового алгоритма:  $E(b_t(x) - y^*(x))^2 = E\varepsilon_t^2(x)$ .

Средняя среднеквадратичная ошибка по всем базовым алгоритмам:

$$E_{avg} = \frac{1}{T} E \sum_{t=1}^T \varepsilon_t^2(x).$$

Предположим, что ошибки несмещены и некоррелированы:  $E\varepsilon_t(x) = 0, E\varepsilon_t\varepsilon_u = 0, t \neq u$ .

# Зачем нужно ансамблирование

Рассмотрим часто применяемый на практике метод простого голосования:

$a(x) = \frac{1}{T} \sum_{t=1}^T b_t(x)$ , где  $T$  – число базовых алгоритмов регрессии,  $b_t(x)$  – сами базовые алгоритмы (для классификации с sign все рассматривается аналогично).

Если  $y^*(x)$  – истинная функция ответа, то среднеквадратичная ошибка для базового алгоритма:  $E(b_t(x) - y^*(x))^2 = E\varepsilon_t^2(x)$ .

Средняя среднеквадратичная ошибка по всем базовым алгоритмам:

$$E_{avg} = \frac{1}{T} E \sum_{t=1}^T \varepsilon_t^2(x).$$

Предположим, что ошибки несмещены и некоррелированы:  $E\varepsilon_t(x) = 0, E\varepsilon_t\varepsilon_u = 0, t \neq u$ .

Найдем среднеквадратичную ошибку для  $a(x)$ :

$$E_{ens} = E(a(x) - y^*(x))^2 = E\left(\frac{1}{T} \sum_{t=1}^T b_t(x) - y^*(x)\right)^2 = E\left(\frac{1}{T} \sum_{t=1}^T \varepsilon_t\right)^2 = \frac{1}{T^2} E\left(\sum_{t=1}^T \varepsilon_t^2 + \sum_{t \neq u} \varepsilon_t \varepsilon_u\right) = \frac{1}{T^2} E \sum_{t=1}^T \varepsilon_t^2(x) = \frac{1}{T} E_{avg}.$$



# Зачем нужно ансамблирование

Рассмотрим часто применяемый на практике метод простого голосования:

$a(x) = \frac{1}{T} \sum_{t=1}^T b_t(x)$ , где  $T$  – число базовых алгоритмов регрессии,  $b_t(x)$  – сами базовые алгоритмы (для классификации с sign все рассматривается аналогично).

Если  $y^*(x)$  – истинная функция ответа, то среднеквадратичная ошибка для базового алгоритма:  $E(b_t(x) - y^*(x))^2 = E\varepsilon_t^2(x)$ .

Средняя среднеквадратичная ошибка по всем базовым алгоритмам:

$$E_{avg} = \frac{1}{T} E \sum_{t=1}^T \varepsilon_t^2(x).$$

Предположим, что ошибки несмещены и некоррелированы:  $E\varepsilon_t(x) = 0, E\varepsilon_t\varepsilon_u = 0, t \neq u$ .

Найдем среднеквадратичную ошибку для  $a(x)$ :

$$E_{ens} = E(a(x) - y^*(x))^2 = E\left(\frac{1}{T} \sum_{t=1}^T b_t(x) - y^*(x)\right)^2 = E\left(\frac{1}{T} \sum_{t=1}^T \varepsilon_t\right)^2 = \frac{1}{T^2} E\left(\sum_{t=1}^T \varepsilon_t^2 + \sum_{t \neq u} \varepsilon_t \varepsilon_u\right) = \frac{1}{T^2} E \sum_{t=1}^T \varepsilon_t^2(x) = \frac{1}{T} E_{avg}.$$

Таким образом, простое голосование позволило уменьшить средний квадрат ошибки в  $T$  раз!





# Стековое обобщение<sup>1</sup>

Предположим, что мы можем обучить  $T$  базовых алгоритмов.  
После этого мы обучаем комбинирующий алгоритм верхнего уровня (**мета-алгоритм**),  
*входом* для которого являются *выходы* базовых.

---

<sup>1</sup>Wolpert D. (1992) "Stacked Generalization"



# Стековое обобщение<sup>1</sup>

Предположим, что мы можем обучить  $T$  базовых алгоритмов.

После этого мы обучаем комбинирующий алгоритм верхнего уровня (**мета-алгоритм**), *входом* для которого являются *выходы* базовых.

## Схема стекового обобщения

- 1 Обучаем по отдельности каждый базовый алгоритм  $b_t(x)$ ,  $t = 1, \dots, T$
- 2 Фиксируем алгоритмы  $b_t(x)$
- 3 Обучаем комбинирующий алгоритм верхнего уровня  $a(x) = a(b_1(x), \dots, b_T(x))$

---

<sup>1</sup>Wolpert D. (1992) "Stacked Generalization"

# Стековое обобщение<sup>1</sup>

Предположим, что мы можем обучить  $T$  базовых алгоритмов.

После этого мы обучаем комбинирующий алгоритм верхнего уровня (**мета-алгоритм**), *входом* для которого являются *выходы* базовых.

## Схема стекового обобщения

- 1 Обучаем по отдельности каждый базовый алгоритм  $b_t(x)$ ,  $t = 1, \dots, T$
- 2 Фиксируем алгоритмы  $b_t(x)$
- 3 Обучаем комбинирующий алгоритм верхнего уровня  $a(x) = a(b_1(x), \dots, b_T(x))$

**Замечание 1.** Простое (или взвешенное) голосование является частным случаем стекового обобщения с необучаемым комбинирующим алгоритмом верхнего уровня.

---

<sup>1</sup>Wolpert D. (1992) "Stacked Generalization"

# Стековое обобщение<sup>1</sup>

Предположим, что мы можем обучить  $T$  базовых алгоритмов.

После этого мы обучаем комбинирующий алгоритм верхнего уровня (**мета-алгоритм**), *входом* для которого являются *выходы* базовых.

## Схема стекового обобщения

- 1 Обучаем по отдельности каждый базовый алгоритм  $b_t(x)$ ,  $t = 1, \dots, T$
- 2 Фиксируем алгоритмы  $b_t(x)$
- 3 Обучаем комбинирующий алгоритм верхнего уровня  $a(x) = a(b_1(x), \dots, b_T(x))$

**Замечание 1.** Простое (или взвешенное) голосование является частным случаем стекового обобщения с необучаемым комбинирующим алгоритмом верхнего уровня.

**Замечание 2.** Стековое обобщение - один из главных методов достижения успехов на Kaggle :)

---

<sup>1</sup>Wolpert D. (1992) "Stacked Generalization"

# Блендинг (Blending)

Ранее мы рассмотрели общую схему, теперь рассмотрим конкретные варианты реализации на практике.

# Блендинг (Blending)

Ранее мы рассмотрели общую схему, теперь рассмотрим конкретные варианты реализации на практике.

- 1 Разбиваем обучающую выборку на две части

# Блендинг (Blending)

Ранее мы рассмотрели общую схему, теперь рассмотрим конкретные варианты реализации на практике.

- 1 Разбиваем обучающую выборку на две части
- 2 На одной части обучаем базовые алгоритмы

# Блендинг (Blending)

Ранее мы рассмотрели общую схему, теперь рассмотрим конкретные варианты реализации на практике.

- 1 Разбиваем обучающую выборку на две части
- 2 На одной части обучаем базовые алгоритмы
- 3 На второй получаем ответы базовых алгоритмов и обучаем мета-алгоритм



# Блендинг (Blending)

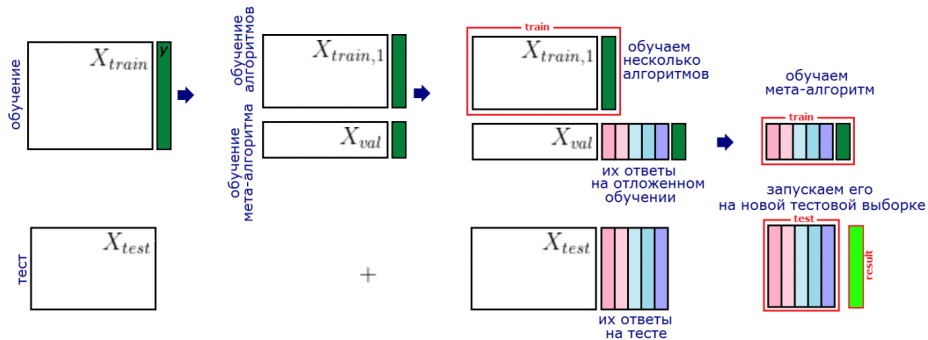
Ранее мы рассмотрели общую схему, теперь рассмотрим конкретные варианты реализации на практике.

- 1 Разбиваем обучающую выборку на две части
- 2 На одной части обучаем базовые алгоритмы
- 3 На второй получаем ответы базовых алгоритмов и обучаем мета-алгоритм
- 4 На тесте сначала получаем выходы базовых алгоритмов, к которым затем применяем мета-алгоритм



# Блендинг – визуализация

Рассмотрим схему<sup>2</sup> блендинга:



<sup>2</sup><https://dyakonov.org>

Проблема классического блендинга: ни базовые алгоритмы, ни мета-алгоритм не видят всей обучающей выборки. Поэтому можно немного усовершенствовать подход (подход “вширь”).

- 1 Разбиваем обучающую выборку на две части



Проблема классического блендинга: ни базовые алгоритмы, ни мета-алгоритм не видят всей обучающей выборки. Поэтому можно немного усовершенствовать подход (подход “вширь”).

- 1 Разбиваем обучающую выборку на две части
- 2 На одной части обучаем базовые алгоритмы

Проблема классического блендинга: ни базовые алгоритмы, ни мета-алгоритм не видят всей обучающей выборки. Поэтому можно немного усовершенствовать подход (подход “вширь”).

- 1 Разбиваем обучающую выборку на две части
- 2 На одной части обучаем базовые алгоритмы
- 3 На второй части получаем их ответы (**мета-признаки**)



Проблема классического блендинга: ни базовые алгоритмы, ни мета-алгоритм не видят всей обучающей выборки. Поэтому можно немного усовершенствовать подход (подход “вширь”).

- 1 Разбиваем обучающую выборку на две части
- 2 На одной части обучаем базовые алгоритмы
- 3 На второй части получаем их ответы (**мета-признаки**)
- 4 Обучаем мета-алгоритм

Проблема классического блендинга: ни базовые алгоритмы, ни мета-алгоритм не видят всей обучающей выборки. Поэтому можно немного усовершенствовать подход (подход “вширь”).

- 1 Разбиваем обучающую выборку на две части
- 2 На одной части обучаем базовые алгоритмы
- 3 На второй части получаем их ответы (**мета-признаки**)
- 4 Обучаем мета-алгоритм
- 5 Повторяем пп. 2-4  $M \geq 2$  раз для других разбиений обучающей выборки



Проблема классического блендинга: ни базовые алгоритмы, ни мета-алгоритм не видят всей обучающей выборки. Поэтому можно немного усовершенствовать подход (подход “вширь”).

- 1 Разбиваем обучающую выборку на две части
- 2 На одной части обучаем базовые алгоритмы
- 3 На второй части получаем их ответы (**мета-признаки**)
- 4 Обучаем мета-алгоритм
- 5 Повторяем пп. 2-4  $M \geq 2$  раз для других разбиений обучающей выборки
- 6 На тесте сначала получаем  $M$  наборов выходов базовых алгоритмов, обученных на разных разбиениях, затем для каждого набора запускаем соответствующий мета-алгоритм





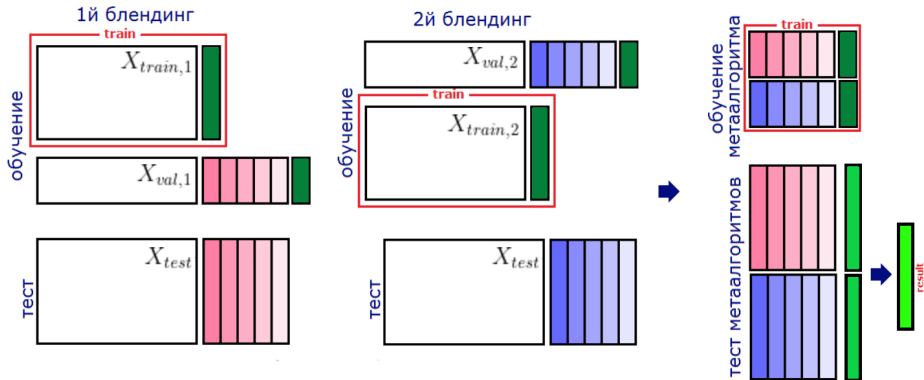
Проблема классического блендинга: ни базовые алгоритмы, ни мета-алгоритм не видят всей обучающей выборки. Поэтому можно немного усовершенствовать подход (подход “вширь”).

- 1 Разбиваем обучающую выборку на две части
- 2 На одной части обучаем базовые алгоритмы
- 3 На второй части получаем их ответы (**мета-признаки**)
- 4 Обучаем мета-алгоритм
- 5 Повторяем пп. 2-4  $M \geq 2$  раз для других разбиений обучающей выборки
- 6 На тесте сначала получаем  $M$  наборов выходов базовых алгоритмов, обученных на разных разбиениях, затем для каждого набора запускаем соответствующий мета-алгоритм
- 7 После чего усредняем  $M$  ответов мета-алгоритмов



# Блендинг – визуализация усовершенствования

Рассмотрим схему усовершенствованного <sup>3</sup> блендинга:



<sup>3</sup><https://dyakonov.org>

# Стекинг (Stacking)

Попытка решения той же проблемы: алгоритмы не видят всей обучающей выборки.

---

<sup>4</sup>Van der Laan, M. J., Polley, E. C., and Hubbard, A. E. (2007). "Super learner"



# Стекинг (Stacking)

Попытка решения той же проблемы: алгоритмы не видят всей обучающей выборки.

Опишем алгоритм **Super Learner**<sup>4</sup>:

- 1 Разбиваем обучающую выборку на  $k$  частей

---

<sup>4</sup>Van der Laan, M. J., Polley, E. C., and Hubbard, A. E. (2007). "Super learner"


# Стекинг (Stacking)

Попытка решения той же проблемы: алгоритмы не видят всей обучающей выборки.

Опишем алгоритм **Super Learner**<sup>4</sup>:

- 1 Разбиваем обучающую выборку на  $k$  частей
- 2 Обучаем базовые алгоритмы с помощью кросс-валидации

---

<sup>4</sup>Van der Laan, M. J., Polley, E. C., and Hubbard, A. E. (2007). "Super learner" 

Попытка решения той же проблемы: алгоритмы не видят всей обучающей выборки.

Опишем алгоритм **Super Learner**<sup>4</sup>:

- ❶ Разбиваем обучающую выборку на  $k$  частей
- ❷ Обучаем базовые алгоритмы с помощью кросс-валидации
  - Обучаем каждый алгоритм на  $k - 1$  частях и тестируем на hold-out части,
  - Повторяем так для каждой из  $k$  частей,
  - Обучаем каждый базовый алгоритм на всей обучающей выборке,

---

<sup>4</sup>Van der Laan, M. J., Polley, E. C., and Hubbard, A. E. (2007). “Super learner”


# Стекинг (Stacking)

Попытка решения той же проблемы: алгоритмы не видят всей обучающей выборки.

Опишем алгоритм **Super Learner**<sup>4</sup>:

- ❶ Разбиваем обучающую выборку на  $k$  частей
- ❷ Обучаем базовые алгоритмы с помощью кросс-валидации
  - Обучаем каждый алгоритм на  $k - 1$  частях и тестируем на hold-out части,
  - Повторяем так для каждой из  $k$  частей,
  - Обучаем каждый базовый алгоритм на всей обучающей выборке,
- ❸ Обучаем мета-алгоритм на hold-out выходах (мета-признаках) базовых алгоритмов по всей обучающей выборке

---

<sup>4</sup>Van der Laan, M. J., Polley, E. C., and Hubbard, A. E. (2007). "Super learner" 

# Стекинг (Stacking)

Попытка решения той же проблемы: алгоритмы не видят всей обучающей выборки.

Опишем алгоритм **Super Learner**<sup>4</sup>:

- ❶ Разбиваем обучающую выборку на  $k$  частей
- ❷ Обучаем базовые алгоритмы с помощью кросс-валидации
  - Обучаем каждый алгоритм на  $k - 1$  частях и тестируем на hold-out части,
  - Повторяем так для каждой из  $k$  частей,
  - Обучаем каждый базовый алгоритм на всей обучающей выборке,
- ❸ Обучаем мета-алгоритм на hold-out выходах (мета-признаках) базовых алгоритмов по всей обучающей выборке
- ❹ На тесте сначала получаем выходы базовых алгоритмов, к которым применяем мета-алгоритм

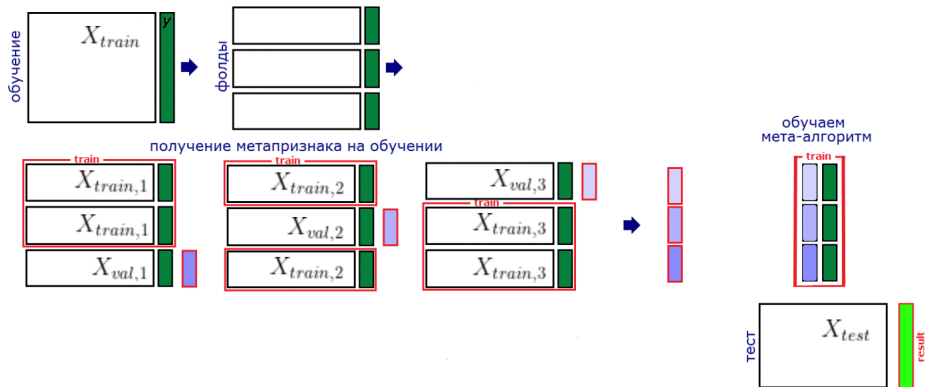
---

<sup>4</sup>Van der Laan, M. J., Polley, E. C., and Hubbard, A. E. (2007). “Super learner”



# Стекинг – визуализация

Рассмотрим схему<sup>5</sup> стекинга:



<sup>5</sup><https://dyakonov.org>

# Замечания о стековых обобщениях

- Блендинг очень прост в реализации

# Замечания о стековых обобщениях

- Блендинг очень прост в реализации
- Усовершенствованный блендинг зачастую не дает прироста по сравнению с обычным усреднением ответов базовых алгоритмов, обученных на всей обучающей выборке



# Замечания о стековых обобщениях

- Блендинг очень прост в реализации
- Усовершенствованный блендинг зачастую не дает прироста по сравнению с обычным усреднением ответов базовых алгоритмов, обученных на всей обучающей выборке
- Стекинг решает проблему обучения базовых алгоритмов на всем обучающем множестве, однако не всегда от этого есть ощутимый прирост



# Замечания о стековых обобщениях

- Блендинг очень прост в реализации
- Усовершенствованный блендинг зачастую не дает прироста по сравнению с обычным усреднением ответов базовых алгоритмов, обученных на всей обучающей выборке
- Стекинг решает проблему обучения базовых алгоритмов на всем обучающем множестве, однако не всегда от этого есть ощутимый прирост
- Стековое обобщение подходит для использования алгоритмов разной природы



# Замечания о стековых обобщениях

- Блендинг очень прост в реализации
- Усовершенствованный блендинг зачастую не дает прироста по сравнению с обычным усреднением ответов базовых алгоритмов, обученных на всей обучающей выборке
- Стекинг решает проблему обучения базовых алгоритмов на всем обучающем множестве, однако не всегда от этого есть ощутимый прирост
- Стековое обобщение подходит для использования алгоритмов разной природы
- В качестве мета-алгоритмов проще всего использовать регрессоры



# Замечания о стековых обобщениях

- Блендинг очень прост в реализации
- Усовершенствованный блендинг зачастую не дает прироста по сравнению с обычным усреднением ответов базовых алгоритмов, обученных на всей обучающей выборке
- Стекинг решает проблему обучения базовых алгоритмов на всем обучающем множестве, однако не всегда от этого есть ощутимый прирост
- Стековое обобщение подходит для использования алгоритмов разной природы
- В качестве мета-алгоритмов проще всего использовать регрессоры
- Выходы базовых алгоритмов часто коррелируют, поэтому лучше использовать недообученные версии этих алгоритмов



# Замечания о стековых обобщениях

- Блендинг очень прост в реализации
- Усовершенствованный блендинг зачастую не дает прироста по сравнению с обычным усреднением ответов базовых алгоритмов, обученных на всей обучающей выборке
- Стекинг решает проблему обучения базовых алгоритмов на всем обучающем множестве, однако не всегда от этого есть ощутимый прирост
- Стековое обобщение подходит для использования алгоритмов разной природы
- В качестве мета-алгоритмов проще всего использовать регрессоры
- Выходы базовых алгоритмов часто коррелируют, поэтому лучше использовать недообученные версии этих алгоритмов
- Можно для обучения мета-алгоритма использовать не только выходы базовых алгоритмов, но и исходные данные; однако так лучше не делать







# Бутстрэп (Bootstrap)





Английская поговорка: *"To pull oneself over a fence by one's bootstraps"*.

Русский аналог: *Мюнхгаузен, вытаскивающий себя за волосы из болота.*

## Определение

**Бутстрэп** - это методика тестирования на основе случайного семплирования (либо само это семплирование) из выборки с **возвращением**.

---

<sup>6</sup>Efron, B. (1979). "Bootstrap methods: Another look at the jackknife"

## Определение

**Бутстрэп** - это методика тестирования на основе случайного семплирования (либо само это семплирование) из выборки с **возвращением**.

- Бутстрэп<sup>6</sup> позволяет оценивать параметры алгоритмов (такие как смещение, разброс, доверительный интервал и т.п.) на основе семплированных выборок

---

<sup>6</sup>Efron, B. (1979). "Bootstrap methods: Another look at the jackknife"

## Определение

**Бутстрэп** - это методика тестирования на основе случайного семплирования (либо само это семплирование) из выборки с **возвращением**.

- Бутстрэп<sup>6</sup> позволяет оценивать параметры алгоритмов (такие как смещение, разброс, доверительный интервал и т.п.) на основе семплированных выборок
- Многократная генерация выборок происходит методом Монте-Карло на базе имеющейся выборки (т.о., из одной выборки генерируем любое число выборок)

---

<sup>6</sup>Efron, B. (1979). "Bootstrap methods: Another look at the jackknife"

## Теорема

При использовании бутстрэпа для генерации выборки той же мощности  $N$ , что и исходная выборка, доля объектов, не попавших в сгенерированную выборку, стремится к  $e^{-1}$  при  $N \rightarrow \infty$ .

## Теорема

При использовании бутстрэпа для генерации выборки той же мощности  $N$ , что и исходная выборка, доля объектов, не попавших в сгенерированную выборку, стремится к  $e^{-1}$  при  $N \rightarrow \infty$ .

**Доказательство.** На каждом шаге все объекты попадают в новую выборку с возвращением равновероятно, т.е. отдельный объект – с вероятностью  $\frac{1}{N}$ . Вероятность того, что объект не попадёт в новую выборку после  $N$  шагов:  $(1 - \frac{1}{N})^N$ . Вспоминаем второй замечательный предел:  
$$\lim_{N \rightarrow \infty} (1 - \frac{1}{N})^N = \lim_{N \rightarrow \infty} ((1 - \frac{1}{N})^{-N})^{-1} = e^{-1}. \text{ Ч.т.д.}$$





## Теорема

При использовании бутстрэпа для генерации выборки той же мощности  $N$ , что и исходная выборка, доля объектов, не попавших в сгенерированную выборку, стремится к  $e^{-1}$  при  $N \rightarrow \infty$ .

**Доказательство.** На каждом шаге все объекты попадают в новую выборку с возвращением равновероятно, т.е. отдельный объект – с вероятностью  $\frac{1}{N}$ . Вероятность того, что объект не попадёт в новую выборку после  $N$  шагов:  $(1 - \frac{1}{N})^N$ .

Вспоминаем второй замечательный предел:

$$\lim_{N \rightarrow \infty} (1 - \frac{1}{N})^N = \lim_{N \rightarrow \infty} ((1 - \frac{1}{N})^{-N})^{-1} = e^{-1}. \text{ Ч.т.д.}$$

**Замечание.** Т.о. можно тестировать алгоритм на оставшихся  $e^{-1} \approx 37\%$  данных.



- **Вопрос.** Что будет, если мы **запретим** при процедуре бутстрэп-семплирования **возвращать** объекты назад в выборку для их возможного выбора еще раз?

---

<sup>7</sup>Breiman, L. (1999). "Pasting small votes for classification in large databases and on-line"

- **Вопрос.** Что будет, если мы **запретим** при процедуре бутстрэп-семплирования **возвращать** объекты назад в выборку для их возможного выбора еще раз?
- **Ответ.** Получим т.н. процедуру **пэстинга**<sup>7</sup> (Pasting).

---

<sup>7</sup>Breiman, L. (1999). "Pasting small votes for classification in large databases and on-line"

- **Вопрос.** Что будет, если мы **запретим** при процедуре бутстрэп-семплирования **возвращать** объекты назад в выборку для их возможного выбора еще раз?
- **Ответ.** Получим т.н. процедуру **пэстинга**<sup>7</sup> (Pasting).

Приведем пример для сравнения различных процедур семплирования.

---

<sup>7</sup>Breiman, L. (1999). "Pasting small votes for classification in large databases and on-line"

- **Вопрос.** Что будет, если мы **запретим** при процедуре бутстрэп-семплирования **возвращать** объекты назад в выборку для их возможного выбора еще раз?
- **Ответ.** Получим т.н. процедуру **пэстинга**<sup>7</sup> (Pasting).

Приведем пример для сравнения различных процедур семплирования.

- Дано: обучающая выборка, состоящая из  $N = 5$  объектов  $X = \{1, 2, 3, 4, 5\}$ ,

---

<sup>7</sup>Breiman, L. (1999). "Pasting small votes for classification in large databases and on-line"

- **Вопрос.** Что будет, если мы **запретим** при процедуре бутстрэп-семплирования **возвращать** объекты назад в выборку для их возможного выбора еще раз?
- **Ответ.** Получим т.н. процедуру **пэстинга**<sup>7</sup> (Pasting).

Приведем пример для сравнения различных процедур семплирования.

- Дано: обучающая выборка, состоящая из  $N = 5$  объектов  $X = \{1, 2, 3, 4, 5\}$ ,
- Проводим процедуру семплирования  $n = 3$  объектов,

---

<sup>7</sup>Breiman, L. (1999). "Pasting small votes for classification in large databases and on-line"

- **Вопрос.** Что будет, если мы **запретим** при процедуре бутстрэп-семплирования **возвращать** объекты назад в выборку для их возможного выбора еще раз?
- **Ответ.** Получим т.н. процедуру **пэстинга**<sup>7</sup> (Pasting).

Приведем пример для сравнения различных процедур семплирования.

- Дано: обучающая выборка, состоящая из  $N = 5$  объектов  $X = \{1, 2, 3, 4, 5\}$ ,
- Проводим процедуру семплирования  $n = 3$  объектов,
- Бутстрэп:  $\{2, 2, 5\}, \{1, 2, 5\}, \{2, 4, 4\}, \dots$

---

<sup>7</sup>Breiman, L. (1999). "Pasting small votes for classification in large databases and on-line"

- **Вопрос.** Что будет, если мы **запретим** при процедуре бутстрэп-семплирования **возвращать** объекты назад в выборку для их возможного выбора еще раз?
- **Ответ.** Получим т.н. процедуру **пэстинга**<sup>7</sup> (Pasting).

Приведем пример для сравнения различных процедур семплирования.

- Дано: обучающая выборка, состоящая из  $N = 5$  объектов  $X = \{1, 2, 3, 4, 5\}$ ,
- Проводим процедуру семплирования  $n = 3$  объектов,
- Бутстрэп:  $\{2, 2, 5\}, \{1, 2, 5\}, \{2, 4, 4\}, \dots$
- Пэстинг:  $\{2, 3, 5\}, \{1, 3, 5\}, \{2, 3, 4\}, \dots$

---

<sup>7</sup>Breiman, L. (1999). "Pasting small votes for classification in large databases and on-line"



- **Вопрос.** Что будет, если мы **запретим** при процедуре бутстрэп-семплирования **возвращать** объекты назад в выборку для их возможного выбора еще раз?
- **Ответ.** Получим т.н. процедуру **пэстинга**<sup>7</sup> (Pasting).

Приведем пример для сравнения различных процедур семплирования.

- Дано: обучающая выборка, состоящая из  $N = 5$  объектов  $X = \{1, 2, 3, 4, 5\}$ ,
- Проводим процедуру семплирования  $n = 3$  объектов,
- Бутстрэп:  $\{2, 2, 5\}, \{1, 2, 5\}, \{2, 4, 4\}, \dots$
- Пэстинг:  $\{2, 3, 5\}, \{1, 3, 5\}, \{2, 3, 4\}, \dots$

**Замечание.** Пэстинг имеет очевидную реализацию: сначала случайно перемешиваем объекты, затем берем из них  $n$  первых.

---

<sup>7</sup>Breiman, L. (1999). "Pasting small votes for classification in large databases and on-line"

- При бутстрэпе мы можем получить практически неограниченное количество подвыборок (за счет разрешения семплирования объектов с повторением),

# Сравнение пэстинга и бутстрэпа

- При бутстрэпе мы можем получить практически неограниченное количество подвыборок (за счет разрешения семплирования объектов с повторением),
- При пэстинге мы можем получить гораздо меньше подвыборок (поскольку все элементы должны быть различны),

# Сравнение пэстинга и бутстрэпа

- При бутстрэпе мы можем получить практически неограниченное количество подвыборок (за счет разрешения семплирования объектов с повторением),
- При пэстинге мы можем получить гораздо меньше подвыборок (поскольку все элементы должны быть различны),
- Пэстинг при размере выборки, совпадающей по порядку с размером исходной обучающей выборки ( $n \sim N$ ), практически не имеет никакого смысла,



# Сравнение пэстинга и бутстрэпа

- При бутстрэпе мы можем получить практически неограниченное количество подвыборок (за счет разрешения семплирования объектов с повторением),
- При пэстинге мы можем получить гораздо меньше подвыборок (поскольку все элементы должны быть различны),
- Пэстинг при размере выборки, совпадающей по порядку с размером исходной обучающей выборки ( $n \sim N$ ), практически не имеет никакого смысла,
- Пэстинг имеет смысл применять, когда нам важно, чтобы объекты не повторялись.





Вспомним разложение ошибки на разброс и смещение:  $\sigma^2 + \text{variance}(a) + \text{bias}^2(f, a)$ .

---

<sup>8</sup>Breiman L. (1994). "Bagging Predictors".

Вспомним разложение ошибки на разброс и смещение:  $\sigma^2 + \text{variance}(a) + \text{bias}^2(f, a)$ .  
Также мы видели, что простое усреднение  $T$  алгоритмов позволяет теоретически уменьшить разброс в  $T$  раз, при этом не влияя на смещение.

---

<sup>8</sup>Breiman L. (1994). "Bagging Predictors".





Вспомним разложение ошибки на разброс и смещение:  $\sigma^2 + \text{variance}(a) + \text{bias}^2(f, a)$ .

Также мы видели, что простое усреднение  $T$  алгоритмов позволяет теоретически уменьшить разброс в  $T$  раз, при этом не влияя на смещение.

Это и есть главная идея бэггинга<sup>8</sup>:

---

<sup>8</sup>Breiman L. (1994). "Bagging Predictors".

Вспомним разложение ошибки на разброс и смещение:  $\sigma^2 + \text{variance}(a) + \text{bias}^2(f, a)$ .

Также мы видели, что простое усреднение  $T$  алгоритмов позволяет теоретически уменьшить разброс в  $T$  раз, при этом не влияя на смещение.

Это и есть главная идея бэггинга<sup>8</sup>:

- уменьшить разброс алгоритма,

---

<sup>8</sup>Breiman L. (1994). "Bagging Predictors".

Вспомним разложение ошибки на разброс и смещение:  $\sigma^2 + \text{variance}(a) + \text{bias}^2(f, a)$ .

Также мы видели, что простое усреднение  $T$  алгоритмов позволяет теоретически уменьшить разброс в  $T$  раз, при этом не влияя на смещение.

Это и есть главная идея бэггинга<sup>8</sup>:

- уменьшить разброс алгоритма,
- как следствие, бороться с переобучением.

---

<sup>8</sup>Breiman L. (1994). "Bagging Predictors".

Вспомним разложение ошибки на разброс и смещение:  $\sigma^2 + \text{variance}(a) + \text{bias}^2(f, a)$ .

Также мы видели, что простое усреднение  $T$  алгоритмов позволяет теоретически уменьшить разброс в  $T$  раз, при этом не влияя на смещение.

Это и есть главная идея бэггинга<sup>8</sup>:

- уменьшить разброс алгоритма,
- как следствие, бороться с переобучением.

## Определение

**Бэггинг (Bootstrap AGGregatING)** - это метод ансамблирования, основанный на:

- 1 бутстрэп-семплировании для каждого обучения базового алгоритма,
- 2 последующем усреднении ответов уже обученных базовых алгоритмов методом простого голосования.

<sup>8</sup>Breiman L. (1994). "Bagging Predictors".

# Алгоритм бэггинга

- Дано: обучающая выборка  $X^m$  мощности  $m$ .
- Цель: обучить ансамбль из  $T$  классификаторов  $b_t(x)$ ,  $t = 1, \dots, T$ .



# Алгоритм бэггинга

- Дано: обучающая выборка  $X^m$  мощности  $m$ .
- Цель: обучить ансамбль из  $T$  классификаторов  $b_t(x)$ ,  $t = 1, \dots, T$ .

## Алгоритм

- 1 Формируем  $T$  выборок  $X_t^m$ ,  $t = 1, \dots, T$  мощности  $m$  с помощью бутстрэп-семплирования,

# Алгоритм бэггинга

- **Дано:** обучающая выборка  $X^m$  мощности  $m$ .
- **Цель:** обучить ансамбль из  $T$  классификаторов  $b_t(x)$ ,  $t = 1, \dots, T$ .

## Алгоритм

- 1 Формируем  $T$  выборок  $X_t^m$ ,  $t = 1, \dots, T$  мощности  $m$  с помощью бутстрэп-семплирования,
- 2 На каждой выборке  $X_t^m$ ,  $t = 1, \dots, T$  обучаем свой алгоритм  $b_t(x)$ ,



- **Дано:** обучающая выборка  $X^m$  мощности  $m$ .
- **Цель:** обучить ансамбль из  $T$  классификаторов  $b_t(x)$ ,  $t = 1, \dots, T$ .

## Алгоритм

- 1 Формируем  $T$  выборок  $X_t^m$ ,  $t = 1, \dots, T$  мощности  $m$  с помощью бутстрэп-семплирования,
- 2 На каждой выборке  $X_t^m$ ,  $t = 1, \dots, T$  обучаем свой алгоритм  $b_t(x)$ ,
- 3 Результат применения – усреднение (для регрессии) или голосование (для классификации).





Оказывается, можно использовать бутстрэп-семплирование не только для обучающей выборки, но и для признаков!

---

<sup>9</sup>Ho T. K. (1998). "The Random Subspace Method for Constructing Decision Forests"



Оказывается, можно использовать бутстрэп-семплирование не только для обучающей выборки, но и для признаков!

Это – **метод случайных подпространств**<sup>9</sup>.

---

<sup>9</sup>Но Т. К. (1998). “The Random Subspace Method for Constructing Decision Forests”

Оказывается, можно использовать бутстрэп-семплирование не только для обучающей выборки, но и для признаков!

Это – **метод случайных подпространств**<sup>9</sup>.

Т.о., случайные деревья из прошлой лекции – это объединение:

- Бэггинга для работы с выборкой и алгоритмами,

---

<sup>9</sup>Но Т. К. (1998). “The Random Subspace Method for Constructing Decision Forests”

Оказывается, можно использовать бутстрэп-семплирование не только для обучающей выборки, но и для признаков!

Это – **метод случайных подпространств**<sup>9</sup>.

Т.о., случайные деревья из прошлой лекции – это объединение:

- Бэггинга для работы с выборкой и алгоритмами,
- Метода случайных подпространств для работы с признаковым пространством.

---

<sup>9</sup>Но Т. К. (1998). “The Random Subspace Method for Constructing Decision Forests”

## Плюсы бэггинга

- Уменьшает разброс и, как следствие, борется с переобучением,

## Плюсы бэггинга

- Уменьшает разброс и, как следствие, борется с переобучением,
- Ошибки базовых алгоритмов взаимно компенсируются,

## Плюсы бэггинга

- Уменьшает разброс и, как следствие, борется с переобучением,
- Ошибки базовых алгоритмов взаимно компенсируются,
- Объекты-выбросы могут не попасть в некоторые обучающие подвыборки,

## Плюсы бэггинга

- Уменьшает разброс и, как следствие, борется с переобучением,
- Ошибки базовых алгоритмов взаимно компенсируются,
- Объекты-выбросы могут не попасть в некоторые обучающие подвыборки,
- Хорошо работает для нестабильных алгоритмов (нейронные сети),



## Плюсы бэггинга

- Уменьшает разброс и, как следствие, борется с переобучением,
- Ошибки базовых алгоритмов взаимно компенсируются,
- Объекты-выбросы могут не попасть в некоторые обучающие подвыборки,
- Хорошо работает для нестабильных алгоритмов (нейронные сети),
- Легко распараллеливается.

# Плюсы и минусы бэггинга

## Плюсы бэггинга

- Уменьшает разброс и, как следствие, борется с переобучением,
- Ошибки базовых алгоритмов взаимно компенсируются,
- Объекты-выбросы могут не попасть в некоторые обучающие подвыборки,
- Хорошо работает для нестабильных алгоритмов (нейронные сети),
- Легко распараллеливается.

## Минусы бэггинга

- Не борется со смещением,

# Плюсы и минусы бэггинга

## Плюсы бэггинга

- Уменьшает разброс и, как следствие, борется с переобучением,
- Ошибки базовых алгоритмов взаимно компенсируются,
- Объекты-выбросы могут не попасть в некоторые обучающие подвыборки,
- Хорошо работает для нестабильных алгоритмов (нейронные сети),
- Легко распараллеливается.

## Минусы бэггинга

- Не борется со смещением,
- Каждый базовый алгоритм видит всего 63% обучающих данных,

# Плюсы и минусы бэггинга

## Плюсы бэггинга

- Уменьшает разброс и, как следствие, борется с переобучением,
- Ошибки базовых алгоритмов взаимно компенсируются,
- Объекты-выбросы могут не попасть в некоторые обучающие подвыборки,
- Хорошо работает для нестабильных алгоритмов (нейронные сети),
- Легко распараллеливается.

## Минусы бэггинга

- Не борется со смещением,
- Каждый базовый алгоритм видит всего 63% обучающих данных,
- Не очень хорошо работает для стабильных алгоритмов (метод К-ближайших соседей),

# Плюсы и минусы бэггинга

## Плюсы бэггинга

- Уменьшает разброс и, как следствие, борется с переобучением,
- Ошибки базовых алгоритмов взаимно компенсируются,
- Объекты-выбросы могут не попасть в некоторые обучающие подвыборки,
- Хорошо работает для нестабильных алгоритмов (нейронные сети),
- Легко распараллеливается.

## Минусы бэггинга

- Не борется со смещением,
- Каждый базовый алгоритм видит всего 63% обучающих данных,
- Не очень хорошо работает для стабильных алгоритмов (метод К-ближайших соседей),
- Нужно держать в памяти  $T$  базовых алгоритмов,



# Плюсы и минусы бэггинга

## Плюсы бэггинга

- Уменьшает разброс и, как следствие, борется с переобучением,
- Ошибки базовых алгоритмов взаимно компенсируются,
- Объекты-выбросы могут не попасть в некоторые обучающие подвыборки,
- Хорошо работает для нестабильных алгоритмов (нейронные сети),
- Легко распараллеливается.

## Минусы бэггинга

- Не борется со смещением,
- Каждый базовый алгоритм видит всего 63% обучающих данных,
- Не очень хорошо работает для стабильных алгоритмов (метод К-ближайших соседей),
- Нужно держать в памяти  $T$  базовых алгоритмов,
- Нужно делать  $T$  запусков.



# Плюсы и минусы бэггинга

## Плюсы бэггинга

- Уменьшает разброс и, как следствие, борется с переобучением,
- Ошибки базовых алгоритмов взаимно компенсируются,
- Объекты-выбросы могут не попасть в некоторые обучающие подвыборки,
- Хорошо работает для нестабильных алгоритмов (нейронные сети),
- Легко распараллеливается.

## Минусы бэггинга

- Не борется со смещением,
- Каждый базовый алгоритм видит всего 63% обучающих данных,
- Не очень хорошо работает для стабильных алгоритмов (метод К-ближайших соседей),
- Нужно держать в памяти  $T$  базовых алгоритмов,
- Нужно делать  $T$  запусков.

**Замечание.** Последние два минуса характерны в целом для ансамблирования.







Попробуем теперь бороться не только с разбросом, но и со смещением.

---

<sup>10</sup>Schapire R. E. (1990). "The Strength of Weak Learnability".

Попробуем теперь бороться не только с разбросом, но и со смещением.  
Главная идея бустинга<sup>10</sup>:

---

<sup>10</sup>Schapire R. E. (1990). "The Strength of Weak Learnability".

Попробуем теперь бороться не только с разбросом, но и со смещением.

Главная идея бустинга<sup>10</sup>:

- Отвечает на вопрос: “Может ли набор слабых обучающих алгоритмов создать сильный обучающий алгоритм?”,

---

<sup>10</sup>Schapire R. E. (1990). “The Strength of Weak Learnability”.

Попробуем теперь бороться не только с разбросом, но и со смещением.

Главная идея бустинга<sup>10</sup>:

- Отвечает на вопрос: “Может ли набор слабых обучающих алгоритмов создать сильный обучающий алгоритм?”,
- Борется не только с разбросом, но и со смещением алгоритма.

---

<sup>10</sup>Schapire R. E. (1990). “The Strength of Weak Learnability”.

Попробуем теперь бороться не только с разбросом, но и со смещением.

Главная идея бустинга<sup>10</sup>:

- Отвечает на вопрос: “Может ли набор слабых обучающих алгоритмов создать сильный обучающий алгоритм?”,
- Борется не только с разбросом, но и со смещением алгоритма.

## Определение

**Бустинг (Boosting)** - это метод ансамблирования, основанный на:

- 1 взвешенном голосовании композиции,
- 2 последовательном выборе нового классификатора на основе ошибок предыдущих.

<sup>10</sup>Schapire R. E. (1990). “The Strength of Weak Learnability”.

# Типы бустинга

Разные виды бустинга можно описать в зависимости от:

- Функции потерь,

# Типы бустинга

Разные виды бустинга можно описать в зависимости от:

- Функции потерь,
- Множества выходных значений базовых классификаторов.



# Типы бустинга

Разные виды бустинга можно описать в зависимости от:

- Функции потерь,
- Множества выходных значений базовых классификаторов.

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



# Типы бустинга

Разные виды бустинга можно описать в зависимости от:

- Функции потерь,
- Множества выходных значений базовых классификаторов.

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

Разные виды бустинга можно описать в зависимости от:

- Функции потерь,
- Множества выходных значений базовых классификаторов.

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  $L : \mathbb{R} \rightarrow \mathbb{R}$  – гладкая функция от  $y_i a(x_i)$



# Типы бустинга

Разные виды бустинга можно описать в зависимости от:

- Функции потерь,
- Множества выходных значений базовых классификаторов.

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  $L : \mathbb{R} \rightarrow \mathbb{R}$  – гладкая функция от  $y_i a(x_i)$

## Gradient Boosting

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

# Бустинг для бинарной классификации

Пусть  $X^m = \{(x_i, y_i)_{i=1}^m\}$ ,  $y_i \in Y = \{+1, -1\}$ ,  $b_t : X \rightarrow \{-1, 0, +1\}$ . Значение  $b_t(x) = 0$  вводится для сигнализации неопределенности в классификации (аналогия: нахождение внутри полосы для SVM).



# Бустинг для бинарной классификации

Пусть  $X^m = \{(x_i, y_i)_{i=1}^m\}$ ,  $y_i \in Y = \{+1, -1\}$ ,  $b_t : X \rightarrow \{-1, 0, +1\}$ . Значение  $b_t(x) = 0$  вводится для сигнализации неопределенности в классификации (аналогия: нахождение внутри полосы для SVM).

- Алгоритм классификации – взвешенное голосование:  $a(x) = \text{sign}(\sum_{t=1}^T \alpha_t b_t(x))$ ,



# Бустинг для бинарной классификации

Пусть  $X^m = \{(x_i, y_i)_{i=1}^m\}$ ,  $y_i \in Y = \{+1, -1\}$ ,  $b_t : X \rightarrow \{-1, 0, +1\}$ . Значение  $b_t(x) = 0$  вводится для сигнализации неопределенности в классификации (аналогия: нахождение внутри полосы для SVM).

- Алгоритм классификации – взвешенное голосование:  $a(x) = \text{sign}(\sum_{t=1}^T \alpha_t b_t(x))$ ,
- Эмпирический риск – число ошибок на  $X^m$ :  
$$R_T = \sum_{i=1}^m [y_i \sum_{t=1}^T \alpha_t b_t(x_i) < 0]$$



# Бустинг для бинарной классификации

Пусть  $X^m = \{(x_i, y_i)_{i=1}^m\}$ ,  $y_i \in Y = \{+1, -1\}$ ,  $b_t : X \rightarrow \{-1, 0, +1\}$ . Значение  $b_t(x) = 0$  вводится для сигнализации неопределенности в классификации (аналогия: нахождение внутри полосы для SVM).

- Алгоритм классификации – взвешенное голосование:  $a(x) = \text{sign}(\sum_{t=1}^T \alpha_t b_t(x))$ ,
- Эмпирический риск – число ошибок на  $X^m$ :  
$$R_T = \sum_{i=1}^m [y_i \sum_{t=1}^T \alpha_t b_t(x_i) < 0]$$

Основные идеи обучения:

- Заморозка  $\alpha_1 b_1(x_i), \dots, \alpha_{t-1} b_{t-1}(x_i)$  при добавлении  $\alpha_t b_t(x_i)$ ,



# Бустинг для бинарной классификации

Пусть  $X^m = \{(x_i, y_i)_{i=1}^m\}$ ,  $y_i \in Y = \{+1, -1\}$ ,  $b_t : X \rightarrow \{-1, 0, +1\}$ . Значение  $b_t(x) = 0$  вводится для сигнализации неопределенности в классификации (аналогия: нахождение внутри полосы для SVM).

- Алгоритм классификации – взвешенное голосование:  $a(x) = \text{sign}(\sum_{t=1}^T \alpha_t b_t(x))$ ,
- Эмпирический риск – число ошибок на  $X^m$ :  
$$R_T = \sum_{i=1}^m [y_i \sum_{t=1}^T \alpha_t b_t(x_i) < 0]$$

Основные идеи обучения:

- Заморозка  $\alpha_1 b_1(x_i), \dots, \alpha_{t-1} b_{t-1}(x_i)$  при добавлении  $\alpha_t b_t(x_i)$ ,
- Использовать аппроксимированный Э.Р.





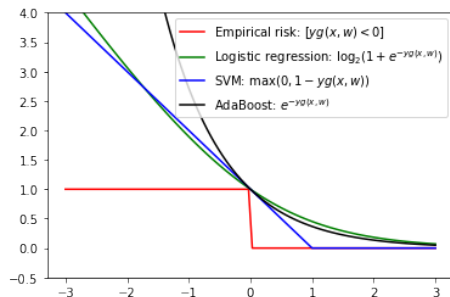
# Бустинг для бинарной классификации

Пусть  $X^m = \{(x_i, y_i)_{i=1}^m\}$ ,  $y_i \in Y = \{+1, -1\}$ ,  $b_t : X \rightarrow \{-1, 0, +1\}$ . Значение  $b_t(x) = 0$  вводится для сигнализации неопределенности в классификации (аналогия: нахождение внутри полосы для SVM).

- Алгоритм классификации – взвешенное голосование:  $a(x) = \text{sign}(\sum_{t=1}^T \alpha_t b_t(x))$ ,
- Эмпирический риск – число ошибок на  $X^m$ :  
 $R_T = \sum_{i=1}^m [y_i \sum_{t=1}^T \alpha_t b_t(x_i) < 0]$

Основные идеи обучения:

- Заморозка  $\alpha_1 b_1(x_i), \dots, \alpha_{t-1} b_{t-1}(x_i)$  при добавлении  $\alpha_t b_t(x_i)$ ,
- Использовать аппроксимированный Э.Р.



Аппроксимация Э.Р. с помощью функции потерь  $e^{-y_i a(x_i)}$ :

$$R_T \leq \tilde{R}_T = \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)}$$



Аппроксимация Э.Р. с помощью функции потерь  $e^{-y_i a(x_i)}$ :

$$R_T \leq \tilde{R}_T = \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)}$$

- Вектор весов (взвешиваем объекты)  $W^m = (w_1, \dots, w_m)$ :

$$w_i = e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} \Rightarrow \tilde{R}_{T-1} = \sum_{i=1}^m w_i,$$

- Нормировка:  $\tilde{w}_i = \frac{w_i}{\sum_{j=1}^m w_j} \Rightarrow \sum_{i=1}^m \tilde{w}_i = 1, \tilde{w}_i \geq 0$



Аппроксимация Э.Р. с помощью функции потерь  $e^{-y_i a(x_i)}$ :

$$R_T \leq \tilde{R}_T = \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)}$$

- Вектор весов (взвешиваем объекты)  $W^m = (w_1, \dots, w_m)$ :  
 $w_i = e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} \Rightarrow \tilde{R}_{T-1} = \sum_{i=1}^m w_i$ ,
- Нормировка:  $\tilde{w}_i = \frac{w_i}{\sum_{j=1}^m w_j} \Rightarrow \sum_{i=1}^m \tilde{w}_i = 1, \tilde{w}_i \geq 0$
- Вероятностный вектор  $U^m = (u_1, \dots, u_m)$ :  $\sum_{i=1}^m u_i = 1, u_i \geq 0$ ,
- Взвешенное число правильных классификаций алгоритма  $b(x)$  по 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 вектору  $U^m$ :  
 $N(b; U^m) = \sum_{i=1}^m u_i [b(x) \neq y_i]$
- Взвешенное число отказов от классификации:  $1 - P - N$ .



# Основная теорема бустинга

Пусть  $A$  – достаточно богатое семейство базовых алгоритмов.

## Теорема

Если для любого вероятностного вектора  $U^m$  существует алгоритм  $b \in A$ , т.ч.  $P(b; U^m) > N(b; U^m)$ , то минимум аппроксимированного Э.Р.  $\tilde{R}_T$  достигается на:

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



# Основная теорема бустинга

Пусть  $A$  – достаточно богатое семейство базовых алгоритмов.

## Теорема

Если для любого вероятностного вектора  $U^m$  существует алгоритм  $b \in A$ , т.ч.  $P(b; U^m) > N(b; U^m)$ , то минимум аппроксимированного Э.Р.  $\tilde{R}_T$  достигается на:

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

**Замечание.** В этом случае  $\alpha_T > 0$ .



Если  $b \in \{-1, 0, +1\}$ , то верно тождество  $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$ .



Если  $b \in \{-1, 0, +1\}$ , то верно тождество  $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$ .

$$\begin{aligned} \tilde{R}_T &= \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} = \\ &= \sum_{i=1}^m w_i (e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]) = \end{aligned}$$





Если  $b \in \{-1, 0, +1\}$ , то верно тождество  $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$ .

$$\begin{aligned}\tilde{R}_T &= \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} = \\ &= \sum_{i=1}^m w_i (e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]) = \\ &= e^{-\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = -y_i] + \sum_{i=1}^m w_i [b_T(x_i) = 0] =\end{aligned}$$



Если  $b \in \{-1, 0, +1\}$ , то верно тождество  $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$ .

$$\begin{aligned} \tilde{R}_T &= \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} = \\ &= \sum_{i=1}^m w_i (e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]) = \\ &= e^{-\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = -y_i] + \sum_{i=1}^m w_i [b_T(x_i) = 0] = \\ &= (e^{-\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = -y_i] + \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = 0]) \sum_{i=1}^m w_i = \end{aligned}$$

Если  $b \in \{-1, 0, +1\}$ , то верно тождество  $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$ .

$$\begin{aligned} \tilde{R}_T &= \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} = \\ &= \sum_{i=1}^m w_i (e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]) = \\ &= e^{-\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = -y_i] + \sum_{i=1}^m w_i [b_T(x_i) = 0] = \\ &= (e^{-\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = -y_i] + \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = 0]) \sum_{i=1}^m w_i = \\ &= (e^{-\alpha_T} P + e^{\alpha_T} N + 1 - P - N) \tilde{R}_{T-1}. \end{aligned}$$



Если  $b \in \{-1, 0, +1\}$ , то верно тождество  $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$ .

$$\begin{aligned} \tilde{R}_T &= \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} = \\ &= \sum_{i=1}^m w_i (e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]) = \\ &= e^{-\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = -y_i] + \sum_{i=1}^m w_i [b_T(x_i) = 0] = \\ &= (e^{-\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = -y_i] + \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = 0]) \sum_{i=1}^m w_i = \\ &= (e^{-\alpha_T} P + e^{\alpha_T} N + 1 - P - N) \tilde{R}_{T-1}. \end{aligned}$$

$$\tilde{R}_T \rightarrow \min_{\alpha_T} \Rightarrow \frac{\partial \tilde{R}_T}{\partial \alpha_T} = 0.$$



# Доказательство

Если  $b \in \{-1, 0, +1\}$ , то верно тождество  $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$ .

$$\begin{aligned}\tilde{R}_T &= \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} = \\ &= \sum_{i=1}^m w_i (e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]) = \\ &= e^{-\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = -y_i] + \sum_{i=1}^m w_i [b_T(x_i) = 0] = \\ &= (e^{-\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = -y_i] + \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = 0]) \sum_{i=1}^m w_i = \\ &= (e^{-\alpha_T} P + e^{\alpha_T} N + 1 - P - N) \tilde{R}_{T-1}.\end{aligned}$$

$$\tilde{R}_T \rightarrow \min_{\alpha_T} \Rightarrow \frac{\partial \tilde{R}_T}{\partial \alpha_T} = 0.$$

$$\frac{\partial \tilde{R}_T}{\partial \alpha_T} = (-e^{-\alpha_T} P + e^{\alpha_T} N) \tilde{R}_{T-1} = 0 \Rightarrow e^{-\alpha_T} P = e^{\alpha_T} N \Rightarrow e^{2\alpha_T} = \frac{P}{N} \Rightarrow \alpha_T = \frac{1}{2} \ln \frac{P(b_T; \tilde{W}^m)}{N(b_T; \tilde{W}^m)}.$$



Если  $b \in \{-1, 0, +1\}$ , то верно тождество  $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$ .

$$\begin{aligned} \tilde{R}_T &= \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} = \\ &= \sum_{i=1}^m w_i (e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]) = \\ &= e^{-\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = -y_i] + \sum_{i=1}^m w_i [b_T(x_i) = 0] = \\ &= (e^{-\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = -y_i] + \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = 0]) \sum_{i=1}^m w_i = \\ &= (e^{-\alpha_T} P + e^{\alpha_T} N + 1 - P - N) \tilde{R}_{T-1}. \end{aligned}$$

$$\tilde{R}_T \rightarrow \min_{\alpha_T} \Rightarrow \frac{\partial \tilde{R}_T}{\partial \alpha_T} = 0.$$

$$\frac{\partial \tilde{R}_T}{\partial \alpha_T} = (-e^{-\alpha_T} P + e^{\alpha_T} N) \tilde{R}_{T-1} = 0 \Rightarrow e^{-\alpha_T} P = e^{\alpha_T} N \Rightarrow e^{2\alpha_T} = \frac{P}{N} \Rightarrow \alpha_T = \frac{1}{2} \ln \frac{P(b_T; \tilde{W}^m)}{N(b_T; \tilde{W}^m)}.$$

Для поиска  $b_T(x)$  подставим найденное  $\alpha_T$  в формулу для  $\tilde{R}_T$ :



Если  $b \in \{-1, 0, +1\}$ , то верно тождество  $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$ .

$$\begin{aligned} \tilde{R}_T &= \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} = \\ &= \sum_{i=1}^m w_i (e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]) = \\ &= e^{-\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = -y_i] + \sum_{i=1}^m w_i [b_T(x_i) = 0] = \\ &= (e^{-\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = -y_i] + \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = 0]) \sum_{i=1}^m w_i = \\ &= (e^{-\alpha_T} P + e^{\alpha_T} N + 1 - P - N) \tilde{R}_{T-1}. \end{aligned}$$

$$\tilde{R}_T \rightarrow \min_{\alpha_T} \Rightarrow \frac{\partial \tilde{R}_T}{\partial \alpha_T} = 0.$$

$$\frac{\partial \tilde{R}_T}{\partial \alpha_T} = (-e^{-\alpha_T} P + e^{\alpha_T} N) \tilde{R}_{T-1} = 0 \Rightarrow e^{-\alpha_T} P = e^{\alpha_T} N \Rightarrow e^{2\alpha_T} = \frac{P}{N} \Rightarrow \alpha_T = \frac{1}{2} \ln \frac{P(b_T; \tilde{W}^m)}{N(b_T; \tilde{W}^m)}.$$

Для поиска  $b_T(x)$  подставим найденное  $\alpha_T$  в формулу для  $\tilde{R}_T$ :

$$\begin{aligned} \tilde{R}_T &= (\sqrt{\frac{N}{P}} P + \sqrt{\frac{P}{N}} N + 1 - P - N) \tilde{R}_{T-1} = (1 - (P - 2\sqrt{PN} + N)) \tilde{R}_{T-1} = \\ &= (1 - (\sqrt{P} - \sqrt{N})^2) \tilde{R}_{T-1} \rightarrow \min_{b_T} \Rightarrow \end{aligned}$$



Если  $b \in \{-1, 0, +1\}$ , то верно тождество  $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$ .

$$\begin{aligned} \tilde{R}_T &= \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} = \\ &= \sum_{i=1}^m w_i (e^{-\alpha_T [b_T(x_i) = y_i]} + e^{\alpha_T [b_T(x_i) = -y_i]} + [b_T(x_i) = 0]) = \\ &= e^{-\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = -y_i] + \sum_{i=1}^m w_i [b_T(x_i) = 0] = \\ &= (e^{-\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = -y_i] + \sum_{i=1}^m \tilde{w}_i [b_T(x_i) = 0]) \sum_{i=1}^m w_i = \\ &= (e^{-\alpha_T} P + e^{\alpha_T} N + 1 - P - N) \tilde{R}_{T-1}. \end{aligned}$$

$$\tilde{R}_T \rightarrow \min_{\alpha_T} \Rightarrow \frac{\partial \tilde{R}_T}{\partial \alpha_T} = 0.$$

$$\frac{\partial \tilde{R}_T}{\partial \alpha_T} = (-e^{-\alpha_T} P + e^{\alpha_T} N) \tilde{R}_{T-1} = 0 \Rightarrow e^{-\alpha_T} P = e^{\alpha_T} N \Rightarrow e^{2\alpha_T} = \frac{P}{N} \Rightarrow \alpha_T = \frac{1}{2} \ln \frac{P(b_T; \tilde{W}^m)}{N(b_T; \tilde{W}^m)}.$$

Для поиска  $b_T(x)$  подставим найденное  $\alpha_T$  в формулу для  $\tilde{R}_T$ :

$$\begin{aligned} \tilde{R}_T &= (\sqrt{\frac{N}{P}} P + \sqrt{\frac{P}{N}} N + 1 - P - N) \tilde{R}_{T-1} = (1 - (P - 2\sqrt{PN} + N)) \tilde{R}_{T-1} = \\ &= (1 - (\sqrt{P} - \sqrt{N})^2) \tilde{R}_{T-1} \rightarrow \min_{b_T} \Rightarrow \end{aligned}$$

$$b_T = \arg \max_{b \in A} \sqrt{P(b; \tilde{W}^m)} - \sqrt{N(b; \tilde{W}^m)} \text{ (т.к. } N < P \leq 1 \text{ по условию Теоремы). Ч.т.д.}$$



# Следствие 1: Сходимость

## Теорема

Если на каждом шаге  $t$  можно добиться выполнения

$\sqrt{P(b_t; \widetilde{W}^m)} - \sqrt{N(b_t; \widetilde{W}^m)} = \beta_t \geq \beta$  при некотором  $0 < \beta \leq 1$ , то за конечное число шагов будет построен алгоритм, не допускающий ни единой ошибки на обучающем множестве.

# Следствие 1: Сходимость

## Теорема

Если на каждом шаге  $t$  можно добиться выполнения

$\sqrt{P(b_t; \widetilde{W}^m)} - \sqrt{N(b_t; \widetilde{W}^m)} = \beta_t \geq \beta$  при некотором  $0 < \beta \leq 1$ , то за конечное число шагов будет построен алгоритм, не допускающий ни единой ошибки на обучающем множестве.

**Доказательство.**

$$R_T \leq \tilde{R}_T = (1 - (\sqrt{P} - \sqrt{N})^2) \tilde{R}_{T-1} \leq (1 - \beta^2) \tilde{R}_{T-1} \leq \dots \leq (1 - \beta^2)^{T-1} \tilde{R}_1.$$

# Следствие 1: Сходимость

## Теорема

Если на каждом шаге  $t$  можно добиться выполнения

$\sqrt{P(b_t; \widetilde{W}^m)} - \sqrt{N(b_t; \widetilde{W}^m)} = \beta_t \geq \beta$  при некотором  $0 < \beta \leq 1$ , то за конечное число шагов будет построен алгоритм, не допускающий ни единой ошибки на обучающем множестве.

**Доказательство.**

$$R_T \leq \tilde{R}_T = (1 - (\sqrt{P} - \sqrt{N})^2) \tilde{R}_{T-1} \leq (1 - \beta^2) \tilde{R}_{T-1} \leq \dots \leq (1 - \beta^2)^{T-1} \tilde{R}_1.$$

Для любого  $0 < \beta \leq 1$  и любого  $\tilde{R}_1$  будет существовать такое  $T$ , что  $R_T < 1$ .

# Следствие 1: Сходимость

## Теорема

Если на каждом шаге  $t$  можно добиться выполнения

$\sqrt{P(b_t; \widetilde{W}^m)} - \sqrt{N(b_t; \widetilde{W}^m)} = \beta_t \geq \beta$  при некотором  $0 < \beta \leq 1$ , то за конечное число шагов будет построен алгоритм, не допускающий ни единой ошибки на обучающем множестве.

**Доказательство.**

$$R_T \leq \widetilde{R}_T = (1 - (\sqrt{P} - \sqrt{N})^2) \widetilde{R}_{T-1} \leq (1 - \beta^2) \widetilde{R}_{T-1} \leq \dots \leq (1 - \beta^2)^{T-1} \widetilde{R}_1.$$

Для любого  $0 < \beta \leq 1$  и любого  $\widetilde{R}_1$  будет существовать такое  $T$ , что  $R_T < 1$ .

Э.Р.  $R_T$  – это число ошибок на обучающем множестве (т.е. неотрицательное целое число)

$\Rightarrow R_T = 0$ . Ч.т.д.



## Следствие 2: Классический AdaBoost

Рассмотрим более частную ситуацию, когда базовый алгоритм не сигнализирует о неопределенности:  $b_t : X \rightarrow \{-1, +1\}$ . Тогда  $P + N = 1$ .

---

<sup>11</sup>Freund Y. and Schapire R.E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting"

## Следствие 2: Классический AdaBoost

Рассмотрим более частную ситуацию, когда базовый алгоритм не сигнализирует о неопределенности:  $b_t : X \rightarrow \{-1, +1\}$ . Тогда  $P + N = 1$ .

В этом случае конкретный алгоритм бустинга называется AdaBoost<sup>11</sup> (**Adaptive Boosting**).

---

<sup>11</sup>Freund Y. and Schapire R.E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting"

## Следствие 2: Классический AdaBoost

Рассмотрим более частную ситуацию, когда базовый алгоритм не сигнализирует о неопределенности:  $b_t : X \rightarrow \{-1, +1\}$ . Тогда  $P + N = 1$ .

В этом случае конкретный алгоритм бустинга называется AdaBoost<sup>11</sup> (**Adaptive Boosting**).

### Теорема

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

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

<sup>11</sup>Freund Y. and Schapire R.E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting"

## Алгоритм

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



## Алгоритм

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

Для  $t = 1, \dots, T$

- Обучение базового алгоритма  $b_t = \arg \min_{b \in A} N(b; \widetilde{W}^m),$



## Алгоритм

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

Для  $t = 1, \dots, T$

- Обучение базового алгоритма  $b_t = \arg \min_{b \in A} N(b; \widetilde{W}^m),$
- Вычисление нового веса  $\alpha_t = \frac{1}{2} \ln \frac{1 - N(b_t; \widetilde{W}^m)}{N(b_t; \widetilde{W}^m)},$

## Алгоритм

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

Для  $t = 1, \dots, T$

- 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  $w_i := w_i e^{-\alpha_t y_i b_t(x_i)}, i = 1, \dots, m,$

## Алгоритм

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

Для  $t = 1, \dots, T$

- 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  $w_i := w_i e^{-\alpha_t y_i b_t(x_i)}, i = 1, \dots, m,$
- Перенормировка весов  $w_i := \frac{w_i}{\sum_{j=1}^m w_j}, i = 1, \dots, m.$



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

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

- Вес объекта  $x_i$  увеличивается в  $e^{\alpha_t}$  раз, когда  $b_t$  допускает на нем ошибку (т.к.  $\alpha_t > 0$ ),



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

- Вес объекта  $x_i$  увеличивается в  $e^{\alpha_t}$  раз, когда  $b_t$  допускает на нем ошибку (т.к.  $\alpha_t > 0$ ),
- Вес объекта  $x_i$  уменьшается в  $e^{\alpha_t}$  раз, когда  $b_t$  правильно его классифицирует,



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

- Вес объекта  $x_i$  увеличивается в  $e^{\alpha_t}$  раз, когда  $b_t$  допускает на нем ошибку (т.к.  $\alpha_t > 0$ ),
- Вес объекта  $x_i$  уменьшается в  $e^{\alpha_t}$  раз, когда  $b_t$  правильно его классифицирует,
- Т.о. наибольший вес накапливается у тех объектов, которые чаще оказывались трудными для предыдущих алгоритмов.







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