Введение в искусственный интеллект. Машинное обучение

Лекция 9. Ансамблирование моделей. Три метода на букву “Б”

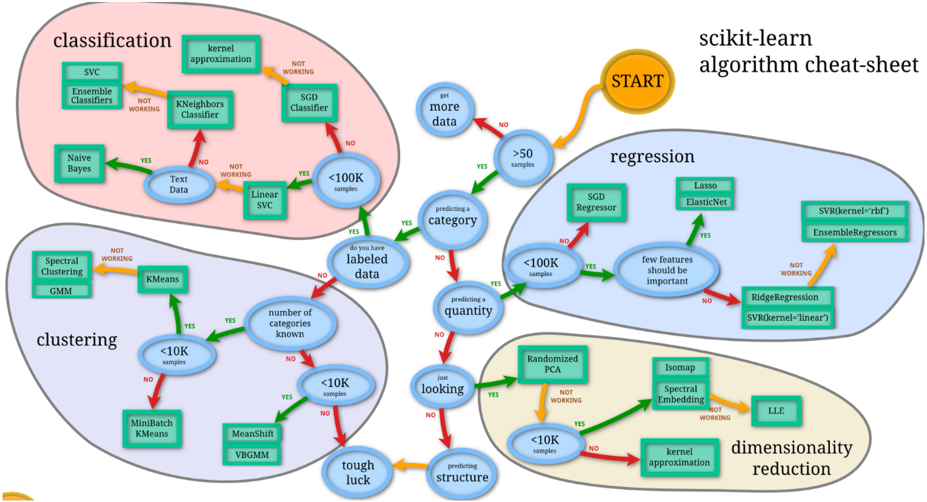
МаТИС

19 апреля 2019г.

План лекции

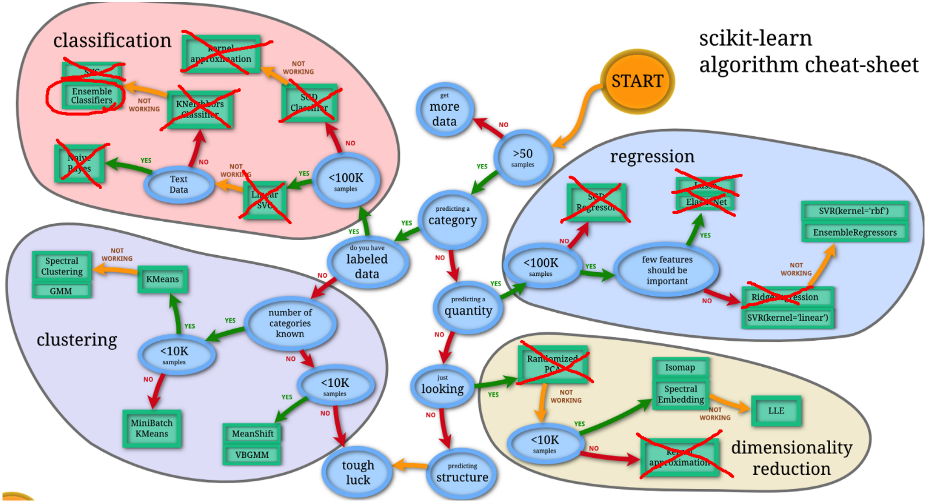
1. Стековое обобщение
   - Блендинг
   - Стекинг
2. Бутстрэп
3. Бэггинг
4. Бустинг с дискретными базовыми алгоритмами

Дорожная карта Scikit-Learn[1]



[1] https://scikit-learn.org/stable/tutorial/machine_learning_map/

Дорожная карта Scikit-Learn[1]



[1] https://scikit-learn.org/stable/tutorial/machine_learning_map/

Ансамблирование

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 не бесконечен: состоит из конкретного конечного множества альтернативных моделей.

Ансамблирование

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 не бесконечен: состоит из конкретного конечного множества альтернативных моделей.
Основные представители:

- Стековое обобщение (stacked generalization)
- Бэггинг (bagging)
- 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 матожидание среднеквадратичной ошибки:
$E(b_t(x) - y(x))^2 = E\varepsilon_t^2(x)$.
Средняя ошибка базовых алгоритмов: $E_{avg} = \frac{1}{T} E \sum_{t=1}^{T} \varepsilon_t^2(x)$.

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 матожидание среднеквадратичной ошибки:
$E(b_t(x) - y(x))^2 = E\varepsilon_t^2(x)$.
Средняя ошибка базовых алгоритмов: $E_{avg} = \frac{1}{T}E\sum_{t=1}^{T}\varepsilon_t^2(x)$.
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 матожидание среднеквадратичной ошибки:
$E(b_t(x) - y(x))^2 = E\varepsilon_t^2(x)$.
Средняя ошибка базовых алгоритмов: $E_{avg} = \frac{1}{T} E\sum_{t=1}^{T} \varepsilon_t^2(x)$.
Предположим, что ошибки несмещены и некоррелированы: $E\varepsilon_t(x) = 0, E\varepsilon_t\varepsilon_u = 0, t \neq u$.
Найдем среднеквадратичную ошибку для $a(x)$:
$E_{ens} = E(a(x) - y(x))^2 = E(\frac{1}{T}\sum_{t=1}^{T} b_t(x) - y(x))^2 = E(\frac{1}{T}\sum_{t=1}^{T}\varepsilon_t)^2 =$
$\frac{1}{T^2}E(\sum_{t=1}^{T}\varepsilon_t^2 + \sum_{t\neq u}\varepsilon_t\varepsilon_u) = \frac{1}{T^2}E\sum_{i=1}^{T}\varepsilon_i^2(x) = \frac{1}{T}E_{avg}$.

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 матожидание среднеквадратичной ошибки:
$E(b_t(x) - y(x))^2 = E\varepsilon_t^2(x)$.
Средняя ошибка базовых алгоритмов: $E_{avg} = \frac{1}{T} E \sum_{t=1}^{T} \varepsilon_t^2(x)$.
Предположим, что ошибки несмещены и некоррелированы: $E\varepsilon_t(x) = 0, E\varepsilon_t\varepsilon_u = 0, t \neq u$.
Найдем среднеквадратичную ошибку для $a(x)$:
$E_{ens} = E(a(x) - y(x))^2 = E(\frac{1}{T}\sum_{t=1}^{T} b_t(x) - y(x))^2 = E(\frac{1}{T}\sum_{t=1}^{T} \varepsilon_t)^2 =$
$\frac{1}{T^2} E(\sum_{t=1}^{T} \varepsilon_t^2 + \sum_{t \neq u} \varepsilon_t \varepsilon_u) = \frac{1}{T^2} E \sum_{i=1}^{T} \varepsilon_i^2(x) = \frac{1}{T} E_{avg}$.
Таким образом, простое голосование позволило уменьшить средний квадрат ошибки в T раз!

Стековое обобщение[2]

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

[2] Wolpert D. (1992) "Stacked Generalization"

Стековое обобщение[2]

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

[2] Wolpert D. (1992) "Stacked Generalization"

Стековое обобщение[2]

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 $b_t(x), t = 1, \ldots, T$
2. Фиксируем алгоритмы $b_t(x)$
3. Обучаем комбинирующий алгоритм верхнего уровня $a(x) = a(b_1(x), \ldots, b_T(x))$

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

[2] Wolpert D. (1992) "Stacked Generalization"

Стековое обобщение[2]

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 $b_t(x), t = 1, \ldots, T$
2. Фиксируем алгоритмы $b_t(x)$
3. Обучаем комбинирующий алгоритм верхнего уровня $a(x) = a(b_1(x), \ldots, b_T(x))$

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 :)

[2] 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.

- Разбиваем выборку на две части

Блендинг (Blending)

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

- Разбиваем выборку на две части
- На одной части обучаем базовые алгоритмы

Блендинг (Blending)

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

- Разбиваем выборку на две части
- На одной части обучаем базовые алгоритмы
- На второй части обучаем мета-алгоритм

Блендинг (Blending)

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

- Разбиваем выборку на две части
- На одной части обучаем базовые алгоритмы
- На второй части обучаем мета-алгоритм
- На тесте сначала получаем выходы базовых алгоритмов, к которым применяем мета-алгоритм

Блендинг (Blending)

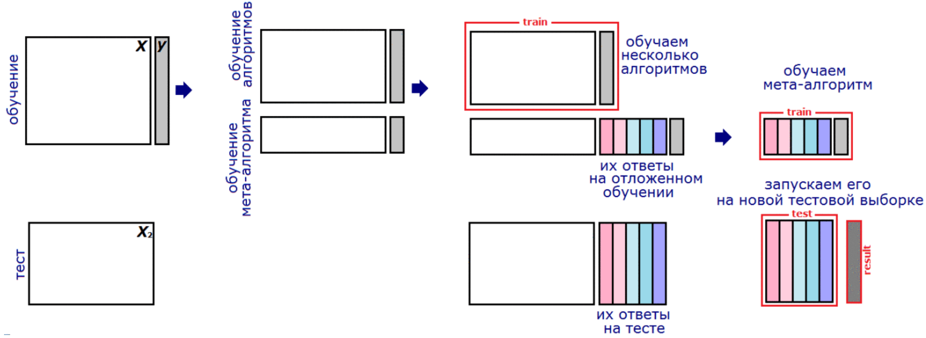
Ранее мы рассмотрели общую схему, теперь рассмотрим конкретные варианты реализации на практике.

- Разбиваем выборку на две части
- На одной части обучаем базовые алгоритмы
- На второй части обучаем мета-алгоритм
- На тесте сначала получаем выходы базовых алгоритмов, к которым применяем мета-алгоритм

Замечание. Понятно, что можем делить на больше число частей и затем усреднять или конкатенировать выходы.

Блендинг – визуализация

Рассмотрим схему[3] блендинга:



[3] https://dyakonov.org

Стекинг (Stacking)

Проблема блендинга: базовые алгоритмы не видят всей обучающей выборки. Поэтому можно усложнить схему.

Стекинг (Stacking)

Проблема блендинга: базовые алгоритмы не видят всей обучающей выборки. Поэтому можно усложнить схему.

- Разбиваем выборку на две части

Стекинг (Stacking)

Проблема блендинга: базовые алгоритмы не видят всей обучающей выборки. Поэтому можно усложнить схему.

- Разбиваем выборку на две части
- На одной части обучаем первый базовый алгоритмы и получаем его выход на второй части

Стекинг (Stacking)

Проблема блендинга: базовые алгоритмы не видят всей обучающей выборки. Поэтому можно усложнить схему.

- Разбиваем выборку на две части
- На одной части обучаем первый базовый алгоритмы и получаем его выход на второй части
- На другой части обучаем второй базовый алгоритмы и получаем его выход на первой части

Стекинг (Stacking)

Проблема блендинга: базовые алгоритмы не видят всей обучающей выборки. Поэтому можно усложнить схему.

- Разбиваем выборку на две части
- На одной части обучаем первый базовый алгоритмы и получаем его выход на второй части
- На другой части обучаем второй базовый алгоритмы и получаем его выход на первой части
- Обучаем мета-алгоритм на выходах базовых алгоритмов

Стекинг (Stacking)

Проблема блендинга: базовые алгоритмы не видят всей обучающей выборки. Поэтому можно усложнить схему.

- Разбиваем выборку на две части
- На одной части обучаем первый базовый алгоритмы и получаем его выход на второй части
- На другой части обучаем второй базовый алгоритмы и получаем его выход на первой части
- Обучаем мета-алгоритм на выходах базовых алгоритмов
- На всей выборке обучаем третий базовый алгоритм

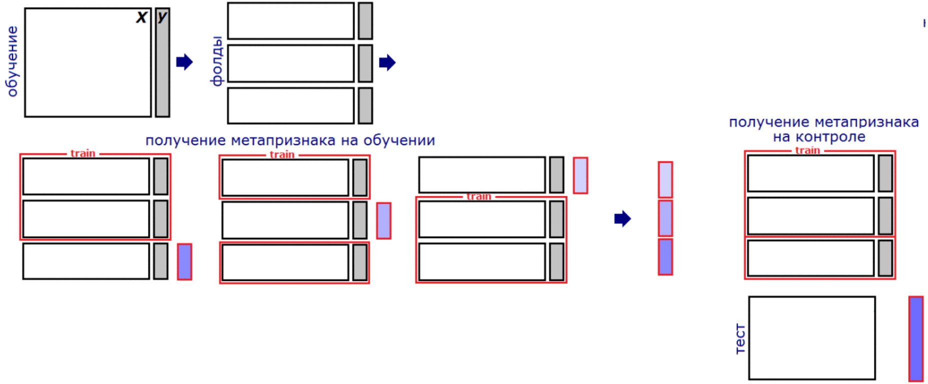
Стекинг (Stacking)

Проблема блендинга: базовые алгоритмы не видят всей обучающей выборки. Поэтому можно усложнить схему.

- Разбиваем выборку на две части
- На одной части обучаем первый базовый алгоритмы и получаем его выход на второй части
- На другой части обучаем второй базовый алгоритмы и получаем его выход на первой части
- Обучаем мета-алгоритм на выходах базовых алгоритмов
- На всей выборке обучаем третий базовый алгоритм
- На тесте получаем выход третьего алгоритма и к нему применяем мета-алгоритм

Стекинг – визуализация

Рассмотрим схему[4] стекинга:



[4] https://dyakonov.org

Замечания о стековых обобщениях

- Блендинг очень прост в реализации

Замечания о стековых обобщениях

- Блендинг очень прост в реализации
- 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
- 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
- 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
- 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
- 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)

Бутстрэп (Bootstrap)

Английская поговорка: "To pull oneself over a fence by one's bootstraps".
Русский аналог: Мюнхгаузен, вытаскивающий себя за волосы из болота.

Бутстрэп

Определение

Бутстрэп - это методика тестирования на основе случайного семплирования из выборки с возвращением.

[5] Efron, B. (1979). "Bootstrap methods: Another look at the jackknife"

Бутстрэп

Определение

Бутстрэп - это методика тестирования на основе случайного семплирования из выборки с возвращением.

- Бутстрэп[5] позволяет оценивать параметры алгоритмов (такие как смещение, разброс, доверительный интервал и т.п.) на основе семплированных выборок.

[5] Efron, B. (1979). "Bootstrap methods: Another look at the jackknife"

Бутстрэп

Определение

Бутстрэп - это методика тестирования на основе случайного семплирования из выборки с возвращением.

- Бутстрэп[5] 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)

[5] Efron, B. (1979). "Bootstrap methods: Another look at the jackknife"

О семплировании с возвращением

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 $N \to \infty$.

О семплировании с возвращением

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 $N \to \infty$.

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:
$\lim_{N\to\infty}(1 - \frac{1}{N})^N = \lim_{N\to\infty}\left((1 - \frac{1}{N})^{-N}\right)^{-1} = e^{-1}$. Ч.т.д.

О семплировании с возвращением

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 $N \to \infty$.

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:
$\lim_{N\to\infty}(1-\frac{1}{N})^N = \lim_{N\to\infty}\left((1-\frac{1}{N})^{-N}\right)^{-1} = e^{-1}$. Ч.т.д.
Замечание. Т.о. можно тестировать алгоритм на оставшихся $e^{-1} \approx 37\%$ данных.

Бэггинг

Вспомним разложение ошибки на разброс и смещение: $\sigma^2 + variance(a) + bias^2(f, a)$.
Простое усреднение T алгоритмов позволяет теоретически уменьшить разброс в T раз, при этом не влияя на смещение.

[6] Breiman L. (1994). “Bagging Predictors”.

Бэггинг

Вспомним разложение ошибки на разброс и смещение: $\sigma^2 + variance(a) + bias^2(f, a)$.
Простое усреднение T алгоритмов позволяет теоретически уменьшить разброс в T раз, при этом не влияя на смещение.
Это и есть главная идея бэггинга[6]:

[6] Breiman L. (1994). "Bagging Predictors".

Бэггинг

Вспомним разложение ошибки на разброс и смещение: $\sigma^2 + variance(a) + bias^2(f, a)$.
Простое усреднение T алгоритмов позволяет теоретически уменьшить разброс в T раз, при этом не влияя на смещение.
Это и есть главная идея бэггинга[6]:

- уменьшить разброс алгоритма,

[6] Breiman L. (1994). "Bagging Predictors".

Бэггинг

Вспомним разложение ошибки на разброс и смещение: $\sigma^2 + variance(a) + bias^2(f, a)$.
Простое усреднение T алгоритмов позволяет теоретически уменьшить разброс в T раз, при этом не влияя на смещение.
Это и есть главная идея бэггинга[6]:

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

[6] Breiman L. (1994). "Bagging Predictors".

Бэггинг

Вспомним разложение ошибки на разброс и смещение: $\sigma^2 + variance(a) + bias^2(f, a)$.
Простое усреднение T алгоритмов позволяет теоретически уменьшить разброс в T раз, при этом не влияя на смещение.
Это и есть главная идея бэггинга[6]:

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

Определение

Бэггинг (**B**ootstrap **AGG**regat**ING**) - это метод ансамблирования, основанный на:

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

[6] 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 $b_t(x)$,

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 $b_t(x)$,
3. Результат применения – усреднение (для регрессии) или голосования (для классификации).

Пример бэггинга

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

[7] Ho T. K. (1998). "The Random Subspace Method for Constructing Decision Forests"

Пример бэггинга

Оказывается, можно использовать бутстрэп-семплирование не только для обучающей выборки, но и для признаков!
Это – метод случайных подпространств[7].

[7] Ho T. K. (1998). “The Random Subspace Method for Constructing Decision Forests”

Пример бэггинга

Оказывается, можно использовать бутстрэп-семплирование не только для обучающей выборки, но и для признаков!
Это – метод случайных подпространств[7].
Т.о., случайные деревья из прошлой лекции – это объединение:

- Бэггинга для работы с выборкой,

[7] Ho T. K. (1998). "The Random Subspace Method for Constructing Decision Forests"

Пример бэггинга

Оказывается, можно использовать бутстрэп-семплирование не только для обучающей выборки, но и для признаков!
Это – метод случайных подпространств[7].
Т.о., случайные деревья из прошлой лекции – это объединение:

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

[7] Ho T. K. (1998). "The Random Subspace Method for Constructing Decision Forests"

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.

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 K-ближайших соседей).

Бустинг

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

[8] Schapire R. E. (1990). “The Strength of Weak Learnability”.

Бустинг

Попробуем теперь бороться не только с разбросом, но и со смещением.
Главная идея бустинга[8]:

[8] Schapire R. E. (1990). "The Strength of Weak Learnability".

Бустинг

Попробуем теперь бороться не только с разбросом, но и со смещением.
Главная идея бустинга[8]:

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

[8] Schapire R. E. (1990). “The Strength of Weak Learnability”.

Бустинг

Попробуем теперь бороться не только с разбросом, но и со смещением.
Главная идея бустинга[8]:

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

[8] Schapire R. E. (1990). “The Strength of Weak Learnability”.

Бустинг

Попробуем теперь бороться не только с разбросом, но и со смещением.
Главная идея бустинга[8]:

- 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.

[8] Schapire R. E. (1990). "The Strength of Weak Learnability".

Типы бустинга

Разные виды бустинга можно описать в зависимости от:

- аппроксимации функции потерь,

Типы бустинга

Разные виды бустинга можно описать в зависимости от:

- аппроксимации функции потерь,
- множества выходных значений базовых классификаторов.

Типы бустинга

Разные виды бустинга можно описать в зависимости от:

- аппроксимации 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:

- аппроксимации 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:

- аппроксимации 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 – гладкая функция от отступа $L(y_i a(x_i))$

Типы бустинга

Разные виды бустинга можно описать в зависимости от:

- аппроксимации 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 – 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 $L(y_i, a(x_i))$

Бустинг для бинарной классификации

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

Бустинг для бинарной классификации

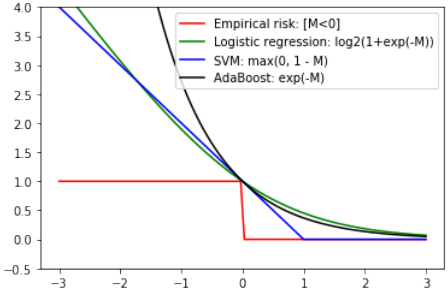
Пусть $X^m = (x_i, y_i)_{i=1}^m, y_i \in Y = \{+1, -1\}, b_t : X \to \{-1, 0, +1\}$. Значение $b_t(x) = 0$ вводится для сигнализации неопределенности в классификации (аналогия: нахождение внутри полосы для SVM).

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

- Алгоритм классификации – взвешенное голосование: $a(x) = \operatorname{sign}(\sum_{t=1}^T \alpha_t b_t(x))$,
- 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 $\alpha_1 b_1(x_i), \ldots, \alpha_{t-1} b_{t-1}(x_i)$ при добавлении $\alpha_t b_t(x_i)$,
- Использовать аппроксимированный Э.Р.

Обозначения

Аппроксимация Э.Р.:

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

Обозначения

Аппроксимация Э.Р.:

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

- Вектор весов (взвешиваем объекты) $W^m = (w_1, \dots, w_m)$: $w_i = e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)}$
- Нормировка: $\widetilde{w}_i = \frac{w_i}{\sum_{j=1}^{m} w_j}$

Обозначения

Аппроксимация Э.Р.:

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

- Вектор весов (взвешиваем объекты) $W^m = (w_1, \dots, w_m)$: $w_i = e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)}$
- Нормировка: $\widetilde{w}_i = \frac{w_i}{\sum_{j=1}^m w_j}$
- 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: $N(b; U^m) = \sum_{i=1}^m u_i[b(x) = -y_i]$
- Взвешенное число отказов от классификации: $1 - P - N$.

Основная теорема бустинга

Пусть A – достаточно богатое семейство базовых алгоритмов.

Теорема

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

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

Доказательство

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

Доказательство

Если $b \in \{-1, 0, +1\}$, то верно тождество $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$.
$\widetilde{R}_T = \sum_{i=1}^{m} e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} =$
$\sum_{i=1}^{m} w_i \left(e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]\right) =$

Доказательство

Если $b \in \{-1, 0, +1\}$, то верно тождество $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$.
$\widetilde{R}_T = \sum_{i=1}^{m} e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} =$
$\sum_{i=1}^{m} w_i \left(e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]\right) =$
$e^{-\alpha_T} \sum_{i=1}^{m} w_i[b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^{m} w_i[b_T(x_i) = -y_i] + \sum_{i=1}^{m} w_i[b_T(x_i) = 0] =$

Доказательство

Если $b \in \{-1, 0, +1\}$, то верно тождество $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$.
$\widetilde{R}_T = \sum_{i=1}^{m} e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} =$
$\sum_{i=1}^{m} w_i \left(e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]\right) =$
$e^{-\alpha_T} \sum_{i=1}^{m} w_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^{m} w_i [b_T(x_i) = -y_i] + \sum_{i=1}^{m} w_i [b_T(x_i) = 0] =$
$(e^{-\alpha_T} \sum_{i=1}^{m} \widetilde{w}_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^{m} \widetilde{w}_i [b_T(x_i) = -y_i] + \sum_{i=1}^{m} \widetilde{w}_i [b_T(x_i) = 0]) \sum_{i=1}^{m} w_i =$

Доказательство

Если $b \in \{-1, 0, +1\}$, то верно тождество $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$.
$\widetilde{R}_T = \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} =$
$\sum_{i=1}^m w_i \left(e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]\right) =$
$e^{-\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = -y_i] + \sum_{i=1}^m w_i [b_T(x_i) = 0] =$
$(e^{-\alpha_T} \sum_{i=1}^m \widetilde{w}_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m \widetilde{w}_i [b_T(x_i) = -y_i] + \sum_{i=1}^m \widetilde{w}_i [b_T(x_i) = 0]) \sum_{i=1}^m w_i =$
$(e^{-\alpha_T} P + e^{\alpha_T} N + 1 - P - N) \widetilde{R}_{T-1}$.

Доказательство

Если $b \in \{-1, 0, +1\}$, то верно тождество $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$.
$\widetilde{R}_T = \sum_{i=1}^{m} e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} =$
$\sum_{i=1}^{m} w_i \left(e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]\right) =$
$e^{-\alpha_T} \sum_{i=1}^{m} w_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^{m} w_i [b_T(x_i) = -y_i] + \sum_{i=1}^{m} w_i [b_T(x_i) = 0] =$
$(e^{-\alpha_T} \sum_{i=1}^{m} \widetilde{w}_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^{m} \widetilde{w}_i [b_T(x_i) = -y_i] + \sum_{i=1}^{m} \widetilde{w}_i [b_T(x_i) = 0]) \sum_{i=1}^{m} w_i =$
$(e^{-\alpha_T} P + e^{\alpha_T} N + 1 - P - N)\widetilde{R}_{T-1}$.
$\widetilde{R}_T \to \min_{\alpha_T, b_T} \Rightarrow \frac{\partial \widetilde{R}_T}{\partial \alpha_T} = 0$.

Доказательство

Если $b \in \{-1, 0, +1\}$, то верно тождество $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$.
$\widetilde{R}_T = \sum_{i=1}^m e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} =$
$\sum_{i=1}^m w_i \left(e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]\right) =$
$e^{-\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m w_i [b_T(x_i) = -y_i] + \sum_{i=1}^m w_i [b_T(x_i) = 0] =$
$(e^{-\alpha_T} \sum_{i=1}^m \widetilde{w}_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m \widetilde{w}_i [b_T(x_i) = -y_i] + \sum_{i=1}^m \widetilde{w}_i [b_T(x_i) = 0]) \sum_{i=1}^m w_i =$
$(e^{-\alpha_T} P + e^{\alpha_T} N + 1 - P - N) \widetilde{R}_{T-1}$.
$\widetilde{R}_T \to \min_{\alpha_T, b_T} \Rightarrow \frac{\partial \widetilde{R}_T}{\partial \alpha_T} = 0$.
$\frac{\partial \widetilde{R}_T}{\partial \alpha_T} = (-e^{-\alpha_T} P + e^{\alpha_T} N) \widetilde{R}_{T-1} = 0 \Rightarrow e^{-\alpha_T} P = e^{\alpha_T} N \Rightarrow e^{2\alpha_T} = \frac{P}{N} \Rightarrow \alpha_T = \frac{1}{2} \ln \frac{P(b_T; \widetilde{W^m})}{N(b_T; \widetilde{W^m})}$.

Доказательство

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

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

$\sum_{i=1}^m w_i \left(e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]\right) =$

$e^{-\alpha_T} \sum_{i=1}^m w_i[b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m w_i[b_T(x_i) = -y_i] + \sum_{i=1}^m w_i[b_T(x_i) = 0] =$

$\left(e^{-\alpha_T} \sum_{i=1}^m \widetilde{w}_i[b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^m \widetilde{w}_i[b_T(x_i) = -y_i] + \sum_{i=1}^m \widetilde{w}_i[b_T(x_i) = 0]\right) \sum_{i=1}^m w_i =$

$(e^{-\alpha_T} P + e^{\alpha_T} N + 1 - P - N)\widetilde{R}_{T-1}$.

$\widetilde{R}_T \to \min_{\alpha_T, b_T} \Rightarrow \frac{\partial \widetilde{R}_T}{\partial \alpha_T} = 0$.

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

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

Доказательство

Если $b \in \{-1, 0, +1\}$, то верно тождество $e^{-\alpha b} = e^{-\alpha}[b = 1] + e^{\alpha}[b = -1] + [b = 0]$.
$\widetilde{R}_T = \sum_{i=1}^{m} e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)} e^{-y_i \alpha_T b_T(x_i)} =$
$\sum_{i=1}^{m} w_i \left(e^{-\alpha_T}[b_T(x_i) = y_i] + e^{\alpha_T}[b_T(x_i) = -y_i] + [b_T(x_i) = 0]\right) =$
$e^{-\alpha_T} \sum_{i=1}^{m} w_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^{m} w_i [b_T(x_i) = -y_i] + \sum_{i=1}^{m} w_i [b_T(x_i) = 0] =$
$(e^{-\alpha_T} \sum_{i=1}^{m} \widetilde{w}_i [b_T(x_i) = y_i] + e^{\alpha_T} \sum_{i=1}^{m} \widetilde{w}_i [b_T(x_i) = -y_i] + \sum_{i=1}^{m} \widetilde{w}_i [b_T(x_i) = 0]) \sum_{i=1}^{m} w_i =$
$(e^{-\alpha_T} P + e^{\alpha_T} N + 1 - P - N)\widetilde{R}_{T-1}$.
$\widetilde{R}_T \to \min_{\alpha_T, b_T} \Rightarrow \frac{\partial \widetilde{R}_T}{\partial \alpha_T} = 0$.
$\frac{\partial \widetilde{R}_T}{\partial \alpha_T} = (-e^{-\alpha_T} P + e^{\alpha_T} N)\widetilde{R}_{T-1} = 0 \Rightarrow e^{-\alpha_T} P = e^{\alpha_T} N \Rightarrow e^{2\alpha_T} = \frac{P}{N} \Rightarrow \alpha_T = \frac{1}{2} \ln \frac{P(b_T; \widetilde{W^m})}{N(b_T; \widetilde{W^m})}$.
Для поиска $b_T(x)$ подставим найденное α_T в формулу для $\widetilde{R}_T$:
$\widetilde{R}_T = (\sqrt{\frac{N}{P}} P + \sqrt{\frac{P}{N}} N + 1 - P - N)\widetilde{R}_{T-1} = (1 - (P - 2\sqrt{PN} + N))\widetilde{R}_{T-1} =$
$(1 - (\sqrt{P} - \sqrt{N})^2)\widetilde{R}_{T-1} \to \min_{b_T} \Rightarrow b_T = \arg\max_{b \in A} \sqrt{P(b; \widetilde{W^m})} - \sqrt{N(b; \widetilde{W^m})}$ (т.к. $P > N$). 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 $\beta > 0$, 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 $\beta > 0$, то за конечное число шагов будет построен алгоритм, не допускающий ни единой ошибки на обучающем множестве.

Доказательство.

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

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 $\beta > 0$, то за конечное число шагов будет построен алгоритм, не допускающий ни единой ошибки на обучающем множестве.

Доказательство.
$R_T \leq \widetilde{R}_T = (1 - (\sqrt{P} - \sqrt{N})^2)\widetilde{R}_{T-1} \leq (1 - \beta^2)\widetilde{R}_{T-1} \leq \cdots \leq (1 - \beta^2)^{T-1}\widetilde{R}_1$.
Для любого $0 < \beta \leq 1$ 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 $\beta>0$, то за конечное число шагов будет построен алгоритм, не допускающий ни единой ошибки на обучающем множестве.

Доказательство.
$R_T\leq\widetilde{R}_T=(1-(\sqrt{P}-\sqrt{N})^2)\widetilde{R}_{T-1}\leq(1-\beta^2)\widetilde{R}_{T-1}\leq\cdots\leq(1-\beta^2)^{T-1}\widetilde{R}_1$.
Для любого $0<\beta\leq 1$ будет существовать такое T, что $R_T<1$.
Э.Р. – это число ошибок на обучающем множестве (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: $b_t : X \to \{-1, +1\}$. Тогда $P + N = 1$.

[9] 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: $b_t : X \to \{-1, +1\}$. Тогда $P + N = 1$.
В этом случае конкретный алгоритм бустинга называется AdaBoost[9] (**Ada**ptive **Boost**ing).

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:

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

[9] Freund Y. and Schapire R.E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting"

Алгоритм AdaBoost

Алгоритм

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

Алгоритм AdaBoost

Алгоритм

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

Для $t = 1, \ldots, T$

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

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 $\alpha_t = \frac{1}{2} \ln \frac{1 - N(b_t; \widetilde{W^m})}{N(b_t; \widetilde{W^m})},$

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

Алгоритм AdaBoost

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 $w_i = \frac{w_i}{\sum_{j=1}^m w_j}, i = 1, \dots, m.$

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

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

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

- 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 накапливается у тех объектов, которые чаще оказывались трудными для предыдущих алгоритмов.

О значении алгоритма AdaBoost

- В 2003 году создатели алгоритма AdaBoost Фройнд и Шапире получили премию Гёделя (вручается за выдающиеся труды по логике и теоретической информатике),

[10] Viola and Jones (2001). “Robust Real-time Object Detection”

О значении алгоритма AdaBoost

- В 2003 году создатели алгоритма AdaBoost Фройнд и Шапире получили премию Гёделя (вручается за выдающиеся труды по логике и теоретической информатике),
- В 2001 году[10] был создан алгоритм, позволяющий обнаруживать объекты на изображениях (прежде всего человеческое лицо) в реальном времени.

[10] Viola and Jones (2001). "Robust Real-time Object Detection"

О значении алгоритма AdaBoost

- В 2003 году создатели алгоритма AdaBoost Фройнд и Шапире получили премию Гёделя (вручается за выдающиеся труды по логике и теоретической информатике),
- В 2001 году[10] был создан алгоритм, позволяющий обнаруживать объекты на изображениях (прежде всего человеческое лицо) в реальном времени.
  - Математической основой послужил модифицированный AdaBoost,

[10] Viola and Jones (2001). "Robust Real-time Object Detection"

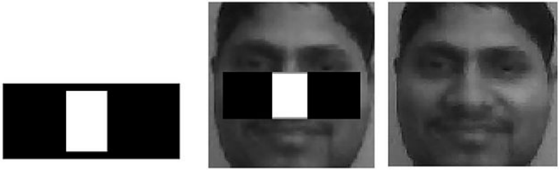
О значении алгоритма AdaBoost

- В 2003 году создатели алгоритма AdaBoost Фройнд и Шапире получили премию Гёделя (вручается за выдающиеся труды по логике и теоретической информатике),
- В 2001 году[10] был создан алгоритм, позволяющий обнаруживать объекты на изображениях (прежде всего человеческое лицо) в реальном времени.
  - Математической основой послужил модифицированный AdaBoost,
  - Этот алгоритм детекции был лидирующим для детекции лиц на протяжении более 10 лет (до начала широкого применения сверточных нейросетей).

[10] Viola and Jones (2001). "Robust Real-time Object Detection"

О значении алгоритма AdaBoost

- В 2003 году создатели алгоритма AdaBoost Фройнд и Шапире получили премию Гёделя (вручается за выдающиеся труды по логике и теоретической информатике),
- В 2001 году[10] был создан алгоритм, позволяющий обнаруживать объекты на изображениях (прежде всего человеческое лицо) в реальном времени.
  - Математической основой послужил модифицированный AdaBoost,
  - Этот алгоритм детекции был лидирующим для детекции лиц на протяжении более 10 лет (до начала широкого применения сверточных нейросетей).



[10] Viola and Jones (2001). "Robust Real-time Object Detection"

Источники

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