Градиентный бустинг

Методы анализа данных

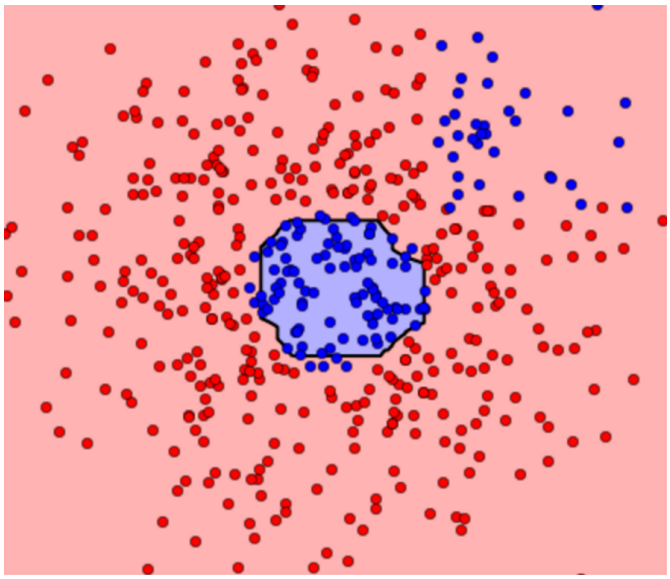
Лекция 8

Москва, МФТИ, 2020

Емчинов Александр Владимирович

Недостатки случайного леса

- Обучение глубоких деревьев требует очень много вычислительных ресурсов, особенно в случае большой выборки или большого числа признаков.
- Если ограничить глубину решающих деревьев в случайном лесе, то они уже не смогут улавливать сложные закономерности в данных. Это приведет к тому, что сдвиг будет слишком большим.



Недостатки случайного леса

Процесс построения деревьев является ненаправленным: каждое следующее дерево в композиции никак не зависит от предыдущих. Из-за этого для решения сложных задач необходимо огромное количество деревьев.

Бустинг: основная идея

Бустинг — это подход к построению композиций, в рамках которого:

- Базовые алгоритмы строятся последовательно, один за другим.
- Каждый следующий алгоритм строится таким образом, чтобы исправлять ошибки уже построенной композиции.

Бустинг на примере задачи регрессии

Дана задача регрессии, в качестве ошибки используется среднеквадратичная ошибка:

$$MSE(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} (a(x_i) - y_i)^2 .$$

Необходимо обучить неглубокое решающее дерево:

$$b_1(x) = \text{argmin}_b \frac{1}{\ell} \sum_{i=1}^{\ell} (b(x_i) - y_i)^2 .$$

Задача минимизации квадратичной ошибки решается градиентным спуском.

Бустинг на примере задачи регрессии

Второй алгоритм должен быть обучен таким образом, чтобы композиция первого и второго алгоритмов:

$$b_1(x_i) + b_2(x_i)$$

имела наименьшую из возможных ошибку на обучающей выборке:

$$b_2(x) = \operatorname{argmin}_b \frac{1}{\ell} \sum_{i=1}^{\ell} (b_1(x_i) + b(x_i) - y_i)^2 = \operatorname{argmin}_b \frac{1}{\ell} \sum_{i=1}^{\ell} (b(x_i) - (y_i - b_1(x_i)))^2 .$$

Алгоритм $b_2(x)$ улучшает качество работы алгоритма $b_1(x)$.

Бустинг на примере задачи регрессии

На N шаге очередной алгоритм $b_N(X)$ будет определяться следующим образом:

$$b_N(x) = \text{argmin}_b \frac{1}{\ell} \sum_{i=1}^{\ell} \left(b(x_i) - \left(y_i - \sum_{n=1}^{N-1} b_n(x_i) \right) \right)^2 .$$

Процесс продолжается до тех пор, пока ошибка композиции $b_1(x) + \ldots + b_N(x)$ не будет устраивать.

Градиентный бустинг

В градиентном бустинге строящаяся композиция

$$a_N(x) = \sum_{n=1}^{N} b_n(x)$$

является суммой базовых алгоритмов $b_i(x)$.

Градиентный бустинг

Пусть задана функция потерь $L(y, z)$, где y — истинный ответ, z — прогноз алгоритма на некотором объекте. Примерами возможных функций потерь являются:

- среднеквадратичная ошибка (в задаче регрессии):

$$L(y, z) = (y - z)^2$$

- логистическая функция потерь (в задаче классификации):

$$L(y, z) = \log(1 + \exp(-yz)).$$

Инициализация

В начале нужно ее инициализировать построить первый базовый алгоритм $b_0(x)$. Можно использовать:

- алгоритм $b_0(x) = 0$, который всегда возвращает ноль (в задаче регрессии);
- более сложный $b_0(x) = \frac{1}{l}\sum_{i=1}^{l} y_i$, который возвращает средний по всем элементам обучающей выборки истинный ответ (в задаче регрессии);
- алгоритм $b_0(x) = \text{argmax}_{y \in Y} \sum_{i=1}^{l} [y_i = y]$, который всегда возвращает метку самого распространенного класса в обучающей выборке (в задаче классификации).

Обучение базовых алгоритмов

Пусть к некоторому моменту обучены $N-1$ алгоритмов $b_1(x), \ldots, b_{N-1}(x)$ композиция имеет вид:

$$a_{N-1}(x) = \sum_{n=1}^{N-1} b_n(x).$$

К композиции добавляется алгоритм $b_N(x)$. Этот алгоритм обучается так:

$$\sum_{i=1}^{\ell} L\left(y_i, a_{N-1}(x_i) + b(x_i)\right) \to \min_b.$$

Обучение базовых алгоритмов

Определить, какие значения $s_1, ..., s_l$ должен принимать алгоритм $b_N(x_i) = s_i$ на объектах обучающей выборки, чтобы ошибка на обучающей выборке была минимальной:

$$F(s) = \sum_{i=1}^{\ell} L\left(y_i, a_{N-1}(x_i) + s_i\right) \to \min_s,$$

Обучение базовых алгоритмов

Необходимо найти вектор сдвигов s, который будет минимизировать функцию $F(s)$. Направление наискорейшего убывания функции задается направлением антиградиента

$$s = -\nabla F = \begin{pmatrix} -L'_z(y_1, a_{N-1}(x_1)), \\ \dots \\ -L'_z(y_\ell, a_{N-1}(x_\ell)) \end{pmatrix}.$$

Обучение базовых алгоритмов

Обучение $b_N(x)$, таким образом, представляет собой задачу обучения на размеченных данных, в которой $(x_i, s_i)_{i=1}^{l}$ — обучающая выборка, и используется, например, квадратичная функция ошибки:

$$b_N(x) = \operatorname{argmin}_b \frac{1}{\ell} \sum_{i=1}^{\ell} (b(x_i) - s_i)^2 .$$

Описание алгоритма градиентного бустинга

- **Инициализация:** инициализация композиции $a_0(x) = b_0(x)$, то есть построение простого алгоритма b_0 .
- **Шаг итерации:**
  - **Вычисляется вектор сдвига**

$$s = -\nabla F = \begin{pmatrix} -L'_z(y_1, a_{N-1}(x_1)), \\ \dots \\ -L'_z(y_\ell, a_{N-1}(x_\ell)) \end{pmatrix}.$$

  - **Строится алгоритм**

$$b_n(x) = \operatorname{argmin}_b \frac{1}{\ell} \sum_{i=1}^{\ell} (b(x_i) - s_i)^2,$$

параметры которого подбираются таким образом, что его значения на элементах обучающей выборки были как можно ближе к вычисленному вектору оптимального сдвига s.

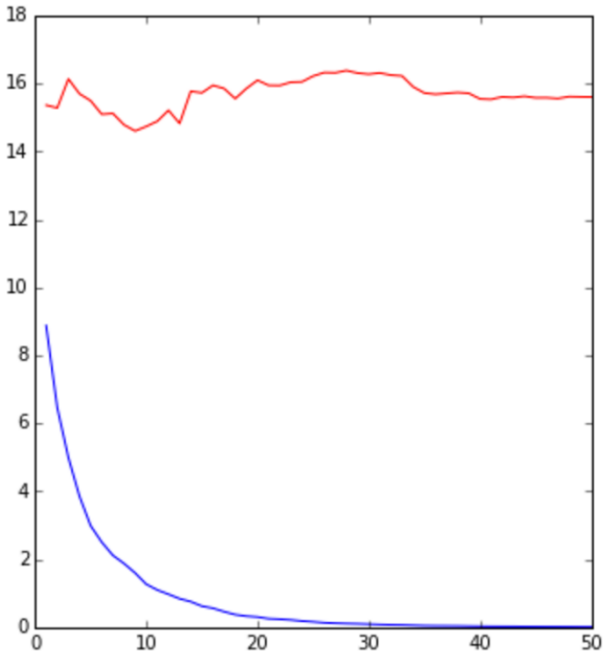
Описание алгоритма градиентного бустинга

- **Шаг итерации:**
  - **Алгоритм $b_n(x)$ добавляется в композицию**

$$a_n(x) = \sum_{m=1}^{n} b_m(x)$$

- Если не выполнен критерий останова (об этом будет рассказано далее), то выполнить еще один шаг итерации. Если критерий останова выполнен, остановить итерационный процесс.

Проблема переобучения градиентного бустинга



По мере увеличения числа деревьев ошибка на обучающей выборке постепенно уходит в 0.
Это связано с тем, что базовый алгоритм пытается приблизить вектор антиградиента на обучающей выборке.

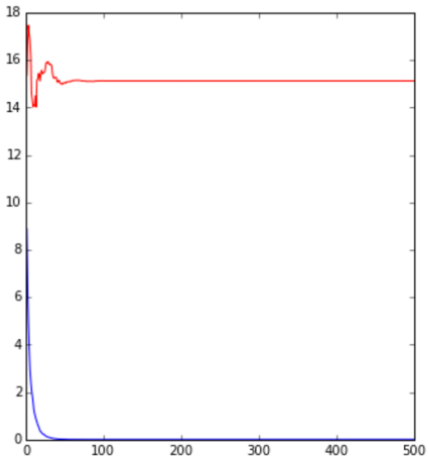
Сокращение размера шага

Чтобы решить эту проблему, нужно чуть-чуть смещаться в сторону вектора:

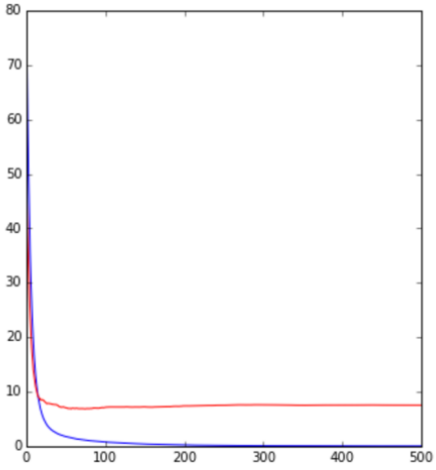
$$a_N(x) = a_{N-1}(x) + \eta b_N(x),$$

где $\eta \in (0, 1]$ — длина шага.

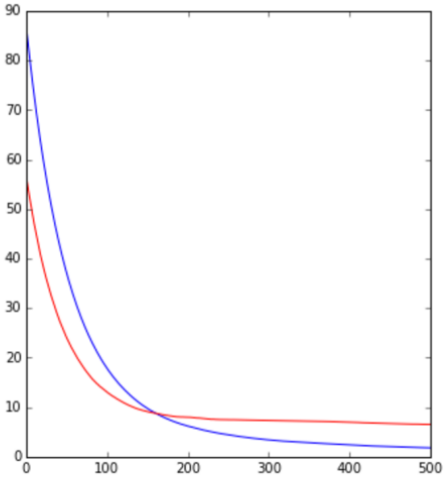
Сокращение размера шага



(a) Случай $\eta = 1$.



(b) Случай $\eta = 0.1$.



(c) Случай $\eta = 0.01$.

Сокращение размера шага

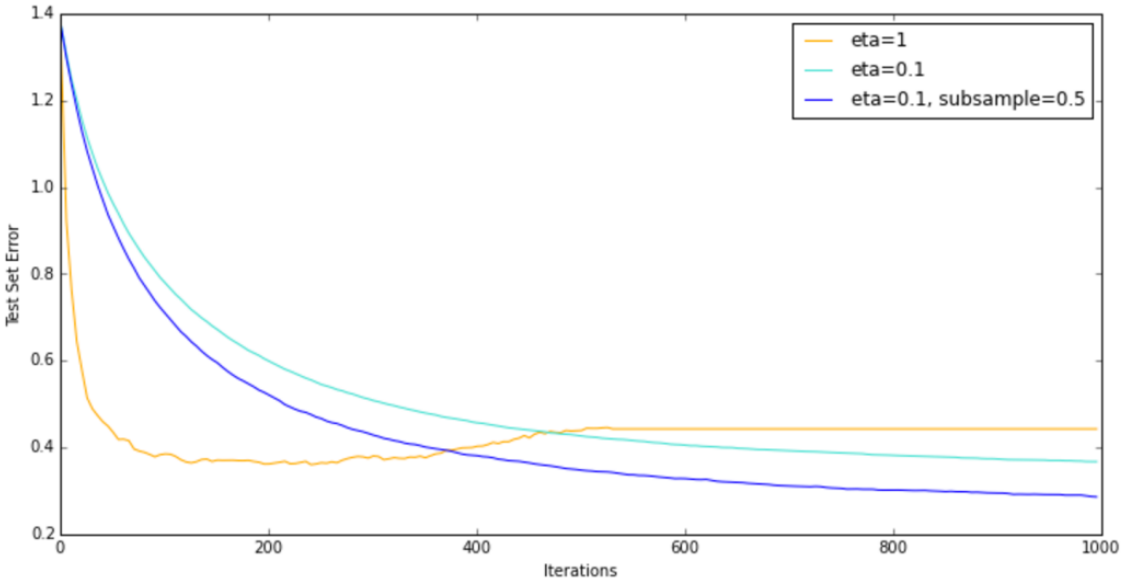
- Чем меньше размер шага, тем больше нужно базовых алгоритмов, чтобы достичь хорошего качества, и тем больше времени занимает процесс.
- Чем меньше размер шага, тем лучшего качества можно достичь.

Подбор гиперпараметров

1. Зафиксировать размер шага *eta* и подбирать число итераций *N*
2. Зафиксировать число итераций *N* и подбирать размер шага η

Стохастический градиентный бустинг

Бэггинг — еще один подход к борьбе с переобучением градиентного бустинга, который заключается в том, что каждый базовый алгоритм обучается не на всей выборке, а на некоторой ее случайной подвыборке.



Стохастический градиентный бустинг

- **Оранжевая кривая:** без применения сокращения шага и без бэггинга. Кривая обладает явно выраженным минимумом, после чего алгоритм начинает переобучаться.
- **Бирюзовая кривая:** при использовании сокращения шага с параметром $\eta = 0.1$ без бэггинга. В этом случае требуется больше итерации, чтобы достичь хорошего качества, но и при этом достигается более низкое значение ошибки. То есть метод сокращения шага действительно работает.
- **Синяя кривая:** при использовании сокращения шага с параметром $\eta = 0.1$ и бэггинга с размером подвыборки равным половине обучающей выборки. Форма кривой совпадает совпадает с формой бирюзовой кривой, но при этом за такое же количество итераций алгоритм достигает более низкой ошибки.

Градиентный бустинг для регрессии

- функционал ошибки в регрессии — это среднеквадратичная ошибка:

$$MSE(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} (a(x_i) - y_i)^2 .$$

- функция потерь, которая измеряет ошибку для одного объекта:

$$L(y, z) = (z - y)^2, \qquad L'_z(y, z) = 2(z - y),$$

Градиентный бустинг для регрессии

Вектор сдвигов

$$s = \begin{pmatrix} -2\Big(a_{N-1}(x_1) - y_1\Big) \\ \dots \\ -2\Big(a_{N-1}(x_\ell) - y_\ell\Big) \end{pmatrix}.$$

Градиентный бустинг для классификации

В задаче бинарной классификации ($\mathbb{Y} = 1, +1$) выбором для функции потерь является логистическая функция потерь:

$$\sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i a(x_i)\right)\right),$$

Если $a(x) > 0$, классификатор относит объект x к классу $+1$, а при $a(x) \leq 0$ – к классу -1.

$$L(y, z) = \log(1 + \exp(-yz)), \qquad L'_z(y, z) = -\frac{y}{1 + \exp(yz)}.$$

Градиентный бустинг для классификации

Вектор сдвигов s

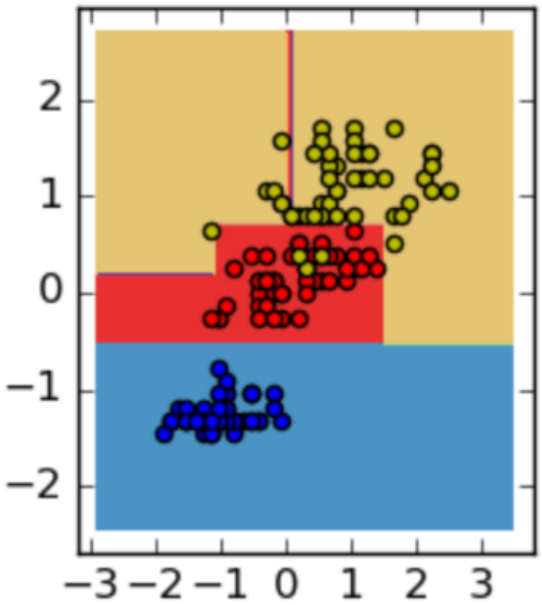
$$s = \begin{pmatrix} \frac{y_1}{1+\exp(y_1 a_{N-1}(x_1))} \\ \dots \\ \frac{y_\ell}{1+\exp(y_\ell a_{N-1}(x_\ell))} \end{pmatrix}.$$

После того, как вычислен алгоритм a N (x), можно оценить вероятности принадлежности объекта x к каждому из классов:

$$P(y=1|x) = \frac{1}{1+\exp\left(-a_N(x)\right)}, \qquad P(y=-1|x) = \frac{1}{1+\exp\left(a_N(x)\right)}.$$

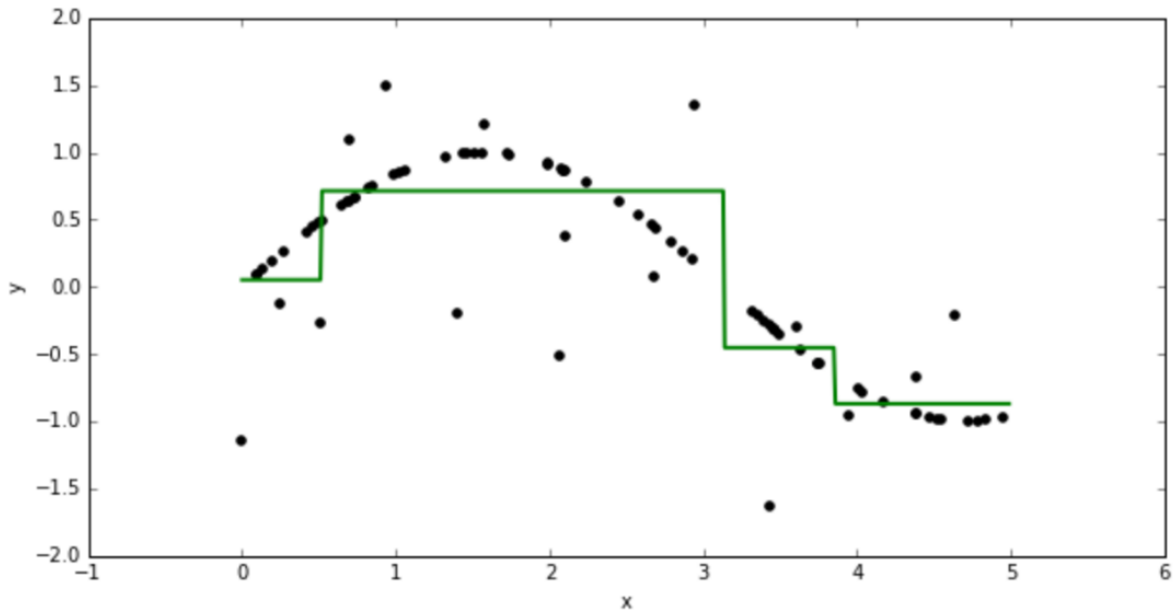
Градиентный бустинг для решающих деревьев

Поверхности, которые восстанавливают решающие деревья в задаче классификации



Поверхности, которые восстанавливают решающие деревья

В задаче регрессии



Поверхности восстанавливающие решающие деревья

Решающее дерево, таким образом, разбивает все пространство объектов на J областей:

$$R_1, R_2, ..., R_J,$$

в каждой из которых дерево возвращает постоянное предсказание. Пусть b_j — предсказание дерева в области R_j , тогда решающее дерево $b(x)$ можно записать в следующем виде:

$$b(x) = \sum_{j=1}^{J} [x \in R_j] \ b_j.$$

Градиентный бустинг для решающих деревьев

В градиентном бустинге новый базовый алгоритм b N прибавляется к уже построенной композиции:

$$a_N(x) = a_{N-1}(x) + b_N(x)$$

Если базовые алгоритмы — это решающие деревья

$$b_N(x) = \sum_{j=1}^{J} [x \in R_{Nj}] b_{Nj},$$

Тогда новая композиция a_N

$$a_N(x) = a_{N1}(x) + \sum_{j=1}^{J} [x \in R_{Nj}] b_{Nj}$$

Градиентный бустинг для решающих деревьев

Можно подобрать каждый прогноз b_{N_j} , где N — номер дерева, j — номер листа в этом дереве, таким образом, чтобы он был оптимальным с точки зрения исходной функции потерь:

$$\sum_{i=1}^{\ell} L\left(y_i, a_{N-1}(x) + \sum_{j=1}^{J} [x \in R_{Nj}]\, b_{Nj}\right) \to \min_{b_1,\ldots,b_J}$$

Можно показать, что данная задача распадается на J подзадач:

$$b_{Nj} = \operatorname{argmin}_{\gamma \in R} \sum_{x_i \in \mathbb{R}_j} L\left(y_i, a_{N-1}(x) + \gamma\right).$$