KARPOV.COURSES >>>
КОНСПЕКТ

Конспект > 20 урок > Градиентный бустинг. Biase-variance tradeoff

>Бустинг

До этого в композициях базовые алгоритмы обучались независимо. В этом есть свой плюс: так как весь процесс можно сделать параллельным.

Идея Бустинга: а что, если базовые алгоритмы строить не независимо, а последовательно? Чтобы каждая последующая модель исправляла ошибки предыдущей.

Пусть решаем задачу обучения с учителем и построили какой-то изначальный алгоритм:

$$b_1(x) = argmin \sum_{n}^{i} (L(y_i, b_1(x_i))$$

Можно замерить ошибки такого алгоритма на каждом объекте i:

$$s_i^{(1)} = y_i - b_1(x_i)$$

Следующий алгоритм должен исправлять ошибки первого. Сделаем ошибки базового алгоритма новым таргетом для следующего алгоритма. т.е обучим еще одну модель на сдвигах $s_i(1)$

$$b_2(x) = argmin \sum_{n}^{i} (L(s_i^{(1)}, b_2(x_i))$$

Как получить финальную композицию?

Чтобы получить финальный прогноз, нужно просуммировать результаты всех моделей:

$$a^2(x) = \sum_2^i (b_i(x)) = b_1(x) + b_2(x)$$

На практике двух моделей недостаточно, поэтому новые модели добавляются по такому же принципу.

Формула в общем виде выглядит следующим образом:

$$b_N = argmin \sum_i^n L\left(s_i^{(N-1)}, \ b_N(x_i)\right) = argmin \sum_i^n L\left(y_i - \sum_j^{N-1} b_j(x_i), \ b_N(x_i)\right)$$

Графически бустинг можно определить следующим образом:

>Градиентный бустинг

Пусть мы оптимизируем какую-то метрику с функцией потерь L. Ансамблевая модель - это сумма базовых моделей с некоторыми весами.

Давайте договоримся, что теперь, когда строим градиентный бустинг, мы не просто суммируем модели, а еще умножаем каждую модель на какую-то константу.

$$a^{(N)}(x) = \sum_i^N y_i * b_i(x)$$

Чтобы понять, как работает градиентный бустинг, давайте пойдем от обратного:

Пусть мы построили композицию $a^{N-1}(x)$ из N - 1 базовых алгоритмов. Как мы будем выбирать следующую модель?

Обычно мы минимизируем сумму Loss по каждому объекту. После того, как мы добавили следующую модель, какой прогноз будет давать композиция?

$$\sum_i^n L\left(y_i,\ a^{(N-1)}(x_i) + \gamma_N \cdot b_N(x_i)\right) \to \min_{b_N, \gamma_N}$$

Давайте посмотрим на формулу и подумаем, какие значения должна возвращать новая базовая модель.

Пусть это какие-то числа s_i, но какими они должны быть?

$$\sum_i^n L\left(y_i,\ a^{(N-1)}(x_i) + s_i\right) \to \min_{s_1, \dots, s_n}$$

Вообще, с добавлением нового алгоритма мы хотим как можно сильнее минимизировать функционал выше, то есть сделать шаг в сторону антиградиента.

Тогда желаемые s_i:

$$s_i = -\nabla_z \sum_i^n L(y_i, z_i) \Big|_{z_i = a^{(N-1)}(x_i)}$$

Тогда давайте подбирать $b_N(x)$ по следующей схеме:

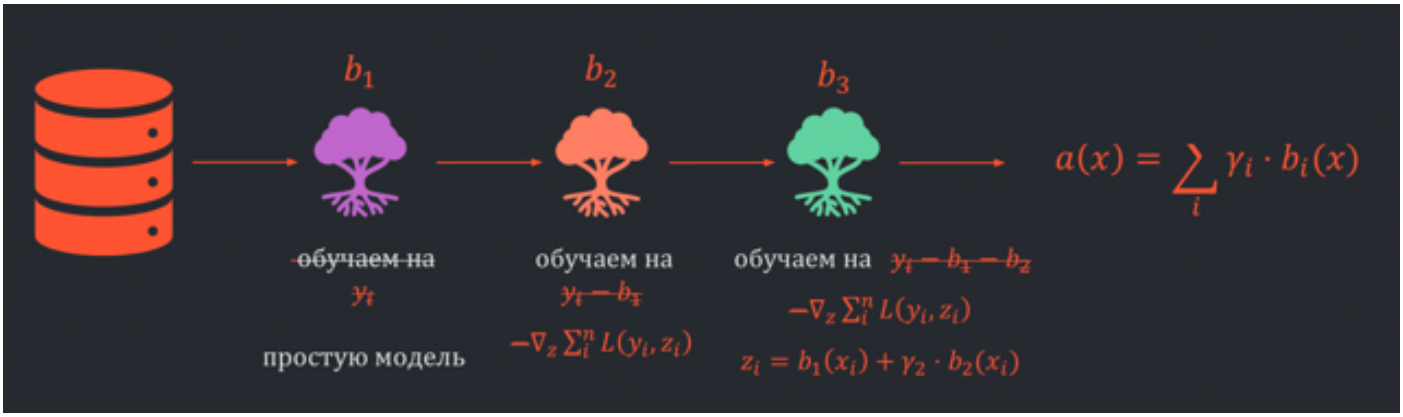
$$b_N(x) = argmin \sum_i^n (b_N(x_i) - s_i)^2$$

Заметим, что здесь используем квадратичную функцию потерь. Можно использовать и другие, но, как правило, данной функции потерь вполне достаточно.

После того, как новый алгоритм $b_N(x)$ найден, осталось найти вес этой базовой модели. Здесь мы уже в явном виде оптимизируем изначальный функционал ошибки по единственному параметру y_N, что будет легко сделать. После всех действий добавляем новую модель к ансаблю:

$$a_N(x) = \gamma_N \cdot b_N(x) + \sum_i^{N-1} \gamma_i \cdot b_i(x)$$

Графически градиентный бустинг можно определить следующим образом:



>Bias-variance tradeoff

Мы разобрали несколько подходов ансамблирования. Осталось понять, как их сравнивать между собой.

Мы умеем строить модели из различных семейств. Для разных выборок алгоритмы разных семейств могут или побеждать друг друга, или друг другу проигрывать.

Можно выделить три фактора, которые формируют ошибки на наших алгоритмах:

1. Сложность распределения данных p(x,y)
2. Сходство модели с истинной зависимостью
3. Сила и богатство семейства, из которого выбираются алгоритмы

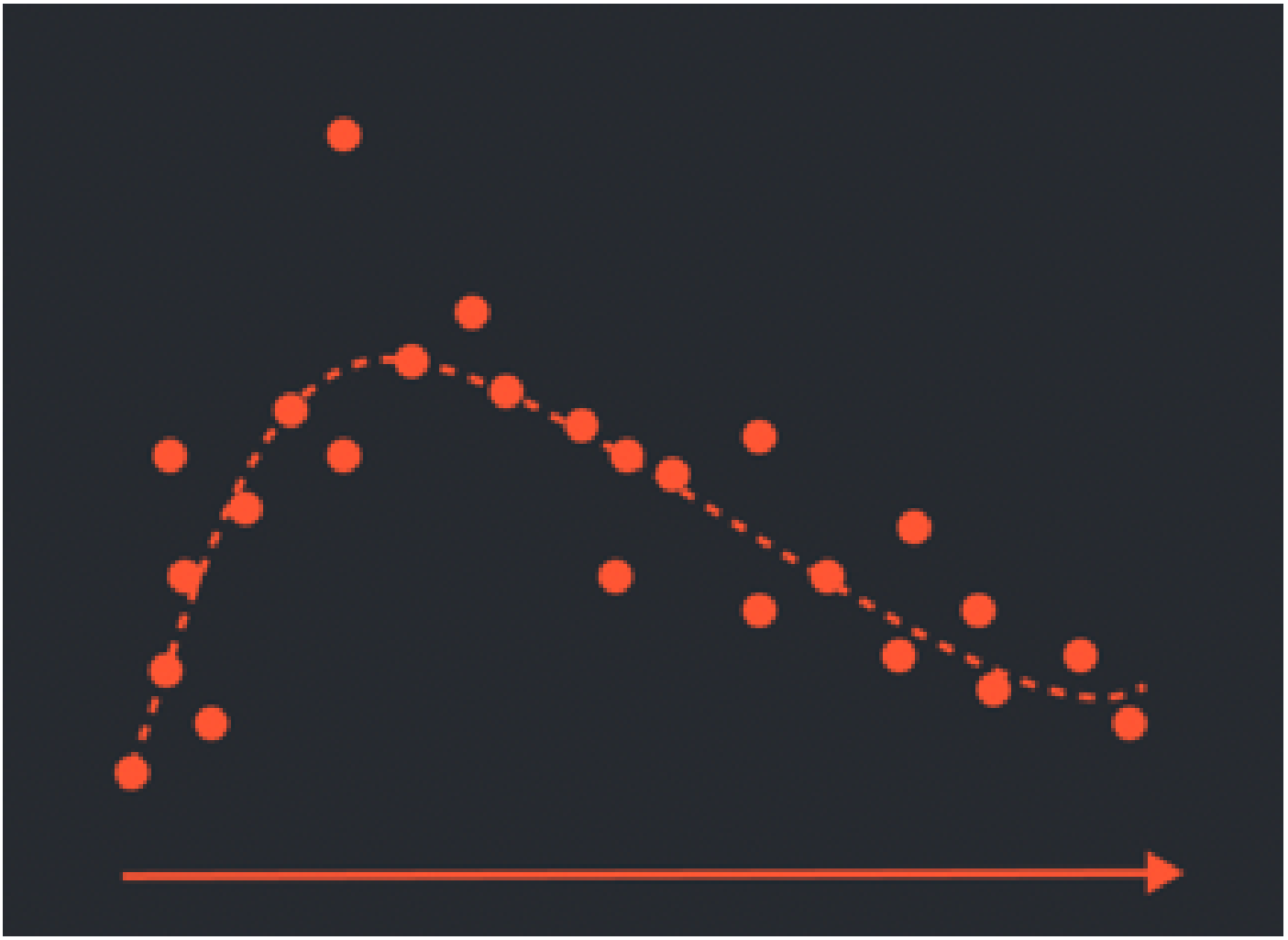
Давайте попробуем формально описать типы этих ошибок. Пусть решаем задачу регрессии минимизируя MSE. Пусть так же есть некоторый метод обучения \pi. Для этого метода можно расписать MSE ошибку

$$L(\pi) = E_{x,y}\left[\left(y - E\left[y|x\right]\right)^2\right] + E_x\left[\left(E_x[\pi(X)] - E[y|x]\right)^2\right] + E_x\left[E_X\left[\left(\pi(X) - E_X[\pi(X)]\right)^2\right]\right]$$

Первое слагаемое выражения называется шумом, второе - смещением, третье - разбросом.

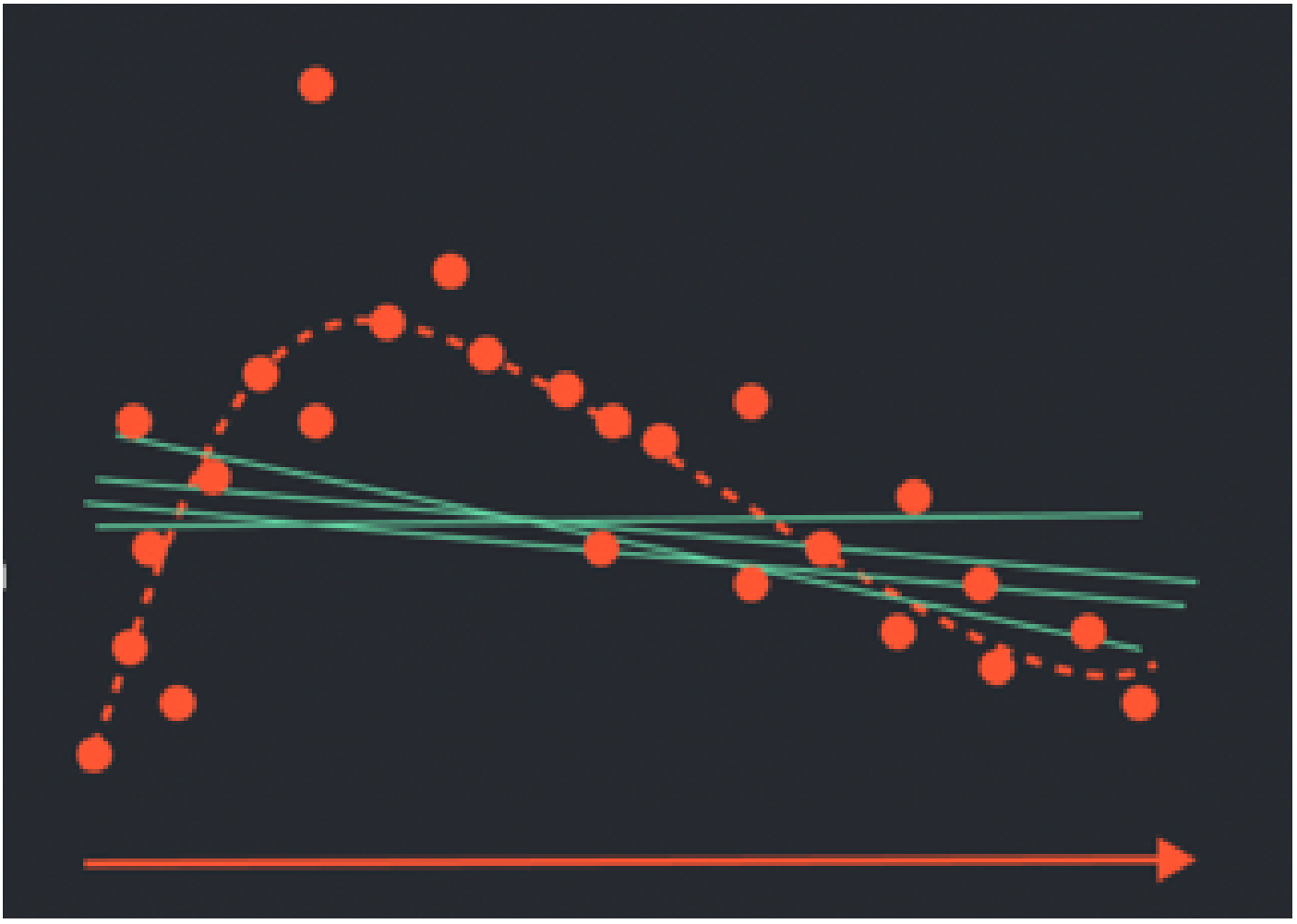
Рассмотрим первый показатель: шум

Шум - ошибка самой лучшей модели, которую мы никак не можем контролировать. Даже если мы найдем идеальную модель, всегда будут неточности из-за шума. Например:



Рассмотрим смещение:

Давайте проделаем несколько раз следующую процедуру: сгенерируем подвыборку X^{k}. Обучим модель на этой выборке. Замерим на каждом объекте ошибку алгоритма, а затем усредним по всем полученным моделям.



Смещение оценивает следующий фактор: насколько хорошо в среднем мы можем приблизить лучшую модель, то есть оценить истинную зависимость. Иными словами, смещение оценивает мощность алгоритма.

Рассмотрим разброс:

Проделаем несколько раз следующую процедуру: сгенерируем подвыборку X^{k}. Обучим модель на этой выборке. Затем посчитаем выборочную дисперсию прогнозов всех моделей по каждому объекту.

Разброс помогает оценить, как сильно меняются прогнозы модели в зависимости от полученной выборки. Иными словами разброс оценивает чувствительность построенных алгоритмов.