Градиентный бустинг

Материалы: Афанасьев Глеб, Авдеев Роман. Конспект: Морозов Иван, Ахтырченко Михаил

1 Вводная часть

Имеется некоторая модель. Необходимо её обучить, чтобы она давала хорошие предсказания. На рис. 1 по горизонтальной оси отложена сложность модели N (количество параметров модели). По вертикальной оси отложена ошибка *loss*.

На обучающей выборке при увеличении N ошибка уменьшается. Если модель очень сложная, ошибку можно свести к нулю. Это значит, что модель запомнила всю обучающую выборку и выдаёт на ней ту целевую метку, которая ей присуща.

На валидации ошибка ведёт себя иначе. Сначала она падает, но в некоторый момент N' модель начинает переобучаться, и ошибка начинает расти.

Если обучение остановилось при значении N, меньшем N', то говорят, что модель *недообучена*; если значение N оказалось больше N', то *переобучена* (хорошо предсказывает обучающую выборку, но на валидации выдаёт большую ошибку). Нужно определить значение N'.

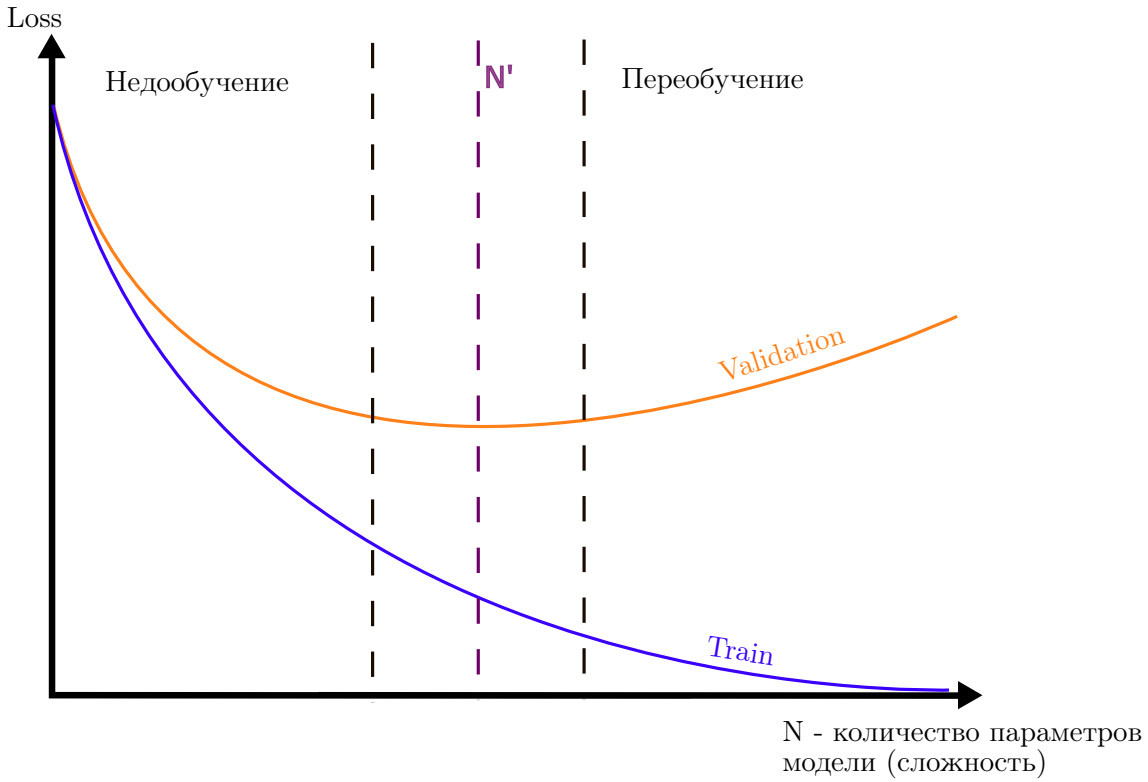

Рис. 1: Зависимость ошибки модели на обучающей и валидационной выборке от сложности модели.

Необходимо более формально определить понятия «переобучение» и «недообучение».

2 Bias-Variance decomposition

Предполагается взаимосвязь между целевой меткой y и переменной x: $y = f(x) + \varepsilon$, где ε - шум. Шум предсказать никак нельзя, нужно предсказать функцию $f(x)$.

Пусть имеется модель $\hat{f}(x)$, которая была обучена на обучающей выборке. Нужно оценить, насколько хороша эта модель: степень её переобученности/недообученности. Рассматривается квадратичная ошибка (или среднеквадратичная, поскольку домножение квадратичного функционала на константу, большую нуля, не изменяет его поведение с точки зрения оптимизации).

Рассматривается не конкретная выборка, а всевозможные обучающие выборки для этой задачи.

$$\underbrace{\mathbb{E}_{XY}\left\{\left[\hat{f}(x) - y(x)\right]^2\right\}}_{\text{ошибка на объекте}} = \underbrace{\left(\mathbb{E}_{XY}\left\{\hat{f}(x)\right\} - f(x)\right)^2}_{\text{bias(сдвиг)}} + \underbrace{\mathbb{E}_{XY}\left\{\left(\hat{f}(x) - \mathbb{E}_{XY}\left(\hat{f}(x)\right)\right)^2\right\}}_{\text{variance(разброс)}} + \underbrace{\mathbb{E}\varepsilon^2}_{\text{шум}} \tag{1}$$

В (1) x считается фиксированным, т. е. рассматривается ошибка на конкретном объекте.

Первое слагаемое в правой части равенства характеризует, насколько модель хороша: усреднение всех предсказаний модели на объекте x, которые были получены в результате обучения модели на всевозможных выборках с последующим оцениванием, насколько далеко полученная величина будет находиться от истинной функции $f(x)$. Это слагаемое называется *bias(сдвиг)*.

Второе слагаемое выражает усреднение по всем выборкам отклонения предсказания модели при условии, что её обучили на выборке, от среднего предсказания модели. Показывается насколько зависит предсказание модели от того, на какой выборке она обучалась. Это слагаемое называется *variance(разброс)*.

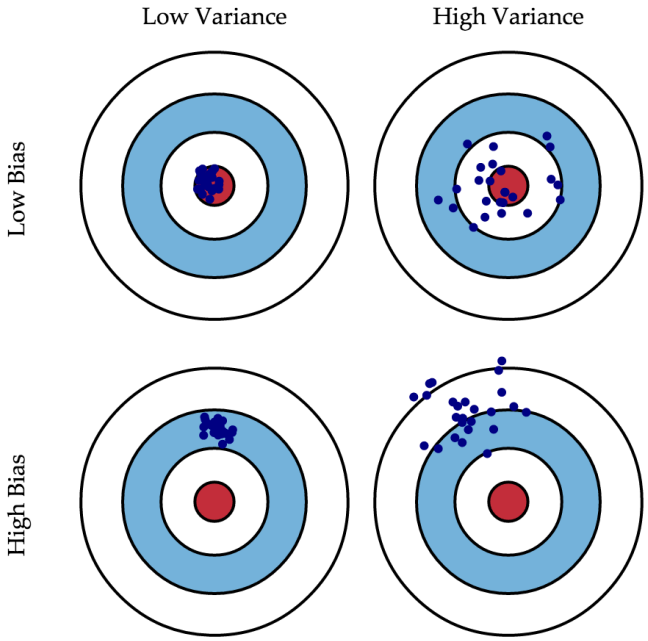

Рис. 2: Иллюстрация сдвига и разброса для различных моделей.

На рис. 2 иллюстрируются величины сдвига и разброса для различный моделей, истинное значение - центры мишеней. Если модель недообучена, то в ошибке будет доминировать сдвиг - возможно, модель не плохая, но она предсказывает не то; если переобучена (сложная, перепараметризованная), то будет доминировать разброс - для конкретной (обучающей) выборки предсказывается точное значение, но

при малейшем изменении обучающей выборки из-за большого количества параметров предсказание будет далеко от центра.

Таким образом, было одним из возможных способов формально определено, что такое переобучение и недообучение. Для борьбы с ними существуют **ансамблевые методы обучения**.

3 Ансамбли (ансамблевые методы обучения)

Имеется несколько обученных моделей:

$$\left.\begin{array}{c} y_1 = f_1(x) \\ y_2 = f_2(x) \\ ... \\ y_m = f_m(x) \end{array}\right\} \tag{2}$$

Целевая метка предсказывается не как предсказание одной из этих моделей, а как функция G от всех этих моделей:

$$y = G(f_1(x), f_2(x), ..., f_m(x)). \tag{3}$$

Выбирая G, можно изменять сложность модели. Если все модели слишком сложные, то можно выбрать простую G, например, усреднение, упростив таким образом ансамбль:

$$y = \frac{1}{m}\sum_{i=1}^{m} f_i(x). \tag{4}$$

Если модели простые, то выбором более сложной G можно усложнить ансамбль.

Пример для многоклассовой классификации. Имеется классификатор, идеально решающий задачу классификации для 2 классов. Применением этого классификатора последовательно отделяется один класс от всех, потом второй класс от всех, потом третий класс от всех и т. д. Исходя из голосования классификатора на каждом шаге объект относится к определённому классу.

Рассматривается взвешенное усреднение:

$$y = \frac{1}{m}\sum_{i=1}^{m} w_i * f_i(x). \tag{5}$$

G - это линейная регрессия/классификация. Как обучать такой объект?

1 способ. На рис. 3 изображена таблица, в левой части которой расположена обучающая выборка, содержащая n объектов с k признаками, для каждого объекта известна истинный ответ (столбец y). Модели $f_1(x), f_2(x), ..., f_m(x)$ были обучены по этой выборке, и с помощью этих моделей были предсказаны метки l_{ij}, $i = 1, ..., n$, $j = 1, ..., m$ для исходной выборки, расположенные в правой части таблицы. На метках l_{ij} обучается финальная модель G.

Проблема такого подхода заключается в том, что модели были обучены на обучающей выборке, потом эта же выборка предсказывается с помощью обученных моделей, т. е. происходит утечка целевой переменной. На обучающей выборке при таком способе будут слишком позитивные прогнозы, в отличие от теста.

2 способ. На рис. 4 изображена модифицированная таблица, в которой обучающая выборка разделена на 2 части: по n_0 и $n - n_0$ объектов соответственно. Все m моделей обучаются на первой части выборки из n_0 объектов, после чего обученные модели предсказывают вторую часть объектов, порождая метки l_{ij}, $i = n_0 + 1, ..., n$, $j = 1, ..., m$ Модель G обучается на метках l_{ij}, $i = n_0 + 1, ..., n$, $j = 1, ..., m$.

	X				y	$f_1(x)$	$f_2(x)$	$\dots$	$f_m(x)$	G
Object count	x_{11}	x_{12}	$\dots$	x_{1k}	y_1	l_{11}	l_{12}	$\dots$	l_{1m}	
	x_{21}	x_{22}	$\dots$	x_{2k}	y_2	l_{21}	l_{22}	$\dots$	l_{2m}	
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	
	x_{n1}	x_{n2}	$\dots$	x_{nk}	y_n	l_{n1}	l_{n2}	$\dots$	l_{nm}	
	Feature count				True answer	Predict answer				

Рис. 3

	X				y	$f_1(x)$	$f_2(x)$	$\dots$	$f_m(x)$	G
f_i train	x_{11}	x_{12}	$\dots$	x_{1k}	y_1					
	x_{21}	x_{22}	$\dots$	x_{2k}	y_2					
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$					
	$x_{n_0 1}$	$x_{n_0 2}$	$\dots$	$x_{n_0 k}$	y_{n_0}					
G train	$x_{(n_0+1)1}$	$x_{(n_0+1)2}$	$\dots$	$x_{(n_0+1)k}$	y_{n_0+1}	$l_{(n_0+1)1}$	$l_{(n_0+1)2}$	$\dots$	$l_{(n_0+1)m}$	
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	
	x_{n1}	x_{n2}	$\dots$	x_{nk}	y_n	l_{n1}	l_{n2}	$\dots$	l_{nm}	

Рис. 4

3 способ. Рассматривается случай, когда выборка слишком небольшая для того, чтобы обучаться на небольшой части исходных объетов. На рис. 5 изображена модифицированная таблица, в которой обучающая выборка разделена на t равных фолдов. t выбирается из соображений оптимальности временных затрат. Все m моделей обучаются на первых $t-1$ фолдах, после чего обученные модели предсказывают фолд с номером t; далее модели обучаются, например, на 1, 2, ..., $t-2$, t фолдах, а предсказывают $t-1$ фолд, и т. д. Порождаются все метки l_{ij}, $i = 1, ..., n$, $j = 1, ..., m$. На метках l_{ij} обучается финальная модель G. Такой алгоритм называется **стекинг**.

	X				y	$f_1(x)$	$f_2(x)$	$\dots$	$f_m(x)$	G
$(t-1)$ train	x_{11}	x_{12}	$\dots$	x_{1k}	y_1					
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$					
	$x_{n_0 1}$	$x_{n_0 2}$	$\dots$	$x_{n_0 k}$	y_{n_0}					
		$\dots$			$\dots$					
1 predict	$x_{((t-2)n_0+1)1}$	$x_{((t-2)n_0+1)2}$	$\dots$	$x_{((t-2)n_0+1)k}$	$y_{(t-2)n_0+1}$	$l_{((t-2)n_0+1)1}$	$l_{((t-2)n_0+1)2}$	$\dots$	$l_{((t-2)n_0+1)m}$	
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	
	x_{n1}	x_{n2}	$\dots$	x_{nk}	y_n	l_{n1}	l_{n2}	$\dots$	l_{nm}	
						Predicted answer for f_i on last fold				

Рис. 5

Полезная формула.

$$F(x) = \sum_{i=1}^{m} w_i * f_i(x), \tag{6}$$

где $w_i > 0$, $\sum_{i=1}^{m} w_i = 1$.

$$(F(x) - y)^2 = \sum_{i=1}^{m} w_i (f_i(x) - y)^2 - \sum_{i=1}^{m} w_i (f_i(x) - F)^2, \tag{7}$$

т. е. квадратичная ошибка на всей выборке может быть разложена на:

- слагаемое, которое показывает, как хорошо каждый из *base learners* приближает целевую метку
- неположительное слагаемое, показывающее, насколько каждый конкретный *base learner* предсказывает объекты далеко от истинного значения. Это слагаемое по сути означает разброс.

Таким образом, улучшить качество ансамбля можно не только улучшая каждую конкретную модель, но и беря в ансамбль модели, которые наиболее независимы друг от друга (их значения лежат далеко друг от друга). Можно уменьшить ошибку на обучающей выборке, даже если все модели будуь плохими.

4 Градиентный бустинг

Был рассмотрен концепт ансамблевых методов в виде представления метки как функции от нескольких моделей. В рамках этого метода все модели обучаются независимо друг от друга. Это хорошо тем, что может позволить сделать вычисления параллельными. Было бы полезным так организовать обучение моделей, чтобы каждая следующая модель знала об ошибках в существующем ансамбле и пыталась обучиться исправлять ошибки общего ансамбля. Для таких целей был придуман **градиентный бустинг**.

Пример. Рассматривается задача минимизации квадратичного функционала

$$\frac{1}{2} \sum_{i=1}^{l} (a(x_i) - y_i)^2 \to \min_a, \tag{8}$$

где l - количество объектов в обучающей выборке, a - обучаемая модель. Модель имеет вид:

$$a_N(x) = \sum_{i=1}^{N} b_i(x), b_i \in \mathbb{A}, \tag{9}$$

где $\mathbb{A}$ - семейство моделей (линейных, деревьев, нейросетей и т. д.). На начальном этапе нет никакой информации об ошибках, поскольку нет обучаемого ансамбля:

$$b_1(x) = \underset{b \in \mathbb{A}}{\operatorname{argmin}} \Big\{ \frac{1}{2} \sum_{i=1}^{l} (a(x_i) - y_i)^2 \Big\} \tag{10}$$

Далее рассматриваются ошибки алгоритма:

$$s_i^1 = y_i - b_1(x_i) = y_i - a_1(x_i) \tag{11}$$

Следующий алгоритм учитывает ошибки предыдущей модели:

$$b_2(x) = \underset{b \in \mathbb{A}}{\operatorname{argmin}} \Big\{ \frac{1}{2} \sum_{i=1}^{l} (b(x_i) - s_i^1)^2 \Big\} \tag{12}$$

Новый базовый алгоритм прибавляется к уже имеющемуся ансамблю, рассматриваются новые ошибки:

$$s_i^2 = y_i - a_2(x_i) \tag{13}$$

Эти ошибки учитывает третий алгоритм и т. д. Пусть есть ансамбль из N моделей:

$$s_i^N = y_i - a_N(x_i) \tag{14}$$

Стоит заметить, что:

$$s_i^N = -\frac{\partial}{\partial z}\frac{1}{2}(z - y_i)^2\Big|_{z=a_N(x_i)}, \tag{15}$$

т. е. если строить ансамбль моделей таким образом, то каждый раз будет делаться шаг градиентного спуска, новый ансамбль вычисляется исходя из направления антиградиента.

Другой подход. Пусть имеется дифференцируемая функция потерь $L(y, z)$. Строится взвешенная сумма базовых алгоритмов:

$$a_N = \sum_{i=0}^{N} \gamma_i * b_i(x), \tag{16}$$

где $\gamma_i \in \mathbb{R}$, $i = 1, ..., N$ - веса. Нумерация осуществляется с 0, поскольку в отличие от предыдущего примера минимизации квадратичного функционала, где первая базовая модель получалась посредством оптимизации, сейчас выбирается начальное приближение модели, что менее сложно и затратно по времени. Например, $b_0(x) = const$. Если в качестве константы взять 0, то подбор модели $b_1(x)$ будет эквивалентен подбору аналогичной модели в предыдущем примере. Также в качестве начального приближения можно взять наилучшее константное приближение.

Пусть имеется ансамбль из $N - 1$ моделей:

$$a_{N-1} = \sum_{i=0}^{N-1} \gamma_i * b_i(x), \tag{17}$$

Необходимо следующую модель взять такой, чтобы общий ансамбль улучшился. Необходимо решить задачу следующего вида:

$$\sum_{i=0}^{N} L\big(y_i, a_{N-1}(x_i) + \gamma_N b_N(x_i)\big) \to \min_{b_N, \gamma_N} \tag{18}$$

На рис. 6 по горизонтали отложены объекты, по вертикали ответы на этих объектах, причем истинные значения отмечены зелёным цветом, а предсказанные синим.

На каждом объекте можно вычислить ошибки и приблизить предсказания новой модели к истинным объектам. Если в качестве новых $b_N(x)$ брать эти ошибки как константы, то предсказания будет идеальными с нулевой ошибкой на обучающей выборке, однако необходимо предсказывать ответы не дискретно, а вывести зависимость (т. е. подобрать функцию, принимающую на объетах обучающей выборки нужные значения). В предыдущем примере было установлено соответствие междунаправлением антиградиента и прибавлением новых моделей, обученных под ошибки предыдущей модели. При текущем подходе такой переход производиться не будет. Необходимо минимизировать функционал качества L. y_i, $i = 1, ..., N$, заданы, их изменить нельзя. Значит, нужно изменять $a_{N-1}(x_i) + \gamma_N b_N(x_i)$. Очевидно, что к $a_{N-1}(x_i)$ нужно прибавлять антиградиент (шагнуть в направлении антиградиента).

$$s_i = -\frac{\partial L(y_i, z_i)}{\partial z}\Big|_{z=a_{N-1}(x_i)}, \tag{19}$$

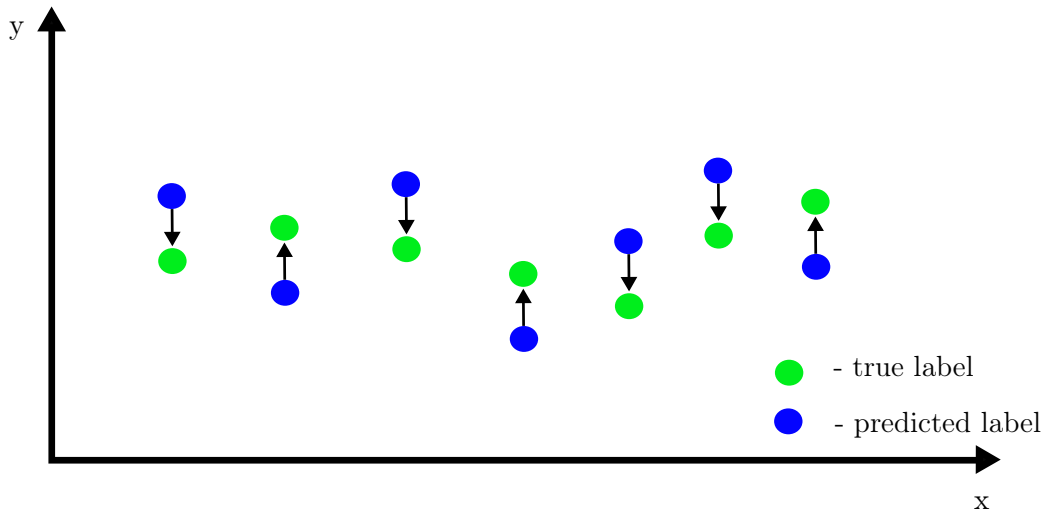

Рис. 6

где s_i - это то, под что обучается каждая следующая модель. В терминологии всей выборки можно записать все s_i в виде вектора:

$$s = \Big(-\frac{\partial L}{\partial z}\Big|_{z=a_{N-1}(x_i)}\Big)_{i=1}^{l} = -\nabla_z \sum_{i=1}^{l} L(y_i, z_i)\Big|_{z_i=a_{N-1}(x_i)}. \tag{20}$$

Если в методах градиентного спуска для линейных моделей спуск осуществляется в пространстве обучающей выборки, т. е. размерность этого пространства равна размерности объетов, то при текущем подходе размерность пространства, в котором делаются шаги градиентного спуска, равна l, т. е. количеству объетов в обучающей выборке. Оптимизируются не веса моделей, а веса при моделях.

$$b_n = \underset{b\in\mathbb{A}}{\operatorname{argmin}}\Big(\sum_{i=0}^{l}(b(x_i) - s_i)^2\Big), \tag{21}$$

где для оптимизации был выбран квадратичный функционал, поскольку вся информация о L заложена в s_i. При подгоне b под s оптимизируется исходный функционал L. Чтобы не усложнять задачу, берётся квадратичный функционал.

$$\gamma_N = \underset{\gamma\in\mathbb{R}}{\operatorname{argmin}}\Big(\sum_{i=0}^{N} L(y_i, a_{N-1}(x_i) + \gamma_i b_N(x_i)\Big), \tag{22}$$

что является задачей одномерной минимизации.

Градиентный бустинг реализует ансамблевый метод предсказания, который подбирает каждую следующую модель таким образом, чтобы она уменьшала ошибки уже существующего ансамбля, исходя из функционала L. Этот метод может переобучиться. Нужна регуляризация. По аналогии с градиентным спуском эту задачу может решить стохастический градиентный бустинг: обучать модели не на всей обучающей выборке, а на подвыборке, либо на подмножестве признаков; можно также регулировать длину шага:

$$a_N = a_{N-1}(x) + \eta * \gamma_N * b_N(x), \tag{23}$$

где η - *learning rate*, т. е. делать шаг не в сторону оптимума, а немного в сторону. Также важен подбор базовых моделей, которые называются *weak learners*, поскольку лучше выбирать модели проще. Сложные (перепараметризованные) модели сразу исправят все ошибки на обучающей выборке, сходимость будет осуществлена за малое число шагов. Естественно, в качестве базовых моделей не следует брать настолько простые модели, как константы. Например, можно брать деревья небольшой глубины. Это позволит избежать переобучения.