Ансамбли алгоритмов

Лекция №12

Проблема (регрессия)

Есть несколько алгоритмов регрессии $C_1, C_2, \ldots, C_k$, которые обладают точностью $t_1, t_2, \ldots, t_k$. Можно ли из них построить новый алгоритм, с большей точностью?

Основные методы:

1) выдача среднего значения;

2) взвешенная сумма;

3) бустинг (когда алгоритм C_k улучшает C_{k-1})

Проблема (классификация)

Есть несколько алгоритмов классификации $C_1, C_2, \ldots, C_k$, которые обладают точностью $t_1, t_2, \ldots, t_k$. Можно ли из них построить новый алгоритм, с большей точностью*?*

Основные методы:

1) голосование по большинству;

2) взвешенное голосование;

3) бустинг (когда алгоритм C_k улучшает C_{k-1})

Голосование по большинству (комитет)

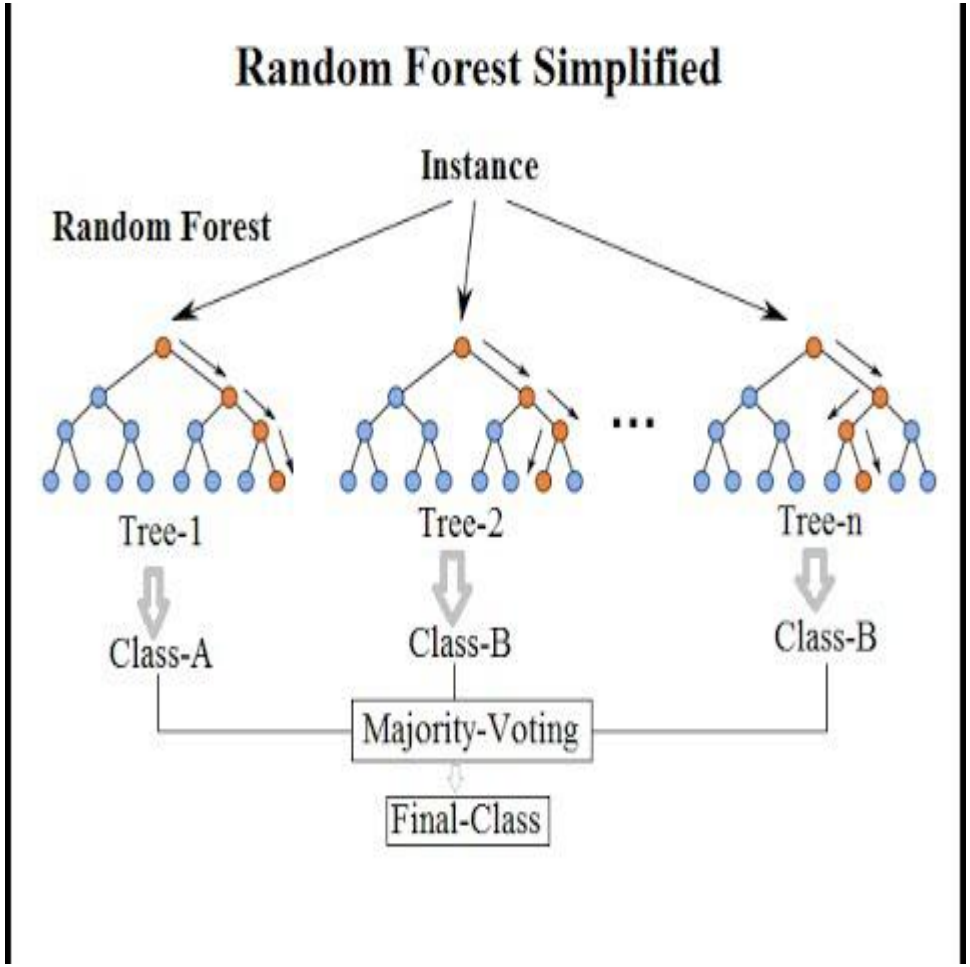
Голосование по большинству

Пусть часть классификаторов $C_1, C_2, \ldots, C_k$ отнесла объект A к классу -1, а другая часть отнесла объект A к классу 1. Итог: класс, который выбрало большинство классификаторов.

Такая схема еще называется комитетом.

Почему комитет обладает большей точностью, чем отдельные классификаторы?

Как известно, RandomForest работает по принципу комитета



https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

Почему комитет обладает большей точностью, чем отдельные классификаторы?

В качестве иллюстрации можно рассмотреть задачку из курса теории вероятностей.

Комитет состоит из 3-х членов. Члены комитета выносят правильное решение с вероятностями 0.6 0.7 0.8 соответственно. Окончательное решение принимается большинством голосов. Найти вероятность ошибки комитета.

Почему это работает?

Решение. Комитет ошибается, когда ошибается 2 или 3 члена комитета. Легко перечислить все случаи, когда комитет ошибается:

1) +--
2) -+-
3) --+
4) ---

Нужно найти вероятности каждого из случаев. Чему они равны?

Почему это работает?

1) +-- 0.6*0.3*0.2=0.036
2) -+- 0.4*0.7*0.2=0.056
3) --+ 0.4*0.3*0.8=0.096
4) --- 0.4*0.3*0.2=0.024

Суммарная вероятность ошибки=0.036+0.056+0.096+0.024=0.212

Таким образом, комитет имеет точность

1-0.212=0.788

Почему это работает?

Эффективность комитета особенно сильно возрастает, если все его члены имеют примерно одинаковую точность (в качестве упражнения можете решить предыдущую задачу, где точность всех членов комитета равна 0.8. Ответ будет=0.896).

Базовые алгоритмы уже должны быть отличными.

Если комитет составить из классификаторов, каждый из которых имеет точность 0.5 то точность комитета будет…

Базовые алгоритмы уже должны быть отличными.

Если комитет составить из классификаторов, каждый из которых имеет точность 0.5 то точность комитета будет 0.5 (останется равной точности членов комитета).

Но есть проблемка

Комитет гарантировано улучшает точность, если результаты работы базовых классификаторов независимы друг от друга (прямо как в наивном Байесе).

А что будет, если классификаторы зависят друг от друга?

В частности, что будет если комитет составлен из клонов одного и того же классификатора?

Комитет из зависимых членов

Пусть комитет состоит из 3-х клонов одного и того же классификатора. Точность классификатора 0.834243 (специально взял страшное число – чтобы вам сложнее считалось)))))))).

Найти точность комитета.

Ваш ответ=…

Комитет из зависимых членов

Решение очень простое. Комитет может выдавать лишь следующие распределения голосов:

1) +++
2) ---

Случай 1) происходит, когда классификатор отвечает правильно. То есть сего вероятность равна 0.834243

Ответ: 0.834243 – то есть комитет бесполезен в данной ситуации.

Что делать с зависимыми классификаторами?

На практике независимость результатов классификаторов друг от друга не всегда выполняется. Но это не повод отказываться от комитета. Дерзайте!

Кроме того, никак нельзя заранее вычислить оптимальный размер комитета.

При добавлении в комитет новых членов точность (теоретически) должна возрастать, но, с другой стороны, появляются зависимость между членами комитета.

Взвешенное голосование AdaBoost

Взвешенное голосование

Пусть классификаторы $C_1, C_2, \ldots, C_k$ относят объект A к классу 1 или -1 (результат работы C_i для объекта A обозначим через $C_i(A)$). Окончательное решение зависит от весов w_i:

если $w_1C_1(A) + w_2C_2(A) + \ldots + w_kC_k(A) < 0$, то А из класса -1.

если $w_1C_1(A) + w_2C_2(A) + \ldots + w_kC_k(A) > 0$, то А из класса 1.

Очень похоже на линейный классификатор (здесь тоже нужно оптимально настроить веса w_i).

Аналогия с лин. классификатором

По аналогии вводим понятие отступа объекта A_i с целевым признаком y_i:

$M_i = y_i(w_1 C_1(A_i) + w_2 C_2(A_i) + \ldots + w_k C_k(A_i))$.

Значения весов находятся из условия минимизации выражения:

$$\sum_{i=n}^{n} [M_i < 0]$$

где n – объем тренировочной выборки, M_i – отступ i-го объекта.

Как и для лин. классиф-ов данное выражение мажорируется дифференцируемой функцией.

AdaBoost

Если в качестве мажорирующей отступ функции выбрать функцию e^{-Mi}, то получим известный алгоритм AdaBoost.

AdaBoost – это итерационный алгоритм, основанных на следующих идеях:

1. Вводится понятие «вес объекта», которое позволяет получить взвешенную ошибку каждого классификатора на тренировочной выборке.

AdaBoost

2. На очередной итерации мы настраиваем вес w_i при классификаторе C_i с минимальной взвешенной ошибкой.

3. Объекты тренир. выборки, на которых ошибается C_i, на следующей итерации приобретают бОльший вес.

Т.о. на каждой итерации мы ищем классификатор, который лучше всех классифицирует «трудные» объекты, то есть объекты с большим весом.

AdaBoost (формальное описание)

Пусть u_j – вес j-го объекта из тренировочной выборки. Вначале все u_j равны 1/n (n – объем тренировочной выборки)

Ошибка *Bad(i)* классификатора C_i – это сумма весов объектов, которые он классифицирует неправильно.

AdaBoost (формальное описание)

1. Найти классификатор C_l с мин. взвешенной ошибкой *Bad(l)*.
2. Вычисляем новый вес классификатора C_l

$$w_l=0.5\ln[(1-Bad(l))/Bad(l)].$$

3. Обновляем веса объектов

$$u_j:=u_j exp(-w_l y_j C_l(A_j))$$

и нормируем их:

$$u_j:=u_j/(u_1+\ldots u_n)$$

Шаги 1-3 нужно повторять пока…

AdaBoost (формальное описание)

Шаги 1-3 нужно повторять пока

1) не надоест;
2) точность на тестовой выборке не стабилизируется;

Итоговый ответ – это правило:

если $w_1C_1(A)+ w_2C_2(A)+…+ w_kC_k(A)<0$, то A из класса -1.

если $w_1C_1(A)+ w_2C_2(A)+…+ w_kC_k(A)>0$, то A из класса 1.

КСТАТИ! Если некоторые объекты постоянно приобретают большие веса, то это, скорее всего, выбросы (еще один способ борьбы с ними).

Пример работы AdaBoost

Объ.	C1	C2	C3	Y
1	1	-1	-1	1
2	-1	1	-1	1
3	-1	-1	1	1
4	-1	1	1	1
5	1	-1	1	1
6	1	1	-1	1
7	1	1	1	1

1-ая итерация:

веса объектов

u=(1/7, 1/7, 1/7, 1/7, 1/7, 1/7, 1/7)

Bad(1)=0.43

Bad(2)=0.43

Bad(3)=0.43

Берем 1-й классификатор. w_1=0.14

Новые веса объектов:

(0.13 0.17 0.17 0.17 0.13 0.13 0.13)

Пример работы AdaBoost

2-ая итерация:

веса объектов

u=(0.13 0.17 0.17 0.17 0.13 0.13 0.13)

Bad(1)=0.5

Bad(2)=0.42

Bad(3)=0.42

Берем 2-й классификатор. w_2=0.17

Новые веса объектов:

(0.15 0.14 0.20 0.14 0.15 0.11 0.11)

Объ.	C1	C2	C3	Y
1	1	-1	-1	1
2	-1	1	-1	1
3	-1	-1	1	1
4	-1	1	1	1
5	1	-1	1	1
6	1	1	-1	1
7	1	1	1	1

Пример работы AdaBoost

Объ.	C1	C2	C3	Y
1	1	-1	-1	1
2	-1	1	-1	1
3	-1	-1	1	1
4	-1	1	1	1
5	1	-1	1	1
6	1	1	-1	1
7	1	1	1	1

3-ая итерация:

веса объектов

u=(0.15 0.14 0.20 0.14 0.15 0.11 0.11)

Bad(1)=0.49

Bad(2)=0.2

Bad(3)=0.4

Берем 3-й классификатор. w_2=0.20

Новые веса объектов:

(0.19 0.19 0.17 0.12 0.13 0.14 0.09)

Пример работы AdaBoost

Если прерваться на 3-й итерации,

то выражение

$w_1C_1(A)+ w_2C_2(A)+w_3C_3(A)$

для объектов трен.выборки будет равно:

-0.23

-0.18

-0.11

0.23

0.18

0.11

0.51

Объ.	C1	C2	C3	Y
1	1	-1	-1	1
2	-1	1	-1	1
3	-1	-1	1	1
4	-1	1	1	1
5	1	-1	1	1
6	1	1	-1	1
7	1	1	1	1

Градиентный бустинг

Бустинг – это?

Это способ улучшить существующий алгоритм предсказания a(X) с помощью нового алгоритма b(X).

Бустинг для задачи регрессии

Пусть для объектов тренировочной выборки был построен алгоритм a(X). Можно составить таблицу:

Объект	Признаки объекта	Y истинный	a(X) – предсказанное значение	Разность
A	…	10	8	2
B	…	5	6	-1
C	…	7	7	0

Бустинг для задачи регрессии

Зафиксируем множество алгоритмов M, среди которых будем искать улучшение для a(X). Например, M={линейные регрессии} или M={деревья регрессии}. M может быть и более узким классом, например M={деревья регрессии высоты не более 3}

Объект	Признаки объекта	Y истинный	a(X) – предсказанное значение	Разность
A	…	10	8	2
B	…	5	6	-1
C	…	7	7	0

Бустинг для задачи регрессии

Алгоритм $b(X) \in M$ будем искать по следующей тренировочной выборке

Объект	Признаки объекта	Разность
A	…	2
B	…	-1
C	…	0

То есть b(X) ищет поправки к ответам алгоритма a(X). ИЛИ: b(X) пытается минимизировать ошибки алгоритма a(X).

Бустинг для задачи регрессии

Получаем таблицу:

Объект	Признаки объекта	Y истинный	a(X)	Разность	b(X)	a(X)+b(X)
A	…	10	8	2	1	9
B	…	5	6	-1	-1	5
C	…	7	7	0	0.5	7.5

Получаем новую модель регрессии:

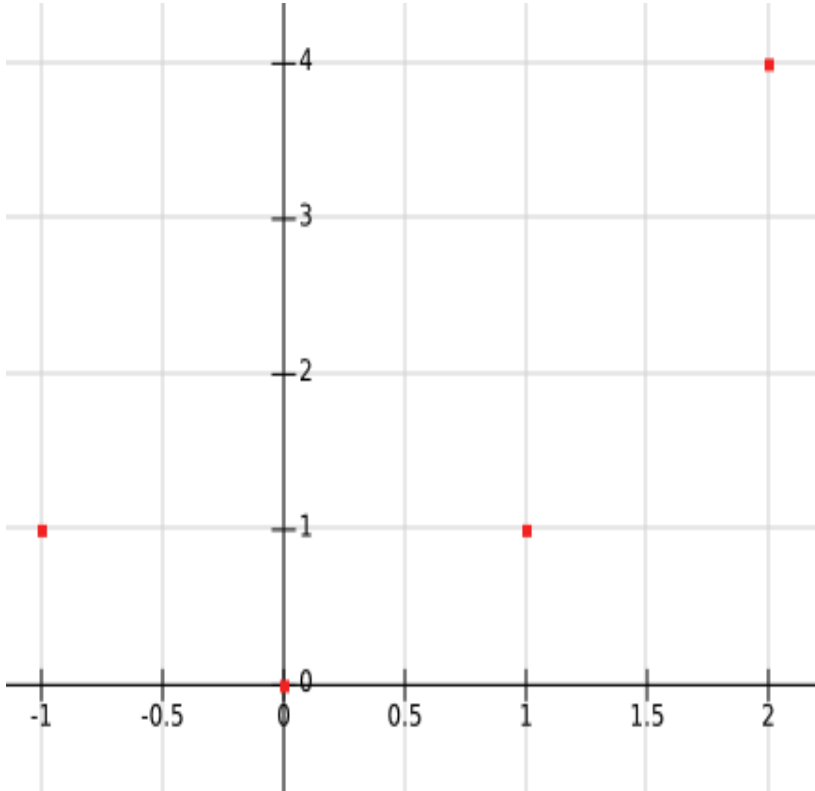
$$a(X)+b(X)$$

Если точность новой модели мала, то процедуру можно повторить.

Бустинг для задачи регрессии

Не всякую модель можно улучшить с помощью бустинга. Пусть a(X) – это модель линейной регрессии, построенной по данным:

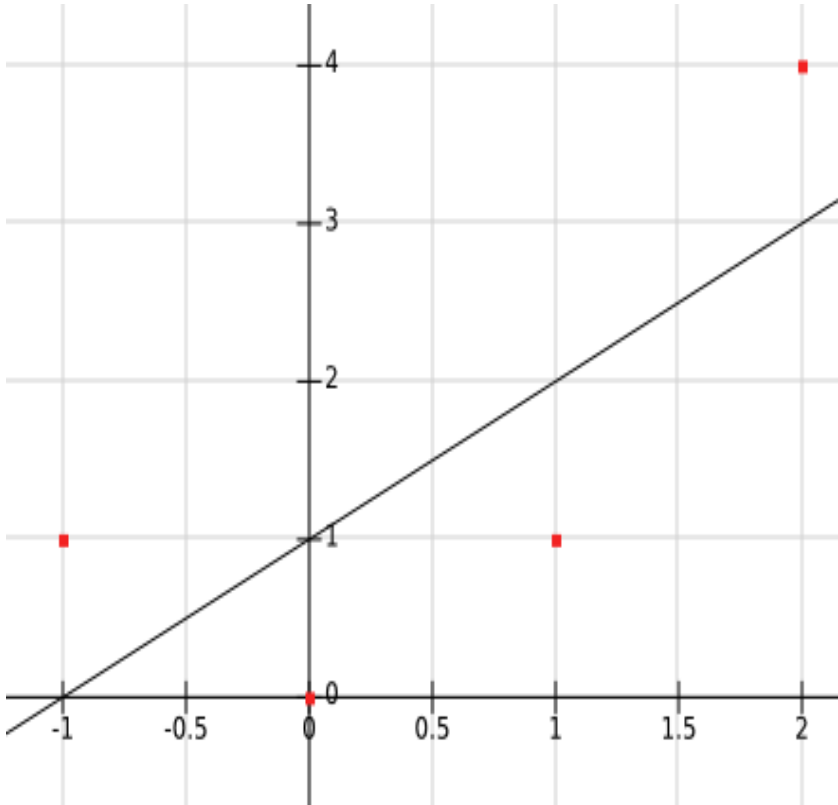
Объект	X	Y
A	-1	1
B	0	0
C	1	1
D	2	4



Бустинг для задачи регрессии

Алгоритм построения линейно регрессии даёт для a(X):

Объект	X	Y	a(X)	Разность
A	-1	1	0	1
B	0	0	1	-1
C	1	1	2	-1
D	2	4	3	1



Бустинг для задачи регрессии

Пусть b(X) ищется среди класса моделей M={модели линейной регрессии}.

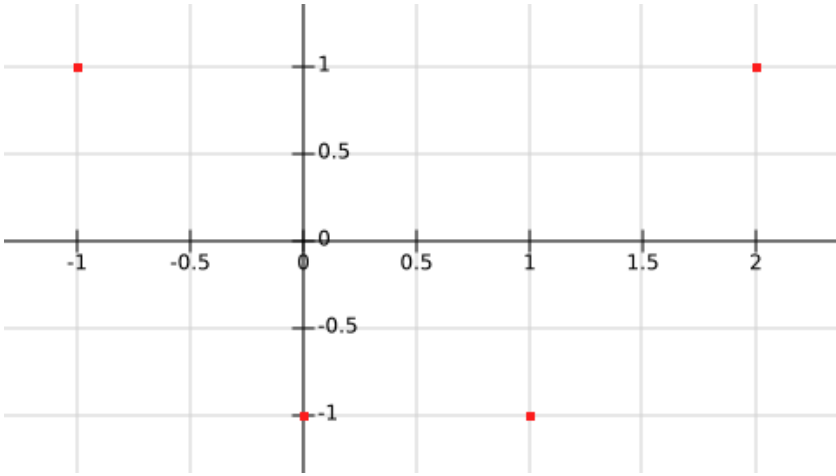
Имеем таблицу для b(X):

Объект	X	Разность (это значение должен b(X) предсказать)
A	-1	1
B	0	-1
C	1	-1
D	2	1

Какая модель лин. регрессии будет выбрана для этих данных в качестве b(X)?

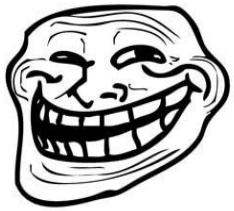
Бустинг для задачи регрессии

Объект	X	Разность (это значение должен b(X) предсказать)
A	-1	1
B	0	-1
C	1	-1
D	2	1



Бустинг для задачи регрессии

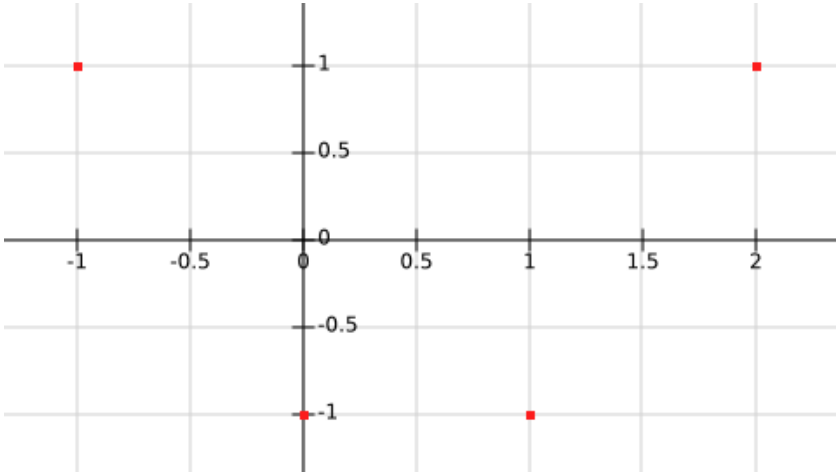
Будет выбрана модель b(X)=0!!!



И улучшить алгоритм a(X) не получится:

a(X)+b(X)=a(X)

Короче: линейную модель нельзя улучшить с помощью линейной!

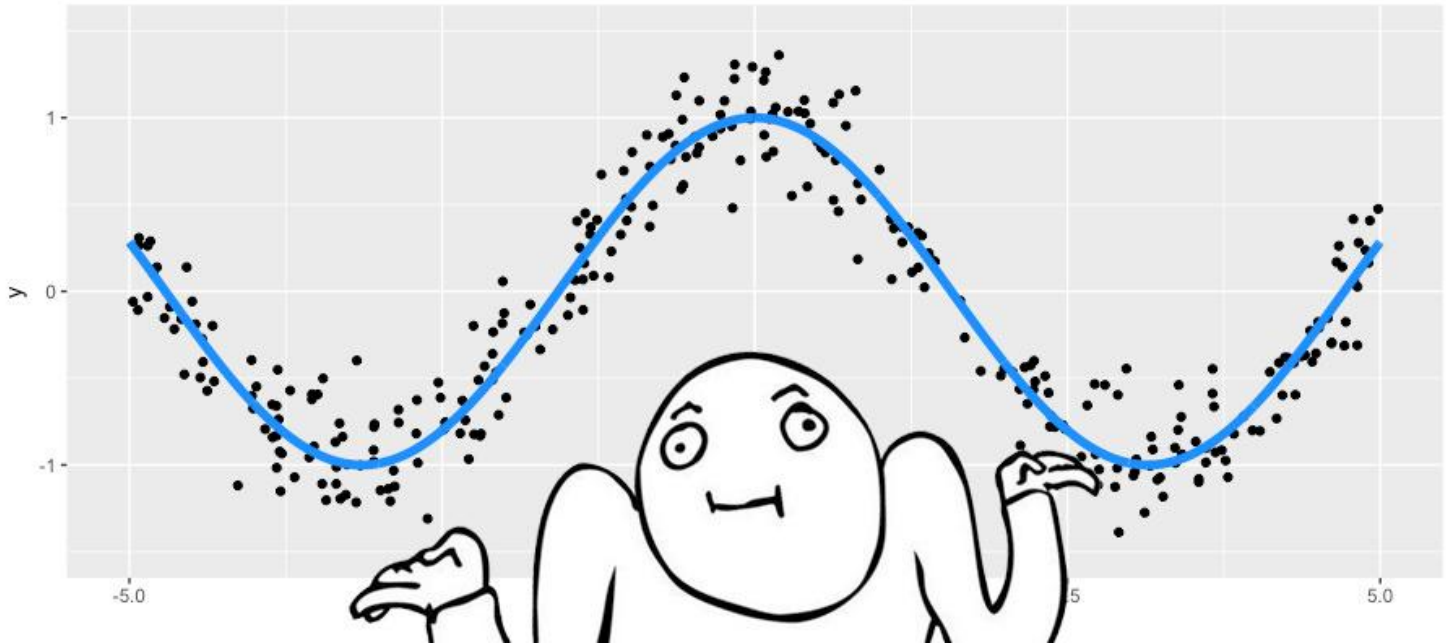


Бустинг для задачи регрессии

Следующий пример взят с https://habrahabr.ru/company/ods/blog/327250/

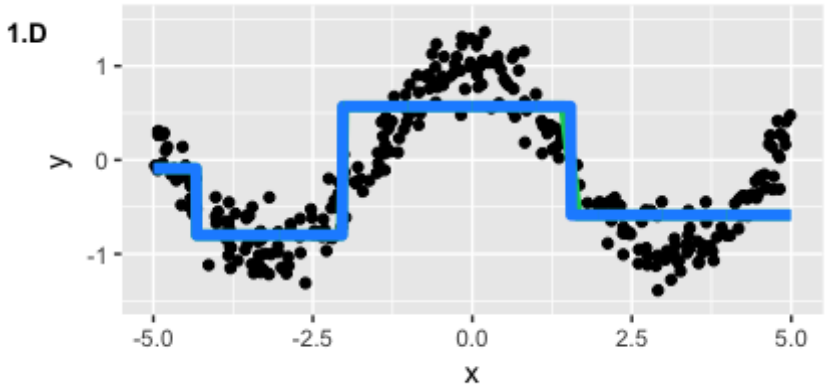
Объекты тренировочной выборки были сгенерированы как

y=cos(x)+{случайная ошибка}

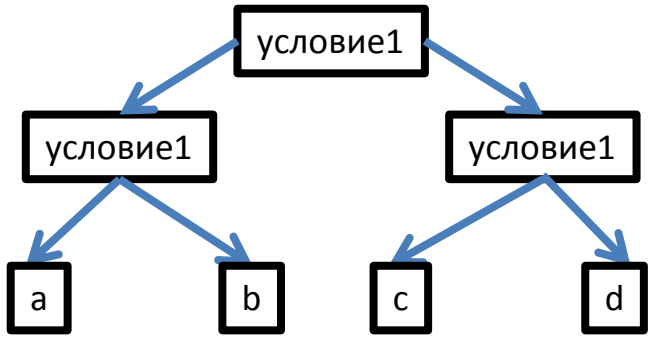


Бустинг для задачи регрессии

Первый алгоритм и все его улучшители будут браться из множества M={деревья высоты 3}. Поскольку дерево такой высоты имеет не более 4 листьев, то оно определяет график с не более чем 4-мя ступеньками.

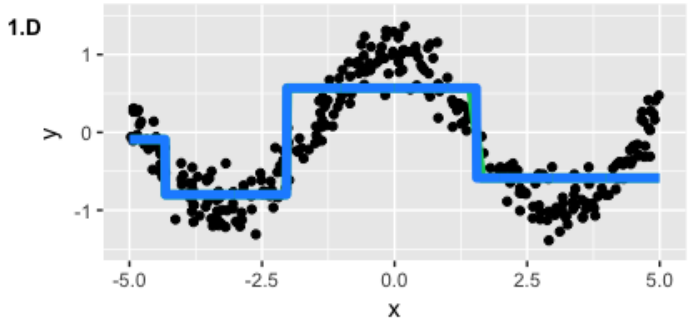


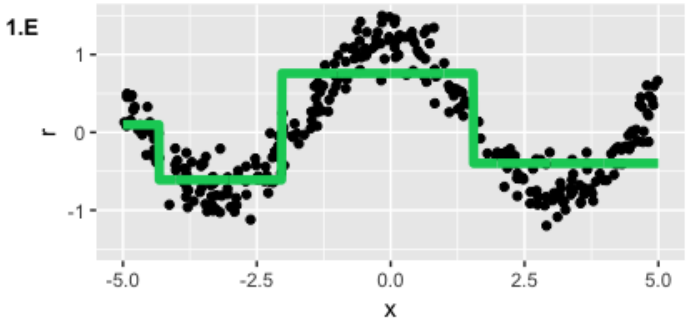
Ответы:

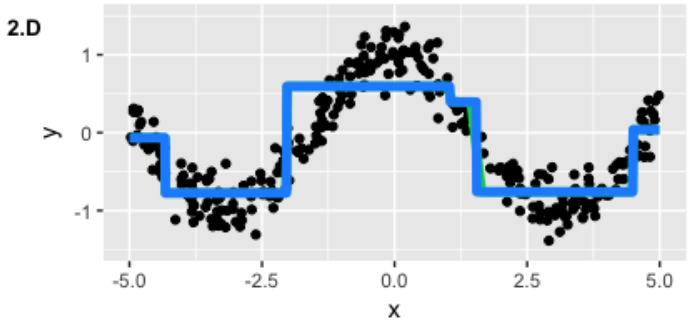


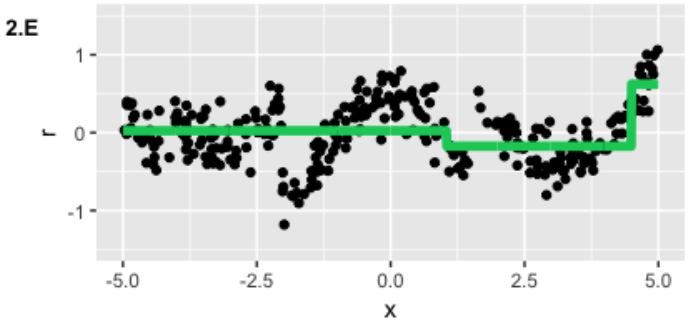
Бустинг для задачи регрессии

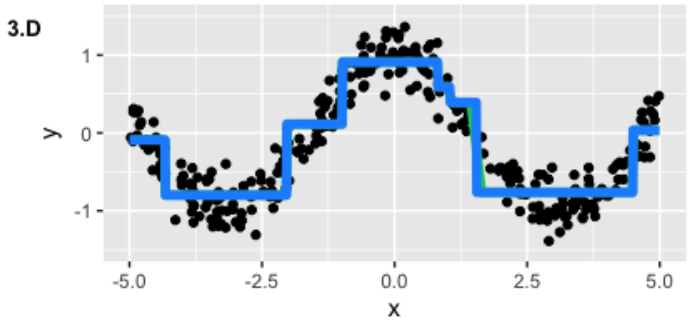
Показаны результаты работы 2-х итераций бустинга

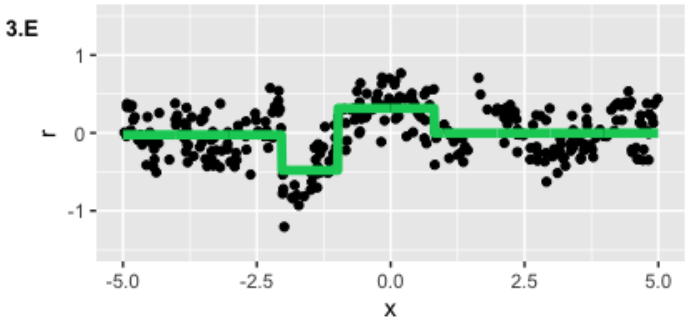












Бустинг для классификации

Бустинг для задач классификации строится как серия алгоритмов регрессии

$$a_1(X),\ldots,a_k(X)$$

с правилом классификации:

если $a_1(X)+\ldots+a_k(X)>0$*, то объект X из класса 1*

если $a_1(X)+\ldots+a_k(X)<0$*, то объект X из класса -1*

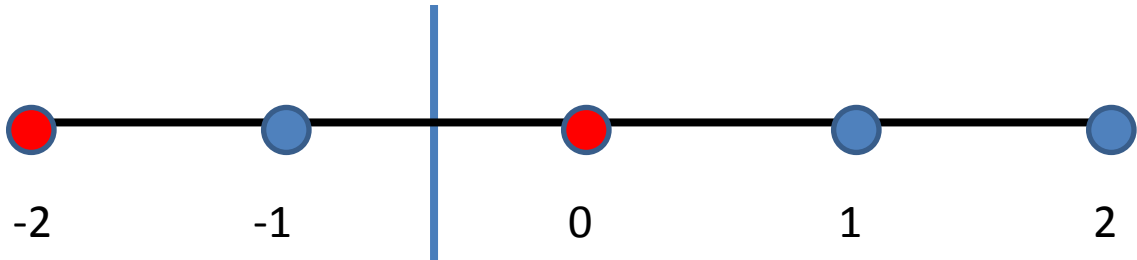
(очень похоже на правило линейного классификатора)

Бустинг для классификации

Первый алгоритм $a_1(X)$ строится обычным способом. Например, для данных из таблицы строится алгоритм $a_1(X)= 0.4x+0.2$.

То есть если $0.4x+0.2>0$, то класс 1, если $0.4x+0.2<0$, то класс -1

	x	y
A	-2	-1
B	-1	1
C	0	-1
D	1	1
E	2	1



Бустинг для классификации

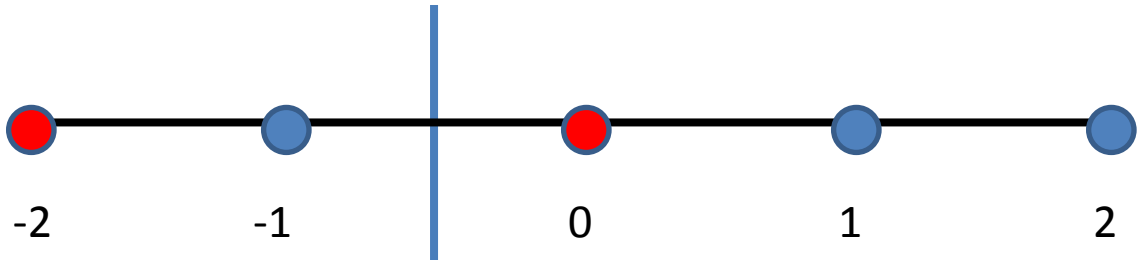
Добавим ответы алгоритма $a_1(X)= 0.4x+0.2$ в таблицу и подумаем, что делать дальше…

Алгоритм ошибается на объектах B,C (см. правило классификации).

Второй алгоритм (улучшающий первый) должен тренироваться на ошибках первого алгоритма.

Но как в задаче классификации корректно определить ошибку алгоритма?

	x	$a_1(x)$	y
A	-2	-0.6	-1
B	-1	-0.2	1
C	0	0.2	-1
D	1	0.6	1
E	2	1	1



Бустинг для классификации

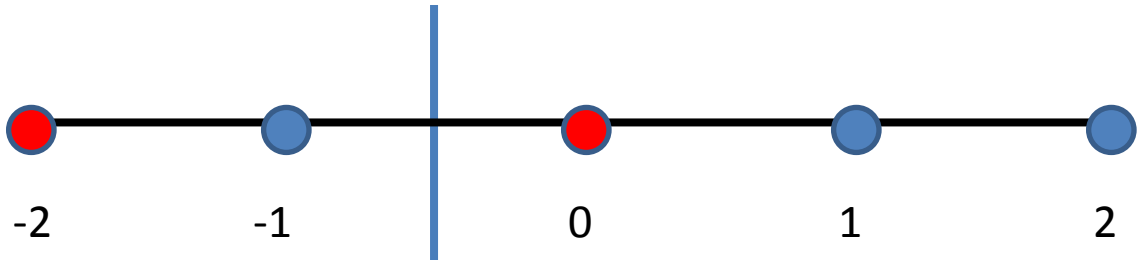
Решение заключается в вычислении отступов объектов и выборе мажорирующей функции.

В бустинге отступ i-го объекта – это

$M_i = y_i a(X)$,

Для данных из таблицы имеем отступы (отрицательный отступ говорит об ошибке классификации)

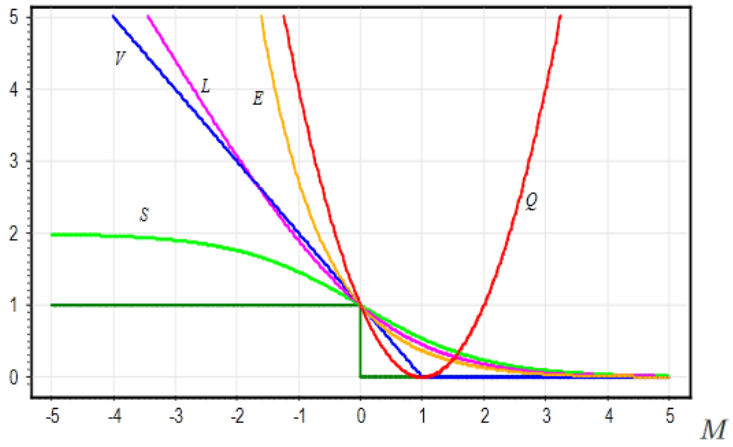
	x	$a_1(x)$	M_i	y
A	-2	-0.6	0.6	-1
B	-1	-0.2	-0.2	1
C	0	0.2	-0.2	-1
D	1	0.6	0.6	1
E	2	1	1	1



Бустинг для классификации

1. выбираем мажорирующую функцию *f(M);*
2. вместо M подставляем в функцию *f(M)* выражение ya;
3. находим производную *f'(M)*, дифференцируя по параметру a;
4. домножаем на -1;
5. вычисляем остаток r_i для i-го объекта, подставляя вместо *y*- метку класса i-го объекта

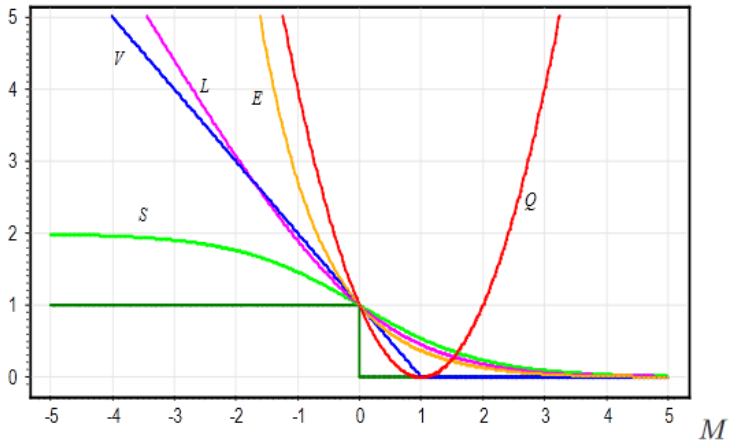
a – ответ алгоритма для i-го объекта (щас будет примерчик).



$Q(M) = (1 - M)^2$ — квадратичная;
$V(M) = (1 - M)_+$ — кусочно-линейная;
$S(M) = 2(1 + e^M)^{-1}$ — сигмоидная;
$L(M) = \log_2(1 + e^{-M})$ — логистическая;
$E(M) = e^{-M}$ — экспоненциальная.

Бустинг для классификации

1. $f(M)=e^{-M}$;
2. имеем $f(M)=e^{-ya}$
3. $f'(M)=-ye^{-ya}$
4. $-f'(M)=ye^{-ya}$
5. *остатки записываем в таблицу*



$Q(M) = (1-M)^2$ — квадратичная;
$V(M) = (1-M)_+$ — кусочно-линейная;
$S(M) = 2(1+e^M)^{-1}$ — сигмоидная;
$L(M) = \log_2(1+e^{-M})$ — логистическая;
$E(M) = e^{-M}$ — экспоненциальная.

	x	$a_1(x)$	M_i	y	r_i
A	-2	-0.6	0.6	-1	-0.55
B	-1	-0.2	-0.2	1	1.22
C	0	0.2	-0.2	-1	-1.22
D	1	0.6	0.6	1	0.55
E	2	1	1	1	0.37

Бустинг для классификации

А теперь строим модель регрессии, используя в качестве целевой переменной вектор остатков r_i.

Получится новая модель регрессии $a_2(X)$ и теперь классификатор выглядит

если $a_1(X)+a_2(X)>0$, то объект X из класса 1

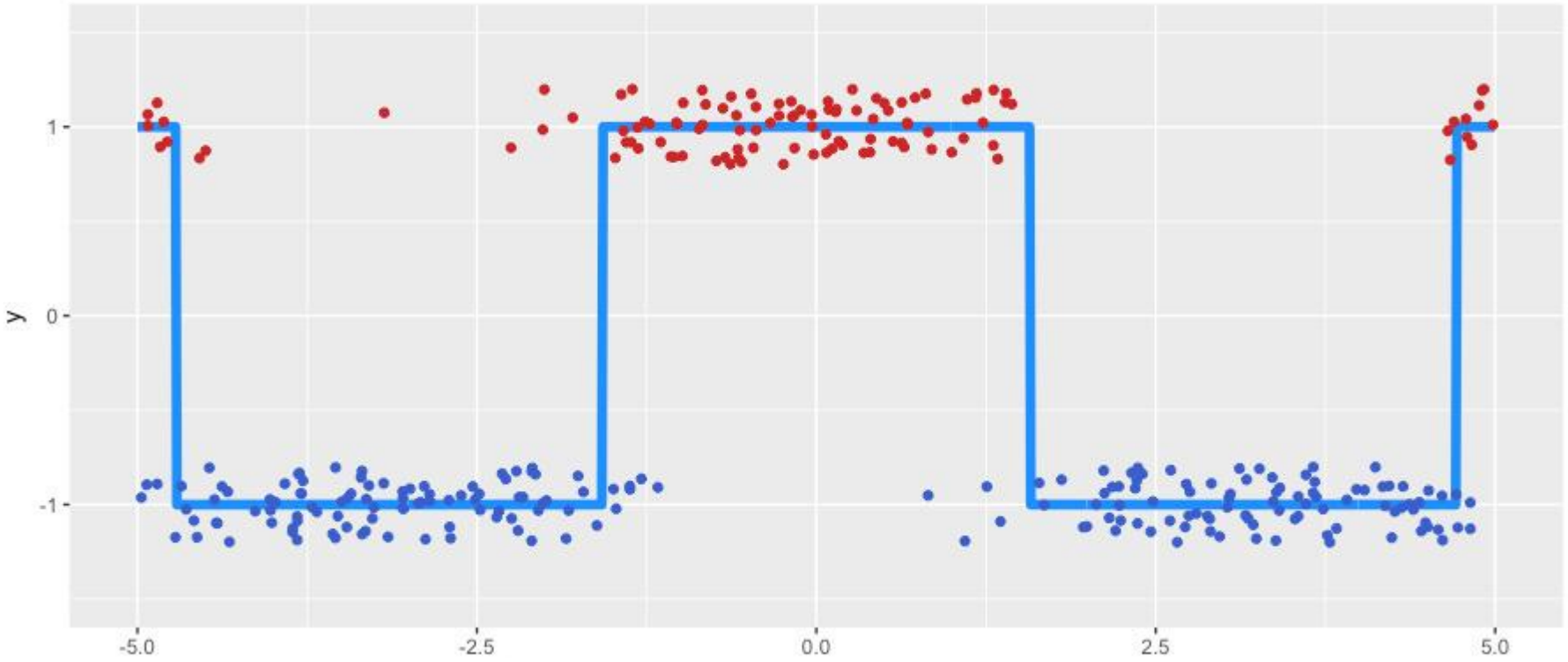
если $a_1(X)+a_2(X)<0$, то объект X из класса -1

итерацию бустинга можно повторить, построив алгоритм $a_3(X)$ и. т. д.

Данный вид бустинга называется градиентным, так как фактически вычисляется градиент функции ошибок (потерь) и новый алгоритм получается как шаг против градиента функции ошибок.

Бустинг для классификации

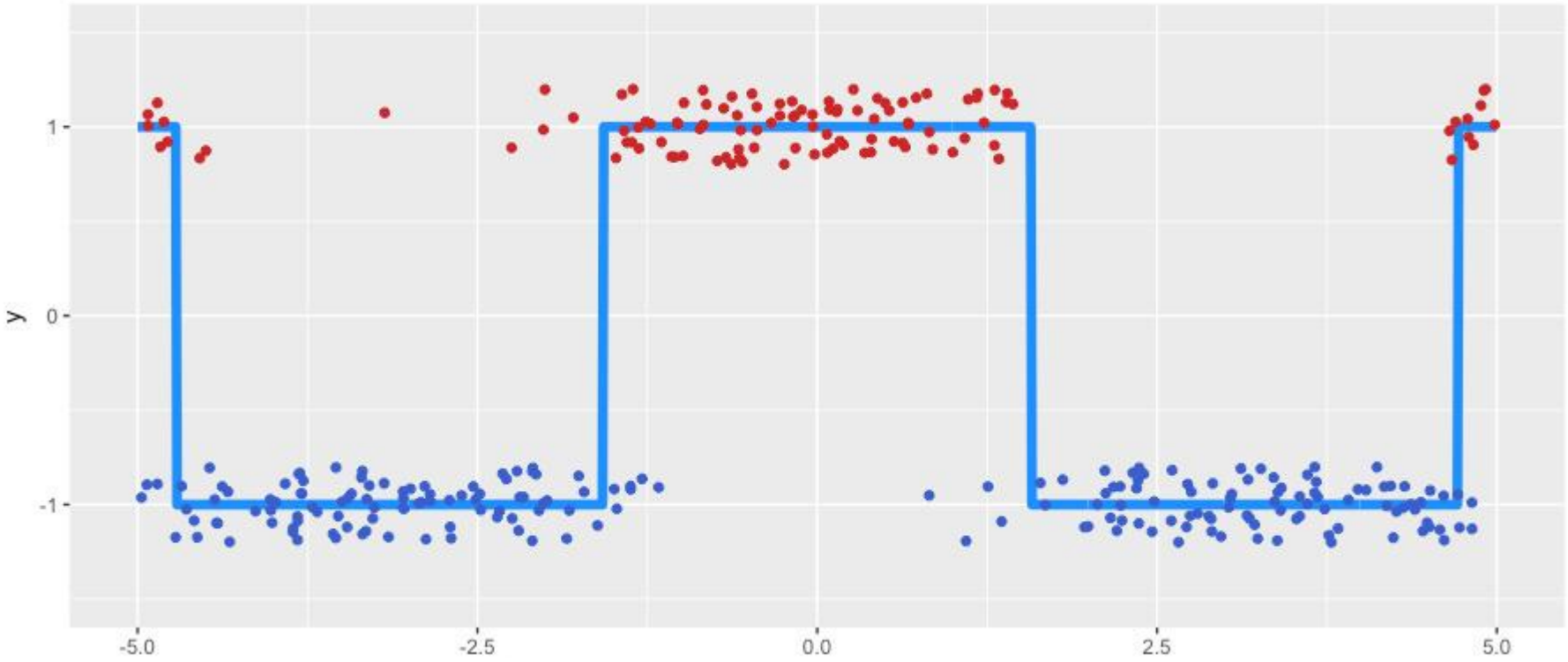
Возьмем данные: класс 1 получается из точек x при *cos(x)>0*; класс -1 получается из точек x при *cos(x)<0*. Добавим к ним немного шума, ошибок и выбросов.



https://habrahabr.ru/company/ods/blog/327250/

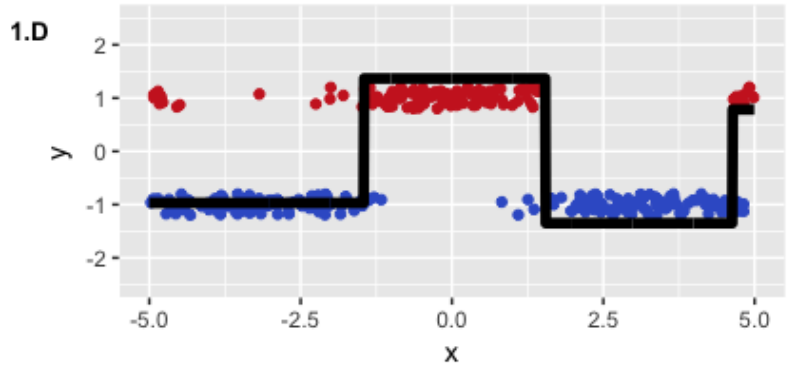
Бустинг для классификации

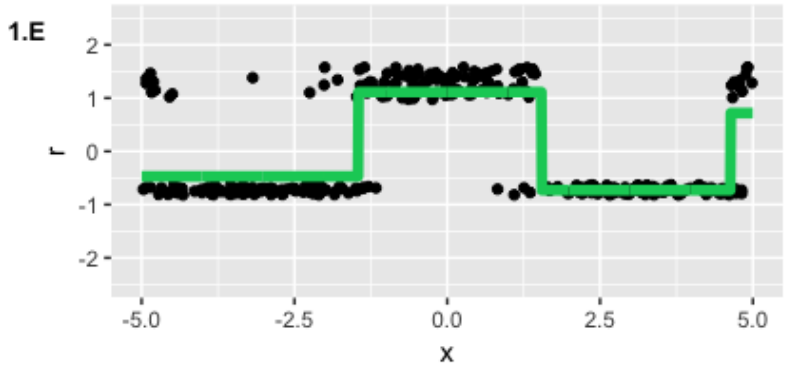
В качестве мажорирующей функции будем брать $L(M)=ln(1+e^{-M})$. Алгоритмы будем искать в классе регрессионных деревьев высоты 3.

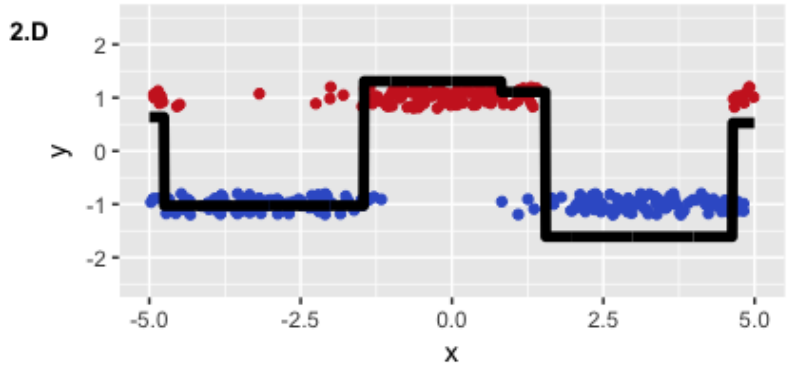


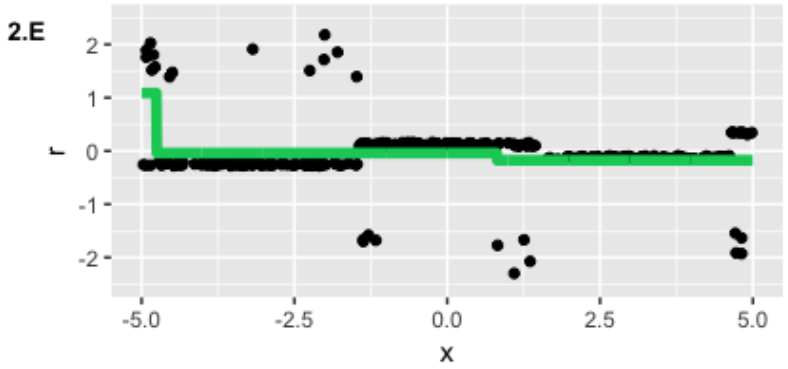
https://habrahabr.ru/company/ods/blog/327250/

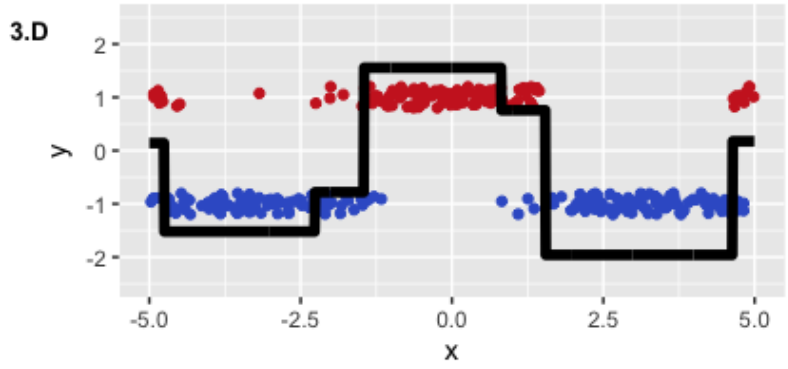
Проведем 3 итерации

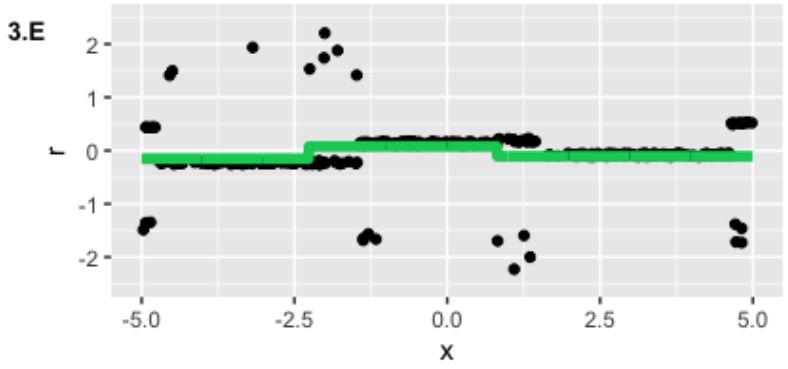












https://habrahabr.ru/company/ods/blog/327250/

XGBoost

Известный метод XGBoost полностью укладывается в описанную выше схему. В качестве моделей XGBoost берет деревья малой высоты (пеньки).

Замечание о величине остатков

Итак, в бустинге каждая новая модель тренируется на остатках r_i предыдущей модели. С теоретической точки зрения это означает, что происходит шаг против градиента функции потерь.

А можем ли мы регулировать длину этого шага? (слишком большие или слишком маленькие шаги – одинаково плохи).

Шаг бустинга

Напомним, что мы строим последовательность алгоритмов регрессии

$$a_1(X),\ldots,a_k(X),$$

которая дает окончательную модель

$$a_1(X)+\ldots+a_k(X)$$

А нельзя ли каждый новый алгоритм добавлять в сумму с некоторым коэффициентом (как в модели взвешенного голосования)?

Шаг бустинга: всё просто!

1. Пусть модели $a_1(X), a_2(X)$ построены по описанным ранее правилам.
2. Пусть a(X)= $a_1(X)+w*a_2(X)$ – новая модель, где вторая модель входит с некоторым весом *w.*
3. Для модели *a(X)* можно вычислить отступы M_i всех объектов тренировочной выборки (это будут выражения, зависимые от *w*).
4. Берем мажорирующую функцию f(M), которая использовалась при построении модели $a_2(X)$.
5. Составляем сумму $F(w)=f(M_1)+\ldots+f(M_n)$ по всем объектам тренировочной выборки.
6. Находим минимум функции F(w) как функции одной переменной.

Шаг бустинга: пример

Пусть в недавнем примере был построен алгоритм $a_2(X)$, улучшающий $a_1(X)$, дающий следующие ответы:

	x	$a_1(x)$	M_i	y	r_i	$a_2(X)$
A	-2	-0.6	0.6	-1	-0.55	-0.09
B	-1	-0.2	-0.2	1	1.22	1.22
C	0	0.2	-0.2	-1	-1.22	-1.22
D	1	0.6	0.6	1	0.55	0.55
E	2	1	1	1	0.37	-0.09

Шаг бустинга: пример

Нужно оптимальным образом выбрать параметр w в композиции алгоритмов

$$a(x)=a_1(x)+wa_2(x)$$

Для этого нужно считать отступы алгоритма $a(x)$.

	x	$a_1(x)$	M_i	y	r_i	$a_2(X)$	M_i для a(x)
A	-2	-0.6	0.6	-1	-0.55	-0.09	-(-0.6-0.09w)
B	-1	-0.2	-0.2	1	1.22	1.22	(-0.2+1.22w)
C	0	0.2	-0.2	-1	-1.22	-1.22	-(0.2-1.22w)
D	1	0.6	0.6	1	0.55	0.55	(0.6+0.55w)
E	2	1	1	1	0.37	-0.09	(1-0.09w)

Шаг бустинга: пример

Используя мажорирующую функцию e^{-M} (какую использовали при построении a_2), просуммировать ее значения для всех объектов:

$$\exp(-0.6 - x \times 0.09) + \exp(0.2 - x \times 1.22) + \exp(0.2 - x \times 1.22) + \exp(-0.6 - x \times 0.46) + \exp(-1 + x \times 0.09)$$

это будет суммарная функция потерь.

	x	$a_1(x)$	M_i	y	r_i	$a_2(X)$	M_i для a(x)
A	-2	-0.6	0.6	-1	-0.55	-0.09	-(-0.6-0.09w)
B	-1	-0.2	-0.2	1	1.22	1.22	(-0.2+1.22w)
C	0	0.2	-0.2	-1	-1.22	-1.22	-(0.2-1.22w)
D	1	0.6	0.6	1	0.55	0.55	(0.6+0.55w)
E	2	1	1	1	0.37	-0.09	(1-0.09w)

Шаг бустинга: пример

Находим точку минимума для

$$\exp(-0.6 - x \times 0.09) + \exp(0.2 - x \times 1.22) + \exp(0.2 - x \times 1.22) + \exp(-0.6 - x \times 0.46) + \exp(-1 + x \times 0.09)$$

численные методы дают ответ w=5.48.

Получаем ответы алгоритма $a(x)=a_1(x)+5.48*a_2(x)$

и ответы для задачи классификации:

	x	$a_1(x)$	M_i	y	r_i	$a_2(X)$	a(x)	ответ бустинга
A	-2	-0.6	0.6	-1	-0.55	-0.09	-1,0932	-1
B	-1	-0.2	-0.2	1	1.22	1.22	6,4856	1
C	0	0.2	-0.2	-1	-1.22	-1.22	-6,4856	-1
D	1	0.6	0.6	1	0.55	0.55	3,614	1
E	2	1	1	1	0.37	-0.09	0,5068	1

Литература

1. https://alexanderdyakonov.wordpress.com/2017/06/09/%D0%B3%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B9-%D0%B1%D1%83%D1%81%D1%82%D0%B8%D0%BD%D0%B3/
2. https://habrahabr.ru/company/ods/blog/327250/

Приложение: стекинг

Стекинг

Это один из самых популярных (и общих) способов построения ансамблей алгоритмов, т.е. использования нескольких алгоритмов для решения одной задачи машинного обучения. Известно, что если обучить несколько разных алгоритмов, то в задаче регрессии их среднее, а в задаче классификации — голосование по большинству, часто превосходят по качеству все эти алгоритмы.

Стекинг

Возникает вопрос: а почему, собственно, мы обязаны использовать либо усреднение либо голосование по большинству?

Можно же ансамблирование доверить очередному алгоритму (т.н. «метаалгоритму») машинного обучения!

Зафиксирует терминологию:

базовые алгоритмы – выдают ответы (метапризнаки), которые используются для настройки метаалгоритма.

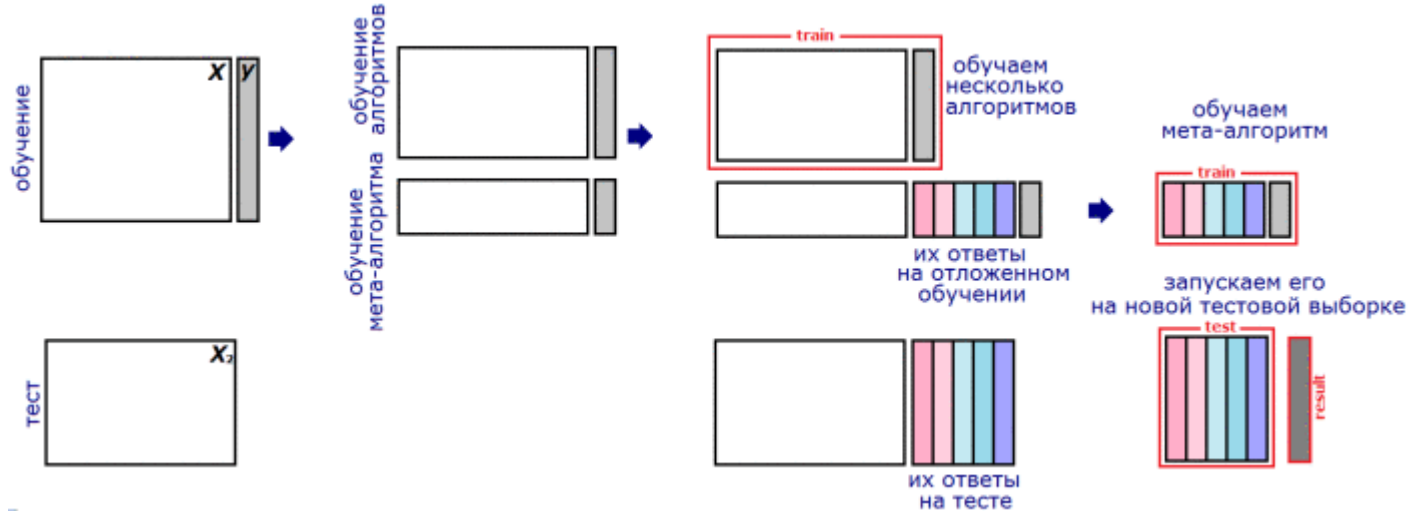
метаалгоритм – использует лишь ответы базовых алгоритмов и выдает окончательный ответ.

Простейшая схема стекинга

Тренировочную выборку делят на две части. На первой обучают базовые алгоритмы. Затем получают их ответы на второй части и на тестовой выборке.

Ответ каждого базового алгоритма – это новый признак (метапризнак). На метапризнаках второй части обучения настраивают метаалгоритм. Затем запускают его на метапризнаках теста и получают ответ.

Простейшая схема стекинга



Зачем еще разбивать тренировочную выборку?

Если этого не делать, то будет переобучение, поскольку в каждом метапризнаке будет «зашита» информация о значении всего целевого вектора.