# Урок 8

# Случайные леса

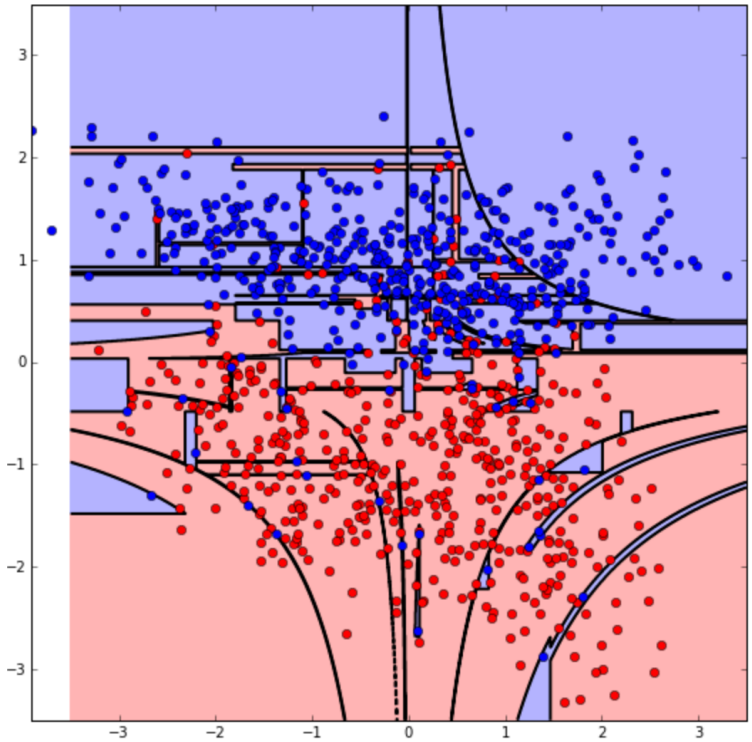
В прошлом уроке изучались решающие деревья и было установлено, что они способны восстанавливать очень сложные закономерности, следовательно, склонны к переобучению. Другими словами, деревья слишком легко подгоняются под обучающую выборку и получаются непригодными для построения прогнозов.

Но оказывается, решающие деревья очень хорошо подходят для объединения в композиции и построения одного непереобученного алгоритма на основе большого количества решающих деревьев.

## 8.1. Композиции деревьев

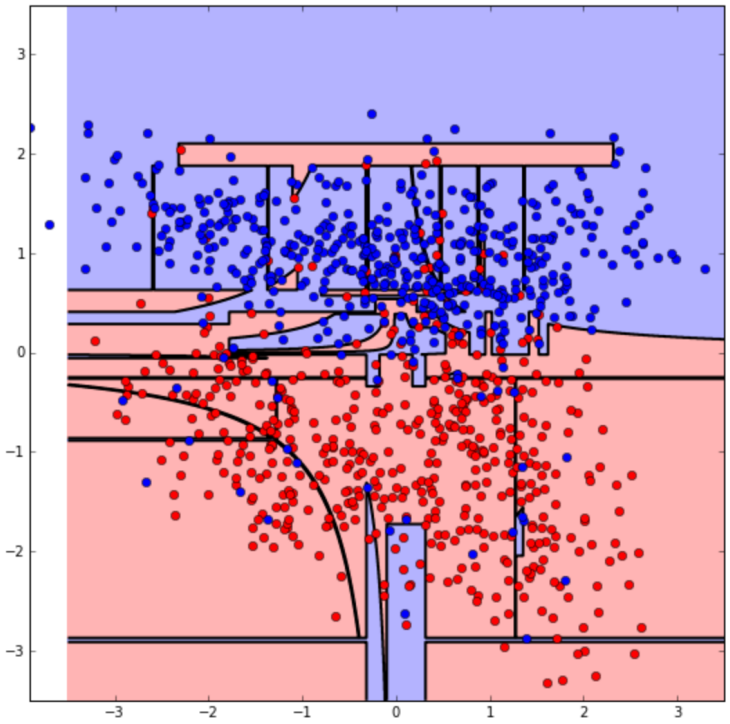
### 8.1.1. Основные недостатки решающих деревьев

Если взять сложную выборку и обучить на ней решающее дерево до конца, то есть пока в каждом из лепестков не останется по одному объекту, получившаяся разделяющая поверхность будет очень сложной:




Даже если какой-то объект попадает в «гущу другого класса», разделяющая поверхность пытается «уловить» его и выдать на нем правильный ответ.

Если немного изменить обучающую выборку, например выкинуть пару объектов, то обученное на получившейся выборке дерево все еще будет характеризоваться изрезанной и переобученной разделяющей поверхностью, но совершенно другой:




Разделяющая поверхность крайне неустойчива к изменению выборки. Другими словами, решающее дерево обладает следующими серьезными недостатками:

- сильно переобучается
- сильно меняется при небольшом изменении выборки

На самом деле, второй пункт можно будет превратить в достоинство с помощью композиции.

### 8.1.2. Композиция алгоритмов

Композиция — это объединение $N$ алгоритмов $b_1(x), ..., b_N(x)$ в один. Идея заключается в том, чтобы обучить алгоритмы $b_1(x), ..., b_N(x)$, а затем усреднить полученные от них ответы:

$$a(x) = \frac{1}{N}\sum_{n=1}^{N} b_n(x).$$

Это выражение непосредственно является ответом в задаче регрессии. В задачах классификации нужно будет взять знак от получившегося выражения:

$$a(x) = \text{sign}\,\frac{1}{N}\sum_{n=1}^{N} b_n(x),$$

Алгоритм $a(x)$, который возвращает среднее или знак среднего, называется композицией $N$ алгоритмов $b_1(x), ..., b_N(x)$, а они сами называются базовыми алгоритмами.

Например, пусть при решении задачи классификации с двумя классами использовались 6 базовых алгоритмов, которые на некотором объекте $x$ выдали следующие ответы:

$$-1, -1, 1, -1, 1, -1.$$

Ответ композиции этих 6 алгоритмов будет:

$$a(x) = \text{sign}\left(-\frac{2}{6}\right) = -1.$$

Попросту говоря, объект был отнесен к классу $-1$, так как за этот вариант «проголосовало» большинство базовых алгоритмов.

### 8.1.3. Рандомизация

Чтобы построить композицию, нужно сначала обучить $N$ базовых алгоритмов, причем их нельзя обучать на всей обучающей выборке, так как в этом случае они получаются одинаковыми, и в их усреднении не будет никакого смысла.

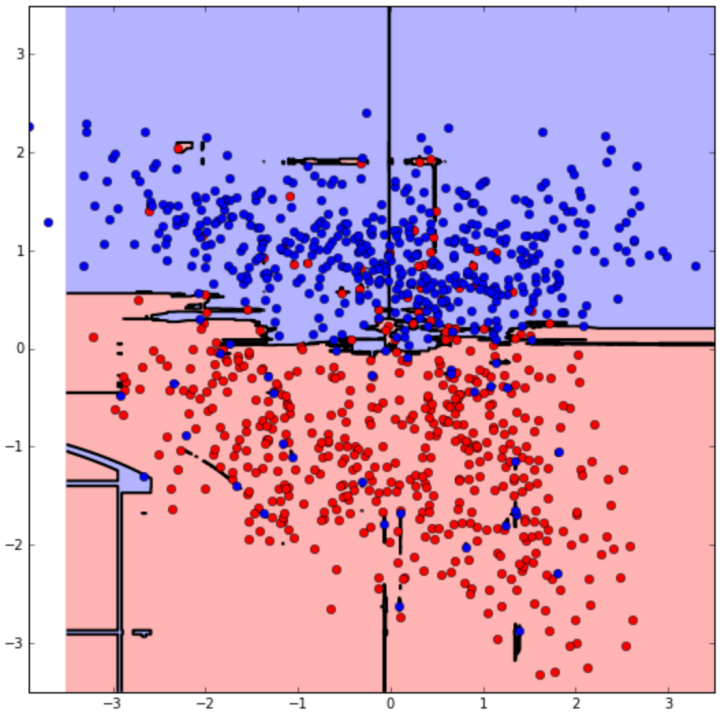
Использовать рандомизацию, то есть обучать базовые алгоритмы на разных подвыборках обучающей выборки, — это один из способов сделать базовые алгоритмы различными. А поскольку решающие деревья сильно меняются даже от небольших изменений обучающей выборки, такая рандомизация значительно повышает различность базовых алгоритмов.

Так называемый бутстрап — один из популярных подходов к построению подвыборок. Он заключается в том, что из обучающей выборки длины $\ell$ выбирают с возвращением $\ell$ объектов. При этом новая выборка также будет иметь размер $\ell$, но некоторые объекты в ней будут повторятся, а некоторые объекты из исходной выборки в нее не попадут. Можно показать, что в бутстрапированной выборке будет содержаться в среднем 63% различных объектов исходной выборки.

Другой подход к рандомизации — генерация случайного подмножества обучающей выборки. Размер этого случайного подмножества является гиперпараметром. Например, можно случайно взять половину исходной выборки и обучить на ней базовый алгоритм. Этот подход несколько проигрывает бутстрапу, так как содержит гиперпараметр, в то время как бутстрап без какой-либо настройки выдает подвыборку.

### 8.1.4. Композиция деревьев

Если с помощью бутстрапа построить 100 базовых решающих деревьев и объединить их в композицию, разделяющая поверхность будет все еще сложная, но уже гораздо менее переобученная:




Разделяющая поверхность уже не подгоняется под большую часть попавших в гущу чужого класса объектов и в целом хорошо разделяет два класса. Увеличением количества базовых алгоритмов можно устранить оставшиеся погрешности.

## 8.2. Смещение и разброс

В этом разделе речь пойдет о разложении ошибки на шум, смещение и разброс. Эта техника позволяет глубже понять причины, почему усреднение алгоритмов позволяет повысить качество.

### 8.2.1. Разложение ошибки: шум, смещение и разброс

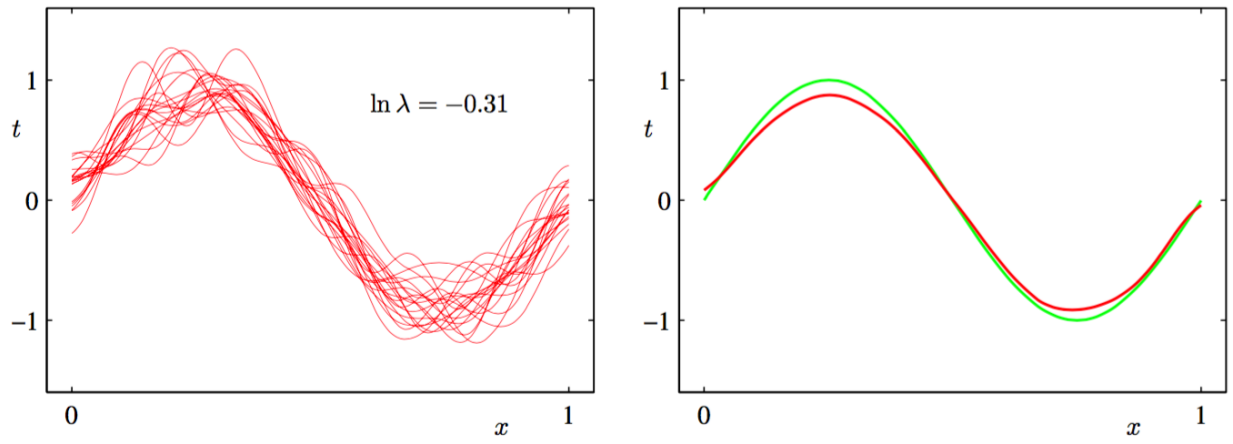
Ошибка алгоритма на новых тестовых данных складывается из трех компонент: шума, смещения и разброса. При этом все они характеризуют разные аспекты данных и модели, с помощью которой решается задача на этих данных:

- **Шум** — компонента ошибки алгоритма, которая будет проявляться даже на идеальной модели в этой задаче. Другими словами, шум является характеристикой данных и будет проявляться, какая бы модель не использовалась.

Пусть обучающая выборка генерируется из некоторого вероятностного распределения. На каждой конкретной обучающей выборке можно обучить некоторую модель и использовать обученную модель на тестовой выборке.

- **Смещение** — отклонение, усредненного по различным обучающим выборкам, прогноза заданной модели от прогноза идеальной модели.
- **Разброс** — дисперсия ответов моделей, обученных по различным обучающим выборкам. Разброс характеризует то, насколько сильно прогноз алгоритма зависит от конкретной обучающей выборки.

Продемонстрировать разложение ошибки на смещение и разброс можно на следующем примере. Рассматривается задача регрессии: требуется аппроксимировать истинную зависимость (изображена на правом графике зеленым) полиномом третьего порядка по обучающей выборке. Обучающая выборка представляет собой 10 случайных точек истинной зависимости, к которым был добавлен случайный шум. На левом графике изображены полиномы, получающиеся для различных обучающих выборок.




Усредненный полином (изображен красной линией на правом рисунке) практически идеально попадает в истинную зависимость, но каждый полином по отдельности существенно от нее отличается. Другими словами, используемое семейство алгоритмов обладает низким смещением, но довольно большим разбросом.

Линейные модели способны восстанавливать только линейные зависимости, а, следовательно, в случае нелинейных задач, которых подавляющее большинство, смещение при использовании таких алгоритмов будет большим. Разброс, наоборот, будет маленьким из-за малого числа параметров, сравнимого с количеством признаков. Вряд ли параметры линейной модели сильно поменяются при незначительном изменении обучающей выборки. Решающие деревья — полная противоположность. Они характеризуются низким смещением, то есть способны восстанавливать сложные закономерности, и большим разбросом: решающие деревья сильно меняются даже при небольших изменениях обучающей выборки.

### 8.2.2. Смещение и разброс композиции алгоритмов

При вычислении композиции базовых алгоритмов (с одинаковым смещением) смещение композиции совпадает со смещением отдельного базового алгоритма. Таким образом, поскольку деревья характеризуются низким

смещением, то же самое будет верно и для композиции деревьев. Следовательно, композиции деревьев тоже способны восстанавливать сложные закономерности.

Разброс композиции уже отличается от разброса одного базового алгоритма:

$$\left(\begin{smallmatrix}\textbf{разброс}\\ \textbf{композиции}\end{smallmatrix}\right) = \frac{1}{N}\left(\begin{smallmatrix}\textbf{разброс одного}\\ \textbf{базового алгоритма}\end{smallmatrix}\right) + \left(\begin{smallmatrix}\textbf{корелляция между}\\ \textbf{базовыми алгоритмами}\end{smallmatrix}\right).$$

Если базовые алгоритмы независимы, то есть их прогнозы не коррелируют между собой, выражение упрощается:

$$\left(\begin{smallmatrix}\textbf{разброс}\\ \textbf{композиции}\end{smallmatrix}\right) = \frac{1}{N}\left(\begin{smallmatrix}\textbf{разброс одного}\\ \textbf{базового алгоритма}\end{smallmatrix}\right).$$

Фактически, композиция достаточного количества некоррелированных алгоритмов может дать идеальный алгоритм. Но, к сожалению, базовые алгоритмы всегда получаются в той или иной степени коррелированы, так как обучаются на подвыборках одной выборки. Таким образом, возникает необходимость уменьшения корелляции базовых алгоритмов.

### 8.2.3. Уменьшение корреляции базовых алгоритмов

Существуют следующие два подхода по уменьшению корреляции базовых алгоритмов:

1. **Беггинг:** Обучение базовых алгоритмов происходит на случайных подвыборках обучающей выборки. Причем чем меньше размер случайной подвыборки, тем более независимыми получаются базовые алгоритмы.

2. **Метод случайных подпространств:** выбирается случайное подмножество признаков (столбцов матрицы «объекты–признаки») и очередной базовый алгоритм обучается только на этих признаках. Доля выбираемых признаков является гиперпараметром этого метода.

Два данных подхода — бэггинг и метод случайных подпространств — можно объединять и использовать одновременно.

## 8.3. Случайные леса

В этом разделе речь пойдет о случайных лесах, которые являются одним из лучших способов объединения деревьев в композиции.

### 8.3.1. Случайный лес

Ранее были получены следующие результаты:
- Ошибка может быть разложена на смещение и разброс.
- Смещение композиции близко к смещению одного базового алгоритма.
- Разброс при построении композиции уменьшается, причем тем сильнее, чем менее коррелированы базовые алгоритмы.

Рассмотренных в прошлый раз способов понижения корреляции между базовыми алгоритмами (бэггинг и метод случайных подпространств) оказывается недостаточно. Чтобы базовые алгоритмы были еще менее скореллированными, имеет смысл сделать случайным их процесс построения.

### 8.3.2. Рандомизация процесса построения решающих деревьев

Процесс построения решающих деревьев представляет собой жадный алгоритм, работающий до выполнения критерия останова.

Пусть на некотором шаге алгоритма необходимо разбить вершину $m$, в которой оказалась выборка $X_m$, на две. В качестве условия разбиения используется сравнение $j$-го признака с порогом $t$:

$$[x^j \leq t].$$

Параметры $j$ и $t$ выбираются исходя из условия минимизации функции ошибки $Q(X_m, j, t)$:

$$Q(X_m, j, t) \to \min_{j,t}.$$

Рандомизировать процесс построения можно, если в задаче поиска оптимальных параметров выбирать $j$ из случайного подмножества признаков размера $q$. Оказывается, что этот подход действительно позволяет сделать деревья менее коррелированными.

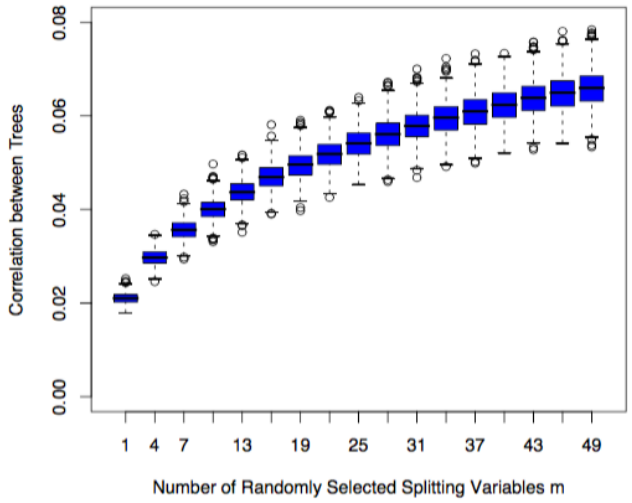



Рис. 8.1: Зависимость корреляции между деревьями от параметра $q$

По графику видно, что чем меньше «простор для выбора лучшего разбиения», то есть чем меньше $q$, тем меньше корреляции между получающимися решающими деревьями. Случай $q = 1$ соответствует абсолютно случайному выбору признака.

Для $q$ есть некоторые рекомендации, которые неплохо работают на практике:

- В задаче регрессии имеет смысл брать $q = d/3$, то есть использовать треть от общего числа признаков.
- В задаче классификации имеет смысл брать $q = \sqrt{d}$.

### 8.3.3. Алгоритм построения случайного леса

Чтобы построить случайный лес из $N$ решающих деревьев, необходимо:

1. Построить с помощью бутстрапа $N$ случайных подвыборок $\tilde{X}_n$, $n = 1, ..., N$.
2. Каждая получившаяся подвыборка $\tilde{X}_n$ используется как обучающая выборка для построения соответствующего решающего дерева $b_n(x)$. Причем:
   - Дерево строится, пока в каждом листе окажется не более $n_{min}$ объектов. Очень часто деревья строят до конца ($n_{min} = 1$), чтобы получить сложные и переобученные решающие деревья с низким смещением.
   - Процесс построения дерева рандомизирован: на этапе выбора оптимального признака, по которому будет происходить разбиение, он ищется не среди всего множества признаков, а среди случайного подмножества размера $q$.
   - Следует обратить особое внимание, что случайное подмножество размера $q$ выбирается заново каждый раз, когда необходимо разбить очередную вершину. В этом состоит основное отличие такого подхода от метода случайных подпространств, где случайное подмножество признаков выбиралось один раз перед построением базового алгоритма.
3. Построенные деревья объединяются в композицию:
   - В задачах регрессии $a(x) = \frac{1}{N}\sum_{n=1}^{N} b_n(x)$;
   - В задачах классификации $a(x) = \text{sign}\, \frac{1}{N}\sum_{n=1}^{N} b_n(x)$.

Одна из особенностей случайных лесов: они не переобучаются при росте числа базовых алгоритмов.

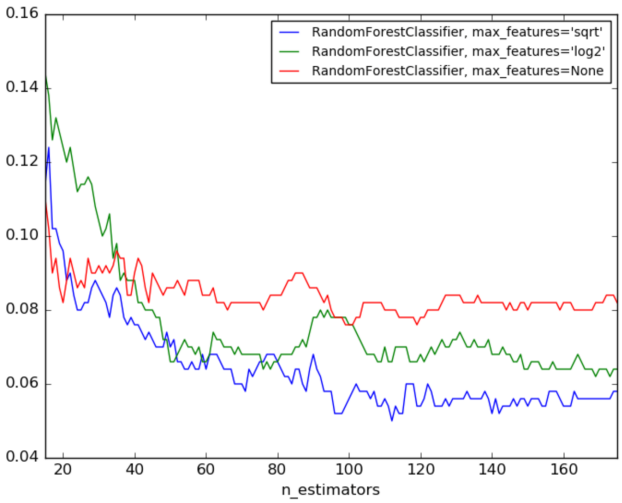



Рис. 8.2: Зависимость качества случайного леса от значения параметра $q$.

По графику видно, что ошибка на тесте сначала уменьшается с ростом числа базовых алгоритмов, а затем выходит на асимптоту. Не происходит роста ошибки при росте числа базовых алгоритмов.

## 8.4. Трюки со случайными лесами

Случайный лес обладает рядом интересных особенностей.

### 8.4.1. Возможность распараллеливания

Поскольку каждое дерево обучается независимо от всех остальных базовых решающих деревьев, его можно обучать на отдельном ядре или отдельном компьютере.

Фактически, данная задача допускает идеальное распараллеливание: скорость вычислений пропорциональна количеству задействованных вычислительных ядер.

### 8.4.2. Оценивание качества случайного леса

Каждое дерево из случайного леса обучается на бутстрапированной выборке, в которую попадают приблизительно 63% объектов полной выборки. Таким образом, около 37% объектов выборки не использовались при обучении этого дерева, а значит их можно использовать для оценки обобщающей способности случайного леса.

Такой подход носит название out-of-bag и позволяет оценивать качество леса без использования отложенной выборки или кросс-валидации. Формула для оценки качества случайного леса из $N$ деревьев в рамках подхода out-of-bag имеет вид:

$$OOB = \sum_{i=1}^{\ell} L\left(y_i, \frac{1}{\sum_{n=1}^{N}[x_i \notin X_n]} \sum_{n=1}^{N}[x_i \notin X_n] b_n(x_i)\right).$$

Эта формула устроена следующим образом. Для каждого объекта $x_i$ из обучающей выборки вычисляется средний прогноз по тем деревьям, в обучающую выборку которых не входит объект $x_i$:

$$\frac{1}{\sum_{n=1}^{N}[x_i \notin X_n]} \sum_{n=1}^{N}[x_i \notin X_n] b_n(x_i).$$

Для полученного прогноза вычисляется значение ошибки. В качестве оценки качества случайного леса используется сумма таких значений для всех элементов выборки.

Также с помощью случайных лесов и out-of-bag можно отбирать наиболее важные признаки. Об этом пойдет речь в следующем курсе.