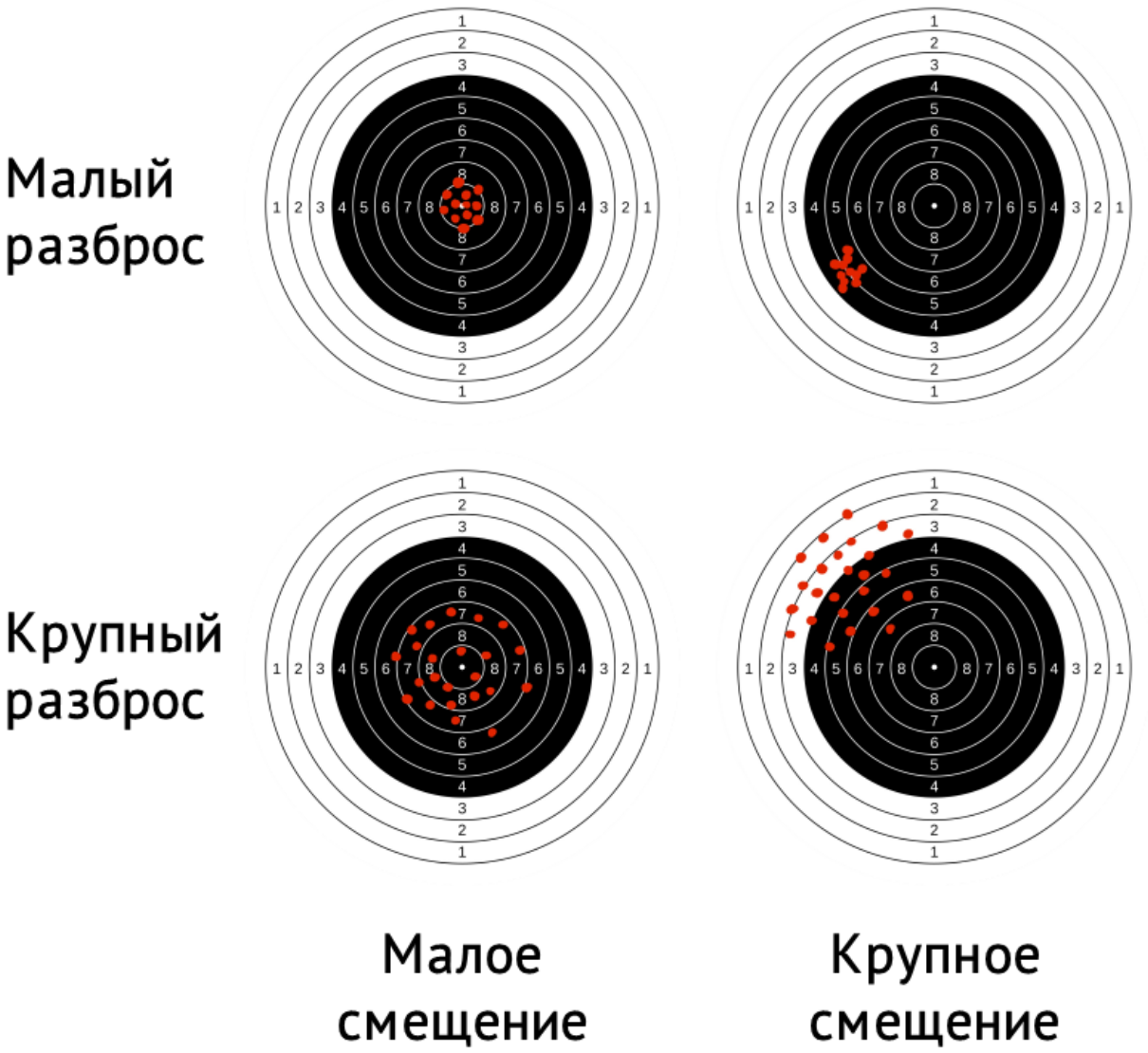
Лекция 9. Ансамблирование алгоритмов

Все время до этого мы говорили о том, как наилучшим образом построить конкретный алгоритм, который решал бы поставленную задачу. Но что если при обучении удалось построить несколько моделей, которые несколько отличаются в точности но, например, каждая из этих моделей отлично справляется на определенной подвыборке из нашего датасета? А что если попробовать объединить между собой эти модели?

Прежде чем мы перейдем к подходам по объединению моделей, давайте обсудим, а что именно мы можем улучшить, если сразу несколько моделей будут давать прогноз вместе?

Смещение и разброс

Разброс -дисперсия ответов алгоритма
Смещение - матожидание разности между истинным ответом и выданным алгоритмом.



Разброс и смещение можно проиллюстрирвать на примере мишени. Идеальная ситуация - это когда малый разброс и малое смещение, т.е. попадание четко в яблочко, применимо к машинному обучению - модель с постоянным мастерством прогнозирует верный класс.

Но в реальности так получается далеко не всегда. Например, ответы могут быть сгруппированы около другой точки (с малым разбросом, но большим смещением), или же точки могут в среднем попадать в цель, но при этом иметь довольно сильный разброс. Идеальной стратегией видится обучить алгоритм малому разбросу, а потом уже немного подвинуть ответы в сторону правильного алгоритма. Однако подобный подход предполагает переход на более сложную

модель, т.к. мы знаем о смещении только для конкретного набора и данных и лучшее что мы можем сделать - просто подобрать константу смещения, но подобные выводы уже могут быть невалидны при переходе к новым данным. В таком случае правильнее переходить к другой модели, где уже в свою очередь сложно спрогнозировать какой именно будет разброс.

Давайте предположим, что мы переходим к более сложной модели. В этом случае зависимость ошибки от сложности модели можно описать зависимостью на слайде. Эмпирический риск на контрольной выборке сначала снижается, но после прохождения некоторого уровня оптимальной сложности начинает расти. Чаще всего это является следствием уменьшения смещения и увеличением разброса.

Для простых моделей характерно недообучение, т.е. модель не может выцепить характерные для задачи зависимости и обладают большим смещением. Сложные модели в свою очередь склонны к переобучению, т.е. очень сильно подстраиваемся под обчающую выборку.

А что если нам взять несколько простых алгоритмов, у каждого из которых есть свое смещение и разброс и попробовать усреднить их результаты? Так и появилась идея ансамблей.

Ансамбль (Ensemble) - алгоритм, который состоит из нескольких алгоритмов машинного обучения, чаще всего, простых алгоритмов. Алгоритмы, из которых состоит ансамбль, называются **базовыми алгоритмами (base learners).**

Ансамблирование (ensemble learning) - процесс обучения ансамбля.

Пример для регрессии - усреднение ответов нескольких алгоритмов:

$$\hat{y} = \frac{f_1(x) + f_2(x) + ... + f_n(x)}{n}$$

Пример для классификации - голосование большинством:

$$\hat{y} = mode(f_1(x), f_2(x), ..., f_n(x))$$

Основные виды ансамблей

Подход	**Комментарий**
Бэггинг (bagging)	Независимое обучение алгоритмов и агрегация их ответов при использование (голосование, усреднение, …)
Бустинг (boosting)	Последовательное обучение базовых алгоритмов, при котором каждый следующий алгоритм обучается с учётом ошибок уже обученных алгоритмов.
Стекинг (stacking)	Независимое обучение сильных моделей, которые в дальнейшем передают свои прогнозы как признаки для верхнеуровневой модели.

Бэггинг

Идея **бэггинга** (**bagging**, **bootstrap aggregation**) заключается в следующем. Пусть обучающая выборка состояла из n объектов. Выберем из неё n примеров равновероятно, с возвращением. Этот этап называется бутстреп - генерация подвыборок сэмплированием с возвращением.

Метод бутстрэпа заключается в следующем. Пусть имеется выборка X размера N. Равномерно возьмем из выборки N
 объектов с возвращением. Это означает, что мы будем N
 раз выбирать произвольный объект выборки (считаем, что каждый объект «достается» с одинаковой вероятностью
$\frac{1}{N}$), причем каждый раз мы выбираем из всех исходных N

объектов. Можно представить себе мешок, из которого достают шарики: выбранный на каком-то шаге шарик возвращается обратно в мешок, и следующий выбор опять делается равновероятно из того же числа шариков. Отметим, что из-за возвращения среди них окажутся повторы. Обозначим новую выборку через X_1. Повторяя процедуру M раз, сгенерируем M подвыборок $\{X_1, X_2, ..., X_M\}$. Теперь мы имеем достаточно большое число выборок и можем оценивать различные статистики исходного распределения.

Итак, получим новую выборку X_1, в которой некоторых элементов исходной выборки не будет, а какие-то могут войти несколько раз. С помощью некоторого алгоритма b обучим на этой выборке модель $b_1(x) = b(x, X_1)$. Повторим процедуру: сформируем вторую выборку X_2 из n элементов с возвращением и с помощью того же алгоритма обучим на ней модель $b_2(x) = b(x, X_2)$. Повторив процедуру m раз, получим m моделей, обученных на m выборках. Теперь, чтобы сформировать финальный прогноз, нам достаточно усреднить прогнозы всех моделей:

Рассмотрим задачу регрессии с базовыми алгоритмами $b_1(x), b_2(x), ..., b_m(x)$. Как мы уже рассматривали ранее, для регрессии функцией потерь будет следующее выражение:

$$L(b_i(x), y(x)) = \epsilon_i(x) = (b_i(x) - y(x))^2$$

Запишем математическое ожидание среднеквадратичной ошибки:

$$E_1 = \frac{1}{N} E_x \sum_{i=1}^{N} \epsilon_i^2(x)$$

Пусть ошибки несмещены и некоррелированы

$$E_x \epsilon_i(x) = 0, \;\; E_x \epsilon_i(x) \epsilon_j(x) = 0, i \neq j$$

Построим теперь новую функцию регрессии, которая будет усреднять ответы построенных нами функций:

$\hat{y} = \frac{\sum_{i=1}^{m} b_i(x)}{m}$

Найдем ее среднеквадратичную ошибку:

$$E_m = E_x(\frac{1}{m}\sum_{i=1}^{m}(b_i(x) - y(x))^2 =$$

$$= E_x(\frac{1}{m}\sum_{i=1}^{m}\epsilon_i)^2 =$$

$$= \frac{1}{m^2}E_x(\sum_{i=1}^{m}\epsilon_i^2(x) + \sum_{i\neq j}\epsilon_i(x)\epsilon_j(x)) = \frac{1}{m}E_1$$

Таким образом, усреднение ответов позволило уменьшить средний квадрат ошибки в m раз!

Бэггинг позволяет снизить дисперсию (разброс) обучаемого классификатора, уменьшая величину, на сколько ошибка будет отличаться, если обучать модель на разных наборах данных, или другими словами, предотвращает переобучение. Эффективность бэггинга достигается благодаря тому, что базовые алгоритмы, обученные по различным подвыборкам, получаются достаточно различными, и их ошибки взаимно компенсируются при голосовании, а также за счёт того, что объекты-выбросы могут не попадать в некоторые обучающие подвыборки.

Самым распространенным примером бэггинга является алгоритм Random Forest или Случайный лес. В данном подходе в роли базовых алгоритмов выступают решающие деревья.

Random Forest

Случайный лес строится по следующему правилу:

1. Обучение:

   а. При помощи бутстрепа формируется случайная подвыборка X_i того же размера, что и X.

   б. В процессе обучения каждого дерева **в каждой вершине** случайно выбираются $n < N$признаков, где N - полное число признаков, и среди них жадным алгоритмом выбирается оптимальный предикат.

2. Прогнозирование

   a. Для прогноза ансамбля при решении задачи регрессии, усредняем ответы каждого дерева, для классификации берём самый популярный класс.

Когда мы говорили про деревья решений, то обсуждали параметры отдельного дерева, а когда мы обучаем сразу несколько деревьев, то возникает закономерный вопрос: а по какому принципу подбирать параметры для ансамбля?

Насколько глубокие деревья строить?

Как мы уже проговорили, ошибка модели, которую мы оптимизиурем, состоит из смещения и разброса. Разброс мы уменьшаем с помощью процедуры бэггинга. На смещение бэггинг не влияет, но конечно мы нацелены на его уменьшение. Поэтому в идеале мы должны предъявлять такое требование к каждому из базовых алгоритмов. У неглубоких деревьев малое число параметров, поэтому дерево способно запомнить только базовые статистики датасета. Поэтому предсказание на тестовом объекте будет стабильным (низкий разброс), но не точным (высокое смещение). Глубокие же деревья способны к более сильной обобщающей способности, поэтому предсказание на тестовом объекте будет сильнее меняться в зависимости от обучающей подвыборки, зато в среднем будет близко к истине (высокая дисперсия, низкое смещение). На основании этих выводов предпочтительнее использовать более глубокие деревья, но при этом критично важно проверять модель на переобучение.

А какое количество деревьев использовать?

Увеличение числа элементарных алгоритмов в ансамбле уменьшает разброс. Лучшим вариантом будет построить график ошибки от числа деревьев и ограничить размер леса в тот момент, когда ошибка перестанет значимо уменьшаться и выйдет на некоторую константу.

Вторым практическим ограничением на количество деревьев может быть время работы ансамбля. Однако есть положительное свойство случайного леса: случайный лес можно строить и применять параллельно, что сокращает время работы, если у нас есть возможность распараллелить обучение на несколько потоков. Но тем не менее, все же лучше предварительно оценить насколько сильное увеличение влияет на ошибку и выделить некоторую оптимальную зону, где необходимо искать количество базовых решающих деревьев.

Бустинг

В бэггинге и случайном лесе базовые алгоритмы обучаются независимо друг от друга на разных подвыборках, а в бустинге обучение происходит последовательно. Каждый последующий алгоритм мы обучаем так, чтобы он исправлял ошибки предыдущих и повышал метрики качества всего ансамбля. Как следствие, итоговый ансамбль в основном стремится уменьшить смещение, по сравнению с отдельным базовым алгоритмом, при этом уменьшение разброса также будет происходить, но в меньшей степени. Основная цель бустинга – уменьшение смещения, поэтому базовыми алгоритмами чаще всего выступают алгоритмы с высоким смещением и небольшим разбросом. В отличии от Случайного леса, в данном случае более приоритетно использовать деревья небольшой глубины- не больше 3-4 уровней. Ещё одной важной причиной для выбора моделей с высоким смещением в качестве базовых является то, что такие модели, как правило, быстрее учатся. Это важно для их последовательного обучения, которое может стать очень дорогим по времени, если на каждой итерации будет учиться сложная модель. На текущий момент основным видом бустинга с точки зрения применения на практике является **градиентный бустинг.**

С точки зрения математики это довольно сложный алгоритм, который мы не будем рассматривать в рамках данного курса.

Стекинг

Основные ограничения предыдущих алгоритмов заключаются в следующем:

1. Базовые алгоритмы должны быть из одного семейства (например, только SVM или только Решающие деревья)
2. Общий результат ансамбля - агрегация базовых алгоритмов посредством усреднения или голосования.

Стекинг - алгоритм, который учитывает эти недочеты, а именно:

1. Базовые алгоритмы могут совершенно разные
2. Результаты базовых алгоритмов являются признаками для модели более высокого уровня. Т.е. финальное решающее правило получается не методом агрегации, а при помощи ещё одного алгоритма машинного обучения

Обучение стекинга проходит в несколько этапов:

1. общая выборка разделяется на тренировочную и тестовую
2. базовые алгоритмы обучаются на тренировочной подвыборке. При этом мы стараемся строить как можно более сложную модель, минимизируя разброс и смещение.
3. Прогнозы каждой модели передаем в мета-модель как метапризнаки. Также, помимо метапризнаков, метамодель может принимать на вход и исходное признаковое пространство, если это необходимо в рамках решаемой задачи.

С точки зрения смещения и разброса стекинг не имеет прямой интерпретации, так как не минимизирует напрямую ни ту, ни другую компоненту ошибки. Удачно работающий стекинг просто уменьшает ошибку, и, как следствие, её компоненты тоже будут убывать.

Здесь может возникнуть вполне резонный вопрос: если в качестве базовых алгоритмов мы можем использовать любой набор, то какой алгоритм выбрать в роли метамодели? Ответ один: необходимо выбрать тот, который будет наилучшим образом оптимизировать выбранный функционал качества. Основное требование, которое стоит учесть, это то, что выбранный алгоритм должен хорошо работать с выбранными метриками что базовых алгоритмов, что финальной метамодели.

Подытожим

Ошибку модели грубо можно разложить на смещение и сброс. В лоб повлиять на эти составляющие мы не можем, т.к. при изменении подхода, мы можем улучшить одно слагаемое, но при этом ухудшить другое. Одним из хороших выходов является объединить несколько базовых алгоритмов. Существует множество различных подходов к ансамблированию, основные из них это бэггинг - усреднение ответов нескольких параллельно обучающихся алгоритмов, бустниг - усреднение ответов нескольких последовательно обучающихся алгоритмов и стекинг - построение метамодели, признаками которой являются результаты работы базовых алгоритмов.