Ансамбли моделей

Идея

1. Обучаем много моделей
2. Аггрегируем их ответы
3. Профит!

Проблемы?

Мотивация

1. Мудрость толпы
2. Теорема Кондорсе о присяжных

Теорема Кондорсе о присяжных

Если каждый член жюри присяжных имеет независимое мнение, и если вероятность правильного решения члена жюри больше 0.5, то тогда вероятность правильного решения присяжных в целом возрастает с увеличением количества членов жюри, и стремится к единице. Если же вероятность быть правым у каждого из членов жюри меньше 0.5, то вероятность принятия правильного решения присяжными в целом монотонно уменьшается и стремится к нулю с увеличением количества присяжных.

3. Покупка качества за железо

Идеи аггрегации

- усреднение
- линейные комбинации
- смеси экспертов
- бустинг
- стэкинг
- логические правила
- ...

Наблюдение

Ансамбли работают лучше, если их компоненты (отображения, базовые алгоритмы) различны, еще лучше — если исправляют ошибки друг друга.

Источники разнообразия

- Предобработка
  - декомпозиция данных: по признакам / по объектам
  - различные преобразования признаков
- Сэмплирование
  - по объектам (bootstrap, undersampling, oversampling)
  - по признакам
- Алгоритмы одного семейства с разными параметрами
- Разные семейства алгоритмов

Bootstrap

Имеется выборка из N объектов. Выбираем равновероятно N объектов с возвращением. Повторяем M раз. Оцениваем статистики (что-то делаем).

Bagging

Bootstrap aggregation.

1. С помощью bootstrap генерируем M выборок
2. Обучаем модели независимо на каждой выборке
3. Получаем независимое предсказание каждой модели
4. Принимаем окончательное решение (усреднение или голосование)

Bagging. Математика

- Смещение композиции равно смещению одного алгоритма
- **Если предположить, что базовые алгоритмы некоррелированы**, то дисперсия композиции в M раз меньше дисперсии отдельного алгоритма.

Bagging. Свойства

- Простой и достаточно мощный метод
- Чем более независимы отдельные решающие функции в ансамбле тем лучше
- Хорошо работает, если в базовых алгоритмах есть случайность и базовые алгоритмы из разных семейств
- Легко параллелится

Bagging. Преимущества и недостатки

+ легко реализовать
+ эффективен
+ снижает разброс
+ предотвращает переобучение
− довольно простой класс отображений

Random Forest

Решающие деревья могут иметь низкое смещение, но легко переобучаются — хороший выбор для бэггинга.
Рандомизация по

1. объектам (бутстрап)
2. признакам (подмножество признаков в узле)

Используются глубокие деревья (высокая дисперсия, низкое смещение).
Практическая рекомендация — брать корень из числа всех признаков для классификации и треть признаков для регрессии.
Нельзя переобучить числом деревьев.

Boosting. Идея

- Зачем обучать независимо?
- Зачем обучать глубокие деревья?

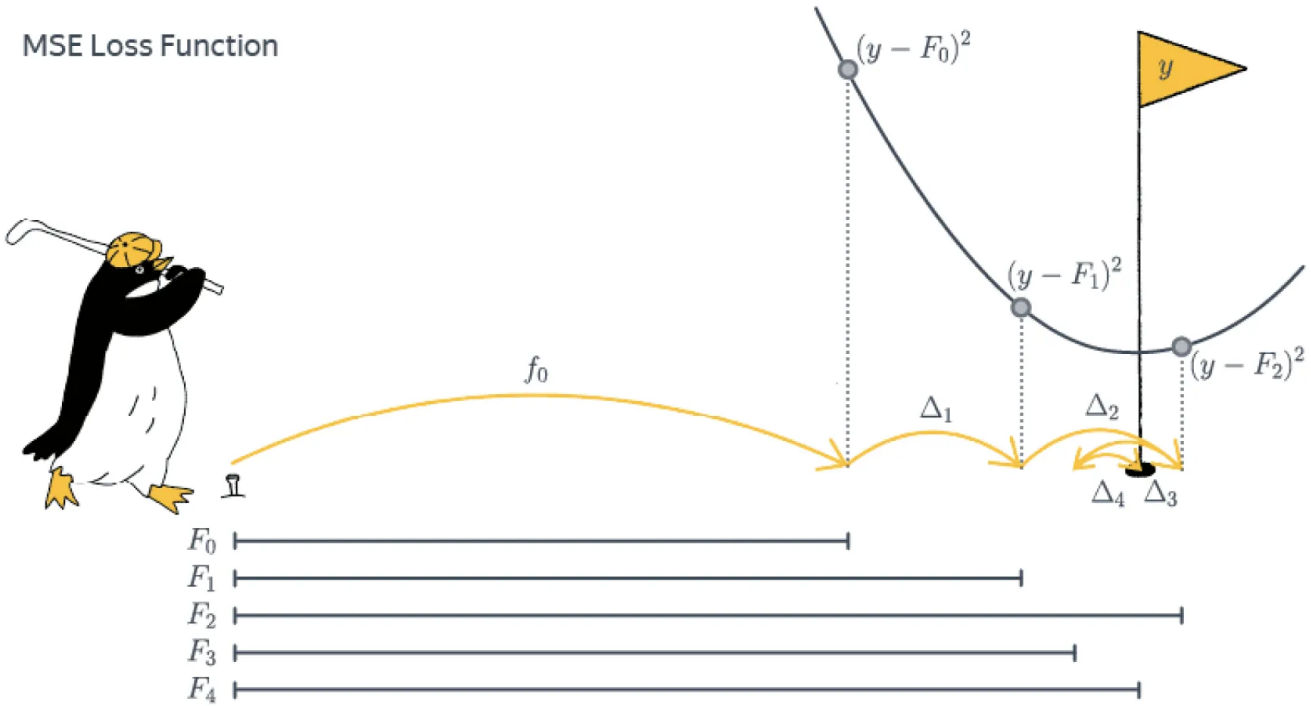
Будем жадно строить ансамбль, в которой каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов.

Boosting

Метод взвешенного усреднения моделей. В качестве базовых алгоритмов часто выбирают алгоритмы с высоким смещением и небольшим разбросом.

- AdaBoost.
- Градиентный бустинг

Градиентный бустинг над решающими деревьями



Bagging vs Boosting

Используют M базовых классификаторов

- Бустинг использует последовательное обучение
- Бэггинг использует параллельное обучение

Генерируют несколько наборов данных для обучения

- Бустинг определяет вес данных, чтоб утяжелить тяжелые случаи
- Бэггинг имеет невзвешенные данные

Принимают окончательное решение, усредняя N классификаторов

- В бустинге определяются веса для них
- В бэггинге они равнозначны

Уменьшают дисперсию и обеспечивают более высокую стабильность

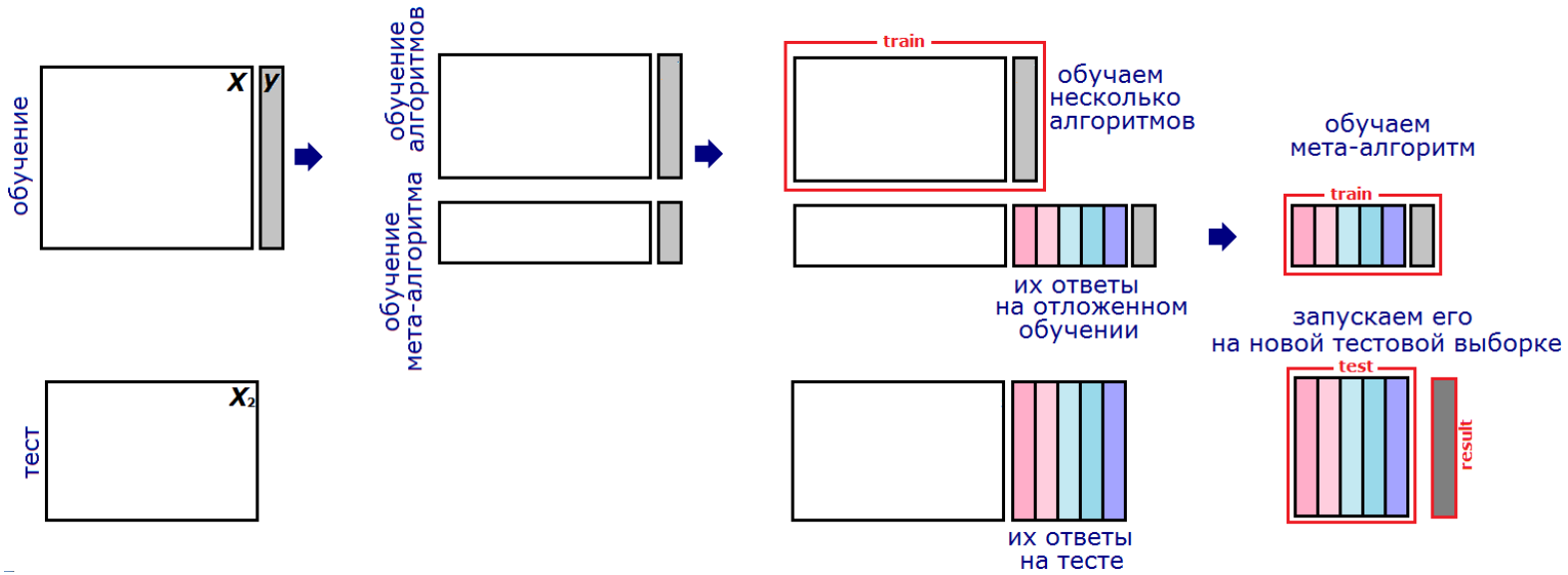
- Бэггинг может решить проблему переобучения
- Бустинг пытается уменьшить смещение, но может увеличить проблему переобучения

Базовые модели

- Бэггинг — сложные (низкое смещение, высокая дисперсия)
- Бустинг — простые (высокое смещение, низкая дисперсия)

Blending

Обучаем модель на выходах моделей.



Недостаток: ни базовые алгоритмы, ни метаалгоритм не используют всю обучающую выборку

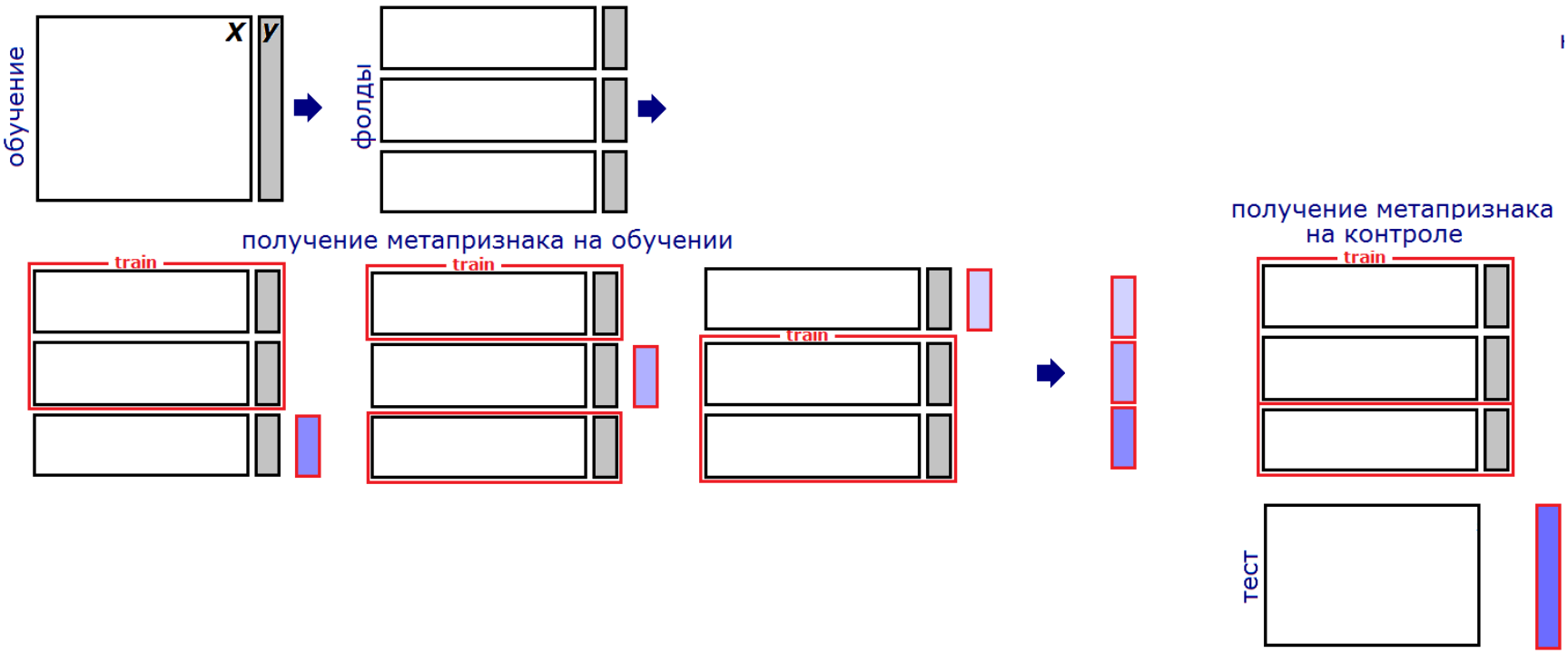
Стэкинг

Боремся за использование всей обучающей выборки

- разобьем выборку на фолды
- перебирая фолды обучаем базовый алгоритм на всех кроме одного
- на оставшемся получаем оценки (метапризнаки)
- для получения аналогичных оценок на тестовой выборке базовые алгоритмы обучают на всей обучающей выборке

Метапризнаки на обучении и на тесте разные!

Стэкинг


обучение
x
y
фолды
получение метапризнака на обучении
train
train
train
получение метапризнака
на контроле
train
тест

Stacking

- Вычислительно затратный
- Легко переобучиться
- Важно и нужно использовать базовые алгоритмы разной природы
- Разные признаковые пространства
- Недооптимизация базовых алгоритмов
- Сильная регуляризация в базовых алгоритмах
- Иногда хорошо работает обучение не на целевой признак, а на разницу между целевым и каким-то другим
- Feature Engineering над метапризнаками (попарные произведения)