Композиции деревьев

Артём Филатов

23 января 2018 г.

Где деревья проигрывают?

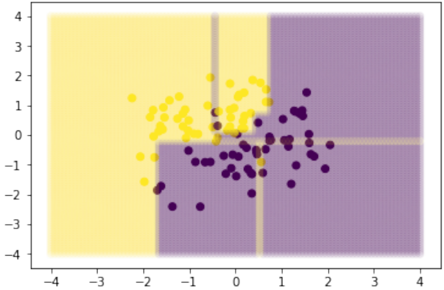

Рис.: Решающее дерево на плоскости

Построение композиции

Простая идея: объеденим деревья в композиции и построим один сложный классификатор.

Но смешивать нужно различные алгоритмы! Как этого добиться?

- Обучим различные алгоритмы (различные гиперпараметры)
- Обучим на разных подвыборках
- Обучим на различных признаках
- ...

Алгоритм построения случайного леса

Случайный лес объединяет две идеи из вышеперечисленных: слабые алгоритмы (деревья) обучаются на различных подвыборках и на различных признаках.

- Подвыборка на которой будет строится данные дерево - это подвыборка с повторением из исходной выборки того же размера (*бутстрэп*).
- На каждом ветвлении дерева подмножество оптимизированных признаков выбирается случайно.
- В итоге, предсказания всех деревьев усредняются.

Псевдокод

Algorithm 1: Построение случайного леса

Data: $X, y, num\_trees, \ldots$
Result: Random Forest
forest = {};
for *i in* num_trees **do**
  sample X_{bs}, y_{bs} from X, y;
  build random tree with X_{bs}, y_{bs};
  add tree to forest ;

Bias/Variance Анализ

Мат. ожидание ошибки любого по алгоритма можно разложить следующим образом

$$\mathbb{E}[(y - \hat{f}(x))^2] = \mathrm{Bias}[\hat{f}(x)]^2 + \mathrm{Var}(\hat{f}(x)) + \sigma^2$$

Идея Leo Brieman'a была в том, что усреднение алгоритмов может снизить $\mathrm{Var}(\hat{f}(x))$, не изменяя смещение (но только тогда, когда деревья максимально "независимы").

Особенности случайного леса

Примеры использования

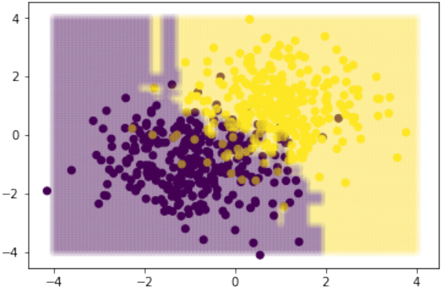

Рис.: Случайный лес на плоскости

Как мы ещё можем объединить слабые алгоритмы?

Идея: пусть каждое следующее дерево исправляет ошибки предыдущего ансамбля деревьев.

Пусть мы имеем ансамбль деревьев $H_n = \sum_{i=1}^{n} h_i$. Тогда

$$h_{i+1} = \mathrm{argmin}_h \sum_i (H_n(x_i) - y_i)^2$$

Алгоритм градиентного бустинга

Algorithm 2: Построение случайного леса

Data: $X, y, num\_trees, \ldots$

Result: Gradient Boosting

ensemble $= 0$; **for** *i in num_trees* **do**

  build new tree with $(X, y - \mathrm{ensemble}(X))$;

  add tree to ensemble ;

Особенности бустинга

Обобщение на любую функцию потерь

Заметим, что обучаясь на величину ошибки, мы на самом деле обучаемся на градиент функции потерь на предыдущем шаге (отсюда и название!).

$$\nabla \sum_i (y_i - H_n(x_i))^2 = \begin{pmatrix} \vdots \\ 2(H_n(x_i) - y_i) \\ \vdots \end{pmatrix}$$

Таким образом, мы можем взять произвольную функцию потерь и с помощью ее градиента обучать следующий алгоритм.

Таким образом мы получаем бустинг для классификации!

Функции потерь для задачи классификации

Популярные модификации

- xgboost: регуляризация деревьев, апроксимация функции потерь с помощью разложения Тейлора.
- adaboost: экспоненциальная функция потерь, веса на каждом объекте.
- catboost: oblivious decision trees.

Пример

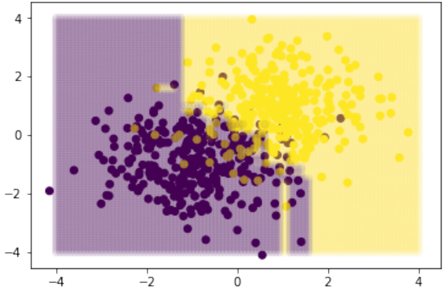

Рис.: Градиентный бустинг на плоскости

Пример

http://arogozhnikov.github.io

Резюме