Random forest (случайный лес)

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Random forest

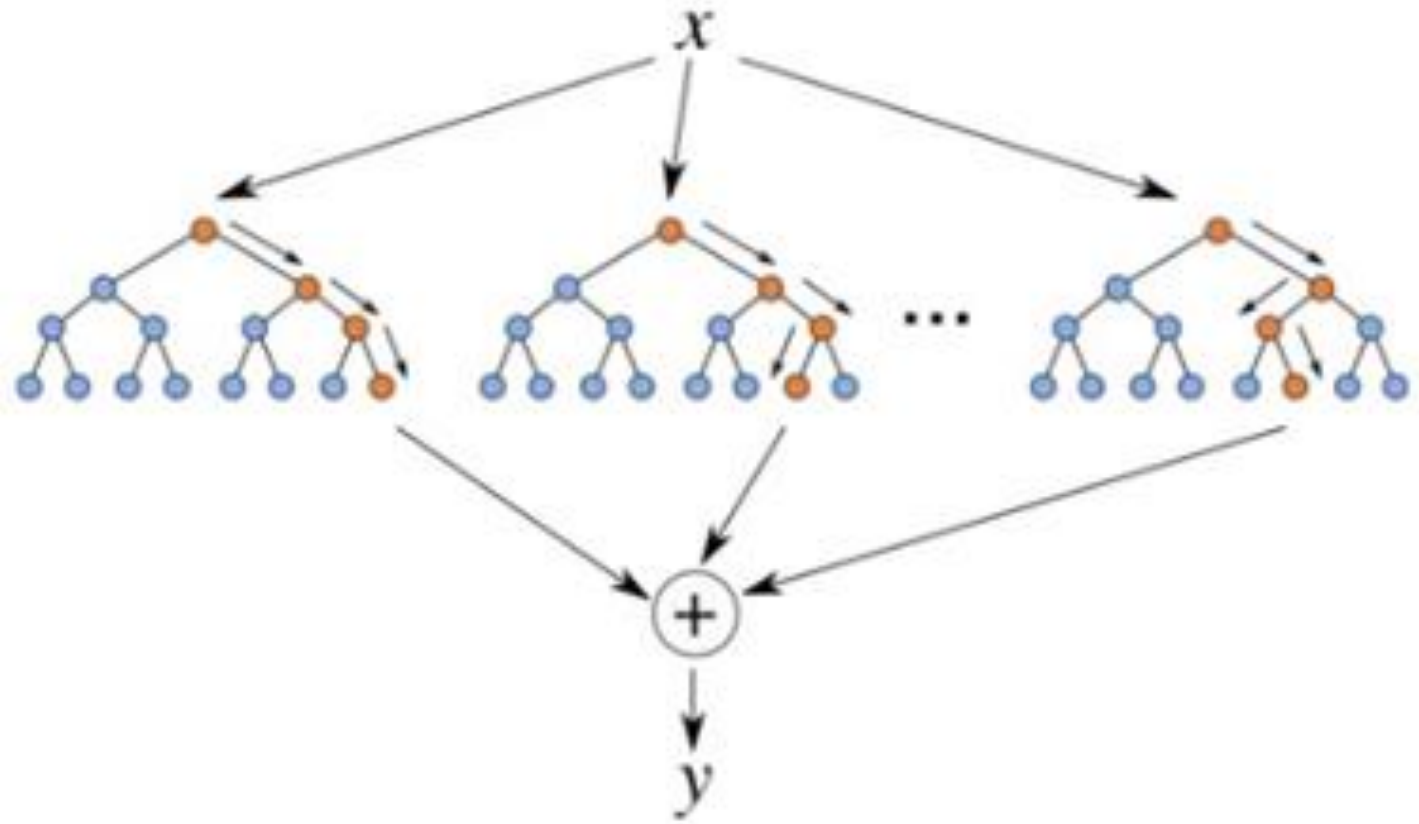
Random forest (с англ. — «случайный лес») — алгоритм машинного обучения, предложенный Лео Брейманом и Адель Катлер, заключающийся в использовании комитета (ансамбля) решающих деревьев. Алгоритм сочетает в себе две основные идеи: метод бэггинга Бреймана, и метод случайных подпространств, предложенный Tin Kam Ho. Алгоритм применяется для задач классификации, регрессии и кластеризации. Основная идея заключается в использовании большого ансамбля решающих деревьев, каждое из которых само по себе даёт очень невысокое качество классификации, но за счёт их большого количества результат получается хорошим.

Пусть обучающая выборка состоит из *N* образцов, размерность пространства признаков равна *M*, и задан параметр *m* (в задачах классификации обычно $m \approx \sqrt{M}$ как неполное количество признаков для обучения.

Наиболее распространённый способ построения деревьев комитета следующий:

1. Сгенерируем случайную подвыборку с повторениями размером N из обучающей выборки. (Таким образом, некоторые образцы попадут в неё два или более раза, в среднем $N^*(1-1/N)^N$, а примерно N/e образцов не войдут в неё вообще). Те образцы, которые не попали в выборку, называются out-of-bag (неотобранные).
2. Построим решающее дерево, классифицирующее образцы данной подвыборки, причём в ходе создания очередного узла дерева будем выбирать набор признаков, на основе которых производится разбиение (не из всех *M* признаков, а лишь из *m* случайно выбранных). Выбор наилучшего из этих m признаков может осуществляться различными способами. В оригинальном коде Бреймана используется критерий Джини, применяющийся также в алгоритме построения решающих деревьев CART. В некоторых реализациях алгоритма вместо него используется критерий прироста информации.[3]
3. Дерево строится до полного исчерпания подвыборки и не подвергается процедуре прунинга (англ. pruning — отсечение ветвей) (в отличие от решающих деревьев, построенных по таким алгоритмам, как CART или C4.5).

Классификация объектов проводится путём голосования: каждое дерево комитета относит классифицируемый объект к одному из классов, и побеждает класс, за который проголосовало наибольшее число деревьев. Оптимальное число деревьев подбирается таким образом, чтобы минимизировать ошибку классификатора на тестовой выборке. В случае её отсутствия, минимизируется оценка ошибки out-of-bag: тех образцов, которые не попали в обучающую подвыборку за счёт повторений (их примерно N/e).



RandomForest: синтаксис

Импортируем класс, содержащий метод классификации

```
from sklearn.ensemble import RandomForestClassifier
```

Создадим экземпляр класса

```
RC = RandomForestClassifier(n_estimators=100, max_features=10)
```

Обучим модель на обучающем выборке, а затем прогнозируем ожидаемое значение на тестовой выборке

```
RC = RC.fit(X_train, y_train)
y_predict = RC.predict(X_test)
```

Используйте RandomForestRegressor для регрессии.
Настройте параметры перекрестной проверки

Добавим еще больше случайности

Иногда желательна дополнительная случайность не входящая в модель Случайного леса

- Решение: выбирайте объекты (примеры) случайным образом и создавайте разбиения случайным образом - не выбирайте жадно

- Концепция «Extra Random Trees»

ExtraTreesClassifier: синтаксис

Импортируем класс, содержащий метод классификации

```
from sklearn.ensemble import ExtraTreesClassifier
```

Создадим экземпляр класса

```
EC = ExtraTreesClassifier(n_estimators=100, max_features=10)
```

Обучим модель на обучающем выборке, а затем прогнозируем ожидаемое значение на тестовой выборке

```
EC = EC.fit(X_train, y_train)
y_predict = EC.predict(X_test)
```

Используйте ExtraTreesRegressor для регрессии

Полный список параметров случайного леса для задачи регрессии:

class sklearn.ensemble.RandomForestRegressor

- (n_estimators — число деревьев в "лесу" (по умолчанию – 100)
- criterion — функция, которая измеряет качество разбиения ветки дерева (по умолчанию — "mse" , так же можно выбрать "mae")
- max_features — число признаков, по которым ищется разбиение. Вы можете указать конкретное число или процент признаков, либо выбрать из доступных значений: "auto" (все признаки), "sqrt", "log2". По умолчанию стоит "auto".
- max_depth — максимальная глубина дерева (по умолчанию глубина не ограничена)
- min_samples_split — минимальное количество объектов (примеров), необходимое для разделения внутреннего узла. Можно задать числом или процентом от общего числа объектов (по умолчанию — 2)

Полный список параметров случайного леса для задачи регрессии:

- min_samples_leaf — минимальное число объектов в листе. Можно задать числом или процентом от общего числа объектов (по умолчанию — 1)
- min_weight_fraction_leaf — минимальная взвешенная доля от общей суммы весов (всех входных объектов), которая должна быть в листе (по умолчанию имеют одинаковый вес)
- max_leaf_nodes — максимальное количество листьев (по умолчанию нет ограничения)
- min_impurity_split — порог для остановки наращивания дерева (по умолчанию 1e-7)
- bootstrap — применять ли бустрэп для построения дерева (по умолчанию True)
- oob_score — использовать ли out-of-bag объекты для оценки R^2 (по умолчанию False)
- n_jobs — количество ядер для построения модели и предсказаний (по умолчанию 1, если поставить -1, то будут использоваться все ядра)
- random_state — начальное значение для генерации случайных чисел (по умолчанию его нет, если хотите воспроизводимые результаты, то нужно указать любое число типа int verbose — вывод логов по построению деревьев (по умолчанию 0)
- warm_start — использует уже натренированную модель и добавляет деревья в ансамбль (по умолчанию False)

Для задачи классификации все почти то же самое, мы приведем только те параметры, которыми RandomForestClassifier отличается от RandomForestRegressor

class sklearn.ensemble.RandomForestClassifier

- (criterion — поскольку у нас теперь задача классификации, то по умолчанию выбран критерий "gini" (можно выбрать "entropy")
- class_weight — вес каждого класса (по умолчанию все веса равны 1, но можно передать словарь с весами, либо явно указать "balanced", тогда веса классов будут равны их исходным частям в генеральной совокупности; также можно указать "balanced_subsample", тогда веса на каждой подвыборке будут меняться в зависимости от распределения классов на этой подвыборке)

Наиболее важные параметры

- Число деревьев — **n_estimators**. Чем больше деревьев, тем лучше качество, но время настройки и работы RF также пропорционально увеличиваются
- Число признаков для выбора расщепления — **max_features**. При увеличении max_features увеличивается время построения леса, а деревья становятся «более однообразными». По умолчанию он равен sqrt(n) в задачах классификации и n/3 в задачах регрессии.
- Максимальная глубина деревьев — **max_depth**. Рекомендуется использовать максимальную глубину (кроме случаев, когда объектов слишком много и получаются очень глубокие деревья, построение которых занимает значительное время). При использовании неглубоких деревьев изменение параметров, связанных с ограничением числа объектов в листе и для деления, не приводит к значимому эффекту (листья и так получаются «большими»). Неглубокие деревья рекомендуют использовать в задачах с большим числом шумовых объектов (выбросов).

Плюсы случайного леса

- имеет высокую точность предсказания, на большинстве задач будет лучше линейных алгоритмов; точность сравнима с точностью бустинга
- практически не чувствителен к выбросам в данных из-за случайного сэмплирования
- не чувствителен к масштабированию (и вообще к любым монотонным преобразованиям) значений признаков, связано с выбором случайных подпространств
- не требует тщательной настройки параметров, хорошо работает «из коробки». С помощью «тюнинга» параметров можно достичь прироста от 0.5 до 3% точности в зависимости от задачи и данных
- способен эффективно обрабатывать данные с большим числом признаков и классов
- одинаково хорошо обрабатывает как непрерывные, так и дискретные признаки
- редко переобучается, на практике добавление деревьев почти всегда только улучшает композицию, но на валидации, после достижения определенного количества деревьев, кривая обучения выходит на асимптоту

Плюсы случайного леса

- для случайного леса существуют методы оценивания значимости отдельных признаков в модели
- хорошо работает с пропущенными данными; сохраняет хорошую точность, если большая часть данных пропущена
- предполагает возможность сбалансировать вес каждого класса на всей выборке, либо на подвыборке каждого дерева
- вычисляет близость между парами объектов, которые могут использоваться при кластеризации, обнаружении выбросов или (путем масштабирования) дают интересные представления данных
- возможности, описанные выше, могут быть расширены до неразмеченных данных, что приводит к возможности делать кластеризацию и визуализацию данных, обнаруживать выбросы
- высокая параллелизуемость и масштабируемость

Минусы случайного леса

- в отличие от одного дерева, результаты случайного леса сложнее интерпретировать
- нет формальных выводов (p-values), доступных для оценки важности переменных
- алгоритм работает хуже многих линейных методов, когда в выборке очень много разреженных признаков (тексты, Bag of words)
- случайный лес не умеет экстраполировать, в отличие от той же линейной регрессии (но это можно считать и плюсом, так как не будет экстремальных значений в случае попадания выброса)

Минусы случайного леса

- алгоритм склонен к переобучению на некоторых задачах, особенно на зашумленных данных
- для данных, включающих категориальные переменные с различным количеством уровней, случайные леса предвзяты в пользу признаков с большим количеством уровней: когда у признака много уровней, дерево будет сильнее подстраиваться именно под эти признаки, так как на них можно получить более высокое значение оптимизируемого функционала (типа прироста информации)
- если данные содержат группы коррелированных признаков, имеющих схожую значимость для меток, то предпочтение отдается небольшим группам перед большими
- больший размер получающихся моделей (требуется дополнительная память для хранения модели)