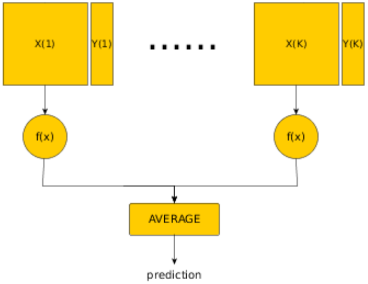
Методы построения ансамблей

Виктор Китов

victorkitov.github.io



Курс поддержан фондом 'Интеллект'

Победитель конкурса VK среди курсов по IT

Содержание

1. Стэкинг
2. Композиции на разных обучающих подвыборках

Алгоритм стэкинга

- Рассмотрим обучающую выборку , базовые модели $f_1(x), ...f_M(x)$ и $G(\cdot)$.

Алгоритм стэкинга

- Рассмотрим обучающую выборку , базовые модели $f_1(x), ...f_M(x)$ и $G(\cdot)$.
- Обучение $f_1(x), ...f_M(x)$ и $G(\cdot)$ на одинаковой выборке вызывает переобучение.

Алгоритм стэкинга:

1. Инициализируем обучающую выборки $G(\cdot)$: $T' = \{\}$
2. Разобьем обучающую выборку $T = \{(x_n, y_n)\}_{n=1}^N$ на K блоков: $T_1, T_2, ...T_K$.
3. для $k = 1, 2, ...K$:
   обучим $f_1(x), ...f_M(x)$ на $T \backslash T_k$
   для каждого $(x, y) \in T_k$:
   дополним T' объектом $([f_1(x), ...f_M(x)], y)$
4. Обучим $G(\cdot)$ на T'.
5. Перенастроим $f_1(x), ...f_M(x)$ на всей T.

Расширения стэкинга

Кроме прогнозов $f_1(x), ...f_M(x)$ агрегирующая функция может зависеть от:

- исходных признаков x
  - в разных частях признакового пространства-разная агрегация
- внутренних представлений f_m (дискриминантных функций, вероятностей).

Линейный стэкинг (блендинг)

- Линейный стэкинг (блендинг, blending) :

$$f(x)=\sum_{m=1}^{M}w_m f_m(x)$$

$$\sum_{n=1}^{N}\left(\sum_{m=1}^{M}w_m f_m(x_n)-y_n\right)^2\to\min_{\mathrm{w}}$$

- $f_1(x),...f_M(x)$ зависимы (предсказывают один и тот же y) => нестабильная оценка.
- Более устойчивая оценка:

$$\begin{cases}\sum_{n=1}^{N}\left(\sum_{m=1}^{M}w_m f_m(x_n)-y_n\right)^2+\lambda\sum_{m=1}^{M}\left(w_m-\frac{1}{M}\right)^2\to\min_{\mathrm{w}}\\ w_1\geq 0,...w_M\geq 0\end{cases}$$

Использование стэкинга (регрессия) 1

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import Ridge
from sklearn.ensemble import StackingRegressor
from sklearn.metrics import accuracy_score
from sklearn.metrics import brier_score_loss

X_train, X_test, Y_train, Y_test =
    get_demo_classification_data()

# Инициализируем базовые модели
knn = KNeighborsRegressor(n_neighbors=100)
tree_model = DecisionTreeRegressor()
```

Использование стэкинга (регрессия) 2

```
# Инициализируем стэкинг для регрессии
ensemble = StackingRegressor(
      estimators=
        [('K nearest neighbors', knn),
         ('decision tree', tree_model)],
      final_estimator=Ridge(),
      cv=3,           # количество блоков кросс-валидации
      n_jobs=-1)      # используем все ядра процессора

# обучение базовых и мета-модели:
ensemble.fit(X_train, Y_train)
Y_hat = ensemble.predict(X_test) # построение прогнозов
print(f'Средний модуль ошибки (MAE): \
   {mean_absolute_error(Y_test, Y_hat):.2f}')
```

Использование стэкинга (классификация) 1

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import StackingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import brier_score_loss

X_train, X_test, Y_train, Y_test =
    get_demo_classification_data()

# Инициализируем базовые модели
knn = KNeighborsClassifier(n_neighbors=100)
tree_model = DecisionTreeClassifier()
```

Использование стэкинга (классификация) 2

```
# Инициализируем стэкинг для классификации
ensemble = StackingClassifier(
    estimators=[('K nearest neighbors', knn),
                ('decision tree', tree_model)],
    final_estimator=LogisticRegression(),
    cv=3,          # количество блоков кросс-валидации
    n_jobs=-1)     # используем все ядра процессора

# обучение базовых и мета-модели:
ensemble.fit(X_train, Y_train)
Y_hat = ensemble.predict(X_test) # построение прогнозов
print(f'Точность прогнозов: \
    {100*accuracy_score(Y_test, Y_hat):.1f}%')

P_hat = ensemble.predict_proba(X_test) # вер-ти классов
loss = brier_score_loss(Y_test, P_hat[:,1])
print(f'Ошибка Бриера: {loss:.2f}')
```

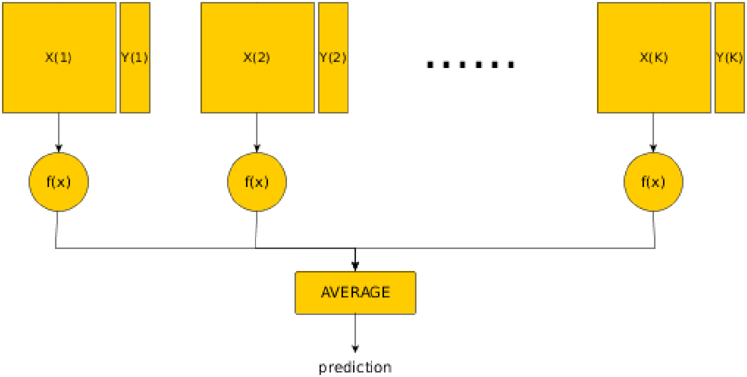
Больше информации. Полный код.

Содержание

1. Стэкинг
2. Композиции на разных обучающих подвыборках

Усреднение по выборкам

Усреднение по выборкам



Усреднение по выборкам

Усреднение по выборкам: если модель переобучается на выборке (X, Y) можно усреднять множество моделей, обученных на разных реализациях обучающих выборок $(X_k, Y_k),\ k = 1, 2, ... M$.

$$bias_{X,Y}\left[\frac{1}{M}\sum_{k=1}^{M} f(x, X_k, Y_k)\right] = y(x) - \mathbb{E}_{X,Y}\left[\frac{1}{M}\sum_{k=1}^{M} f(x, X_k, Y_k)\right]$$
$$= y(x) - \frac{1}{M}\sum_{k=1}^{M}\mathbb{E}_{X,Y} f(x, X_k, Y_k) = y(x) - \frac{1}{M}\sum_{k=1}^{M}\mathbb{E}_{X,Y} f(x, X, Y)$$
$$= y(x) - \mathbb{E}_{X,Y} f(x, X, Y)$$

т.е. смещение=смещению 1-го алгоритма.

Усреднение по выборкам: дисперсия

$$\mathrm{Var}_{X,Y}\left[\frac{1}{M}\sum_{k=1}^{M}f(x,X_k,Y_k)\right]=\mathbb{E}_{X,Y}\left[\frac{1}{M}\sum_{k=1}^{M}f(x,X_k,Y_k)-\mathbb{E}_{X,Y}\left[\frac{1}{M}\sum_{k=1}^{M}f(x,X_k,Y_k)\right]\right]^2$$

$$=\mathbb{E}_{X,Y}\left[\frac{1}{M}\sum_{k=1}^{M}\left(f(x,X_k,Y_k)-\mathbb{E}_{X,Y}f(x,X_k,Y_k)\right)\right]^2=$$

$$=\frac{1}{M^2}\mathbb{E}_{X,Y}\left[\sum_{k=1}^{M}\left(f(x,X_k,Y_k)-\mathbb{E}_{X,Y}f(x,X_k,Y_k)\right)\right]^2=$$

$$=\frac{1}{M^2}\sum_{k=1}^{M}\mathbb{E}_{X,Y}\left[f(x,X_k,Y_k)-\mathbb{E}_{X,Y}f(x,X_k,Y_k)\right]^2+$$

$$+\frac{1}{M^2}\sum_{k_1\neq k_2}\mathbb{E}_{X,Y}\left[\left(f(x,X_{k_1},Y_{k_1})-\mathbb{E}_{X,Y}f(x,X_{k_1},Y_{k_1})\right)\left(f(x,X_{k_2},Y_{k_2})-\mathbb{E}_{X,Y}f(x,X_{k_2},Y_{k_2})\right)\right]$$

$$=\frac{1}{M^2}\sum_{k=1}^{M}\mathrm{Var}_{X,Y}\left[f(x,X_k,Y_k)\right]+\frac{1}{M^2}\sum_{k_1\neq k_2}\mathrm{cov}\left[f(x,X_{k_1},Y_{k_1}),f(x,X_{k_2},Y_{k_2})\right]$$

Усреднение по выборкам: дисперсия

При нескоррелированных $f(x, X_k, Y_k)$:

$$\frac{1}{M^2}\sum_{k=1}^{K}\mathrm{Var}_{X,Y}\left[f(x,X_k,Y_k)\right]+$$
$$+\frac{1}{M^2}\sum_{k_1\neq k_2}\mathrm{cov}\left[f(x,X_{k_1},Y_{k_1}),f(x,X_{k_2},Y_{k_2})\right]=$$
$$=\frac{1}{M}\mathrm{Var}_{X,Y}\left[f(x,X_k,Y_k)\right]$$

Дисперсия в M раз меньше.

Усреднение по выборкам: дисперсия

При частичной скоррелированности, дисперсия тоже уменьшается (используем $\operatorname{cov}(x,y) \le \sqrt{\operatorname{Var}(x)}\sqrt{\operatorname{Var}(y)}$):

$$\frac{1}{M^2}\sum_{k=1}^{K}\operatorname{Var}_{X,Y}\left[f(x,X_k,Y_k)\right]+$$

$$+\frac{1}{M^2}\sum_{k_1\ne k_2}\operatorname{cov}\left[f(x,X_{k_1},Y_{k_1}),f(x,X_{k_2},Y_{k_2})\right]=$$

$$=\frac{1}{M^2}\sum_{k_1=1}^{M}\sum_{k_2=1}^{M}\operatorname{cov}\left[f(x,X_{k_1},Y_{k_1}),f(x,X_{k_2},Y_{k_2})\right]\le$$

$$\le\frac{1}{M^2}\sum_{k_1=1}^{M}\sum_{k_2=1}^{M}\sqrt{\operatorname{Var}_{X,Y}\left[f(x,X_{k_1},Y_{k_1})\right]}\sqrt{\operatorname{Var}_{X,Y}\left[f(x,X_{k_2},Y_{k_2})\right]}\le$$

$$\le\operatorname{Var}_{X,Y}\left[f(x,X,Y)\right]$$

Бэггинг & метод случайных подпространств

На практике дана единственная (X, Y). Как генерировать $(X_1, Y_1), ...(X_M, Y_M)$?

[1] Какова вероятность того, что некоторый объект не попадет в подвыборку? Чему равен предел этой вероятности при $N \to \infty$?

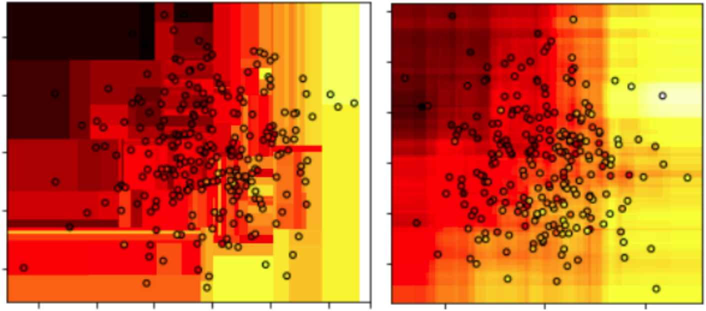
Бэггинг & метод случайных подпространств

На практике дана единственная (X, Y). Как генерировать $(X_1, Y_1), ...(X_M, Y_M)$?

- **Бэггинг (bagging)**:
  - случайный выбор N объектов (с возвращением)[1]
- **Метод случайных подпространств (random subspaces)**:
  - случайный выбор K признаков (без возвращения, $K < N$)
- Можно применять комбинацию методов.

[1] Какова вероятность того, что некоторый объект не попадет в подвыборку? Чему равен предел этой вероятности при $N \to \infty$?

Бэггинг деревьев



Регрессия: одно дерево и бэггинг над деревьями

Бэггинг деревьев

Настройка решающего правила в узле CART:

$$\widehat{i}, \widehat{h} = \arg\max_{f,h \in P(t)} \Delta\phi(t)$$

$P(t)$ для стандартных решающих деревьев:

$P = \{\}$
для каждого i in $\{1, ..., D\}$
    для каждого h из $unique\left\{x_n^i\right\}_{n: x_{n \in t}}$
        $P := P \cup (i, h)$

Бэггинг над решающими деревьями успешно борется с их переобучением.

Использование бэггинга (регрессия)

```
from sklearn.ensemble import BaggingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error

X_train, X_test, Y_train, Y_test =
      get_demo_classification_data()
model = BaggingRegressor(
   DecisionTreeRegressor(), # базовая модель
   max_samples=0.8,  # доля используемых объектов
   max_features=1.0, # доля используемых пр-ков
   n_jobs=-1)  # используем все ядра процессора
model.fit(X_train, Y_train)   # обучение модели
Y_hat = model.predict(X_test) # построение прогнозов
print(f'Средний модуль ошибки (MAE): \
   {mean_absolute_error(Y_test, Y_hat):.2f}')
```

Использование бэггинга (классификация)

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, brier_score_loss

X_train, X_test, Y_train, Y_test =
      get_demo_classification_data()
model = BaggingClassifier(
   DecisionTreeClassifier(), # базовая модель
    n_estimators=100, # число базовых моделей
   max_samples=0.8,  # доля используемых объектов
   max_features=1.0, # доля используемых пр-ков
   n_jobs=-1)  # используем все ядра процессора
model.fit(X_train, Y_train)   # обучение модели
Y_hat = model.predict(X_test) # построение прогнозов
print(f'Точность прогнозов: \
   {100*accuracy_score(Y_test, Y_hat):.1f}%')

P_hat = model.predict_proba(X_test)  # вер-ти классов
loss = brier_score_loss(Y_test, P_hat[:,1])
print(f'Мера ошибки Бриера: {loss:.2f}')
```

Больше информации. Полный код.

Случайный лес, особо случайные деревья

$P(t)$ для случайного леса (random forest, RF):

$P = \{\}$, $K = \alpha D$
сэмплируем $i_1, ... i_K$ случайно из $\{1, ..., D\}$ # без возвращения
для каждого i из $i_1, ... i_K$
    для каждого h из $unique\left\{x_n^i\right\}_{n: x_n \in t}$
        $P := P \cup (i, h)$

$S(t)$ для особо случайных деревьев (extra random trees, ERT):

$S = \{\}$, $K = \alpha D$
сэмплируем $i_1, ... i_K$ случайно из $\{1, ..., D\}$ # с возвращением
для каждого f in $d_1, ... d_K$
    сэмплируем h случайно из $unique\left\{x_n^i\right\}_{n: x_n \in t}$ # без возвращения
    $P := P \cup (f, h)$

Использование случайного леса (регрессия)

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

X_train, X_test, Y_train, Y_test =
    get_demo_regression_data()
model = RandomForestRegressor(
    n_estimators=100, # число базовых моделей
    bootstrap=True, # обучать на подвыборке/исх. выборке
    max_features=0.5, # доля используемых пр-ков
    n_jobs=-1)        # используем все ядра процессора
model.fit(X_train, Y_train)   # обучение модели
Y_hat = model.predict(X_test) # построение прогнозов
print(f'Средний модуль ошибки (MAE): \
   {mean_absolute_error(Y_test, Y_hat):.2f}')
```

Особо случайные деревья - ExtraTreesRegressor.

Использование случайного леса (классификация)

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, brier_score_loss

X_train,X_test,Y_train,Y_test=get_demo_classification_data()
model = RandomForestClassifier(
    n_estimators=100, # число базовых моделей
    bootstrap=True, # обучать на подвыборке/исходной выборке
    max_features=0.5, # доля используемых пр-ков
    n_jobs=-1)        # используем все ядра процессора
model.fit(X_train, Y_train)   # обучение модели
Y_hat = model.predict(X_test) # построение прогнозов
print(f'Точность прогнозов: \
   {100*accuracy_score(Y_test, Y_hat):.1f}%')

P_hat = model.predict_proba(X_test)  # можно предсказывать
    вероятности классов
loss = brier_score_loss(Y_test, P_hat[:,1])  # считаем
    качество Бриера на вероятности положительного класса
print(f'Мера ошибки Бриера: {loss:.2f}')
```

Больше информации. Полный код. См. также ExtraTreesClassifier.

Вневыборочная оценка

Оценка по обучающей выборке - оптимистическая оценка сверху:

$$L = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}\left(\frac{1}{M}\sum_{m=1}^{M} f_m(x_n), y_n\right)$$

Вневыборочная оценка

Оценка по обучающей выборке - оптимистическая оценка сверху:

$$L = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}\left(\frac{1}{M}\sum_{m=1}^{M} f_m(x_n), y_n\right)$$

Вневыборочная оценка (out-of-bag estimate) - если с бэггингом

$$L_{OOB} = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}\left(\frac{1}{|I_n|}\sum_{m\in I_n} f_m(x_n), y_n\right)$$

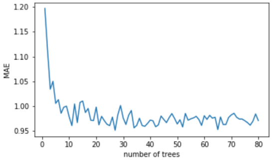
- $I_n \subset \{1, 2, ...M\}$- набор моделей, не использовавших (x_n, y_n) для обучения.
- Не требуется дополнительная валидационная выборка.
- Немного пессимистическая оценка потерь снизу.

Комментарии

- Бэггинг, случайный лес, особо случайные деревья:
  - легко распараллеливаются
  - базовые модели не учатся исправлять ошибки друг друга
- Деревья случайный лес, особо случайные деревья могут строиться на одинаковой обучающей выборке
  - bootstrap=False в sklearn
  - за счет случайности $P(t)$ модели все равно будут получаться разные

Число базовых моделей в ансамбле

- Пусть M=#базовых моделей.
- Типичная зависимость потерь от M:



- Нет переобучения с $\uparrow M$: просто избыточное усреднение по однотипным моделям.
- Настройка: подбор всех параметров с малым M, затем $\uparrow M$.

Заключение

- Разложение на смещение и разброс:
    - простые модели: высокое смещение
    - сложные модели: высокая дисперсия
- Разложение неопределенности:
    - выгодно усреднять разнородные модели
- Композиции:
    - простая агрегирующая модель: борьба с переобучением
    - сложная агрегирующая модель: борьба с недообучением
- Стэкинг: агрегирующая модель и базовые должны обучаться на разных выборках
- Борьба с переобучением: бэггинг, метод случайных подпространств, случайный лес, особо случайные деревья.