Teória učenia súborom metód

Učenie súborom metód (ensemble learning) je prístup využivaný na klasifikáciu a regresiu, pri ktorom sa využije viacero klasifikátorov na vytvorenie lepšieho a stabilnejšieho modelu, aký by bolo možné vytvoriť len pomocou jednoduchého klasifkátora. Je to vlastne postup, v ktorom sa zlučuje viacero základných klasifikátorov pomocou vybranej metódy, a vytvára sa z nich nový model. To ako sa vytvoria základné klasifikátory a akým spôsobom sa zlučujú, záleží od zvolenej metódy učenia súborom metód. Pre všetky však platí, že základný klasfikátor ktorý zlučujú, by mal mať lepšiu úspešnosť ako náhodné hádanie.

Táto teória dáva zmysle aj v prenesení do nášho života. Ak máme urobiť kritické rozhodnutie, vyhľadáme radu experta. Ak urobíme rozhodnutie na základe jeho rady, sme si menej istý našim rozhodnutím, ako keby sme vyhľadali skupinu expertov, zistili každého názor, vyhodnotili ich, a rozhodli sa až potom.

Bagging

Názov bagging je odvodený od Bootstrap aggregating. Princíp tohto učenia súborom metód je založený na predpoklade, že ak na určenie výsledku klasifikácie použijeme výsledky z viacerých klasifikátorov, ktoré vytvoril jeden alebo viac algoritmov na rôznych trénovacích množinách z rovnakej problémovej oblasti, tak tento výsledok by mohol dosiahnuť vyššiu presnosť ako výsledok určený klasifikátorom vytvoreným len na jednej trénovacej množine. Komplikáciou je, že k dispozícií máme najčastejšie iba jednu trénovaciu množinu.

Potrebujeme teda spôsob, akým by sme z jednej trénovacej množiny, dostali toľko trénovacích množín, koľko klasifikátorov chceme zlúčiť. Ak však chceme zlúčiť n klasifikátorov založených na rovnakom algoritme, nemôžeme len vytvoriť n presných kópií trénovacej množiny. Vzniklo by nám totiž n rovnakých klasifikátorov, ktoré keby sme zlúčili, tak by sme nijak nezlepšili výsledok klasifikácie oproti samotnému základnému klasifikátoru.

My totiž potrebujeme dosiahnuť nezávislosť základných klasifikátorov, ktoré chceme zlúčiť. Nezávislosť v tomto prípade znamená, že modely základných klasifikátorov budú od seba odlišné. Teda predikcie, ktoré pomocou nich urobíme, budú odlišné. Ak sa nám to podarí, teória baggingu hovorí, že pri hlasovaní sa nám vyrušia niektoré chyby predikcie, a teda sa zlepší výsledok. Teda presnejšie povedané, základným princípom baggingu je zlučovanie nezávislých klasifikátorov hlasovaním.

Nezávislosť základných klasifikátorov

Poznáme tri najčastejšie spôsoby, akými môžeme dosiahnuť nezávislosť základných klasifikátorov pri baggingu.

- klasický bagging
- feature bagging
- rôzne základné algoritmy

Klasický bagging tento problém rieši tak, že z trénovacej množiny vytvorí m nových. M v tomto prípade reprezentuje počet modelov, z ktorých sa poskladá konečný výsledok. Každá z

týchto množín sa vytvorí náhodným poskladaním z pozorovaní zo základnej množiny, pričom jednotlivé pozorovania sa môžu opakovať.

Môžeme si to predstaviť podobne ako v Tabuľka 1 . V tejto tabuľke môžeme jasne vidieť, že každý klasifikátor bude vlastne vytvorený na mierne odlišných pozorovaniach.

Tabuľka 1: Trénovacie množiny pre klasický bagging

Trénovacia množina	Pozorovanie v množine
T (pôvodná množina)	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
T1	1, 2, 2, 4, 5, 6, 6, 8, 9, 10
T2	1, 1, 3, 4, 5, 5, 7, 8, 10, 10
T3	3, 3, 3, 4, 5, 6, 7, 7, 9, 10
T4	1, 2, 4, 4, 5, 6, 7, 8, 9, 9

Feature bagging (nazývaný aj attribute bagging alebo random subspaces method) je ďalším spôsobom, ako môžeme dosiahnuť nezávislosť základných klasifikátorov. Na rozdiel od klasického baggingu však nebudeme narábať s výskytom pozorovaní v trénovacej množine, ale s atribútmi, ktoré tieto pozorovania budú mať.

Majme trénovaciu množinu pozorovaní takú, že každé pozorovanie má **n** atribútov. My chceme vybrať číslo **p**, pričom platí, že **p** < **n**. Následne náhodne vyberieme p atribútov z množiny všetkých atribútov pozorovaní. Týmto spôsobom nám vznikne trénovacia množina, ktorá má všetky pozorovania ako pôvodná, ale jej pozorovania nemajú všetky atribúty ako pôvodná.

V Tabuľka 2 môžeme vidieť príklad.

Tabuľka 2: Trénovacie množiny pre feature bagging pre p = 4

Trénovacia množina	Atribúty pozorovaní v množine
T (pôvodná množina atribútov)	a, b, c, d, e, f, g, h, i, j
T1	a, b, c, d
T2	h, d, i, a
T3	f, b, j, g
T4	g, a, e, c

Rôzne základné algoritmy je metódou, ktorá je odlišná od predchádzajúcich dvoch, v zásadnom bode. Zatiaľ čo klasický bagging a feature bagging manipulovali s trénovacími údajmi pre základné klasifikátory, pri rôznych základných algoritmoch ponechávame trénovaciu množinu v pôvodnam stave, a iba vytvoríme jej kópiu. Nezávislosť základných klasifkikátorov dosiahne ináč. Už aj sám názov napovedá, ako to asi urobíme. Namiesto toho aby sme základné klasifikátory, vytvorili rovnakým algoritmom, vytvoríme každý základný klasifikátor pomocou iného algoritmu na rovnakej trénovacej množine.

Ako môžeme vidieť, trénovacie množiny pre základné klasifikátory pri baggingu sú vytvorené v podstate naraz. Teda nezávisia od iných faktorov ako napríklad výstup predchádzajúcich klasifikátorov, ale iba od spôsobu akým sa majú vytvoriť. To umožňuje

bagging paralelizovať. Môžme teda celkom jednoducho vytvoriť všetky trénovacie množiny, a paralelne nechať natrénovať jednotlivé základné klasifikátory. Týmto spôsobom môžeme zmierniť nevýhodu, ktorú bagging okrem svojich výhod má. Veľkou nevýhodou baggingu je, že pri jeho využití sme vytvorili namiesto jedného základného klasifikátora viacero. Tým pádom sa nám značne zvýšila časová náročnosť výpočtu (klasifikácie). Medzi ďalšie problémy by sme mohli zaradiť ešte napríklad aj zhoršenie interpretability.

Ako však bagging ako prístup učenia súborom metód zmierňuje problémy bežných klasifikátorov? Chybovosť predpovede klasifikátorov je založená na dvoch faktoroch. Sú nimi vychýlenosť a variancia. Každý z týchto faktorov hovorí o niečom inom.

Variancia

Variancia je v podstate opakom vychýlenosti. Zatiaľ čo vychýlenosť je dôsledkom nerozpoznania kľúčovej charakteristiky, variancia je naopak rozpoznaním nepodstatných charakteristík ako kľúčových. Dôsledkom je stav nazývaný pretrénovanie. Bagging ako prístup učenia súborom metód znižuje varianciu. To dosahuje tým, že vytvára nezávislé základné klasifikátory. Vďaka tomu, že každý klasifikátor z množiny je vytvorený na základe odlišnej tréningovej množiny alebo pomocou iného algoritmu, znižuje sa tým šanca, že finálny výsledok by mohol považovať nepodstatnú charakteristiku za kľúčovú. Ak by ju totiž aj nejaký zo základných klasifikátorov za kľúčovú považoval, je veľká pravdepodobnosť, že ho ostatné základné klasifikátorý prehlasujú ak by na základe tejto chybnej charakteristiky chcel nesprávne zaradiť pozorovanie.

Ukážka kódu pre klasický bagging v jazyku python s použitím knižnice scikit-learn

```
# importneme funkciu na načítanie datasetu
from sklearn.datasets import load_iris
# načítame dataset
iris = load_iris()
# do X si uložme údaje z pozorovaní
X = iris.data
# do Y si uložme zaradenia jednotlivých pozorovaní
Y = iris.target

# Následne si rozdelíme množinu pozorovaní a množinu zaradení na trénovaciu a testovaciu

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=4)

# Importneme rozhodovací strom, ktorý je naimplementovaný v knižnici scikit-learn
from sklearn.tree import DecisionTreeClassifier
# Importneme bagging, ktorý je naimplementovaný v knižnici scikit-learn
from sklearn.ensemble import BaggingClassifier

# Vytvorme si teda bagging klasifikátor, ktorý bude mať ako základný klasifikátor rozhodovací strom
# Tomuto klasifikátoru sa dá nastaviť viacero parametrov
# Nás však teraz zaujíma len to, že chceme aby bolo:
#       - desať základných klasifikátorov (n_estimators)
#       - aby sa pozoravania v kópiach množín mohli opakovať (bootstrap)
#       - aby boli trénovacie množiny rovnako veľké ako pôvodná (max_samples)
bagging = BaggingClassifier(DecisionTreeClassifier(max_depth=1),n_estimators = 10,bootstrap = True, max_samples=1.0)

# Natrénujme teda bagging
bagging.fit(X_train,Y_train)

# Vytvoríme predikciu pre testovaciu množinu
prediction = bagging.predict(X_test)

# importneme funkciu na vyhodnotenie
from sklearn import metrics
print(round(metrics.accuracy_score(Y_test, prediction),2))
```

```
0.98
```