EÖTVÖS LORÁND TUDOMÁNYEGYETEM

TERMÉSZETTUDOMÁNYI KAR

Bartalis Dávid

Adattömörítés szubmoduláris kiválasztással

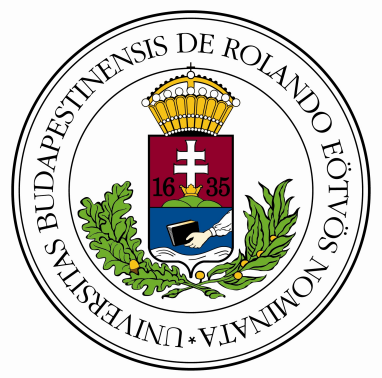
Témavezetők:

Bérczi-Kovács Erika

ELTE, Operációkutatási Tanszék

Béres Ferenc

SZTAKI, Informatikai Kutatólaboratórium



Budapest, 2020

Tartalomjegyzék

1. Bevezetés

A mai világban egyre többször fordul elő, hogy rendkívül nagy mennyiségű adatot használunk különböző modellek tanítására, viszont ahhoz, hogy érdemes eredményt kapjunk nagy számítási teljesítményre, futásidőre van szükség. A szubmoduláris optimalizálás [1] egy megoldás lehet arra, hogy a nagy adathalmazok kezelésével járó számítási terhelést csökkentsük, ugyanis az adathalmazok méretének növelésével általában az ismétlődések száma is nő, viszont ezek a redundáns adatok nem jelentenek hasznot Tehát az adatokban rejlő információ szubmoduláris jellege miatt lehetnek hatékonyak a szubmoduláris maximalizálásra épülő megközelítések. Projektmunkám során témavezetőimmel ezen problémával foglalkoztunk és egy erre fejlesztett programot tanulmányoztunk.

2. Apricot

Az Apricot [2] egy Python csomag, aminek segítségével hatalmas adathalmazból ki lehet választani egy olyan részhalmazt, ami az egész adatsokaságot reprezentálja. Ennek a legfontosabb felhasználása a gépi tanulás során lelhető fel, ugyanis nagyon hasznos, ha a tanulási folyamat során használt teszthalmaz számossága leredukálható úgy, hogy az elért eredmény alig változik. Mindezt szubmoduláris függvények maximalizálásával lehet elérni, amihez az Apricot egy hatékony mohó algoritmust használ. Már több szubmoduláris függvény is implementálva van a csomagban, ezek közül néhányat szeretnék kiemelni. Az, hogy ezek közül éppen mely függvényt célszerű használni, nagy mértékben függ attól, hogy milyen feladattal van dolgunk. Ezt az Apricot programról készült cikk néhány példán keresztül szemlélteti.

2.1. Feature-based / Tulajdonság alapú függvény

Legyen V az alaphalmaz, $X \subseteq V$ halmaz, továbbá definiáljunk összesen D darab tulajdonságot. Minden $d \in D$ tulajdonsághoz tartozik egy súly, mely azt mutatja meg, hogy az adott tulajdonság mennyire fontos. Legyen ez w_d. Azt pedig jelölje $m_d(x)$, hogy az x elemben mennyire található meg a d tulajdonság. Ekkor a feature-based típusú függvények a követke-

zőképpen formalizálhatók:

$$\mathcal{F}(X) = \sum_{d=1}^{D} w_d \phi \left(\sum_{x \in X} m_d(x) \right),$$

ahol a ϕ egy monoton konkáv függvényt jelöl, melyre gyakran használt példa a négyzetgyök, illetve a logaritmus függvény. Hogyha feltesszük, hogy a tulajdonságok értékei, illetve a tulajdonságokhoz tartozó súlyok csak nemnegatív értékeket vesznek fel, akkor ez egy szubmoduláris halmazfüggvény.

2.2. Facility location / Szolgáltató elhelyezési függvény

A facility location / szolgáltató elhelyezési függvény előző jelöléseket használva a következőképpen formalizálható:

$$\mathcal{F}(X) = \sum_{v \in V} \max_{x \in X} \delta(x, v),$$

ahol a $\delta(x, v)$ az x és v elemek közti hasonlóságot méri. A jelen esetben leggyakrabban használt hasonlósági függvények közé tartozik a negatív euklideszi távolság, illetve a koszinusz távolság. Az így definiált függvény szubmoduláris, teljesíti az úgynevezett csökkenő hasznok elvét.

2.3. Max Coverage / Maximális fedés függvény

A maximális lefedettségi probléma az alaphalmaz k darab halmazának kiválasztása úgy, hogy az uniójuk a lehető legnagyobb legyen. Tegyük fel, hogy bináris értékeink vannak, ekkor azon oszlopokat szeretnénk kiválasztani, amiknek van nem nulla elemük.

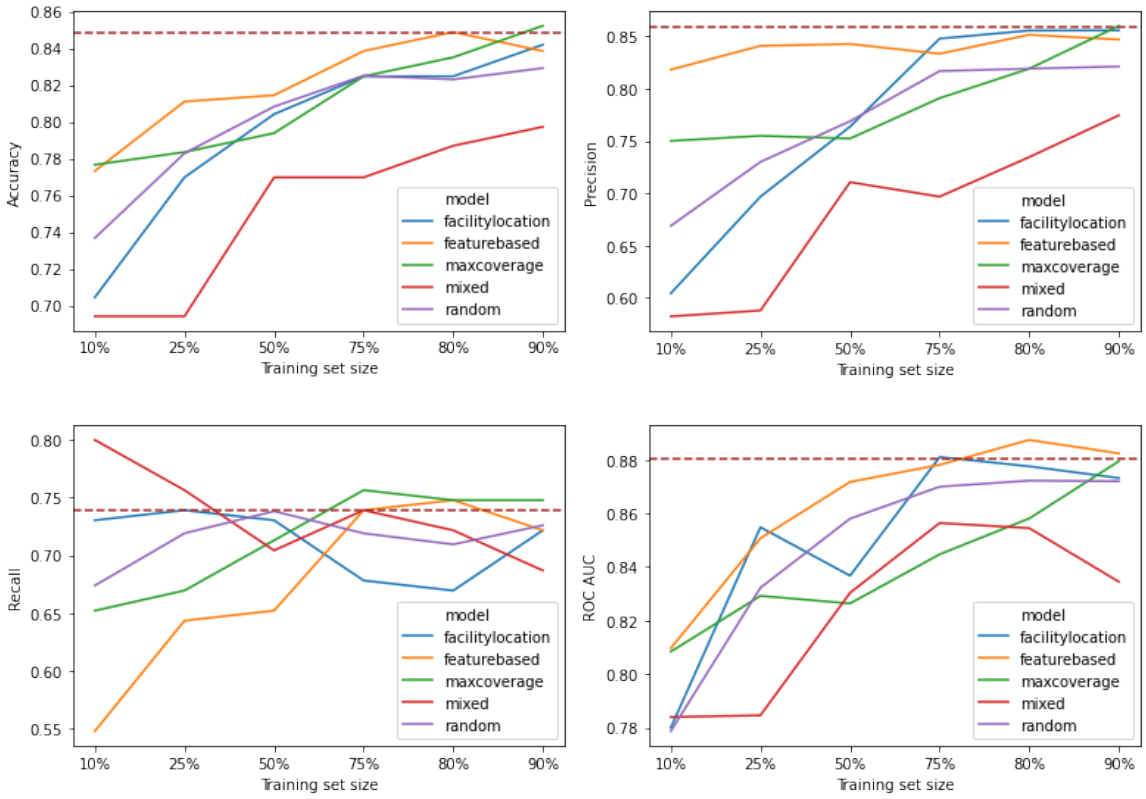
$$\mathcal{F}(X) = \sum_{i=1}^{d} \left(\left(\sum_{x \in X} x_i \right) > 0 \right)$$

Így is egy szubmoduláris függvényt kaptunk.

3. Saját eredmények

Miután elvégeztem néhány adatbányászathoz kapcsolódó kurzust (Intro to Machine Learning [5], Pandas [6]) kipróbáltuk az Apricot csomagot egy Kaggle [4] adathalmazon. Azért válasz-

tottuk a Kaggle-t, mert rengeteg érdekes adathalmazt tartalmaz, illetve sokhoz tartoznak már elkészült notebook-ok, amikből sokat lehet tanulni.
A választott adathalmaz: Titanic: Machine Learning from Disaster [7], ami 881 sorból és 67 oszlopból áll. Minden sorban egy, a Titanicon utazó személy különböző adatai vannak, illetve az, hogy túlélte-e a katasztrófát, vagy sem. A tanulási folyamat során azt szeretnénk elérni, hogy meg tudjuk jósolni a különböző adatokból, hogy az adott ember túlélte-e vagy sem. Ehhez a Gradient Boosting klasszifikációs modell bizonyult a leghatékonyabbnak [8]. Egy training-test vágás után a tanítási adathalmaz sorainak száma 590 volt (az eredeti adathalmaz $\frac{2}{3}$-a), ezt redukáltuk az Apricot-val (10%-ra, 25%-ra, 50%-ra, 75%-ra, 80%-ra, 90%-ra), és ezen a kisebb adaton tanítottuk a modellt, majd megvizsgáltuk, hogy az eredmény hogy változott a kiértékelésre félretett test adathalmazon. A már említett Apricot függvényeket használtuk (feature-based, facility location, max coverage), illetve ezek egy vegyítését (mixed), azaz olyat, amikor mindhárom függvénnyel kiválasztottunk ugyanannyi sort és ezeket egyesítettük, majd ezen tanítottuk a modellt. Emellett a random kiválasztást is kipróbáltuk és azért, hogy ebben az esetben is valós képet kapjunk, többszöri futtatás eredményeit átlagoltuk ki.
Az egyes modellek jóságát 4 különböző metrikával mértük: accuracy (a helyes jóslatok aránya), precision (azok közül, akiket a modell túlélőnek tart, valóban hányan élték túl), recall (azt mutatja, hogy a ténylegesen túlélők közül hányat talált meg a modell), roc auc (annak a valószínűsége, hogy a modell egy random túlélőként címkézett mintát előrébb helyez a rangsorban, mint egy random nem túlélőt). Az eredményeket grafikonokon szemléltettük, melyekről könnyen leolvasható, hogy a legjobb értékeket a feature-based modellel értük el, viszont meglepő eredmény, hogy a tanulási adathalmaz véletlen redukcióját nem sokkal múlja felül az Apricot módszer. Ennek oka lehet például, hogy jelen esetben egy viszonylag kis adathalmazzal dolgoztunk, illetve az is egy ok lehet, hogy nem volt eléggé redundáns az adat. Azt még fontos kiemelni, hogy a szubmoduláris kiválasztás ereje a legjobban az adathalmaz $10 - 25\%$-ára való csökkentésekor látszik. Itt a random módszer rendre gyengébb eredményt mutat, mint például a feature-based.



1. ábra: Elért eredmények

4. Tervek

A továbbiakban szeretnénk több, nagyobb adathalmazon is kipróbálni az Apricot módszert. Ott már érdekes lehet, hogy a futásidőt illetően történik-e javulás és ha igen, akkor mekkora. Hasznos lenne, ha észre tudnánk venni valami szabályszerűséget abban, hogy mely feladatokhoz, illetve kiértékelési metrikákhoz melyik szubmoduláris függvény használata a legjobb. Emellett egy érdekes feladat lenne még egy saját függvény implementálása is.

Hivatkozások

[1] Hui Lin, Jeff Bilmes, A Class of Submodular Functions for Document Summarization, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA, (2011), 510–520

[2] Jacob Schreiber, Jeffrey Bilmes, William Stafford Noble, Apricot: Submodular selection for data summarization in Python, (2019)

[3] https://github.com/jmschrei/apricot

[4] https://www.kaggle.com/

[5] https://www.kaggle.com/learn/intro-to-machine-learning

[6] https://www.kaggle.com/learn/pandas

[7] https://www.kaggle.com/c/titanic

[8] https://www.kaggle.com/yassineghouzam/titanic-top-4-with-ensemble-modeling