AdaBoost algoritmus alkalmazhatóságának vizsgálata

A feasibility study of the AdaBoost algorithm

HORNYÁK Olivér, PhD
egyetemi docens

Miskolci Egyetem, Gépészmérnöki és Informatikai Kar,
Miskolc-Egyetemváros, +36 46 565 000, oliver.hornyak@uni-miskolc.hu, www.ait.iit.uni-miskolc.hu

Abstract

AdaBoost, short for Adaptive Boosting, has emerged as a powerful ensemble learning technique with widespread applications in various domains of machine learning. This paper presents an exploration of AdaBoost, practical implementations, and recent applications. The primary objective of AdaBoost is to improve the performance of weak learners by adaptively assigning weights to training samples and iteratively refining the ensemble model. Furthermore, we discuss the practical considerations when applying AdaBoost, including feature selection or model selection.

Keywords: AdaBoost, algorithm, classification, Artificial intelligence, machine learning

Kivonat

Az Adaptive Boosting, röviden AdaBoost egy összegző tanulási technika, amely népszerűvé vált a gépi tanulás különböző területein való alkalmazásokban. AdaBoost fő célja az un. gyenge tanulók teljesítményének javítása a képzési minták adaptív súlyozásával és az összegző modell iteratív finomításával Ez a cikk az algoritmus paramétereit vizsgálja és áttekinti a tudományos felhasználási területeket. Ismerteti a gyakorlati szempontokat az AdaBoost alkalmazása során, ideértve a modell kiválasztását, a paraméterek meghatározását.

Kulcsszavak: AdaBoost, algoritmus, osztályozás, mesterséges intelligencia, gépi tanulás

1. BEVEZETÉS

A boosting algoritmusok olyan gépi tanulási módszerek, amelyeket arra terveztek, hogy javítsák au úgynevezett gyenge tanulók teljesítményét és növeljék a modellek pontosságát. Ezek a módszerek rendkívül hatékonyak a klasszifikációs és regressziós feladatok megoldásában, és széles körben alkalmazzák a gépi tanulás területén. A boosting lényege az, hogy egymás után tanítja meg a gyenge tanulókat, és kiemeli azokat a példákat vagy tulajdonságokat, amelyeket a korábbi modellek rosszul értelmeztek. Mindegyik iterációban egy új gyenge tanulót hoz létre, amely próbálja korrigálni a modellek korábbi hibáit. Az új tanuló a hibásan osztályozott példákat súlyozva kezeli, így a következő iterációban ezekre a példákra nagyobb hangsúlyt fektet. A legnépszerűbb boosting algoritmusok közé tartozik az AdaBoost [3], a Gradient Boosting [12] (például a XGBoost, LightGBM és a CatBoost), valamint a Stochastic Gradient Boosting [4]. Ezek a módszerek rendkívül eredményesek a valós világban előforduló bonyolult feladatok megoldásában, és a különböző iparágakban, például műszaki területeken, a pénzügyi elemzésben, az egészségügyben és a bioinformatikában is nagy népszerűségnek örvendenek. A boosting algoritmusok ereje abban rejlik, hogy képesek adaptív módon reagálni a tanító adatokban rejlő összetett összefüggésekre, és kiváló predikciós teljesítményt nyújtanak a megfelelő paraméterezéssel és a gyenge tanulók megfelelő kiválasztásával.

2. AZ ADABOOST ALGORITMUS

Az AdaBoost (rövidítve az "Adaptive Boosting"-ot) egy népszerű és hatékony boosting algoritmus a gépi tanulásban, amelyet Yoav Freund és Robert Schapire fejlesztett ki 1996-ban [3]. Az algoritmus lépéseit [8] részletesen leírja. Ez az algoritmus forradalmi jelentőségű volt a gépi tanulásban és a klasszifikációs

feladatok területénAz algoritmus iteratívan próbálja korrigálni a modell korábbi hibáit. A példák súlyait dinamikusan kezeli, igy azokat, amelyeket a korábbi modellek rosszul osztályoztak, nagyobb súllyal veszi figyelembe a következő iterációban.

A gyenge tanuló (weak learner) a boosting algoritmusok kontextusában olyan osztályozó vagy regressziós modell, amelynek teljesítménye csak kissé haladja meg a véletlenszerű szintet, azaz csak kissé jobb, mint a véletlen találgatás. Gyenge tanulók képesek osztályozni vagy előre jelezni az adatokat, de hajlamosak hibázni, és nem rendelkeznek erős predikciós teljesítménnyel. Az AdaBoost és más boosting algoritmusok alapelve az, hogy a gyenge tanulókat kombinálják egy erős osztályozó vagy regressziós modellé. Az egyes gyenge tanulók egyszerű modellek, például döntési fák vagy egyváltozós regressziók lehetnek. Az erős osztályozó vagy regressziós modell, amelyet a boosting folyamat létrehoz, az összes gyenge tanuló kombinációjából származik. A boosting algoritmus dinamikusan súlyozza a példákat és az egyes gyenge tanulókat, hogy kiemelje azokat a példákat, amelyekre a gyenge tanulók rosszul válaszolnak. Így a gyenge tanulók adaptívan javítják a teljesítményüket, és a végső, erős modell eredményesebben osztályozza vagy jelzi előre az adatokat. A gyenge tanulók használata a boosting algoritmusokban azért fontos, mert lehetővé teszi az algoritmusoknak a bonyolult összefüggések feltárását a tanító adatokban, anélkül, hogy túltanulnák azokat.

Az AdaBoost sikere annak köszönhető, hogy képes kezelni a bonyolult és átfedő osztályokat, és javítani a modellek teljesítményét, anélkül, hogy túlzottan túltanítaná a tanító adatokat. Az eredmény egy erős osztályozó, amely rendkívül jó előrejelzési teljesítményt nyújt. Az AdaBoost gyorsan népszerűvé vált a gépi tanulás tudományos közösségében, és később számos továbbfejlesztett változata jelent meg, például a számítógépes látásban és a természetes nyelvfeldolgozásban való alkalmazásokhoz. Az AdaBoost története és sikere szimbolikus a gépi tanulásban a gyenge tanulók erős osztályozóvá történő egyesítése révén, és az algoritmus még mindig fontos szerepet játszik a gépi tanulás gyakorlatában.

1.1. AdaBoost algoritmus paraméterei

A boosting algoritmusnak számos paramétere lehet, amelyek befolyásolják az algoritmus működését és teljesítményét. A konkrét paraméterek és azok jelentése az adott boosting algoritmustól függ, itt néhány általános paramétert említek:

- Learning Rate (Tanulási ráta): Ez a paraméter meghatározza, milyen nagy mértékben korrigálja a modell a hibákat minden iterációban. Nagyobb tanulási ráta gyorsabb konvergenciát eredményezhet, de túl nagy érték instabilitást okozhat. Általában kis értékek (pl. 0,1 vagy kisebb) használatosak.
- Number of Estimators (Becslők száma): Ez a paraméter meghatározza, hogy hány alapmodellt vagy gyenge tanulót fog használni a boosting algoritmus. Minél több becslőt használunk, annál erősebb lesz a modell, de ez növeli a futási időt is. Fontos meghatározni, hogy elegendő becslőt használunk-e ahhoz, hogy a modell megfelelően illeszkedjen a tanító adathalmazhoz, vagy hogy elkerüljük a túltanulást.
- Base Estimator (Alapmodell): Ez a paraméter meghatározza, milyen alapmodellt vagy gyenge tanulót használ a boosting algoritmus. Példák ilyen algoritmusokra a döntési fák, a lineáris regresszió, a SVM vagy a neurális hálózatok.
- Maximum Depth (Maximális mélység): Ha az alapmodell döntési fa, akkor a paraméter meghatározza a fa hierarchiai szintjeinek számát. A nagyobb mélységű fák több információt tanulhatnak a tanító adathalmazból, de emellett növelhetik a modell bonyolultságát és az overfitting kockázatát.
- Subsample (Almintavétel): Ez a paraméter meghatározza a tanító adathalmaznak a különböző iterációk során használt mintáinak a méretét. A kisebb mintavételi arányok csökkenthetik a modell túltanulását és javíthatják az általánosító képességet, de csökkenthetik a tanítás sebességét is.

Fontos megjegyezni, hogy ezek csak néhány példa általános paraméterekre a boosting algoritmusokhoz. A konkrét boosting algoritmustól függően további specifikus paraméterek is létezhetnek, amelyeket a dokumentáció vagy a megfelelő könyvtár ír le.

1.1. Az sklearn specifikus paraméterek

Jelen cikkhez az sklearn programcsomagot használtam [16]. Ez alapértelmezetten a DecisionTreeClassifier osztályt használja, de más osztályokat is meg lehet adni, amelyeket a sklearn támogat. Az n_estimators paraméter meghatározza a gyenge tanulók számát, vagyis hány iterációban fogja a modell hozzáadni az alapmodellt. Általában minél több tanulót használunk, annál erősebb lesz a modell, de

ezzel arányosan növekszik a futási idő is. A learning_rate paraméter meghatározza a tanulási rátát, vagyis mennyire változtatja meg a modell a súlyokat az egyes iterációk során. Alapértelmezett értéke 1.0, de a kisebb értékek (pl. 0,1 vagy kisebb) csökkenthetik a tanulási sebességet. Az algorithm: paraméter meghatározza az AdaBoost algoritmus használatának típusát. Alapértelmezett értéke "SAMME.R", amely jelentőséget rendel a predikciókhoz, de más opciók is elérhetők, mint például a "SAMME", ami a klasszikus AdaBoost algoritmust használja. A random_state: paraméter meghatározza a random generátor kezdőállapotát, ami hatással lehet a tanulás során történő véletlenszerű eseményekre. Azonos random_state érték használata biztosítja a reprodukálhatóságot.

1.2. A tanulási ráta hatása

A tanulási ráta hatását egy kódrészlettel szemléltetem. Az alábbi példakód bemutatja, hogyan befolyásolja a tanulási ráta az AdaBoost algoritmusának teljesítményét. A példában az Iris adathalmazt [17] használom, és az AdaboostClassifier osztályt a scikit-learn könyvtárból.

```
from sklearn.datasets import load_iris
from sklearn.ensemble import AdaboostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Adathalmaz betöltése
iris = load_iris()
X = iris.data
y = iris.target

# Adatok felosztása tanító- és teszthalmazra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Tanító adathalmaz illesztése az AdaboostClassifier modellel különböző tanulási
ráta értékekkel
learning_rates = [0.1, 0.5, 1.0, 1.5, 2.0]
for learning_rate in learning_rates:
    ada_boost = AdaboostClassifier(n_estimators=50, learning_rate=learning_rate,
random_state=42)
    ada_boost.fit(X_train, y_train)
    y_pred = ada_boost.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Learning Rate: {learning_rate}, Accuracy: {accuracy}")
```

A

Az eredmények a következők:

```
Learning Rate: 0.1, Accuracy: 0.9666666666666667
Learning Rate: 0.5, Accuracy: 0.9666666666666667
Learning Rate: 1.0, Accuracy: 1.0
Learning Rate: 1.5, Accuracy: 1.0
Learning Rate: 2.0, Accuracy: 0.9333333333333333
```

Az eredmények azt mutatják, hogy a tanulási ráta változtatása különböző hatással lehet a modell pontosságára. A példában látható, hogy a tanulási ráta 1.0 vagy nagyobb értékek esetén a modell 100%-os pontosságot ér el. Azonban egy túl magas tanulási ráta, például 2.0, kisebb pontosságot eredményezhet. Ez azért történhet, mert egy nagyobb tanulási ráta túl gyorsan változtatja meg a súlyokat, ami negatív hatással lehet a modell konvergenciájára és stabilitására.

1.3. Az algoritmus változtatásának hatása

Az alábbi példakód bemutatja, hogyan befolyásolja az algoritmus paraméter az AdaBoost teljesítményét. Cseréljük ki az algoritmus érdemi részét erre:

```
# Tanító adathalmaz illesztése az AdaboostClassifier modellel különböző
algorithm értékekkel
algorithms = ['SAMME', 'SAMME.R']
for algorithm in algorithms:
    ada_boost = AdaboostClassifier(n_estimators=50, algorithm=algorithm,
random_state=42)
    ada_boost.fit(X_train, y_train)
    y_pred = ada_boost.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Algorithm: {algorithm}, Accuracy: {accuracy}")
```

Az eredmények a következők:

```
Algorithm: SAMME, Accuracy: 0.9666666666666667
Algorithm: SAMME.R, Accuracy: 1.0
```

Az eredmények azt mutatják, hogy az algoritmus változtatása különböző hatással lehet a modell pontosságára. A példában látható, hogy a "SAMME.R" algoritmus 100%-os pontosságot ér el, míg a "SAMME" algoritmus kissé kisebb pontosságot eredményez. Az algorithm paraméter hatása az AdaBoost algoritmikus működésével kapcsolatos. A "SAMME" algoritmus a klasszikus AdaBoost algoritmust használja, amely a klasszifikációra épül és az osztályokra készít előrejelzéseket. A "SAMME.R" algoritmus viszont súlyozott értékekkel dolgozik, amelyek a predikciók valószínűségeit reprezentálják. Ezáltal a "SAMME.R" algoritmus több információt használ fel, és jobb eredményeket érhet el.

1.4 Az n_estimators hatása

Az alábbi példakód bemutatja, hogyan befolyásolja az n_estimators paraméter az AdaBoost algoritmusának teljesítményét.

```
# Tanító adathalmaz illesztése az AdaboostClassifier modellel különböző
n_estimators értékekkel
n_estimators_values = [10, 50, 100, 200, 500]
for n_estimators in n_estimators_values:
    ada_boost = AdaboostClassifier(n_estimators=n_estimators, random_state=42)
    ada_boost.fit(X_train, y_train)
    y_pred = ada_boost.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"n_estimators: {n_estimators}, Accuracy: {accuracy}")
```

Az eredmények a következők:

```
n_estimators: 10, Accuracy: 0.9333333333333333
n_estimators: 50, Accuracy: 1.0
n_estimators: 100, Accuracy: 1.0
n_estimators: 200, Accuracy: 1.0
n_estimators: 500, Accuracy: 1.0
```

Az eredmények azt mutatják, hogy az algoritmus pontossága általában növekszik az n_estimators értékének növekedésével. Ez azt jelenti, hogy minél több gyenge tanulót (alapmodellt) használunk, annál jobb eredményeket érhetünk el.

Azonban érdemes figyelni a túltanulásra is. Az látható, hogy a modell már 50 gyenge tanuló (n_estimators = 50) alkalmazásával is elérte a 100%-os pontosságot, és további tanulók hozzáadása nem növelte tovább a pontosságot. Ez arra utal, hogy az optimális számú gyenge tanuló elérhető volt a 50-es becslő értékkel

3. DISZKRÉT ÉS VALÓS ADABOOST ALGORITMUSOK

A diszkrét AdaBoost és a valós AdaBoost két változata az AdaBoost algoritmusnak, és mindkettőt az osztályozási feladatokban használják. Az alábbiakban összehasonlítom a két algoritmust A diszkrét AdaBoost osztályokra épül, és bináris osztályozási feladatokra szorítkozik. A valós AdaBoost a valós értékekkel működik, amelyek a predikciók súlyozására szolgálnak. Ez lehetővé teszi, hogy az algoritmus ne

csak bináris osztályozást végezzen, hanem többosztályos klasszifikációt is támogasson. Az algoritmusban a rosszul osztályozott példák súlya növekszik, míg a helyesen osztályozottaké csökken, a súlyozásnál valós vagy diszkrét értékeket használva. A súlyok frissítése az algoritmus lépéseinek végrehajtása során történik, figyelembe véve a predikciók valószínűségeit. Osztályozók szempontjából a diszkrét AdaBoost algoritmusban a gyenge tanulók általában bináris osztályozók, például döntési fák. A valós AdaBoost nem korlátozódik a bináris osztályozókra. Gyenge tanulók lehetnek bármilyen típusú osztályozók, amelyek valós értékeket adnak vissza. Robosztusságát tekintve a diszkrét AdaBoost algoritmusra jellemző, hogy érzékeny lehet a zajos vagy kiugró értékekre az adatokban. Kiugrónak tekintjük azt a megfigyelt értéket, ami távol van a többi megfigyelt értéktől A valós AdaBoost robosztusabb a kiugró értéketekkel vagy zajos adatokkal szemben, mivel a valós értékekkel dolgozik. Mindkét algoritmus célja, hogy különböző gyenge tanulókat kombináljon, hogy erős osztályozót hozzon létre. Az AdaBoost algoritmus általános elve a súlyozott példákra és az iteratív módon történő tanulásra épül mindkét változatban. A diszkrét AdaBoost és a valós AdaBoost között úgy célszerű választani, hogy bináris vagy többosztályos osztályozást végezünk-e, és hogy a feladat mennyire érzékeny a zajos adatokra vagy kiugró értékekre.

4. AZ ADABOOST ALGORITMUS FELHASZNÁLÁSI TERÜLETEI

Objektumfelismerés és gépi látás: Az AdaBoost nagyon hatékony az objektumfelismerésben és a gépi látásban [7]. Az a képessége, hogy gyenge osztályozókat kombináljon lehetővé teszi, hogy pontosan és megbízhatóan ismerjen fel objektumokat képeken vagy videókon. Az AdaBoostot alkalmazzák biometriai felismerési feladatokban, például arcfelismerésben [1], ujjlenyomat-azonosításban [10] vagy hangfelismerésben [5]. Az algoritmus segít a jellemzők kiválasztásában és az osztályozók kombinálásában. Az AdaBoostot használják szövegelemzési feladatokban, például spam-szűrésben vagy érzelmek felismerésében a szövegekben. Az algoritmus segít a fontos jellemzők kiválasztásában és az osztályozásban. Az AdaBoostot alkalmazzák a genetikában vagy fehérjék funkcionális elemzésében, például gének osztályozásában. Segít a jellemzők kiválasztásában és az osztályozók erősítésében. Az AdaBoostot alkalmazzák a pénzügyi előrejelzésekben, például részvényárfolyamok vagy pénzügyi mutatók előrejelzésében. Az algoritmus segít a trendek felismerésében és a pénzügyi adatok osztályozásában. Ezen felül az még számos más területen AdaBoostot alkalmazzák, például gépi tanulásban, adatbányászatban, gyógyászatban, környezeti kutatásban és sok más alkalmazási területen, ahol az osztályozás és előrejelzés fontos szerepet játszik.

Az AdaBoost magyar nyelvű alkalmazásaira is vannak példák. Természetes nyelvfeldolgozásra használta az algoritmust [2], szövegfeldolgozási alkalmazást mutat be [5]. A fej térbeli helyzetének meghatározása és a tekintet irányának meghatározását vizsgálta [1]. Gépek meghibásodásának előre jelezhetőségét vizsgálta [8]. A vállalkozói képességre jellemző karakterisztikákat vizsgálta az algoritmussal [6].

5. ÖSSZEFOGLALÁS

A cikkben áttekintettem, hogy az AdaBoost algoritmus paramétereinek megváltoztatása hogyan befolyásolja annak kimenetét. A paraméterek kis mértékben, de befolyásolják a teljesítőképességet. Alapvetően az algoritmus megfelelően robosztus, az osztályzási feladatokban kiválóan alkalmazható. Irodalmi feldolgozás alapján a jellemző felhasználási területeket is áttekintetem a cikkben. A cikknek nem volt tárgya, hogy az adathalmaz kiegyensúlyozottságát vizsgálja. Torz adathalmaz esetén a mintavételezés súlyozása segíthet a segíthet a modellnek jobban teljesíteni az összes osztályon, és objektív döntéseket hozni.

KÖSZÖNETNYILVÁNÍTÁS

A kutatás a Nemzeti Kutatási Fejlesztési és Innovációs Hivatal 2020-1.1.2-PIACI-KFI-2020-00147 pályázati támogatásával valósult meg. A Szerző köszöni a támogatást.

IRODALMI HIVATKOZÁSOK

[1] Bertók, K., Fazekas. A., Sajó L. *A fej térbeli helyzetének meghatározása és a tekintet irányának meghatározása, Képfeldolgozók és Alakfelismerők Országos Konferenciája* (KÉPAF VIII.), 2011-01-25 - 2011-01-28, Szeged, Hungary

[2] Felföldi L *Osztályozók kombinációs rendszerei és alkalmazásuk a természetes nyelvi technológiában*. PhD-értekezés, 2010

[3] Freund, Y., Schapire, R. E. *Experiments with a new boosting algorithm*. In icml (Vol. 96, pp. 148-156). 1996, July).
[4] Friedman, Jerome H. *Stochastic gradient boosting*. Computational statistics & data analysis 38.4 (2002): 367-378.
[5] Gosztolya, G., Busa-Fekete, R. *Calibrating AdaBoost for phoneme classification*. Soft Computing, 23(1), 115-128. 2019
[6] Gosztonyi, M., Judit, C. F. Profiling (Non-) *Nascent Entrepreneurs in Hungary Based on Machine Learning Approache*s. Sustainability, 14(6), 3571. 2022
[7] Guo, L., Ge, P. S., Zhang, M. H., Li, L. H., Zhao, Y. B. *Pedestrian detection for intelligent transportation systems combining AdaBoost algorithm and support vector machine. Expert Systems with Applications*, 39(4), 4274-4286. (2012).
[8] Hornyák, O., Iantovics, L. B. *AdaBoost Algorithm Could Lead to Weak Results for Data with Certain Characteristics*. Mathematics, 11(8), 1801. 2023.
[9] Islam, S. M., Bennamoun, M., & Davies, R. (2008, January). *Fast and fully automatic ear detection using cascaded adaboost*. In 2008 IEEE Workshop on Applications of Computer Vision (pp. 1-6). IEEE.
[10] Liu, M. (2010). *Fingerprint classification based on AdaBoost learning from singularity features*. Pattern Recognition, 43(3), 1062-1070.
[11] Mathanker, S. K., Weckler, P. R., Bowser, T. J., Wang, N., Maness, N. O. *AdaBoost classifiers for pecan defect classification. Computers and electronics in agriculture*, 77(1), 60-68. 2011.
[12] Natekin, Alexey, and Alois Knoll. "Gradient boosting machines, a tutorial." Frontiers in neurorobotics 7 (2013): 21.
[13] Reyzin, L., Schapire, R.E.: *How boosting the margin can also boost classifier complexity*. Proceedings of the 23rd International Conference on Machine Learning (2006)
[14] Thilagavathi, B., K. Suthendran, K. Srujanraju. *Evaluating the AdaBoost algorithm for biometric-based face recognitio*n.. Data Engineering and Communication Technology: Proceedings of ICDECT 2020. Springer Singapore, 2021.
[15] Tharwat, A., Hemedan, A. A., Hassanien, A. E., & Gabel, T. (2018). *A biometric-based model for fish species classification*. Fisheries research, 204, 324-336.
[16] https://scikit-learn.org/stable/about.html (Utolsó letöltés: 2023. 09.15).
[17] UCI Machine Learning Repository: Iris Data Set. archive.ics.uci.edu. (Utolsó letöltés: 2023. 09.15).
[18]