TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Boosting (AdaBoost)

Zadanie na predmet strojové učenie

2023 **QuietXO**

Abstrakt v SJ

Cieľom tohto zadania je implementovať vylepšenú verziu Boosting algoritmu, AdaBoost, v programovacom jazyku Python a to od úplného základu za použitia iba knižnice Numpy. Implementácia tohto algoritmu bola testovaná na medických dátach a to konkrétne predikcia infarktu a identifikácia typu rakoviny, kde výsledky týchto testov vyzerajú sľubné. V tomto dokumente budú zahrnuté dáta na predikciu infarktu.

Kľúčové slová v SJ

Boosting, AdaBoost, Strojové učenie, Rozhodovacie stromy

Abstract

The goal of this assignment is to implement an improved version of the Boosting algorithm, AdaBoost, in the Python programming language from scratch using only the Numpy library. The implementation of this algorithm was tested on medical data, namely the prediction of a heart attack and the identification of the type of cancer, where the results of these tests seem promising. In this document, there will be data for the prediction of a heart attack.

Keywords

Boosting, AdaBoost, Machine Learning, Decision Trees

Obsah

Zoznam obrázkov 4

Zoznam tabuliek 5

Úvod 6

1. Teoretický rozbor Boosting-u a AdaBoost-u 7

1.1. Teoretický rozbor AdaBoost-u 7

1.2. Štruktúra AdaBoost-u [2] 7

1.3. Popis infarktového súboru dát 8

2. Implementácia AdaBoost-u v jazyku Python 9

2.1. Knižnice 9

2.2. Rozhodovací peň 9

2.3. konštruktor AdaBoost modelu 9

2.4. fit funkcia AdaBoost modelu 10

2.5. predict funkcia AdaBoost modelu 11

2.6. show_classification funkcia AdaBoost modelu 12

3. Testovanie rozhodovacieho modelu AdaBoost 13

3.1. Dáta rizika infarktu 13

3.2. Náš preklad dát 13

3.3. Testovanie dát 13

Záver 14

Zoznam použitej literatúry 15

Zoznam obrázkov

Obr. 1 Knižnice 9
Obr. 2 Slabý klasifikátor (rozhodovací peň) 9
Obr. 3 AdaBoost konštruktor 9
Obr. 4 AdaBoost fit – inicializácia váh 10
Obr. 5 AdaBoost fit – vytvorenie slabého klasifikátora 10
Obr. 6 AdaBoost fit – vyhľadávanie najlepšieho príznaku 10
Obr. 7 AdaBoost fit – uloženie slabého klasifikátora 11
Obr. 8 AdaBoost predict 11
Obr. 9 AdaBoost show_classification 12
Obr. 10 príklad AdaBoost show_classification 12
Obr. 11 Dáta rizika Infarktu 13
Obr. 12 Preklad dát 13
Obr. 13 Testovanie dát 13

Zoznam tabuliek

Tab. 1 Popis príznakov infarktového súboru dát [5] 8

Úvod

Cieľom tohto zadania je implementovať vylepšenú verziu Boosting algoritmu, AdaBoost, v programovacom jazyku Python a to od úplného základu za použitia iba knižnice Numpy, ktorá slúži na zjednodušenú prácu s maticami, v ktorých máme uložené dáta.

V prvej časti si vysvetlíme teóriu a fungovanie algoritmu AdaBoost, následne ho v druhej časti implementujeme a nakoniec ho v tretej časti otestujeme na dvoch setoch dát.

1. Teoretický rozbor Boosting-u a AdaBoost-u

Boosting je súborová technika, ktorá sa pokúša vytvoriť silné klasifikátory z množstva slabých klasifikátorov. Na rozdiel od mnohých modelov strojového učenia, ktoré sa zameriavajú na vysokokvalitnú predikciu vykonávanú pomocou jedného modelu, zosilňujúce algoritmy sa snažia zlepšiť predikčnú silu trénovaním postupnosti slabých modelov, z ktorých každý kompenzuje slabé stránky svojich predchodcov. Boosting poskytuje modelom strojového učenia výkon na zlepšenie ich presnosti predpovedí.

1.1. Teoretický rozbor AdaBoost-u

AdaBoost (Adaptive Boosting), je algoritmus strojového učenia, ktorý je založený na sledovaní rozhodovacieho stromu s hĺbkou jedna (je to v podstate les pňov). AdaBoost funguje tak, že kladie väčší dôraz na vysoko klasifikovateľné pravidlá a menšiu váhu na tie, ktoré sú menej dôležité. Algoritmus AdaBoost je vyvinutý na riešenie problému klasifikácie aj regresie.

Myšlienka AdaBoost-u spočíva v kombinácii mnohých slabých klasifikátorov vytvárajúcich silný klasifikátor. Rozhodovacie pne (jeden uzol a dva listy) nie sú skvelé na presnú klasifikáciu, takže to nie je nič iné ako slabý klasifikátor, no niektoré rozhodovacie pne majú vyšší výkon alebo sa lepšie klasifikujú ako iné.

Rozhodovací peň sa robí tak, že sa vezmú do úvahy predchádzajúce chyby v rozhodovacích pňoch a na rozdiel od klasického Boosting-u zoberieme do úvahy pri AdaBoost-e aj váhy jednotlivých zle klasifikovaných dát.

1.2. Štruktúra AdaBoost-u [2]

Začneme s inicializáciou váh $\boldsymbol{\omega_i = \frac{1}{N} \in [0,\ 1]}$, kde N reprezentuje počet dát v sete našich vstupných dát. Následne potrebujeme vyrátať celkovú chybu pre daný rozhodovací peň, kde $chyba = sčítanie\ váh\ zle\ klasifikovaných\ tried$ a túto váhu využijeme pri výpočte sily α tohto rozhodovacieho pňa $\boldsymbol{\alpha_t = \frac{1}{2} ln \frac{1-chyba}{chyba}}$, nakoniec ešte aktualizujeme váhy $\boldsymbol{\omega_i = \omega_{i-1} * e^{\pm\alpha}}$ a normalizujeme aby sa ich súčet rovnal 1. Proces môžeme opakovať pokiaľ nedostaneme nami určený počet rozhodovacích pňov.

Ak máme model natrénovaný, môžeme sa pokúsiť o predikciu na základe vstupných dát pomocou vzorca $\boldsymbol{P_x = sign\left[\sum_{m=1}^{M} \alpha_m * G_m(x)\right]}$, kde x sú naše vstupné príznaky a α_m predstavuje ako veľmi vezmeme daný výstup klasifikátora $G_m(x)$ do úvahy.

1.3. Popis infarktového súboru dát

Na otestovanie tohto AdaBoost modelu boli použité dáta na predikciu infarktu [5], kde daný súbor dát pozostáva s 13 príznakov + výstup (vysoké/nízke riziko infarktu) a 303 záznamov. Príznaky a ich vysvetlenie sa nachádza v Tab. 1.

Tab. 1 Popis príznakov infarktového súboru dát [5]

Index	***Príznak***	***Popis***	***Hodnota***
0	age	Vek pacienta	(roky)
1	sex	Pohlavie pacienta	(1 = muž; 0 = žena)
2	cp	Typ bolesti na hrudníku	0: typická angína
			1: netypická angína
			2: neanginózna bolesť
			3: asymptomatické
3	trtbps	Pokojový krvný tlak	(mm*Hg)
4	chol	Cholesterol získaný pomocou BMI senzora	(v mg/dl)
5	fbs	Hladina cukru v krvi nalačno > 120 mg/dl	(1 = áno; 0 = nie)
6	restecg	Pokojové elektro-kardiografické výsledky	0: normálne
			1: s abnormalitou ST-T vlny
			2: hypertrofia ľavej komory
7	thalachh	Maximálna dosiahnutá srdcová frekvencia	(bpm)
8	exng	Angína vyvolaná cvičením	(1 = áno; 0 = nie)
9	oldpeak	Predchádzajúci vrchol	-
10	slp	Strmosť ST/srdcová frekvencia	[0, 1, 2]
11	caa	Počet hlavných ciev	[0, 1, 2, 3, 4]
12	thall	Miera krvnej poruchy talasémie	[0, 1, 2, 3]
13	output	0 = menšia šanca na infarkt & 1 = väčšia šanca na infarkt	

2. Implementácia AdaBoost-u v jazyku Python

Algoritmus je implementovaný v prostredí Jupyter Notebook, (z vizuálnych dôvodov,) ktorý využíva jazyk Python a vedľajšie funkcie na načítanie dát v samostatnom Python súbore

2.1. Knižnice

```
import DataLoader
import numpy as np
import pandas as pd
```

Obr. 1 Knižnice

- DataLoader – je nami implementovaná knižnica na načítavanie dát do potrebnej podoby
- Numpy – knižnica na prácu s poľami
- Pandas – knižnica na zobrazenie našich dát

2.2. Rozhodovací peň

```
class DecisionStump:
    def __init__(self):
        self.polarity = 1          # It is a positive or negative sample
        self.feature_idx = None    # Which Feature will this Decision Stump use
        self.threshold = None      # The amount by which we choose
        self.alpha = None          # How important this Decision Stump is final prediction

    def predict(self, X):
        n_samples = X.shape[0]
        X_column = X[:, self.feature_idx]
        predictions = np.ones(n_samples)
        if self.polarity == 1:
            predictions[X_column < self.threshold] = -1
        else:
            predictions[X_column > self.threshold] = -1

        return predictions
```

Obr. 2 Slabý klasifikátor (rozhodovací peň)

Slúži ako slabý klasifikátor, ktorý si pamätá svoju rozhodovaciu silu (*alpha*), podľa čoho (*feature_idx*) a akej hodnoty (*threshold*) bude robiť následnú predikciu, ktorej polarita záleží na parametri *polarity*. Funkcia *predict* vie následne po vložení dát na základe týchto parametrov vytvoriť výsledky.

2.3. konštruktor AdaBoost modelu

```
def __init__(self, n_clf=5):
    self.n_clf = n_clf
    self.clfs = []
```

Obr. 3 AdaBoost konštruktor

- n_cls – určuje koľko slabých klasifikátorov chceme použiť na vytvorenie modelu
- clfs – slúži na uloženie vytvorených klasifikátorov

2.4. fit funkcia AdaBoost modelu

```python
def fit(self, X, y):
    """ Train the AdaBoost Model """
    n_samples, n_features = X.shape

    # Initialize weights to 1/N
    w = np.full(n_samples, (1 / n_samples))
```

Obr. 4 AdaBoost fit – inicializácia váh

V tejto časti si zistíme dimenzie našich dát (ich počet + samotné dáta) a inicializujeme počiatočné váhy pre jednotlivé dáta.

```python
    # Iterate through classifiers
    for _ in range(self.n_clf):
        clf = DecisionStump()
        min_error = float("inf")
```

Obr. 5 AdaBoost fit – vytvorenie slabého klasifikátora

Následne už budeme vytvárať objekty slabých klasifikátorov (*clf*) pre ktoré musíme nájsť vhodné parametre, na čo slúži nasledovná časť kódu na obrázku 6.

```python
# Greedy search to find the best threshold and feature
for feature_i in range(n_features):
    X_column = X[:, feature_i]
    thresholds = np.unique(X_column)

    for idx in range(len(thresholds) - 1):
        # Calculate the threshold
        threshold = (thresholds[idx] + thresholds[idx + 1]) / 2

        # Predict with polarity 1
        p = 1
        predictions = np.ones(n_samples)
        predictions[X_column < threshold] = -1

        # Error = sum of misclassified samples weight
        error = sum(w[y != predictions])

        # Swap polarity & fix error
        if error > 0.5:
            error = 1 - error
            p = -1

        # Save the best configuration
        if error < min_error:
            clf.polarity = p
            clf.threshold = threshold
            clf.feature_idx = feature_i
            min_error = error
```

Obr. 6 AdaBoost fit – vyhľadávanie najlepšieho príznaku

Začneme výberom jedného stĺpca príznakov, z ktorého si vyberieme a zotriedime jedinečné hodnoty od najmenšej po najväčšiu a následne testujeme jednotlivé medzi-hodnoty tohto stĺpca a porovnávame ich s výstupom, pomocou čoho vyrátame chybovosť a následne, ak je naša chyba menšia, ako ktorákoľvek predošlá, uložíme si hodnoty tohto slabého klasifikátora. Po dokončení konkrétneho stĺpca príznakov pokračujeme na ďalší, pokiaľ neprejdeme všetkými. (Poznámka: na klasifikáciu nepoužívame Gini index, ale hrubú silu, keďže by mala byť rýchlejšia a rovnako správna.)

```
        # Calculate alpha (How much impact this rule has on final output)
        EPS = 1e-10  # Make sure we never divide by 0
        clf.alpha = 0.5 * np.log((1.0 - min_error + EPS) / (min_error + EPS))

        # Calculate predictions
        predictions = clf.predict(X)

        # Update weights & normalization to one
        w *= np.exp(-clf.alpha * y * predictions)
        w /= np.sum(w)

        # Save classifier
        self.clfs.append(clf)
```

Obr. 7 AdaBoost fit – uloženie slabého klasifikátora

Následne si vypočítame rozhodovaciu silu daného slabého klasifikátora, nájdeme chybné klasifikácie, upravíme a normalizujeme váhy aby sa ich súčet rovnal jednej, uložíme konktrétny slabý klasifikátor do premennej *clfs* a pokračujeme vo vytváraní ďalších pravidiel podľa nami zvoleného počtu.

2.5. predict funkcia AdaBoost modelu

```
def predict(self, X):
    """Create List of predictions for every classifier, sum them & normalize"""
    clf_preds = [clf.alpha * clf.predict(X) for clf in self.clfs]
    y_prediction = np.sum(clf_preds, axis=0)
    y_prediction = np.sign(y_prediction)

    return y_prediction
```

Obr. 8 AdaBoost predict

Ak máme náš model hotový, pomocou tejto funkcie prejdú vložené dáta *X* všetkými rozhodovacími pravidlami ktoré sú vynásobené ich silou a sčítané. Funkcia nám vráti normalizované hodnoty *1* alebo *-1*.

2.6. show_classification funkcia AdaBoost modelu

```python
def show_classification(self, Table, Translate):
    Index = 0
    for clf in self.clfs:
        Index += 1
        if Index < 10:
            print(f'0{Index}. Decision: {Table.columns[clf.feature_idx]} | '
                  f'Threshold: {"Less Than" if clf.polarity == -1 else "Greater Than"} '
                  f'{round(clf.threshold, 3)} is {Translate[1]} | Power of Rule: {round(clf.alpha, 3)}')
        else:
            print(f'{Index}. Decision: {Table.columns[clf.feature_idx]} | '
                  f'Threshold: {"Less Than" if clf.polarity == -1 else "Greater Than"} '
                  f'{round(clf.threshold, 3)} is {Translate[1]} | Power of Rule: {round(clf.alpha, 3)}')
```

Obr. 9 AdaBoost show_classification

Funkcia nám zrozumiteľne vypíše podľa čoho sa náš AdaBoost bude rozhodovať pri klasifikácii daných dát.

```python
# Show the decision-making process
model.Model.show_classification(heart_raw, translate)
```

```
01. Decision: cp | Threshold: Greater Than 0.5 is Positive | Power of Rule: 0.589
02. Decision: caa | Threshold: Less Than 0.5 is Positive | Power of Rule: 0.528
03. Decision: slp | Threshold: Greater Than 1.5 is Positive | Power of Rule: 0.487
04. Decision: sex | Threshold: Less Than 0.5 is Positive | Power of Rule: 0.408
05. Decision: thall | Threshold: Less Than 2.5 is Positive | Power of Rule: 0.37
```

Obr. 10 príklad AdaBoost show_classification

3. Testovanie rozhodovacieho modelu AdaBoost

3.1. Dáta rizika infarktu

```
# Show structure of dataset
heart_raw = pd.read_csv('data/heart.csv')
heart_raw.head()
```

5 rows | 5 rows × 14 columns

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Obr. 11 Dáta rizika Infarktu

3.2. Náš preklad dát

```
# Prepare the dataset
dataset = DataLoader.load_dataset('data/heart.csv')
train_data, test_data, translate = DataLoader.tt_split_dataset(dataset, train=0.8, shuffle=False, data='heart')
X_train, y_train = DataLoader.xy_split_dataset(train_data)
X_test, y_test = DataLoader.xy_split_dataset(test_data)

# Show structure of new dataset
print(f'{train_data[0]}\n{train_data[-1]}\nTranslation: {translate}')
```

```
[63.0, 1.0, 3.0, 145.0, 233.0, 1.0, 0.0, 150.0, 0.0, 2.3, 0.0, 0.0, 1.0, 1]
[47.0, 1.0, 0.0, 110.0, 275.0, 0.0, 0.0, 118.0, 1.0, 1.0, 1.0, 1.0, 2.0, -1]
Translation: {1: 'Positive', -1: 'Negative'}
```

Obr. 12 Preklad dát

3.3. Testovanie dát

```
# Test the AdaBoost Model
model = ModelTesting(X_train, y_train, X_test, y_test)
n_cls = model.optim(Max_Cls=20, Stop=False, Gap=5)
acc = model.accuracy()
pre = model.precision()
rec = model.recall()
f1s = model.f1score()

# Print the Results
print(f'Nuber of classifiers: {n_cls}\n'
      f'Accuracy: {round(acc*100, 2)}% | Precision: {round(pre*100, 2)}% | '
      f'Recall: {round(rec*100, 2)}% | F1 Score: {round(f1s*100, 2)}%')
```

```
Nuber of classifiers: 5
Accuracy: 81.97% | Precision: 84.85% | Recall: 82.35% | F1 Score: 83.58%
```

Obr. 13 Testovanie dát

Optim nám nájde ideálny počet klasifikátorov pre dané dáta s nami určenou horou hranicou.

Záver

AdaBoost je výkonný a široko používaný algoritmus v strojovom učení, ktorý môže zlepšiť výkon mnohých rôznych typov modelov. Hlavnou silnou stránkou AdaBoost je jeho schopnosť spojiť slabé klasifikátory do silného, ktorý dokáže robiť presnejšie predpovede aj v prítomnosti údajov, ktoré obsahujú šum alebo sú zložité. Náš model dosahuje pri testovaní rizika infarktu presnosť okolo 82% a pri identifikácii typu rakoviny 96.5%, čo sú sľubné hodnoty.

Zoznam použitej literatúry

[1]. Starmer, Josh. AdaBoost, Clearly Explained. *StatQuest.* [Online] 14. Jan 2019. Dostupné na internete: <https://www.youtube.com/watch?v=LsK-xG1cLYA>.

[2]. Kurama, Vihar. A Guide to AdaBoost: Boosting to Save The Day. *Paperspace Blog.* [Online] 2020. Dostupné na internete: <https://blog.paperspace.com/adaboost-optimizer>.

[3]. Polzer, Dominik. AdaBoost, Step-by-Step. *Towards Data Science.* [Online] 4. Aug 2022. Dostupné na internete: <https://towardsdatascience.com/adaboost-in-7-simple-steps-a89dc41ec4>.

[4]. Shung, Koo Ping. Accuracy, Precision, Recall or F1? *Towards Data Science.* [Online] 15. Mar 2018. Dostupné na internete: <https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9>.

[5]. Rahman, Rashik. Heart Attack Analysis & Prediction Dataset. [Online] April 2023 Dostupné na internete: <https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset>.

[6]. Taha, Erdem. Cancer Data [Online] Mar 2023. Dostupné na internete: <https://www.kaggle.com/datasets/erdemtaha/cancer-data>.