Ansambli (eng. ensembles)

Delovi materijala su preuzeti iz skripte "Mašinsko učenje" profesora Mladena Nikolića

Ansambli (eng. ensemble) su jedan od najuspešnijih pristupa u mašinskom učenju. Ansambli su skupovi većeg broja modela koji zajednički donose odluke. Ideja koja stoji iza ovog pristupa je da svaki model pravi relativno smisleno predviđanje, pri čemu su greške različitih modela međusobno nezavisne. Uprosečavanjem se nezavisne greške poništavaju, što daje dobru preciznost modela. U slučaju da su podaci predstavljeni u vektorskom obliku, ansambli predstavljaju najpouzdaniji pristup u postizanju visoke preciznosti. Dakle u tom slučaju su precizniji i jednostavniji za obučavanje od neuronskih mreža. Sa druge strane, ako imamo sirove podatke, neuronske mreže su mnogo bolje.
Postoje dve vrste ansambala: prosta agregacija (eng. bagging) - tipičan predstavnik je randomforest, i pojačavanje (eng. boosting) - tipični predstavnici su XGBoost i AdaBoost.

Prosta agregacija (eng. bagging)

Prosta agregacija podrazumeva obučavanje većeg broja modela koji pojedinačno ne moraju imati visoku preciznost, ali čije su greške nezavisne. Prilikom predviđanja, svi modeli nude svoja predviđanja, koja se agregiraju kako bi se dobilo predviđanje ansambla. Ukoliko se radi o regresiji, agregacija se tipično vrši uprosečavanjem (a moguće je koristiti i medijanu), a u slučaju klasifikacije glasanjem. Snaga ovakvog modela počiva na ideji da će se prilikom agregacije greške koje modeli nezavisno prave poništiti. Na primer, ukoliko se rešava problem regresije i ako se modeli posmatraju kao nezavisne slučajne promenljive $X_1, . . , X_m$ sa istom raspodelom koja ima očekivanje μ i disperziju σ^2. Na osnovu centralne granične teoreme važi:

$$\frac{1}{m}\sum_{i=1}^{m} X_i \to \mathcal{N}(\mu, \frac{\sigma^2}{m})$$

Drugim rečima, prosek modela približno ima očekivanje μ i disperziju $\frac{\sigma^2}{m}$, odnosno, prosta agregacija nudi mogućnost smanjenja disperzije bez povećanja sistematskog odstupanja, što vodi povećanju preciznosti. Slična analiza se može izvesti i u slučaju klasifikacije. Naravno, pretpostavka o nezavisnosti slučajnih promenljivih X_i predstavlja idealizovanu pretpostavku koja ne može biti sasvim ispunjena u praksi, što se odražava i na mogućnosti ansambla.
Tipično, rezultati su bolji što je broj modela veći. Ipak, sa povećanjem broja modela, raste i računska zahtevnost obučavanja i predvidanja. Modeli koji se koriste pri agregaciji ne moraju biti iste reprezentacije, iako često jesu.

Slučajna šuma (eng. random forest)

Slučajne šume (eng. randomforest) su jedan od najkorišćenijih algoritama mašinskog učenja. Predstavljaju algoritam proste agregacije i sastoje se od stabala odlučivanja. Mogu se koristiti i za regresiju i za klasifikaciju.
Pretpostavimo da imamo p prediktora. Trening skup delimo na n delova. Obučavamo ukupno n stabala tako što za svako stablo koristimo različiti deo skupa za treniranje, a ne koristimo ni sve prediktore, već samo $\sqrt{p}$ prediktora, pri čemu za svako stablo na slučajan način biramo prediktore koje koristimo. !

[slika1.png](attachment:slika1.png) Dakle, ako se bavimo regresijom, rezultat je srednja vrednost svih predikcija stabala, a ako se bavimo klasifikacijom rezultat je kategorija koju je predvideo najveći broj stabala.

Sledeći kod je većinski preuzet od asistenata Anđelke Zečević i Milana Čugurovića sa časova vežbi iz Mašinskog učenja.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn import model_selection
from sklearn import preprocessing
from sklearn import metrics
```

In [3]:
```
Prvo ćemo učitati podatke i pripremiti ih za treniranje i testiranje.
```

```
  Cell In [3], line 1
    Prvo ćemo učitati podatke i pripremiti ih za treniranje i testiranje.
         ^
SyntaxError: invalid syntax
```

In [ ]:
```
data = pd.read_csv('diabetes.csv')
```

In [ ]:
```
data.head()
```

In [ ]:
```
y = data['Outcome']
```

In [ ]:
```
X = data.drop(columns=['Outcome'], axis=1)
```

In [ ]:
```
X_train, X_test, y_train, y_test = model_selection.train_test_split(
    X, y, test_size=0.33, stratify=y, random_state = 7)
```

In [ ]:
```
scaler = preprocessing.StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Funkcije za rad sa ansamblima su dostupne kroz ensamble paket scikit-learn biblioteke.

In [ ]:
```
from sklearn import ensemble
```

Na nivou ansambla se može zadati broj stabala koja se treniraju (n_estimators parametar), kao i svojstva koja prate stabla (kriterijum homogenosti, maksimalna dubina stabla, maksimalni broj atributa,..). Svojstvom max_samples se može uticati na veličinu podskupa instanci nad kojim se stabla treniraju. Praksa je da se kroz random_state parametar prati slučajnost u podelama kako bi eksperimenti mogli da se reprodukuju.

In [ ]:
```
model_forest = ensemble.RandomForestClassifier(n_estimators=20, max_depth=3, random_stat
```

In [ ]:
```
model_forest.fit(X_train, y_train)
```

In [ ]:
```
y_predicted = model_forest.predict(X_test)
```

In [ ]:
```
metrics.accuracy_score(y_test, y_predicted)
```

Sada i grafik pojedinačnih važnosti atributa izgleda nešto drugačije.

In [ ]:
```
plt.barh(list(X.columns), model_forest.feature_importances_)
plt.show()
```

Važno je napomenuti da se slučajne šume mogu kreirati i nad drugim baznim modelima podešavanjem parametra base_estimator, ali je u praksi najprisutniji rad sa samim stablima odlučivanja.