13. óra

Adatelemzési platformok, Április 10., 7. elméleti óra

Kombinációs modellek

Példa: osztályozási fa 25 féleképpen lehet megoldani, hiba: /epsilon = 0.35.

1. Azt hisszük el amit a többségük mond: de ez nem nagyon javít a döntésben. Ha függenek egymástól az eredmények akkor a közös eredmény is e = 0.35.

Mi van ha függetlenek egymástól?

- Hibást esetek: i = 13
- $e = \sum_{25}^{i=13} 25alattie^i(1-e)^{(25-i)} = 0.06$

(1 - epszilon) / e függvény

Feltételezések

- F1: Véletlennél jobb model: epsilon < 0.5
- F2: Független modelleket szeretnénk, de ez gyakorlatilag kb lehetetlen. Legalább legyenek korrelálatlanak.

Weak learner:

1. Éppen, hogy jobb legyen a véletlennél
2. Legyen gyors (mivel ezekből szeretnénk minél többet lefuttatni)
3. A konfidencia~valószínüség

- SVM, artificial neural network: ezek nem valószínüségeket adnak ki

4. Instabilnak érdemes lennie: a bemenet kicsi változására képes legyen jelentősen megváltoztatnia a kimeneteit.

- KNN, Logisztikus regresszió túl robusztusak ehhez
- Döntési fa ilyen, nagyon gyorsan tudj
    - pl 1 mélységű döntési fa: nem mond meg semmit, de rengeteg ilyen remekül kombinálható

Folyamat

D adathalaz, amiből létrehozunk T darab D_{1..T} adathalmazt C_{1..T} célváltozóval. A kérdés, hogy hogyan kapjuk meg a közös C^* osztályozást.

1. Tanító halmaz manipulációk: Különböző adatokat néznek.
    - Bagging:
    - Boosting:
2. Bemenet attribútomait manipuláljuk: Különböző attribútomokra végezünk különböző méréseket.
    - Random forest algoritmus
3. Osztálycimkék manipulálása
    - Sokosztály -> bináris osztályozás
    - Véletlenszerűen szétválasztjuk a csoportokat
4. Algoritums manipulálása

Bagging

Megfigyelés: ahogy növeljük az adatmintát, növekszik a pontosság. Itt inkább átstrukturáljuk az adathalmazt.

Bootstrapping:

- Minden sornak 1/N valószínüséget adunk
- Visszatevéses mintavétellel egy új D_i-t hozunk létre

Elég nagy elemnagyság után ~63% az esélye egy sornak, hogy bekerül a mintába.

Döntési eljárások

- Többség
- Konfidencia átlagolása
- Súlyozott átlag, ahol a súly a model pontossága
- C*-t egy új model csinálja meg a kimenetek alapján

Példa:
Random Forest használ ilyet

1. D
2. D_i, 1...T
3. C_i gyártás: 1...T
4. C* conf átlag

Random forest light:

- sima decision tree: viszonylag egyértelműen elvágja
- random forest: nem pontos, véletlenszerű vágásokból végez rengeteget és ezzel jól elsimítja az átmeneteket

DE: a random forest egy vágási pontra koncentrál és ha sűrűség nem egy ponton változik, akkor ezt nem fogja jól megmutatni.

- Random foresttel egy megoldás lehet, hogy kivesszük a túl erős oszlopot

Boosting

Azonosítja egy model hibáját és a következő fodulóban arra koncentrál.

"Egyszerű" regressziós boosting

(nem létezik, csak példa): $D_1 -> C_1$

- $r_i = Y_i - C_1(x_i) \in [-1, 1]$
- $D_2\{x_i, r_i\}_1^N : C_1(x_i) + C_2(x_2)$

Könnyű implementálni, de nagyon béna eredményeket hoz, mivel nagyon nehéz megtanulni a hibákat. Folyamatosan túllő és alul lő.

'Súlyozós' boosting

Mindegyik sornak adunk egy súlyt, hogy mennyire fontos {x_i, y_i, w_i}, eleinte W_i = 1/N
Az első modell után megváltoztatja a súlyokat. A hibás találatokat nagyobb súllyal teszi be.

Súlyozási technikák

1. Vannak algoritmusok, amelyekben ez jól van impementálva: KNN, döntési fák. Logisztikus regressziónál nem annyira triviális.
2. Súly alapú mintavétel

Súlyok kezelése:

- AdaBoost
- LogitBoost
- Gradient Boosting Machine (GBM): deep learningen kívüli világon belül ez hozza jelenleg a legjobb eredményeket.
    - pythonban nem a legjobb a default csomag: XGBoost