Elosztott adatbányászat kötelező program

1. Feladatkiírás

A feladat egy elosztott környezetben működő, szabadon választott online tanuló megvalósítása a GoLF keretrendszerben. A megvalósítás és a kiértékelést egy általunk definiált Peersim szimulációban zajlik a GoLF keretrendszer használatával.

2. Fejlesztés menete

A Peersim szimulációs környezet és a GoLF keretrendszer a 2012. április 29-i előadáson lett részletesen bemutatva. Az alábbiakban az előadásra épülő rövid összefoglalót adunk, ami a részletes magyarázat nélkül mutatja be, hogy hogyan használható a rendszer[1].

Töltsük le a gihub.com-ról a GoLF keretrendszert (zip vagy tar.gz) majd csomagoljuk ki egy könyvtárba.

A letöltési URL:
https://github.com/RobertOrmandi/Gossip-Learning-Framework/zipball/student

A csomag magában foglalja az összes használt library-t (így a Peersim-et is), valamint a kurzus számára dedikált kiértékelő szimuláció konfigurációját és az adatbázisokat (tanuló és kiértékelő bontásban)[2]

A forrás letöltése után le kell azt fordítani. Ehhez a projekt könyvtárában ki kell adni a következő parancsot:

```
ant
```

aminek hatására elkészül a bin/gossiplearning.jar állomány.

Implementáljuk a választott tanulót a GoLF API-ban található Model interface megvalósításával.

[1] A mutatott parancsok minden esetben linux parancsok. Windows operációs rendszer esetén a Cygwin környezet használata ajánlott. Beírásuk sortörés nélkül terminál ablakba történik. Feltételezzük, hogy Java, ant és gnuplot telepített.

[2] Ha a letöltést a fent leírtak helyet git használatával szeretnénk megvalósítani, figyeljünk arra, hogy a student branch-et használjuk!

Az implementáció során figyelni kell arra, hogy:

- A model interface clone() metódusát úgy valósítsuk meg, hogy az deep copy-t készítsen az objektumról! Ez azért fontos, mert—a GoLF ajánlás szerint—a modellek másolatai kerülnek küldésre a szimulációban, amit rendszer a clone hívásával valósít meg. Ennek megfelelően ha a clone megvalósítása nem teljes másolatot készít az objektumról, akkor becsaphatjuk magunkat a szimuláció során!

- Az init(String) metódus használatával lehetőség van a konfigurációs állományból paraméterek olvasására. Erről bővebb információt a Peersim dokumentációjában (itt) találunk, példát pedig a GoLF keretrendeszer saját tanulóiban.

- Az update(instance, label) metódus felel az online tanításért. Ehhez a paraméterül kapott tanító példát használhatja (instance - jellemzővektor, label - címke $(0, 1, 2, \ldots, getNumberOfClasses() - 1)$)

- A predict(instance) metódus valósítja meg a predikciót, a paraméterül kapott példára kell egy osztálycímkét visszaadni a modell aktuális állapota alapján.

Helyezzük el a projekt build könyvtárába az elkészült class file-t (célszerű úgy beállítani a fejlesztőkörnyezetet, hogy rögtön oda fordítson).

Az implementációnk teszteléséhez adjuk ki a következő parancsot:

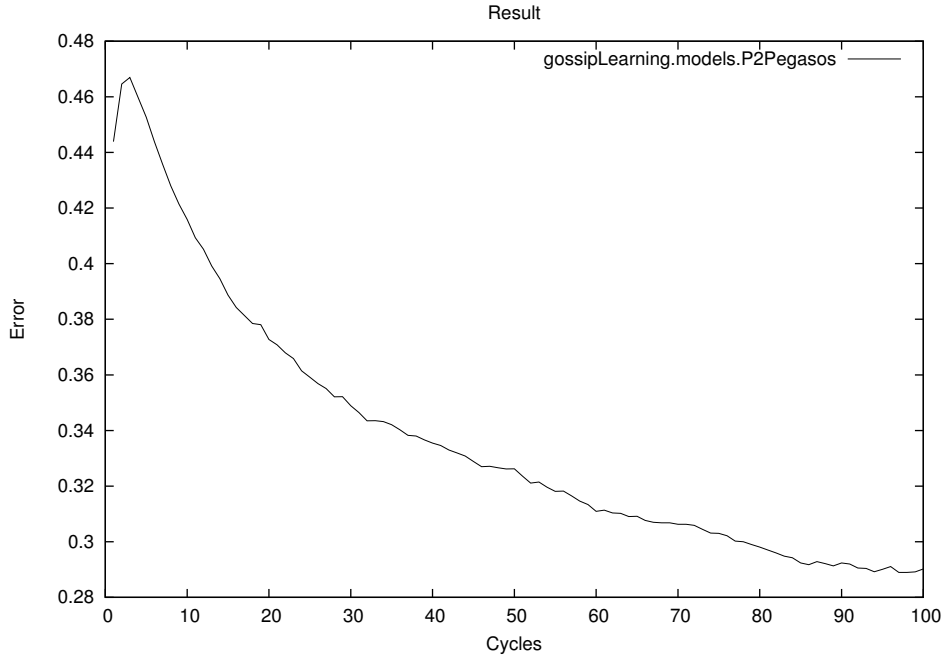
```
../res/script/runAlgorithm.sh className DDMSimulation.txt

    spambase_train.dat spambase_test.dat
```

ahol a ”className„ tartalmazza az általunk megírt osztály nevét (csomagokkal együtt). A fenti parancs a bin könyvtárban lett kiadva. Ha ettől eltérő helyen futtatjuk, akkor értelem szerűen vátoztatni kell a szimuláció konfigurációs file-jának, a tanító és a kiétékelő adatbázis file-jainek elérési útvonalát!

Ez a futás az idő függvényében kirajzolja az algoritmus hiba görbéjét az output.png állományba (szövegesen a képernyőn valamint az output.txt-ben megjelenő adatokból).

Amíg nem rendelkezünk saját implementációval a keretrendszerben megtalálható P2Pegasos algoritmussal is futtathatunk—csak hogy megismerkedjünk a keretrendszerrel—ehhez az alábbi parancsot kell kiadni (szintén a bin könyvtárban)



1. ábra. Példa kimenet a GoLF csomagban található P2Pegasos algoritmus futtattásával. Az ábrán az idő függvényében látható a szimulált P2P hálózat csomópontjai által vétett átlagos predikciós hiba.

```
../res/script/runAlgorithm.sh gossipLearning.models.P2Pegasos

    DDMSimulation.txt spambase_train.dat spambase_test.dat
```

szintén a bin könyvtárból hívva. Ez a hívás az 1 ábrán látható kimenetet generálta.

A fenti példa implementációja is természetesen része csomagnak, itt tekinthető meg.

3. Ajánlott algoritmusok

Az alábbi lista néhány cikket közöl az online tanuló algoritmusokról.

- On-Line Algorithms in Machine Learning [1]
- A New Perceptron Algorithm for Sequence Labeling with Non-local Features [3]

- Online Passive-Aggressive Algorithms [2]

A következőkben néhány online tanulót ajánlunk, amik közzül lehet választani megoldandó feladatot. A lista nem kizárólagos, szóval ha valakinek más ötlete van—e-mail-ben történő egyeztetés után—azon is dolgozhat.

- Winnow
- Perceptron (lineáris döntési függvénnyel)
- Perceptron (bináris döntési függvénnyel)
- Perceptron (szigmoid döntési függvénnyel)
- MostFreq
- Logisztikus regresszió
- Neurálisháló

4. Adminisztratív információk

A programot 2012. június 30-ig kell elkészíteni és bemutatni. Ph.D. hallgatók esetén az index-be való beírás utolsó napjáig tart a határidő.

A kötelező programmal kapcsolatos kérdések:

`ihegedus@inf.u-szeged.hu`
`ormandi@inf.u-szeged.hu`

Jó munkát!

Hivatkozások

[1] Avrim Blum. On-line algorithms in machine learning. In *Developments from a June 1996 seminar on Online algorithms: the state of the art*, pages 306–325, London, UK, UK, 1998. Springer-Verlag.

[2] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, December 2006.

[3] Jun'ichi Kazama and Kentaro Torisawa. A new perceptron algorithm for sequence labeling with non-local features. In *In Proceedings of EMNLP*, 2007.