Luokittelijan kouluttaminen

Näiden tehtävien tarkoituksena on opettaa luokittelijan kouluttaminen “mustana laatikkona” Pythonin scikit-learn -kirjaston avulla.

1. Tutkitaan scikit-learn -kirjaston mukana tulevaa Iris-datajoukkoa. Kyseinen data sisältää mittauksia kolmesta eri kasvilajista, jotka ovat läheistä sukua toisilleen.Tavoitteena on kouluttaa luokittelija ennustamaan, mihin lajiin jokin kasvi kuuluu.

   Tallenna data johonkin muuttujaan:
   ```
   from sklearn.datasets import load_iris
   iris = load_iris()
   ```
   Tarkempaa tietoa kyseisestä metodista:
   http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html

   Datan kuvaus:
   ```
   print iris['DESCR']
   ```

   Mitä *iris.data* ja *iris.target* ovat? Millaisilla piirteillä kasveja on kuvattu?

2. Jaa data kahteen yhtä suureen osaan, jotta voimme käyttää ensimmäistä osaa luokittelijan kouluttamiseen ja toista ennusteiden arvioimiseen. Vinkki: tähän löytyy valmis apufunktio
   http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.train_test_split.html

3. Luo uusi LinearSVC-olio (lineaarinen tukivektoriluokittelija,) opetusparametreilla:
   http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
   Kouluta kyseinen luokittelija.

4. Tee luokittelijalla ennustus jollekin testidatasi esimerkille. Ennustiko luokittelija oikean luokan? Mikäli kyllä, yritä muuttaa piirteiden arvoja niin, että luokittelija tekee väärän ennustuksen.

5. Tee nyt ennustukset kaikille testidatasi esimerkeille ja evaluoi luokittelijan suorituskykyä koko tässä datassa. Apuja: http://scikit-learn.org/stable/modules/model_evaluation.html

6. Tarkastellaan nyt toista datajoukkoa, joka sisältää vanhoja newsgroup-kommentteja. Luokittelijan on tarkoitus oppia, mihin kategoriaan kommentit kuuluvat (esim. moottoripyöräily, jääkiekko, elektroniikka). Kategoriat ovat entuudestaan määriteltyjä, joten kyseessä on tavallinen luokittelutehtävä. Data on valmiiksi jaettu koulutus- ja testijoukkoihin:

```
from sklearn.datasets import fetch_20newsgroups
newsgroups_train = fetch_20newsgroups(subset='train',
remove=('headers', 'footers', 'quotes'))
newsgroups_test = fetch_20newsgroups(subset='test',
remove=('headers', 'footers', 'quotes'))
```

Nyt data onkin tekstiä (en suosittele printtaamaan koko datamäärää) ja se on ensin muutettava piirremuotoon.

7. Muutetaan kommentit TFIDF-vektoreiksi. TFIDF-vektorit ovat yksinkertainen sanojen lukumääriin perustuva tapa esittää tekstidokumentit vektoreina. TFIDF-vektoreiden tarkempi toteutus käydään läpi tiedonhaku-luennolla.

   ```
   from sklearn.feature_extraction.text import TfidfVectorizer
   vectorizer = TfidfVectorizer(norm=None)
   ```

   Vectorizer on myös “sovitettava” dataan.

   ```
   vectorizer.fit(newsgroups_train.data)
   ```

   Nyt voimme muuttaa dokumentit numeerisiksi vektoreiksi.

   ```
   x_train = vectorizer.transform(newsgroups_train.data)
   x_test = vectorizer.transform(newsgroups_test.data)
   ```

   Montako ulottuvuutta piirreavaruudessasi nyt on?
8. Kouluta luokittelija oletusparametreilla kuten aiemminkin ja evaluoi sen suorituskyky testidatalla.

9. Luo kaksi uutta luokittelijaa, toinen pienemmällä ja toinen suuremmalla C-arvolla, kuin alkuperäinen (oletuksena C=1). Miten näiden suorituskyky poikkeaa aiemmasta? Miksi?

Huom! Vaikka näissä tehtävissä käytetään vain kahta datajoukkoa (train ja test), todellisuudessa tarvitsemme kolme, kuten luennoilla kävi ilmi (train, devel ja test). Jos valitsemme C-parametrin siten, että maksimoimme suorituskyvyn devel-datalle, saatamme vahingossa ylisovittaa luokittelijamme siihen. Kolmannen datajoukon on tarkoitus osoittaa, ettei näin ole käynyt (suorituskyvyn pitäisi olla samaa tasoa).