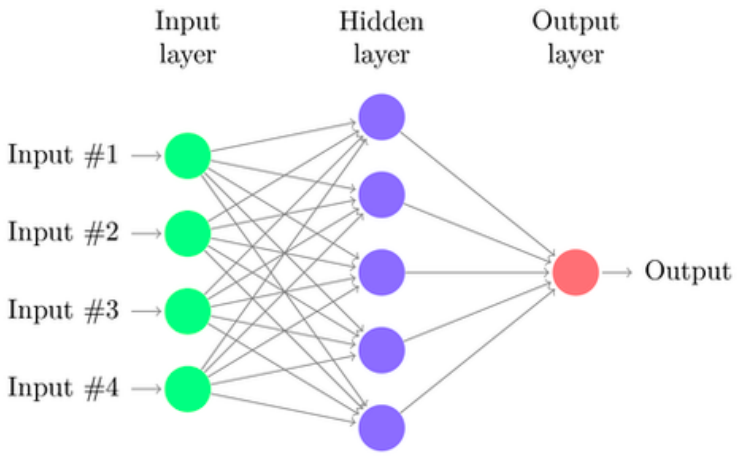
Diseinua

Ieltzu Irazu, Mikel de Velasco, Jorge Nieto

29 de enero de 2010



1 Azaleko Orria / Aurkibidea

2 Aurkezpena

Dokumentu honen bidez, Multilayer Perceptron eta Naive Bayes eredu igarleak konparatzen ditu.

3 Klaseen diseinua

Proiektua hiru atal nagusitan banandu dugu. Lehenengoan, datuak aurreprozezatzen dira, bigarrenean, eredua aterako da eta azkenik instantzia berriak ebaluatzeko jarriko da.

1. **Aurreprozezamendua**
   Atal honetan, bereiziki "R-weka.filters.unsupervised.attribute.Obfuscate" relation-a duten arff fitxategiekin (praktika honetan lan egin behar zirenak) lan egingo dugu. Horregatik behean aipatzen diren filtroak praktika honekin lan egiteko daude.
   Aurre prozezamenduan hurrengo klase eta metodoak erabiliko dira.

   (a) **Aurreprozezadorea:**
   Klase hau, Preprocess.jar exekutagarriak egikarituko duen lehenengo klasea izango da, beraz klase nagusia izango da.
   Bere metodoa hurrengoa da:

   - **main**
     "main" metodoan, argumentu bezala konsolatik train.arff eta dev.arff fitxategiak pasatuko dizkiogu, eta nahi izanez gero (instantzia asko badira) 0-100 arteko zenbaki bat pasatu ahal izango diogu, zenbaki honek instantzia kopuru gutxiagorekin lan egiteko izango da (adb: 70 pasatzen badiogu instantzien %70-rekin lan egingo dugu).
     Klase honetan metodo bakarra egotea erabaki dugu. Wekak dituen liburutegiak filtroen klseak dituztelako eta 'main' metodo honetan, filtro guztiak aplikatuko ditugu:

     i. **Discretize:**
     An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes.

     ii. **InterquartileRange:**
     A filter for detecting outliers and extreme values based on interquartile ranges.

     iii. **Randomize:**
     Randomly shuffles the order of instances passed through it. The random number generator is reset with the seed value whenever a new set of instances is passed in.

     iv. **RemoveUseless:**
     This filter removes attributes that do not vary at all or that vary too much. All constant attributes are deleted automatically, along with any that exceed the maximum percentage of variance parameter. The maximum variance test is only applied to nominal attributes.

     v. **RemoveWithValues:**
     Filters instances according to the value of an attribute.

     vi. **AttributeSelection:**
     A supervised attribute filter that can be used to select attributes. It is very flexible and allows various search and evaluation methods to be combined.

     'Discretize', 'Radomize' eta 'RemoveUseless' "multifilter" batean sartzen dira batera egiteko. 'InterquartileRange' aplikatu ostean bi atributu sortzen ditu, atributu hoiekin 'outlier' instantziak eta 'extreme values' kenduko dira 'RemoveWithValues' eta 'Remove' metodoekin.

Filtro guztiak aplikatu hondoren sortutako instantzia berriak fitxategi batera pasatzen dira, geroago berrerabiltzeko.

2. **Modeloa egitea**
   Atal honetan, entrenamendu multzoak jasoko dira (trainaurre.arff eta devaurre.arff), 'Baseline'-aren ('OneR') eta 'MultilayerPerceptron' aldagaien ekorketa egingo da, geroago bien inferentzia (ez zintzoa, HoldOut (70 / 30), HoldOut (train eta dev) eta 10FoldCrossValidation), azkenik bien modeloak gordeko dira fitxategi batean.
   Modeloak hurrengo klase eta metodoak erabiliko ditu.

   (a) **Modeloa:**
   Klase hau, GetModel.jar exekutagarriak egikarituko duen lehenengo klasea izango da, beraz klase nagusia izango da.
   Bere metodoa hurrengoa da:

   - **main**
     "main" metodoan, argumentu bezala konsolatik trainaurre.arff eta devaurre.arff fitxategiak pasatuko dizkiogu. Fitxategi hauek pasatu behar zaie orden horretan. Horren ondoren (trainaurre instantzien artean klase minoritarioa aterako da. Hori egin eta gero, trainaurre eta devaurre instantziak trainetadev instantzia multzo bakarrean batuko dira. Eta beste aldaera batzuk kalkulatuko dira:
     - trainetadev70 multzoa: trainetadev multzoaren instantzien %70 barruan dituelarik.
     - trainetadev30 multzoa: trainetadev multzoaren instantzien %30 barruan dituelarik.

     Orain 'Baseline' eta 'MultilayerPerceptron' modeloekin hainbat ekintza egingo ditugu:

     i. Parametro ekorketa:
        Klase minoritarioarekiko 'weigthedMeasure'.
        - Baseline:
          * MinBucketSize
          * doNotCheckCapabilities
        - multilayerPerceptron:
          * hiddenLayers
          * training Time

     ii. Bilaketa ez-exhaustiboa

     iii. Inferentzia
        A. Ez-Zintzoa
        B. HoldOut 70-30
        C. HoldOut train eta dev
        D. 10FoldCrossValidation

        Metodo bakoitzarekin fitxategi bat idatziko da **Idazlea** klasearekin eta **fitxategiaEginOneR** eta **fitxategiaEginMultilayerPerceptron** metodoekin hurrenez hurren (evaluationBaseline.txt eta evaluationMultilayerPerceptron.txt).

     Hau guztia egin eta gero modelak idatziko dira **Idazlea** klasearekin eta **modeloaIdatzi** metodoarekin.

   - **minorityClassIndex:**
     Metodo honek instantzia multzo bat jasotzen du eta klase minoritarioaren indezea ateratzen du.

3. **Sailkatzea**
   Sailkatu behar diren instantzia multzo bat jasoko da eta fitxategi berri batean sailkatuko ditu.

(a) **Sailkatzailea**
Klase hau, Classify.jar exekutagarriak egikarituko duen lehenengo klasea izango da, beraz klase nagusia izango da.
Bere metodoa hurrengoa da:

- **main:**
  modeloaPath, testPath eta resultPath jasotzen dira eta **Irakutzailea** klaseak daukan **modeloaKargatu** eta **instantziakIrakurri** metodoekin modeloa eta instantziak kargatuko dira.
  Horren oztean, instantzia guztiak klasifikatuko dira eta emaitza **Idazle** klasearekin eta **idatziInstantzia** metodoarekin resultPath helbidean gordeko da.