Index

Objectius

En aquest PAC es practicaran els conceptes del temari relacionat amb recomanació i clustering, en una vertent clarament pràctica enmarcada en l'ús de llibreries Python.

Descripció de la PAC

En aquesta PAC estudiarem en profunditat l'ús de la llibreria Surprise (http://www.surpriselib.com) com a eina en Python que facilita l'implementació de recomanadors basats en diferents tècniques d'agrupació (clustering).
Farem servir la coneguda base (https://grouplens.org/datasets/movielens/) de dades MovieLens-100k.

Important: Com farem un ús intensiu d'aquesta llibreria, aquesta PAC requereix implícitament un estudi i consulta actius de l'apartat "Documentation" de la pàgina web anteriorment esmentada.

Per a aquesta PAC, se us facilita un fitxer zip amb dos fitxers Python. Aquests dos fitxers hauran de ser completats per l'alumen omplint els diversos forats que s'hi indiquen amb el codi que correspongui.

NO MODIFIQUEU cap nom de funció, variable, ni tampoc dels propis fitxers. La PAC s'ha de completar tot fent servir la base que es proporciona, que en moltes ocasions pot guiar-vos en la manera com dur a terme els exercicis proposats.

Com podeu veure, al fitxer pac1.py tenim diverses definicions de funcions incompletes i un programa molt simple que serà executat (al final del fitxer). Al fitxer IAA_predictors.py tenim, de manera anàloga, la definició d'unes classes
sobre les que treballarem.

Exercici 1 (2 punts)

a) (1 punt) Completeu la definició del mètode predict. Aquest mètode rep un dataset prèviament carregat, un identificador d'usuari i un identificador d'ítem, i la seva finalitat és preparar una sèrie d'algorismes de predicció (basats en diferents tècniques d'agrupació) per tal de dur a terme una predicció de la puntuació (rating) amb què l'usuari uid valorarà l'ítem iid. Com podeu veure al codi, aquest mètode realitzarà aquesta predicció partint dels algorismes SVD, KNNBasic, KNNWithMeans i NormalPredictor. Un cop definit el mètode, dueu a terme les prediccions que aquests algorismes faran de la valoració de l'usuari 777 de l'ítem 333.

Dins el bucle for del mètode predict(), per executar el mateix per a cadascun dels algoritmes de la variable algorithms inicialitzo l'algoritme, després entreno l'algoritme amb el trainset definit i finalment amb el mètode predict() genera la predicció, la qual es retorna s'imprimeix per pantalla.

Converteixo en string l'usuari 777 i l'ítem 333 abans d'utilitzar-los per cridar el mètode predict().

Codi al fitxer pac1.py, mètode predict():

```
def predict(data, uid, iid):

    trainset = data.build_full_trainset()
    # Set the algorithms to predict
    algorithms = (SVD, KNNBasic, KNNWithMeans, NormalPredictor)


    for a in algorithms:

        ########################
        # Init the algorithm and train it with trainset (fit it)
        # Store your prediction in a variable called "pred"
        algo = a()
        algo.fit(trainset)
        pred= algo.predict(uid,iid, r_ui=None, verbose=True)

       ########################

        print(pred)
```

>>predict(data,str(uid),str(iid))

```
####################################
#############Exercise 1##############
####################################
user: 777        item: 333        r_ui = None   est = 3.98   {'was_impossible': False}
user: 777        item: 333        r_ui = None   est = 3.98   {'was_impossible': False}
Computing the msd similarity matrix...
Done computing similarity matrix.
user: 777        item: 333        r_ui = None   est = 3.83   {'actual_k': 40, 'was_impossible': False}
user: 777        item: 333        r_ui = None   est = 3.83   {'actual_k': 40, 'was_impossible': False}
Computing the msd similarity matrix...
Done computing similarity matrix.
user: 777        item: 333        r_ui = None   est = 4.31   {'actual_k': 40, 'was_impossible': False}
user: 777        item: 333        r_ui = None   est = 4.31   {'actual_k': 40, 'was_impossible': False}
user: 777        item: 333        r_ui = None   est = 2.45   {'was_impossible': False}
user: 777        item: 333        r_ui = None   est = 2.45   {'was_impossible': False}
```

b) (1 punt) Consulteu la documentació i expliqueu com proporcionen la seva predicció els algorismes KNNWithMeans i NormalPredictor.

L'algorisme NormalPredictor calcula uns valor aleatoris basats en una distribució normal (gaussiana), la qual s'estima a partir d'un conjunt de dades d'entrenament utilitzant la estimació per màxima verosimilitat. Aquest mètode calcula el valor que maximitza la probabilitat d'obtenir el valor del conjunt d'entrenament.

L'algorisme KNNWithMeans per predir la puntuació que un usuari donara a un ítem, es cerquen els k usuaris que han puntuat ítems més similars a l'usuari del qual volem donar la predicció. A partir de les puntuacions dels k usuaris i similituds amb l'ítem a predir, es donara una puntuació a l'ítem. Els valors més propers s'obtenen a partir d'una funció de similitud com distància euclidiana, correlació de Pearson, MVDM o Hamming.

Exercici 2 (2 punts)

a) (1 punt) Completeu la definició del mètode multiple_cv. Aquest mètode rep un dataset prèviament carregat, i el nombre de vies (o folds, per defecte 5) per tal de dur a terme una valoració creuada dels mateixos algorismes fets servir a l'Exercici 1. Expliqueu les característiques i els avantatges de la validació creuada i justifiqueu per què és important considerar diverses particions del trainset per validar la bondat del nostre sistema.

Característiques i avantatges de la validació creuada i perquè es importants considerar particions del trainset per validar.

La validació creuada es un sistema d'avaluació d'un sistema d'aprenentatge automàtic, que permet mesurar la qualitat dels resultats de l'algoritme utilitzat. Es divideix el dataset en k particions, per tal de calcular la mitjana i desviació de cadascuna de les k particions. Això permet comparar per tal d'estimar el grau de fiabilitat del resultat i millorar-los a partir de modificacions en els mètodes i els seus paràmetres. Els mètodes que hem utilitzat son RMSE (error mitjà quadrat) i MAE (error mitjà absolut), que calculen l'error i quan més petit el resultat es més fiable.

b) (1 punt) Identifiqueu el mètode que dona pitjors resultats i raoneu per què creieu que aquest mètode no és adient. En quines condicions seria més raonable utilitzar-ho com a predictor?

El mètode amb pitjors resultats s'identifica per el valor màxim de la mitjana del mètode d'avaluació, on NormalPredictor ha obtingut una mitjana d'error superior a la resta, amb valors RMSE=1.5203 i MAE=1.2211 respecte a la segona pitjor que es KNNBasic amb RMSE=0.9784 i MAE=0.7728.

Aquest mètode no es adient en casos on sigui requisit una alta fiabilitat com a l'àmbit de la medicina, on les prediccions respecte a malalties pot impactar en la salut de les persones.

Els casos on el mètode NormalPredictor, degut a que necessita menys poder computacional, es més raonable utilitzar-ho en aplicacions on es requereix prediccions en temps real i sigui suficient una predicció correcte.

Codi al fitxer pac1.py, mètode predict():

```
def multiple_cv(data, folds=5):

    # Set the algorithms to cross validate
    algorithms = (SVD, KNNBasic, KNNWithMeans, NormalPredictor)
    measures = ['RMSE', 'MAE']


    for a in algorithms:
        ########################
        # Init the algorithm and perform the cross validation
        algo = a()
        cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True, n_jobs=1)
        ########################
```

```
>>multiple_cv(data,folds=5)
```

```
####################################
#############Exercise 2##############
####################################
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9279  0.9297  0.9410  0.9342  0.9409  0.9347  0.0055
MAE (testset)     0.7311  0.7354  0.7377  0.7380  0.7407  0.7366  0.0032
Fit time          4.61    4.64    4.68    4.74    4.84    4.70    0.08
Test time         0.18    0.13    0.12    0.13    0.13    0.14    0.02
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating RMSE, MAE of algorithm KNNBasic on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9823  0.9736  0.9794  0.9803  0.9767  0.9784  0.0030
MAE (testset)     0.7755  0.7672  0.7739  0.7738  0.7734  0.7728  0.0029
Fit time          0.47    0.52    0.49    0.47    0.45    0.48    0.02
Test time         2.91    2.92    2.86    2.84    2.87    2.88    0.03
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating RMSE, MAE of algorithm KNNWithMeans on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9528  0.9486  0.9484  0.9517  0.9506  0.9504  0.0017
MAE (testset)     0.7493  0.7494  0.7489  0.7485  0.7501  0.7493  0.0005
Fit time          0.47    0.50    0.49    0.50    0.49    0.49    0.01
Test time         3.03    2.95    3.03    3.00    2.98    3.00    0.03
Evaluating RMSE, MAE of algorithm NormalPredictor on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.5230  1.5176  1.5153  1.5180  1.5278  1.5203  0.0045
MAE (testset)     1.2227  1.2172  1.2172  1.2217  1.2267  1.2211  0.0036
Fit time          0.09    0.11    0.11    0.11    0.11    0.11    0.01
Test time         0.13    0.16    0.16    0.16    0.11    0.15    0.02
```

Exercici 3 (2 punts)

Com ja heu observat, les valoracions registrades a MovieLens corresponen a valors discrets en l'interval [1,5], segurament obtinguts després que l'usuari valorés una determinada pel·lícula amb una quantitat d'estrelles.

Imaginem que treballem com a enginyers al portal web d'on s'han extret aquestes puntuacions, i des del departament d'user experience ens comuniquen que han realitzat un estudi que suggereix que hem de canviar aquesta manera de valorar les pel·lícules, de manera que ara, en comptes dels valors de l'1 al 5, seran valorades escollint una de les següents opcions:

["Molt dolenta", "Pobre", "No està malament", "Entretinguda", "Recomanable", "Molt bona", "Brillant"]

Quines consideracions hem de tenir en compte per a poder redissenyar el sistema de recomanació? Ens servirà el conjunt de dades d'entrenament que ja teníem? En cas d'haver d'introduir modificacions a la implementació del sistema, indiqueu-les, tot justificant cadascuna de les decisions que prendríeu.

Quines implicacions tindria el fet que al cap de 2 mesos ens indiquessin que hem d'afegir l'opció "Decepcionant" entre "Pobre" i "No està malament"?

Important: per realitzar aquest exercici no cal implementar cap programa, només un raonament justificat i complet.

Son necessàries les següents consideracions per poder redissenyat el sistema de recomanació:

- Les valoracions actuals son de tipus enter i en la nova son cadenes de caràcters (string).
- Augment de 5 a 6 possibles valors en les valoracions.

El conjunt de dades d'entrenament anterior ens servirà sempre que prèviament es normalitzin els valors a la nova escala utilitzada i s'ajuntin les dades.

Les modificacions a implementar al sistema serien les següents:

- Per a tot el conjunt de dades amb el nou tipus de valoracions, prèviament a entrenar-ho transformaria el tipus de les seves valoracions a un valor discret, per facilitar aprofitar l'actual programa que utilitza valors discrets.
- Un cop tenim el conjunt de dades antic i nou amb valors discrets, tant entrenament com test, normalitzaria les dades perquè tinguessin la mateixa escala de valors. D'aquesta forma l'antic conjunt es podrà incorporar al nou per tal de no perdre l'històric de valoracions i el sistema doni recomanacions més fiables que si ens basem només en el nou conjunt.

Si després de 2 mesos s'han d'afegir una opció entre Pobre i No està malament, significa que s'haurien tornar a normalitzar els conjunts de dades anteriors, per tal que s'adeqüessin al nou rang de valoracions i així utilitzar-los per entrenar el model.

["Molt dolenta", "Pobre", ***«Decepcionant»,*** "No està malament", "Entretinguda", "Recomanable", "Molt bona", "Brillant"]

Exercici 4 (4 punts)

Estudieu detingudament com crear un algorisme de predicció propi

(http://surprise.readthedocs.io/en/stable/building_custom_algo.html) i preneu com a exemple la definició dels algorismes que la llibreria Surprise porta implementats per defecte (per exemple,

https://github.com/NicolasHug/Surprise/blob/master/surprise/prediction_algorithms/knns.py).

En aquest exercici definirem un parell d'algorismes propis. Consulteu el fitxer IAA_predictors.py, on hi ha definida una classe arrel SymmetricAlgo, de la qual heretaran els nostres algorismes, i a continuació tenim definicions incompletes de dos algorismes. En aquest context es demana:

a) (1.5 punts) Completar la definició de la classe Dummy_IAA, de manera que el mètode estimate simplement retorni una valoració a l'atzar d'entre totes les valoracions fetes per l'usuari que es consulta.

Al mètode estimate amb la llista de tuples de self.yr amb els usuaris i valoracions, faig un recorregut per l'usuari u (user_inner_id) per tal de crear una llista 'Listux' amb les valoracions de l'usuari consultat. Només ens resta seleccionar una valoració de la llista 'ListAux' a l'atzar amb el mètode numpy.random.choice.

Codi al fitxer IAA_predictors.py, clase Dummy_IAA, mètode estimate():

>>run_IAAPredictors(data,False, int(uid),int(iid))

```
def estimate(self, u, i):

    if not (self.trainset.knows_user(u) and self.trainset.knows_item(i)):
        raise PredictionImpossible('User and/or item is unkown.')

    x, y = self.switch(u, i)

    est = 0


    ##############################
    ##compute and return estimation
    ##valoració a l'atzar d'entre totes les valoracions fetes per l'usuari que es consulta

    ##Fem un recorregut per self.yr (llista de tuples (user_inner_id, ratings))
    #de l'usuari u i es guarden només les valoracions a ListAux
    #S'agafa a l'atzar una valoració de la llista ListAux.
    ListAux = [(v) for k,v in self.yr[self.trainset.to_inner_uid(str(u))]]
    est = np.random.choice(ListAux)

    return est
```

```
PS D:\dropbox\uoc\iaa\pac1> python pac1.py
##################################
##############Exercise 4##########
##################################
##############Do_cv=False##########
2.0
```

b) (1.5 punts) Completar la definició de la classe KNN_IAA, de manera que a la seva inicialització permeti rebre un paràmetre n, i el mètode estimate faci pràcticament el mateix que el que fa KNNBasic de Surprise, però amb la diferència que dels k_neighbors triem n a l'atzar per calcular el promig ponderat (weighted average). En cas que k_neighbors tingui una mida inferior a n, agafarem el valor de la mida de k_neighbors com a n.

Codi al fitxer IAA_predictors.py, clase KNN_IAA, mètode estimate():

>>run_IAAPredictors(data,False, int(uid),int(iid))

```python
def estimate(self, u, i):

    if not (self.trainset.knows_user(u) and self.trainset.knows_item(i)):
        raise PredictionImpossible('User and/or item is unkown.')

    x, y = self.switch(u, i)

    # Compute similarities between x and x2, where x2 describes all other
    # users that have also rated item Y.
    neighbors = [(self.sim[x, x2], r) for (x2, r) in self.yr[y]]
    k_neighbors = heapq.nlargest(self.k, neighbors, key=lambda t: t[0])

    ###########################################################################

    #Compute weighted average from n random neighbors within k_neighbors
    #First initialize variables
    sum_sim = sum_ratings = actual_k = 0
    kn_length = len(k_neighbors)

    #Obtenim _random_neighbors (n_random_neighbors)
    #Si la mida de k_neigbors es inferior a n, n=kn_length
    if kn_length < self.n:
        n_random_neighbors = self.n
    else:
        #Sino seleccionem un número a l'atzar
        #entre el mínim (self.n)i el màxim de neighbors (kn_length)
        n_random_neighbors = np.random.randint(self.n,kn_length+1)

    #Seleccionem a l'atzar n_random_neihbors de k_neighbors
    k_neighbors_random = random.sample(k_neighbors,n_random_neighbors)

    #Calculem la mitja de k_neighbors_random
    for (sim, r) in k_neighbors_random:
        if sim > 0:
            sum_sim += sim
            sum_ratings += sim * r
            actual_k += 1


    ####################

    if actual_k < self.min_k:
        raise PredictionImpossible('Not enough neighbors.')

    est = sum_ratings / sum_sim

    details = {'actual_k': actual_k}
    return est, details
```

```
PS D:\dropbox\uoc\iaa\pac1> python pac1.py
>>
####################################
##############Exercise 4###############
####################################
##############Do_cv=True###############
##############Do_cv=False##############
Evaluating Dummy_IAA algorithm without cross-validation
2.0
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating KNN_IAA algorithm without cross-validation
(3.8014359950089598, {'actual_k': 34})
```

c) (1 punt) Completeu el mètode run_IAAPredictors de pac1.py, i executeu-ho. Aquest mètode es comportarà de dues maneres diferents, segons el paràmetre do_cv:

• Si és True, realitzarà una validació creuada amb 5 vies (folds).

• Si és False, utilitzarà els paràmetres uid i iid per fer una predicció de valoració de l'ítem iid per part de l'usuari uid.

En ambdós casos, com podeu veure al codi, es faran servir els algorismes Dummy_IAA i KNN_IAA, un darrere l'altre.

Al executar el codi, amb do_cv =False funciona, però amb do_cv =True no.

Adjunto resultat de do_cv=False

```
PS D:\dropbox\uoc\iaa\pac1> python pac1.py
>>
####################################
##############Exercise 4###############
####################################
##############Do_cv=True###############
##############Do_cv=False##############
Evaluating Dummy_IAA algorithm without cross-validation
2.0
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating KNN_IAA algorithm without cross-validation
(3.8014359950089598, {'actual_k': 34})
```