PAC4: Introducció a l'aprenentatge computacional

Presentació

Quarta PAC del curs d'Intel·ligència Artificial

Competències

En aquesta PAC es treballaran les següents competències:

Competències de grau:

- Capacitat d'analitzar un problema amb el nivell d'abstracció adequat a cada situació i aplicar les habilitats i coneixements adquirits per abordar-lo i solucionar-ho.

Competències específiques:

- Conèixer els diferents aspectes d'aprenentatge computacional (aprenentatge supervisat, no supervisat i per reforç).
- Conèixer els fonaments teòrics dels mètodes més representatius.
- Conèixer el tractament dels conjunts de dades per a la correcta validació dels sistemes d'aprenentatge.

Objectius

Aquesta PEC pretén avaluar diferents aspectes d'aprenentatge supervisat i no supervisat.

Descripció de la PAC a realitzar

Arbres de Decisió

Pregunta 1) (3 punts)

Un servidor de correu pretén crear un sistema per detectar correus SPAM que li arriben als seus usuaris a partir de certes característiques: màxima freqüència de repetició de paraules és menor que 2 (freq <2), la longitud de lletres majúscules seguides és més gran que 5 (long> 5), i la màxima freqüència de repetició de paraules més gran que 10 (freq> 10). La següent taula mostra un conjunt d'emails que farem servir com a conjunt d'entrenament per detectar si el correu rebut és SPAM o no.

Clase	freq<2	long>5	freq>10
SPAM	Si	No	No
NO SPAM	No	No	No
SPAM	No	No	Si
SPAM	No	Si	Si
NO SPAM	No	No	No
NO SPAM	No	No	No
SPAM	Si	Si	No
NO SPAM	No	No	No
SPAM	No	No	No
SPAM	No	Si	No
SPAM	No	Si	No
SPAM	Si	Si	No
NO SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No

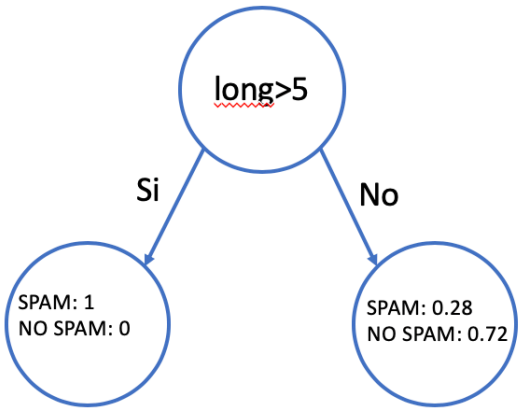
Construir un arbre de decisió a partir d'aquest conjunt calculant la bondat de les particions dels tres atributs binaris.

Solució:

Raiz (Nivell 0)

En el **node arrel** calculem la bondat de les particions dels tres atributs per a tots els elements. L'atribut freq <2 genera dues particions SPAM I NO SPAM on l'atribut No té 8 instàncies amb la classe NO SPAM i 5 instàncies amb SPAM i l'atribut Si només té 3 instàncies amb SPAM. La classe majoritària amb l'atribut No és NO SPAM i amb l'atribut Si és SPAM, i per tant, la bondat de la partició és (8 + 3) /16=0.68765. Fem el mateix procediment per a long> 5 i freq> 10. Per long> 5, l'atribut Si té 5 instàncies amb SPAM mentre que l'atribut No té 8 NO SPAM i 3 SPAM. Agafant les classes majoritàries de cada partició, ens queda (8 + 5) / 16 = 0,8125. Finalment, per freq> 10, No té 8 instàncies amb classe NO SPAM i 6 instàncies amb classe SPAM mentre que l'atribut Si té 2 instàncies de SPAM. Això vol dir que la bondat de freq> 10 és (8 + 2) / 16 = 0.625.

Un cop hem calculat la bondat per a tots els atributs, seleccionem el millor atribut amb la millor bondat, long> 5 amb 0,8125 de bondat, i l'arbre de decisió de nivell 0 és el següent:



Clase	freq<2	long>5	freq>10
SPAM	No	Si	Si
SPAM	Si	Si	No
SPAM	No	Si	No
SPAM	No	Si	No
SPAM	Si	Si	No

Clase	freq<2	long>5	freq>10
SPAM	Si	No	No
NO SPAM	No	No	No
SPAM	No	No	Si
NO SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No
SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No

On la fulla de l'esquerra classifica correctament tots els correus en SPAM, i per tant terminal.

Nivell 1:

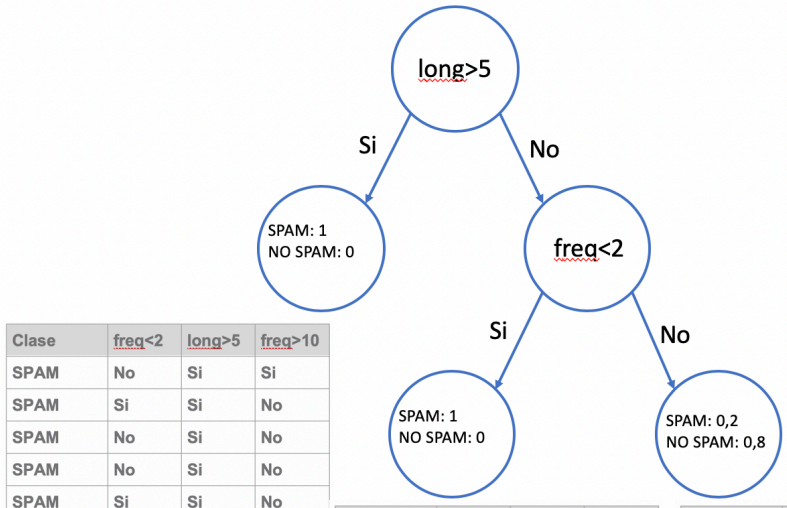
Com el node de l'esquerra és terminal, seguim amb el mateix procediment amb el node de la dreta, calculant la bondat per a cada atribut amb les mostres d'aquest node.

Bondat freq <2: (1 + 8) / 11 = 0,81

Bondat long 5: (0 + 8) / 11 = 0,72

Bondat fred> 10: (1 + 8) / 11 = 0,81

Hem obtingut un empat entre dos dels atributs i podem escollir qualsevol. Triem freq <2 per fer la divisió de les mostres i l'arbre de decisió de nivell 1 és el següent:



Clase	freq<2	long>5	freq>10
SPAM	No	Si	Si
SPAM	Si	Si	No
SPAM	No	Si	No
SPAM	No	Si	No
SPAM	Si	Si	No

Clase	freq<2	long>5	freq>10
SPAM	Si	No	No

Clase	freq<2	long>5	freq>10
NO SPAM	No	No	No
SPAM	No	No	Si
NO SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No
SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No

Nivell 2:

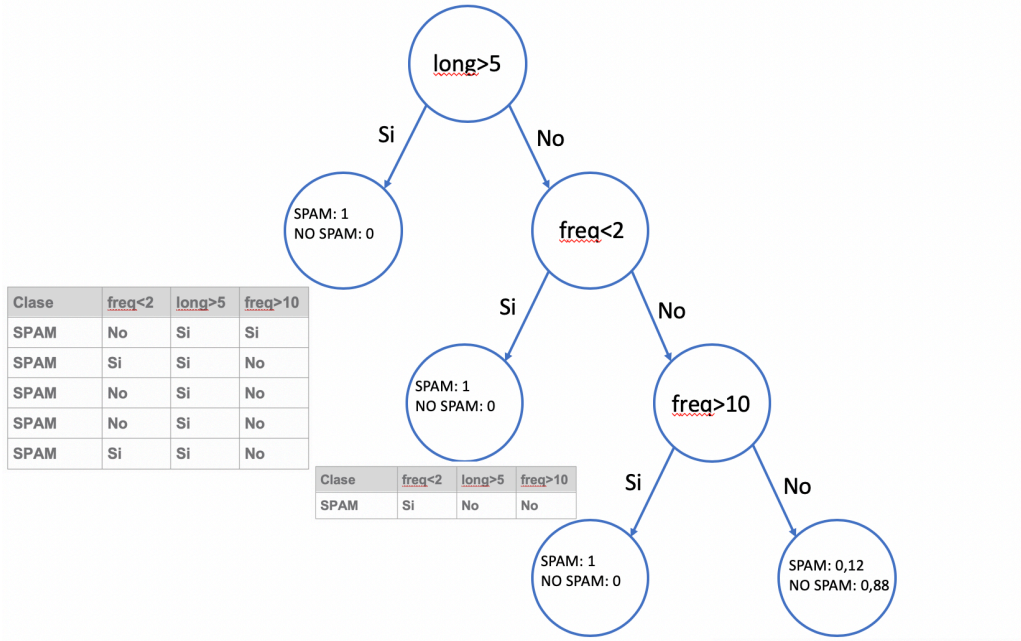
Tal com ha succeït en el nivell 1, el node de la mitjà és terminal perquè és pur completament i seguim calculant la bondat per als atributs de el node de la dreta.

Bondat freq <2: (0 + 8) / 10 = 0,8

Bondat long 5: (0 + 8) / 10 = 0,8

Bondat fred> 10: (1 + 8) / 10 = 0,9

Per tant, agafem la fred> 10 per fer la divisió de el node i obtenir la classificació final dels correus entre SPAM i NO SPAM tal com mostra el següent arbre de decisió amb les probabilitats de cada classe segons el node terminal:



Clase	freq<2	long>5	freq>10
SPAM	No	Si	Si
SPAM	Si	Si	No
SPAM	No	Si	No
SPAM	No	Si	No
SPAM	Si	Si	No

Clase	freq<2	long>5	freq>10
SPAM	Si	No	No

Clase	freq<2	long>5	freq>10
SPAM	No	No	Si

Clase	freq<2	long>5	freq>10
NO SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No
SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No
NO SPAM	No	No	No

Pregunta 2) (2 punt)

En l'exercici anterior s'ha creat un arbre de decisió calculant la bondat de les particions utilitzant els atributs que han estat categoritzats a variables binàries. En moltes aplicacions aquestes dades són ordinaris (nombres enters, reals, etc.). Per exemple, la freqüència d'una paraula en un correu, el nombre mitjà de caràcters per paraula en el cas de detecció de correus SPAM. Considerem els següents dos atributs enters: longitud de majúscules seguides amb rang [1..100] i el nombre màxim de repeticions d'una paraula amb rang [1..20], dues classes: SPAM i NO SPAM, com es calcularia la millor partició de les dades per a la classificació dels correus. (Nota: només s'usa el millor atribut categoritzat, codi Python o pseudo-codi és vàlid).

Solució:

Com hem vist en la solució de l'anterior apartat cal calcular la bondat d'una partició binària, per exemple, freq <2 o freq> 2, que fan la mateixa partició de les dades, però amb valors inversos. Per tant, considerem només el cas de freqüència i

longitud major (>) que cert valor. Per calcular la bondat és necessari el vector de classe SPAM o NO SPAM (y_clase) i un vector binarizado que determina la partició de les dades (x_atributo), per exemple, long> 5. Aquesta funció es pot calcular de la següent manera:

```
def bondad(x_atributo, y_clase):
    n_instancias=len(x_atributo)
    x_0 = np.zeros(2,) # particion igual a 0, con dos clases: 0 o 1
    x_1 = np.zeros(2,) # particion igual a 1, con dos clases: 0 o 1
    for i_idx in range(n_instancias):
        if x_atributo[i_idx]==0:
            y_i = y_clase[i_idx]
            x_0[y_i]+=1
        else:
            y_i = y_clase[i_idx]
            x_1[y_i]+=1
    bondad = (np.max(x_0)+np.max(x_1))/n_instancias
```

Com s'ha calculat anteriorment la bondat en la pregunta 1, la implementació de la bondat és opcional, encara que s'ha de trucar i passar els paràmetres (x_atributos i y_clase).

Un cop tenim definida la funció bondat, s'ha de binarizar cada atribut per a cada valor del seu rang. És a dir, la Binarització consisteix en què tot element major o igual a un valor és igual a 1, i els menors a 0. L'atribut longitud (x_longitud) s'ha d'binarizar per a cada valor entre 1 i 100 mentre que l'atribut freqüència (x_frecuencia) entre 1 i 20. La binarització es pot programar com:

```
def binarizacion(x, thr)
    out = np.zeros(x.shape)
    out[x>=thr]=1
    return out
```

Per calcular la millor partició de les dades però amb atributs sencers, hem de calcular la bondat per a cada partició obtinguda aplicant la Binarització de l'atribut en el rang de l'atribut, i escollir el màxim entre els dos atributs. Aquesta operació es calcula de la següent manera:

```
# y_clase, x_longitud, x_frecuencia se suponen que estan en memoria o pasada por parametros
bondad_longitud = np.zeros(100,)
bonda_frecuencia = np.zeros(20,)

#Se calcula la bondad para todo el rando de longitud
for i_longitud in range(100):
    x_atributo = binarizacion(x_longitud, i_longitud)
    bondad_longitud[i_longitud] = bondad(x_atributo, y_clase)

#Se calcula la bondad para todo el rando de frecuencia
for i_frecuencia in range(20):
    x_atributo = binarizacion(x_frecuencia, i_frecuencia)
    bondad_frecuencia[i_longitud] = bondad(x_atributo, y_clase)

idx_max_longitud_bondad = np.argmax(bondad_longitud)
valor_max_longitud = bondad_longitud[idx_max_longitud_bondad]

idx_max_frecuencia_bondad = np.argmax(bondad_frecuencia)
valor_max_frecuencia = bondad_frecuencia[idx_max_frecuencia_bondad]

print('la particion de los datos se calcula con:')
if valor_max_frecuencia>valor_max_longitud:
    print('frecuencia mayor o igual a '+str(idx_max_frecuencia_bondad))
else:
    print('longitud mayor o igual a '+str(idx_max_longitud_bondad))
```

Pregunta 3) (3 punts)

Utilitzant el notebook de Python que avalua el k nearest neighboor, implementar allò que calgui per a la implementació de l'k-means. (Nota: considerar els clústers com el conjunt de dades de l'kNNs i tots els punts del conjunt de dades com queries).

Solució:

Primer de tot s'ha d'assignar el clúster més proper a cada instància de la base de dades. Això es pot fer usant el knn on el query és la instància actual i la base de dades són tots els centroides, tal com es pot veure a baix.

```
idx2cluster = np.zeros((n_instances,))
for i_instances in range(n_instances):
    idx2cluster[i_instances] = knearestneighbors(data[i_instances,:], clusters, n_neighbors=1)
```

Un cop tenim totes les instàncies assignades als clústers, calculem l'i-th clúster com la mitjana de tots punts assignats al clúster i i així per a tots els clústers.

```
for i_clusters in range(n_clusters):
    clusters[i_clusters,:] = np.mean(data[idx2cluster==i_clusters,:],axis=0)
```

Tota la solució està en el ipython notebook.

Pregunta 4) (2 punts)

Utilitzant el programa Python adjunt, que avalua el kNNs, Arbres de Decisió i xarxes neuronals amb una avaluació simple. Implementar allò que faci falta per poder

avaluar aquests algoritmes utilitzant una validació creuada (cross validation o K-fold cross validation), amb una implementació genèrica per a K i avaluada amb K = 10. A més, calcular la mitjana i la variància de la precisió de cada un dels splits i de cada classificador. (Nota: es recomana usar la classe KFold de sklearn per la seva simplicitat)

Solució:

Primer de tot s'ha de definir la variable amb el nombre de splits K = 10, i generar els splits, i les variables per emmagatzemar la precisió de cada classificador.

```
K=10
kf = KFold(n_splits=K) # Define the split - into 2 folds
# important point to gnerate the splits
#kf.get_n_splits(X)
knn_score = np.zeros((K,))
tree_score = np.zeros((K,))
NN_score = np.zeros((K,))
```

Un cop tenim els splits, iterem pels K splits per entrenar els classificadors i calcular la precisió en l'split de test per calcular la mitjana i la variació estàndard a al final.

```
# K-fold Crossvalidation
from sklearn.model_selection import KFold
K=10
kf = KFold(n_splits=K) # Define the split - into 2 folds
# important point to gnerate the splits
#kf.get_n_splits(X)
knn_score = np.zeros((K,))
tree_score = np.zeros((K,))
NN_score = np.zeros((K,))
for i_iter, (train_index, test_index) in enumerate(kf.split(X)):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    clf_knn.fit(X_train, y_train)
    clf_tree.fit(X_train, y_train)
    clf_NN.fit(X_train, y_train)
    y_pred = clf_knn.predict(X_test)
    print("Fold "+str(i_iter))
    score=accuracy_score(y_test, y_pred)
    knn_score[i_iter]=score
    print("KNN Accuracy:"+str(score))
    y_pred = clf_tree.predict(X_test)
    score=accuracy_score(y_test, y_pred)
    tree_score[i_iter]=score
    print("Decision Tree Accuracy:"+str(score))
    y_pred = clf_NN.predict(X_test)
    score=accuracy_score(y_test, y_pred)
    NN_score[i_iter]=score
    print("NN Accuracy:"+str(score))
print('KNN classifiers accuracy: '+str(np.mean(knn_score))+' +/- '+str(np.std(knn_score)))
print('Tree classifier accuracy '+str(np.mean(tree_score))+' +/- '+str(np.std(tree_score)))
print('NN classifier accuracy: '+str(np.mean(NN_score))+' +/- '+str(np.std(NN_score)))
```

Tota la solució està en el ipython notebook.

Recursos

Per fer aquesta PAC el material és el mòdul 5 i per executar els notebooks en Python de l'exercici va recomanar utilitzar Google Col·lab que és gratuït (https://colab.research.google.com).

Criteris de valoració

Les puntuacions es mostren en cada pregunta de l'enunciat.

Format i data de lliurament

Per a dubtes i aclariments sobre l'enunciat, adreceu-vos al consultor responsable de la vostra aula.

Cal lliurar la solució en un fitxer PDF fent servir una de les plantilles lliurades conjuntament amb aquest enunciat. Adjunteu el fitxer a un missatge a l'apartat Lliurament i Registre d'AC (RAC).

El nom del fitxer ha de ser *CognomsNom*_IA_PAC4 amb l'extensió .pdf (format PDF).

La data límit de lliurament és el: **22/12/2019** (a les 24 hores).

Raoneu la resposta en tots els exercicis. Les respostes sense justificació no rebran puntuació.

Nota: **Propietat intel·lectual**

Sovint és inevitable, en produir una obra multimèdia, fer ús de recursos creats per terceres persones. És per tant comprensible fer-ho en el marc d'una pràctica dels estudis d'Informàtica, sempre i això es documenti clarament i no suposi plagi en la pràctica.

Per tant, en presentar una pràctica que faci ús de recursos aliens, s'ha de presentar juntament amb ella un document en què es detallin tots ells, especificant el nom de cada recurs, el seu autor, el lloc on es va obtenir i el seu estatus legal: si l'obra està protegida pel copyright o s'acull a alguna altra llicència d'ús (Creative Commons, llicència GNU, GPL ...). L'estudiant haurà d'assegurar-se que la llicència que sigui no impedeix específicament seu ús en el marc de la pràctica. En cas de no trobar la informació corresponent haurà d'assumir que l'obra està protegida pel copyright.

Hauran, a més, adjuntar els fitxers originals quan les obres utilitzades siguin digitals, i el seu codi font si correspon.