PAC 2

Solució de la PAC

Exercici 1

Els arxius adjunts contenen dades que es faran servir en posteriors exercicis. En particular, 'car.csv' conté un conjunt gran de dades i tant 'train.csv' com 'test.csv' són uns pocs exemples pertanyents a 'car.csv'.

En aquest exercici heu d'analitzar aquestes dades, decidir si és necessari dur a terme algun tractament previ i si aquest tractament hauria d'esser el mateix per tots tres arxius o no. Abans de prendre aquestes decisions és convenient que llegiu els enunciats de la resta d'exercicis per a saber com s'emprarà cada un dels arxius.

Si és necessari fer el tractament, feu-lo explicant cada una de les decisions preses i mostrau a la documentació les dades de 'train.csv' i 'test.csv' després del tractament. Si no és necessari fer el tractament, justificau molt clarament els motius.

En posteriors exercicis, si es necessita un tractament previ de les dades de 'train.csv', 'test.csv' o 'car.csv' emprareu les dades resultants d'aquest exercici. Si no es necessita tractament previ, heu d'emprar les dades originals.

En primer lloc, descriurem el tractament que s'ha d'aplicar a tots els arxius de la mateixa forma:

- Ja que a cap arxiu hi ha valors absents, no s'han de tractar.
- L'atribut CLASS és la classe i, per tant, no s'ha de tractar. Tot i això, a efectes de representació, substituirem "unacc" per 0 i "acc" per 1.
- L'atribut SAFETY és ordinal amb possibles valors low, med i high. Per a representar-lo adequadament, emprarem codificació additiva de la següent forma: low=00,med=10,high=11.

D'altra banda, tenim que la resta d'atributs són numèrics amb valors dispersos i molt distints entre atributs. En aquest cas aplicarem ranging, tot i que altres opcions com ara estandardització també serien vàlides. Ara bé, considerant el que es demana a la resta d'exercicis, no aplicarem ranging de la mateixa manera a tots ells.

En particular, ja que test.csv conté exemples que es classificaran amb models entrenats amb train.csv, calcularem els màxims i mínims dels atributs segons train.csv i seran aquests màxims i mínims els que emprarem per a fer ranging de test.csv.

Pel que fa a car.csv, ja que és un arxiu únic que s'emprarà per a validacions creuades, hi aplicarem ranging amb els seus propis màxims i mínims.

S'ha de dir que, ja que sabem que test.csv i train.csv s'han extret de car.csv es podria haver pensat en aplicar ranging a tots ells emprant els màxims i mínims de car.csv. Ara bé, és més

correcte fer-ho per separat per tal de restringir el tractament a les dades concretes que s'empraran a cada exercici.

Pel que fa a màxims i mínims emprats per a fer el ranging, tenim el següent:

- Els màxims i mínims de train.csv, també emprats a test.csv són:

	PRICE	MAINT	DOORS	PERSONS
MIN	10000	500	2	2
MAX	40000	2000	5	5

- Els màxims i mínims de car.csv són:

	PRICE	MAINT	DOORS	PERSONS
MIN	10000	500	2	2
MAX	40000	2000	5	5

En aquest cas són els mateixos, però podria no haver estat així. El resultat del tractament descrit per train.csv és:

PRICE	MAINT	DOORS	PERSONS	SAFETY1	SAFETY0	CLASS
0.33	1	0.67	0	1	0	0
1	0.33	0.67	0.67	1	1	1
1	1	0	0	0	0	0
1	0.33	1	0.67	0	0	0
0	0	0.67	0.67	1	0	1
0.67	0.33	0.67	1	1	1	1
0.33	0	0.33	0	0	0	0
0	0.33	0.33	0	1	1	0
0.33	0.33	0	1	1	0	1
0.33	0.33	1	1	1	1	1

I en el cas de test.csv és:

PRICE	MAINT	DOORS	PERSONS	SAFETY1	SAFETY0	CLASS
1	0.34	0	0.67	1	1	1
0.33	0.67	0.33	0	1	0	0

Exercici 2

a) Construïu els models de classificació basats en el veí més proper emprant k=1 i k=3 a partir de l'arxiu "train.csv". És a dir, heu de construir els models 1NN i 3NN. Amb cada un d'aquests dos models heu de classificar els exemples de l'arxiu "test.csv"

Emprarem les dades tractades a l'exercici 1. En primer lloc, calcularem les distàncies de cada exemple de test a cada exemple d'entrenament. Obtenim el següent:

	TEST1	TEST2	CLASS
TRAIN1	1.67	0.47	0
TRAIN2	0.67	1.45	1
TRAIN3	1.70	1.29	0
TRAIN4	1.73	1.56	0

```
TRAIN5      1.60  1.05  1
TRAIN6      0.82  1.53  1
TRAIN7      1.76  1.20  0
TRAIN8      1.25  1.11  0
TRAIN9      1.25  1.11  1
TRAIN10     1.25  1.60  1
```

L'exemple d'entrenament més proper a TEST1 és TRAIN2, el qual és de classe 1. L'exemple d'entrenament més proper a TEST2 és TRAIN1, el qual és de classe 0. Per tant, segons 1NN, TEST1 és de classe 1 i TEST2 és de classe 0. Ja que, segons el ground truth disponible a test.csv, el primer exemple és de classe 1 i el segon de classe 0, tenim una precisió del 100%.

Pel que fa a 3NN, anem a veure els tres exemples més propers de TEST1. Són TRAIN2, TRAIN6 i després TRAIN8, TRAIN9 i TRAIN10 indistintament. Anem a agafar TRAIN8 per ser el primer que apareix a la llista. En aquest cas, les classes dels tres veïns més propers són 1,1,0. La majoria són de classe 1 i, per tant, assignarem la classe 1 a TEST1. Noteu que en aquest cas, seleccionar TRAIN9 o TRAIN10 en comptes de TRAIN8 no hauria suposat cap canvi.

Els tres veïns més propers de TEST2 són TRAIN1, TRAIN5 i TRAIN8 o TRAIN9 indistintament. Anem a seleccionar TRAIN1, TRAIN5 i TRAIN8, que són de classes 0,1,0. La majoria d'exemples són de classe 0 i, per tant, assignarem la classe 0 a TEST2. Noteu que en aquest cas, si haguéssim seleccionat TRAIN9 en comptes de TRAIN8 si hi hauria hagut canvis.

Amb els criteris seguits, 3NN també encerta en els dos exemples i també tenim una precisió de 100%.

b) Construïu el model de classificació basat en k-means per a k=2 a partir de l'arxiu "train.csv". És a dir, heu de construir el model per a 2-means. Un cop tingueu el model heu de classificar els exemples de l'arxiu "test.csv".

Emprarem les dades tractades a l'exercici 1. Comencem aplicant 2-means dues vegades: una per als exemples de cada una de les dues classes. Agafant com a centroides inicials els dos primers exemples de cada cas ens queden els següents centroides per a la classe 0:

```
0.17 0.67 0.50 0.00 1.00 0.50
0.78 0.44 0.44 0.22 0.00 0.00
```

I per a la classe 1:

```
0.67 0.33 0.78 0.89 1.00 1.00
0.17 0.17 0.33 0.83 1.00 0.00
```

Ara hem d'aplicar 1NN al conjunt de test emprant com a dades d'entrenament aquests quatre centroides. Obtenim la següent matriu de distàncies:

	TEST1	TEST2	CLASS
CENT00	1.32	0.55	0
CENT01	1.57	1.14	0
CENT10	0.87	1.49	1
CENT11	1.36	0.99	1

El veí més proper a TEST1 és CENT10, de classe 1. Per tant, a TEST1 li assignam la classe 1. El veí més proper a TEST2 és CEN00, de classe 0. Per tant, a TEST2 li assignam la classe 0. Ja que totes les classes assignades coincideixen amb les del ground truth, tenim una precisió del 100%.

c) Construïu un arbre de decisió a partir de l'arxiu "train.csv". Un cop tingueu l'arbre, classifiqueu amb ell els exemples de l'arxiu "test.csv".

Ja que els arbres de decisió no treballen amb distàncies, no s'han de normalitzar els atributs. Per tant, treballarem directament amb les dades de "train.csv" i "test.csv".

El primer que s'ha de fer és calcular els punts de tall per als atributs numèrics:

ATRIBUT	TALL	BONDAT
PRICE	15000	0.50
PRICE	25000	0.50
PRICE	35000	0.60
MAINT	750	0.50
MAINT	1500	0.70
DOORS	2.5	0.50
DOORS	3.5	0.70
DOORS	4.5	0.50
PERSONS	3	0.90
PERSONS	4.5	0.80

Pel que fa a l'atribut ordinal, podríem tractar cas per cas cada valor o bé considerar-lo numèric per tal de determinar llindars com en els casos anteriors. Anem a considerar-lo numèric, assignant low=0, mid=1, high=2. Així, els punts de tall i les bondats són:

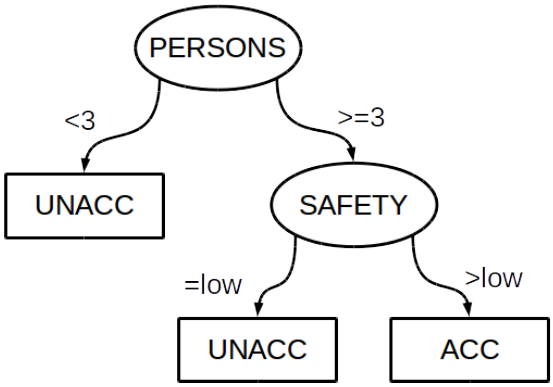
ATRIBUT	TALL	BONDAT
SAFETY	0.5	0.8
PRICE	1.5	0.7

Com es pot veure, l'atribut que ens proporciona una bondat major és PERSONS amb el punt de tall 3. Si observem les dades de l'arxiu train.csv veurem que tots els exemples amb menys de 3 a aquest atribut són de classe 0. Per tant, podríem tancar aquesta branca de l'arbre. Dels exemples

amb més de 3 a l'atribut, tots són de classe 1 excepte un. Anem doncs a repetir el procés amb aquests exemples.

ATRIBUT	TALL	BONDAT
PRICE	15000	0.67
PRICE	25000	0.67
PRICE	35000	0.83
MAINT	750	0.67
DOORS	3	0.67
DOORS	4.5	0.83
PERSONS	4.5	0.67
SAFETY	0.5	1.00
SAFETY	1.5	0.67

En aquest cas obtenim un 100% de bondat amb l'atribut SAFETY i un llindar de 0.5. És a dir, entre els valors low i mid. Si miram els exemples d'aquesta darrera passa, tots els que tenen SAFETY=low són de classe 0 i la resta són de classe 1. Per tant, ja tenim l'arbre complet. Gràficament seria aquest:



Anem ara a classificar els exemples de test. El primer exemple té PERSONS=4. Per tant, anem per la branca de la dreta. Després veim que SAFETY=HIGH, per tant també anem per la branca de la dreta i el classifiquem com ACC (=1).

El segon exemple té PERSONS=2. Per tant, anem per la branca de l'esquerra i el classifiquem com UNACC (=0).

Veim que hem encertat en tots els exemples de test i, per tant, obtenim una precisió del 100%.

d) Apliqueu PCA per a reduir la dimensionalitat dels conjunts anteriors conservant el 95% de la variància. Utilitzeu ara 1NN i 3NN sobre els conjunts reduïts de la mateixa forma que en el primer apartat. Compareu els resultats. Indiqueu clarament les diferències en aplicar PCA sobre el conjunt d'entrenament i sobre el de test.

Emprarem les dades tractades a l'exercici 1. Si apliquem PCA a les dades d'entrenament obtenim les següents variàncies acumulades:

```
0.38 0.61 0.78 0.89 0.98 1.00
```

Per a conservar el 95% de variància necessitem 5 atributs. Noteu que 5 atributs no són tots els atributs, ja que a les dades tractades hem convertit l'atribut SAFETY en dos atributs binaris.

El resultat de projectar el conjunt d'entrenament seria:

```
-1.02 -0.53 1.41  1.04  -1.33
0.98  1.30  0.66 -0.25  0.70
-3.23 0.61  0.86 -0.56  0.28
-0.68 2.00  -1.40 0.53  -0.42
1.03  -1.27 -1.18 0.14  -0.75
1.56  0.85  0.44  -0.50 0.29
-1.36 -0.87 -1.68 0.43  0.90
0.50  -1.66 0.89  0.65  1.12
0.11  -1.01 -0.19 -1.93 -0.60
2.12  0.59  0.20  0.45  -0.19
```

El resultat de projectar el conjunt de test amb les matrius obtingudes pel conjunt d'entrenament seria:

```
0.93 0.91 1.10 -0.66 0.21
0.30 0.23 0.92 0.07 -0.63
```

La matriu de distàncies seria:

```
            TEST1 TEST2 CLASS
TRAIN1      3.35  2.00  0
TRAIN2      0.87  1.88  1
TRAIN3      4.18  3.72  0
TRAIN4      3.44  3.12  0
TRAIN5      3.39  2.69  1
TRAIN6      0.93  1.83  1
TRAIN7      4.21  3.63  0
TRAIN8      3.06  2.65  0
TRAIN9      2.88  2.61  1
TRAIN10     1.93  2.06  1
```

L'exemple més proper a TEST1 és TRAIN2, que és de classe 1. L'exemple més proper a TEST2 és TRAIN6 que és de classe 1. Per tant, 1NN assignaria classe 1 a tots dos exemples i això suposa una precisió del 50%.

Els tres exemples més propers a TEST1 són TRAIN2, TRAIN6 i TRAIN10, tots tres de classe 1. Per tant, s'assignaria la classe 1 a TEST1. Els tres exemples més propers a TEST2 són TRAIN6, TRAIN2 i TRAIN1, de classes 1, 1, i 0 respectivament. En conseqüència, TEST2 es classificaria com a classe 1. Així, 3NN ens dóna el mateix resultat que 1NN i la mateixa precisió del 50%.

En aquest cas, PCA ha conduït a pitjors resultats en l'aplicació de 1NN i 3NN.

Exercici 3

L'objectiu d'aquest exercici és la construcció d'un model amb un conjunt de dades realista. El conjunt de dades realista és el proporcionat per 'car.csv', el qual conté 1728 exemples de cotxes. Per a la construcció del model haureu de programar en Python emprant la biblioteca sklearn (http://scikit-learn.org)

La biblioteca sklearn proporciona iteradors com *KFold*, *RepeatedKFold*, *GroupKFold o StratifiedKFold* especificament dissenyats per a dur a terme validacions creuades (*cross-validation*). Llegiu-ne la documentació i concentrau-vos en un d'ells. També proporciona implementacions d'algorismes com kNN, SVM, xarxes neuronals, etcètera.

Emprant l'iterador de validació creuada escollit i les dades de 'car.csv', feu una validació creuada de, almenys, kNN, SVM i algun altre classificador inclòs en sklearn de la vostra elecció. Justifiqueu cada decisió que prengueu.

Mostrau els principals resultats obtinguts i indiqueu les conclusions que se'n poden extreure.

En aquest exercici emprarem les dades tractades a l'exercici 1. Hem escollit realitzar les proves sol·licitades amb un 10-fold cross validation bàsic. Per això hem emprat l'iterador KFold de sklearn. Els algorismes que hem provat són: KNN, SVC i NuSVC (ambdós són SVM), MLT (perceptró multicapa) i GaussianNB.

	KNN	SVC	NUSVC	MLP	GNB
T. MODEL(MITJANA)	0.0016s	0.0306s	0.0658s	1.1654s	0.0016s
T. MODEL(DESVIACIÓ)	0.0019s	0.0039s	0.0023s	0.2118s	0.0005s
T. CLASS (MITJANA)	0.0048s	0.0022s	0.0041s	0.0006s	0.0003s
T. CLASS (DESVIACIÓ)	0.0006s	0.00007s	0.00005s	0.0011s	0.0002s
ACCURACY	87.905%	92.188%	87.5%	92.419%	79.861%
PRECISION	86.014%	83.538%	77.963%	87.283%	59.815%
RECALL	71.236%	92.085%	81.274%	87.452%	100%
CLASS. CORRECTES	1519	1593	1512	1597	1380
CLASS. INCORRECTES	209	135	216	131	348
MATRIU DE CONFUSIÓ	1150 60 149 369	1116 94 41 477	1091 119 97 421	1144 66 65 453	862 348 0 518

Com es pot veure, qui millors resultats dóna en termes de precisió (accuracy) és el perceptró multicapa (MLP). Si bé és cert que el temps d'entrenament és el més gran, amb diferència, el temps de classificació, que és el que tindria més pes s'hagués d'utilitzar el model, és dels més baixos.

Exercici 4

Realitzeu una valoració global comparant els mètodes dels diferents exercicis i redacteu unes conclusions globals sobre l'aplicació dels mètodes a aquests conjunts de dades. Els criteris de correcció de la PAC invaliden una A si tots els processos no estan ben justificats i comentats.

En aquest exercici s'espera que extreieu conclusions generals sobre els altres exercicis. Aquestes conclusions dependran dels resultats obtinguts. A mode d'exemple, enumerarem algunes de les qüestions sobre les que es podrien argumentar:

- La possibilitat de que sigui pràctica l'aplicació d'algun dels mètodes sobre el problema donat. En cas que no ho sigui, indicar què faltaria afegir.
- Comparativa dels mètodes emprats. Enumeració dels avantatges i inconvenients en funció de la precisió, l'eficiència, les categories, els models, ...
- La representació del problema. Com es comporten els atributs? És una bona representació? Com afecta el preprocés de les dades al funcionament dels algorismes?
- Avantatges dels models que generen els diferents mètodes. Comparativa dels models generats durant tot l'exercici.
- En general, intent de justificació o explicació dels resultats que es van obtenint, fixant-se no només en la precisió.
- Com es comporten els algorismes en funció del nombre d'exemples d'entrenament de que es disposen?
- Quin cost computacional té cada un dels mètodes, tant pel que fa a l'entrenament com per a la classificació?

En comparar els resultats és important notar que els conjunts de dades emprats tenen diferent número d'atributs.

Exercici 5

L'algorisme kNN es basa en determinar els exemples del conjunt d'entrenament més propers a l'exemple que volem classificar. Aquest criteri de proximitat requereix de l'ús d'una distància la qual, habitualment, és la distància euclidiana. Ara bé, existeixen moltes distàncies distintes a l'euclidiana, com ara la distància Manhattan, la distància Mahalanobis o la distància Levenshtein.

Cerqueu informació sobre aquestes distàncies i analitzeu el seu possible ús en kNN. Es podrien emprar? Quins avantatges i inconvenients presentarien respecte de la distància euclidiana? Alguna d'elles seria particularment útil en algun tipus d'aplicació?

No s'acceptaran respostes que continguin text copiat. S'espera que reflexioneu sobre el que es demana i que us expresseu amb les vostres paraules, fins i tot per a definir termes.

La distància Manhattan representa la distància recorreguda sobre una graella a la qual només es poden realitzar moviments sobre dos eixos perpendiculars. La distància de Mahalanobis és una mesura de distància entre un punt i una distribució estadística, típicament una Normal. De fet, la distància euclidiana és el cas particular de Mahalanobis en que la Normal té la identitat com a matriu de covariància. Finalment, la distància Levenshtein és una mesura de les edicions (afegir/eliminar/canviar) que s'han de fer en una cadena de text per a aconseguir-ne una altra.

La distància Manhattan podria tenir una utilitat en problemes de classificació geomètrics. Per exemple, si la intenció fos classificar ubicacions dins d'una ciutat en base al camí que s'hauria de recórrer per arribar-hi a través dels carrers. També s'ha de dir que es podria emprar en substitució de la distància euclidiana en molts de casos per tal d'aprofitar que el seu temps de càlcul és més petit.

La distància de Mahalanobis requereix d'una distribució per tal de poder-se calcular. Podria ser difícil emprar-la directament en el context de kNN, ja que a kNN s'hi calculen distàncies entre punts individuals. Sí podria tenir sentit en el context de K-Means ja que cada categoria conté un conjunt d'exemples dels quals se'n podria calcular la distribució.

La distància de Levenshtein podria emprar-se per a detectar paraules semblants. De fet, és la distància que sovint empren els correctors ortogràfics. Per tant, en un entorn on s'hagués d'identificar un vocabulari reduït, podria emprar-se Levenshtein dins kNN per a decidir a quina de les paraules del vocabulari es correspon una paraula que volguem classificar.