Algorytmy eksploracji danych

plan laboratorium dla kierunku Informatyka studia zaoczne

Marcin Korzeń

Wydział Informatyki, ZUT,
Katedra Metod Sztucznej Inteligencji i Matematyki Stosowanej
`mkorzen@zut.edu.pl`

1 Wprowadzenia do programów Weka i Matlab

Weka

1. Uruchmić Wekę: (Start -> Programy -> Weka -> Weka), lub
2. Pobrac akutualną wersję ze strony:
   `http://www.cs.waikato.ac.nz/ml/weka/index_downloading.html`,
   a zainstalować i następnie uruchomić.
3. Wybrać (Explorer)
4. Wczytać pik z danymi z katalogu (./data) (Preprocess -> Open file)
5. Odczytać i porównać wyniki klasyfikacji dla różnych klasyfikatorów (Classify -> choose), np. naiwny klasyfikator Bayes'a, regresja logistyczna, sieć neuronowa perceptronowa, drzewka decyzyjne (zwizualizować drzewko decyzyjne)
6. Zapoznać się narzędziami do klasteryzacji (Cluster -> choose)
7. Reguły asocjacyjne (Associate)
8. Selekcja atrybutów (Select attributes), przykładowo (InfoGainAttributeEval + Ranker) to znana z ćwiczeń metoda selekcji bazująca na informacji wzajemnej $I(X, Y)$.

Matlab

Przydatne polecenia `.*`, `unique`, `load`, `csvread`, `textread`, `readArff`
(do pobrania ze strony: `http://wikizmsi.zut.edu.pl/kursy`).

1. Pobrać z repozytoriun KDD UCI dowolny zbiór z danymi dyskretnymi i wczytać go do Matlaba
2. Napisać funkcję `[xi, pi, ni]=freq(x)`, która dla zadanej kolumny danych x dyskretnych zwróci: unikalne wartości xi, ich estymowane prawdopodobieństwa pi oraz częstości ni, przykładowe wywołanie powinno zwrócić:

```
>> [xi, ni] = freq([1 1 2 2 1 4 2 4 4 1]')

xi =

     1
     2
```

```
     4

ni =

     4
     3
     3
```

ale również:

```
>> [xi, ni] = freq({'a1'; 'a1'; 'a2'; 'a2'; 'a1'; 'a4'; 'a2'; 'a4'})

xi =

'a1'
'a2'
'a4'

ni =

     3
     3
     2
```

3. Napisać funkcję `[xi, yi, ni] = freq2(x,y)`, która dla zadanych kolumn danych x i y zwróci: unikalne wartości xi, yi oraz łączny rozkład częstości/liczności ni.

```
[xi, pi, ni] = freq2([1 1 2 2 1 4 2 4 4 1]' , [1 1 1 1 1 2 2 2 2 2]')

xi =

     1
     2
     4

yi =

     1     2

ni =

     3     1
     2     1
     0     3
```

4. Wykorzystując powyższe funkcje, napisać funkcje, które wyliczą: entropię `h=entropy(x)` oraz przyrost informacji `i=info(x,y)`
5. Dokonać selekcji/stopniowania atrybutów z wykorzystaniem kryterium przyrostu informacji.

2 Python

2.1 scipy.stats, scipy.io i pandas

```
from scipy.io import arff
import pandas
import numpy as np

dd, meta = arff.loadarff("UCI/autos.arff")
attr_1=[x[1] for x in dd]
print attr_1

# objekty DataFrame i Series
df = pandas.DataFrame(dd)
attr_1 = df['make']
attr_1 = df.make
print attr_1.values()
print attr_1.value_counts()

print df.columns
print df.corr()
print df.values[:,2:10]
gr = df.groupby('make')
print gr.groups
print gr['city-mpg','highway-mpg'].mean()
print gr['fuel-type'].value_counts()
# tabele numpy
dd_np = df.values
dd_np = np.array(df)
print pandas.crosstab(dd_np[:,2], dd_np[:,3])
```

1. Wczytać dane codeautos.arff
2. Określić typy poszczególnych atrybutów
3. Podać częstości atrybutów dysktetnych oraz średnie i odchylenia standardowe atrybutów ciągłych
4. Wykreślić histogramy i wykresy pudełkowe dla wybranych atrybutów ciągłych
5. Poszukać zależności pomiędzy zmiennymi ciągłymi
6. Poszukać zależności pomiędzy zmiennymi dyksretnymi

2.2 scikit-learn

Struktura kasyfikatora s **scikit-learn**:

```
class MojWlasnyClassifier(object):
    def __init__(self, params):
        self.params = params
    def predict_proba(self, x):
        p =
        return p
    def predict(self, x):
```

```
        y =
        return y
    def fit(self, xu, yu):
        pass
```

Przykładowo drzewko decyzyjne oraz eksport danych do **graphviz** a:

```
clf = tree.DecisionTreeClassifier(max_depth=3)
clf = clf.fit(dd_np[:,9:11], dd_np[:,3])
print clf.predict(dd_np[:,9:11])
# eksport danych  do graphviza
dot_data = StringIO()
tree.export_graphviz(clf, out_file=dot_data, feature_names=df.columns
    [9:11])
print dot_data.getvalue()
```

3 Selekcja zmiennych

Cel ćwiczenia

Celem ćwiczenia jest przebadanie podstawowych metod selekcji zmiennych.

Zadania

1. Wczytać i przygotować do analizy następujące zbiory danych: `zoo`, `cpu_act` (uwaga: wymagana binaryzacja względem mediany), `waveform-5000` (uwaga: wymagana binaryzacja względem mediany), `reuters` (uwaga: wymagana binaryzacja zmiennych: $X_i = 0$, $X_i > 0$).
2. Napisać funkcję `entropy` wyznaczającą entropię atrybutu, lub entropię łączną zestawu atrybutów.
3. Napisać funkcję `infogain` wyznaczającą przyrost informacji wykorzystując jedną z zależności: $I_H(X,Y) = H(X) + H(Y) - H(XY)$, $I(Y,X) = H(Y) - H(Y|X)$.
4. Napisać funkcję `binaryzacja` binaryzującą atrybut w taki sposób, aby dostarczał on jak największa ilość informacji o atrybucie decyzyjnym
5. Dla wybranych danych wybrać, w zależności od rozmiaru danych, po kilka/kilkanaście atrybutów dostarczających jak największą ilość informacji na temat atrybutu decyzyjnego.
6. Powtórzyć zadanie posługując się współczynnikiem korelacji Pearsona.
7. Porównać wyniki z analizą składowych głównych: patrząc na wielkość współczynników w początkowych wektorach własnych.
8. Zwizualizować dane `waveform-5000` rzutując na zmienne wyselekcjonowane przyrostem informacji.
9. Przedstawić wnioski dotyczące eksperymentu.

4 Wizualizacja danych

Cel ćwiczenia

Celem ćwiczenia jest wykorzystanie analizy składowych głównych do wizualizacji danych wielowymiarowych

Zadania

1. Wczytać i przygotować do analizy następujące zbiory danych: `zoo`, `waveform5000`, `vote`.
2. Dokonać transformacji PCA, wykorzystując jedną z metod: `cov` +`eig` (**Matlab**, `numpy`), `pca`, `ppca` (**netlab**), `princomp`, `sklearn.decomposition.pca.PCA`. `sklearn.decomposition.pca.ProbabilisticPCA` (**csikit-learn**)
3. Przedstawić graficznie wartości własne macierzy kowariancji – wariancje składowych głównych posortowane od najbardziej istotnej.
4. Wybrać liczbę składowych głównych która wyjaśnia 90% zmienność zmiennych oryginalnych.
5. Zwizualizować dane w rzucie na dwie i trzy pierwsze składowe główne (na osobnych rysunkach), zaznaczając każdą z klas decyzyjnych innym kolorem.
6. Przedstawić wnioski np.: jaka jest jakość wizualizacji w porównaniu z losową dwu– lub trzy–wymiarową projekcją danych oryginalnych.

5 Optymalny klasyfikator Bayesowski, naiwny klasyfikator Bayesa

Naiwny klasyfikator Bayesa, podejście proste — wszystkie atrybuty dyskretne

Wykorzystując dwie funkcje z poprzednich zajęć (`freq`, `jointfreq`) należy zbudować naiwny klasyfikator Bayesa. Można to zrobić wykonując kolejno punkty:

1. Uczenie klasyfikatora. Napisać funkcje `bc = learnBayes(X,d)`, która dla danych uczących $\{X, d\}$ dobierze parametry klasyfikatora. Parametrami klasyfikatora są: rozkład decyzji oraz rozkłady warunkowe $p(X_i|d)$, `bc` może być klasą lub strukturą, która zawiera następujące pola:

   - `bc.name = 'naiveBayes'` – nazwa klasyfikatora
   - `bc.freqdec` – rozkład decyzji
   - `bc.freqcond` – potrzebne rozkład warunkowe

   Należy wziąć pod uwagę sposób identyfikacji prawdopodobieństw dla zdarzeń, które nie występują w danych (rozważyć dwa warianty czystościowy oraz poprawka Laplace'a).

2. Klasyfikacja. Napisać funkcję `[dec, pdx] = classify(bc,x)`, która dla zadanego modelu – struktury `bc` oraz zadanej próbki `x` zwróci: podejmie decyzję (wskaże najbardziej prawdopodobną klasę decyzyjną oraz prawdopodobieństwa decyzji $p(d|x)$)

3. Testowanie. Napisać funkcję obliczającą: dokładność klasyfikacji oraz błąd modelu: `[acc,err] = bayesError(bc,X,d)`. (uwaga: błąd modelu należy wyznaczać na uprzednio przygotowanym zbiorze testowym)

4. Testowanie. Napisać funkcję szacująca błąd modelu metodą krosswalidacji: `err = bayesCross(bc,X,d)`.

Naiwny klasyfikator Bayesa, atrybuty ciągłe i dyskretne

Dodać do powyższego modelu obsługę atrybutów ciągłych. Estymację prawdopodobieństw należy zrobić wykorzystując algorytm EM. (wskazówka: można posłużyć się pakietem `netlab`).

Optymalny klasyfikator Bayesa

Opisane podejście dotyczy w zasadzie atrybutów ciągłych. Należy dokonać estymacji rozkładów warunkowych ale wielowymiarowych, nie posługując się założeniem "naiwnym". To znaczy od oszacowania:

$$Pr(c_i|\mathbf{x}) \propto Pr(c_i)p(\mathbf{x}|c_i),$$

używamy $Pr(c_i)$ tak jak poprzednio, ale $p(\mathbf{x}|d_i)$ estymujemy łącznie dla wszystkich atrybutów. Do tego celu można wykorzystać algorytm EM oraz posłużyć się pakietem `netlab`. Ważne jest, aby uprzednio założyć postać funkcyjną rozkładu łącznego pod warunkiem decyzji np. jako rozkład normalny lub mieszankę rozkładów normalnych. Należy napisać cztery funkcje jak poprzednio.

6 Złożóność próbkowa klasyfikatora

Cel ćwiczenia

Złożoność próbkowa jest to minimalny rozmiar próby wystarczający do nauczenia klasyfikatora, aby osiągnąć zadany błąd klasyfikacji. Celem ćwiczenia jest porównanie kilku wybranych klasyfikatorów ze względu na złożoność próbkową.

Zadania

1. Wczytać i przygotować do analizy następujące zbiory danych: `zoo`, `waveform5000`, `vote`.
2. Napisać funkcję, która dzieli zbiór danych na część uczącą i testującą w zadanej proporcji α.
3. Dla każdego z klasyfikatorów (naiwny Bayes, naiwny bayes + poprawka Laplace'a, regresja logistyczna) i dla każdego zbioru danych wyznaczyć dokładność klasyfikacji jako funkcję parametru α (proporcja rozmiaru próby uczącej w stosunku do całego zbioru danych). Przy ustalonym α wykonać 50 lub 100 powtórzeń i wynik uśrednić.
4. Dla każdego zbioru danych przedstawić osobne wykresy, na których wykreślić zmianę dokładności klasyfikacji w funkcji parametru α dla wszystkich klasyfikatorów.
5. Odczytać z wykresu i przedstawić w tabeli minimalną liczbę próbek uczących przy której dokładność klasyfikacji się ustala.
6. Przedstawić wnioski dotyczące eksperymentu, np.: czy niezależnie od zbioru danych niektóre klasyfikatory mają mniejszą złożoność próbkową? czy poprawka Laplace'a przy estymacji prawdopodobieństwa poprawia jakość uczenia, dla małych prób.

Przydatne pakiety i funkcje

Matlab: **netlab**, `randperm`.
Python: `sklearn.cross_validation.KFold`, `cross_validation.ShuffleSplit`, `sklearn.metrics`

7 Wskaźniki jakości klasyfikacji

Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z podstawowymi wskaźnikami jakości klasyfikacji jak: dokładność klasyfikacji, czułość, specyficzność, precyzja, krzywa ROC, AUC. W tym zadaniu rozważamy klasyfikatory, które umożliwiają predykcje prawdopodobieństwa decyzji np.: naiwny klasyfikator bayesowski, regresja logistyczna, k-NN, sieć neuronowa.

Zadania

1. Wczytać bazę `Reuters-21578`.
2. Wybrać atrybut decyzyjny z grupy tagów: `PLACES`, `TOPICS`, `PEOPLE`, może to być na przykład kraj Japonia, ważne żeby atrybut nie był zbyt częsty. Tabela `WORDS` zawiera atrybuty (słowa) warunkowe.
3. Wybrać podzbiór zbioru atrybutów, tak aby możliwa była dalsza analiza (ten krok nie jest wymagany, jednak niektóre algorytm mogą działać wolno, skomentować to we wnioskach), zaproponować własną metodę selekcji.
4. Do testowania użyć „hold out" kroswalidacji, uśrednionej w 50 iteracjach, podział danych na uczące i testujące stosunku 1 : 1.
5. Wyniki prezentować w postaci:
   (a) W tabeli: wiersze miara jakości (dokładność klasyfikacji, czułość, specyficzność, precyzja, F_1, zbalansowana dokładność, AUC), kolumny kolejne klasyfikatory minimum cztery różne.
   (b) Na jednym wykresie przedstawić krzywe ROC dla różnych klasyfikatorów i wybranego zbioru testowego. Zaznaczyć na krzywych ROC punkty o najwyższej dokładności klasyfikacji. Czy można graficznie na tej podstawie porównać klasyfikatory?
6. Przedstawić wnioski dotyczące eksperymentu, np.: którą miarą warto się posługiwać? który klasyfikator jest najlepszy z punktu widzenia automatycznej selekcji dokumentów? Jaki był czas uczenia, predykcji dla różnych klasyfikatorów?

Przydatne pakiety i funkcje

Matlab: **Statistics Toolbox**, **netlab**, `randperm`, `perfcurve`, `confusionmat`, `cvpartition`, `mlp`, `newff`, `glm`, `glmfit`, `knn`, `knnsearch`

Python: **scikit-learn**, `sklearn.cross_validation.KFold`, `cross_validation.ShuffleSplit`, `sklearn.metrics`, `sklearn.neighbors.KNeighborsClassifier`, `sklearn.linear_model.LogisticRegression`

8 Algorytm k-NN

Cel ćwiczenia

Celem ćwiczenia jest analiza działania algorytmu k-NN. W ćwiczeniu będzie badany wpływ parametru klasyfikatora k na złożoność klasyfikatora, dla szycznych danych testowych.

Zadania

1. Wygenerować dwa zestawy danych testowych:
   (a) kwadrat podzielony na dwie równe części dowolną prostą, sześcian podzielony na dwie równe części dowolną płaszczyzną, kostkę 4,5,6,10 wymiarową podzieloną analogicznie hiperpłaszczyzną (wskazówka: taka hiperpłaszczyzna musi przechodzić przez punkt $(\frac{1}{2}, \ldots, \frac{1}{2})$).
   (b) kwadrat podzielony na dwie równe części dowolną prostą, szachownica składająca się z 4, 9, 16, 25 kwadratów.

   Dane zaburzyć w sposób losowy np. dodając do współrzędnych punków szum gaussowski. Rozmiar danych N przyjąć możliwie duży.
2. Dla każdego ze zbiorów danych w obrębie zestawu danych, oraz dla k zmieniającego się od 1 do rozmiaru próby uczącej (z rozsądnym krokiem) wykonać:
   (a) Podzielić dane na dwie równe części: uczącą i testującą.
   (b) Na części uczącej nastroić klasyfikator k-NN, a następnie wyznaczyć dokładność klasyfikacji na próbie testowej (jako funkcję zależną od zbioru danych oraz od liczby sąsiadów k).
3. Prezentacja wyników. Dla każdego z dwóch zestawów danych: przedstawić na jednym wykresie dokładność w funkcji parametru k, dla wszystkich zbiorów danych w obrębie zestawu. Dwa rysunki na każdym z nich tyle krzywych ile jest zbiorów danych w obrębie zestawu danych.
4. Dla każdego zbioru danych wykresów odczytać maksymalną liczbę k (k_{opt}), po której następuje gwałtowna utrata dokładności modelu, a następnie wyznaczyć: $\alpha = \frac{k_{opt}}{N_u}$. Przedstawić w tabeli uzyskane wartości k_{opt}, α oraz liczbę $\frac{2}{\alpha}$, pogrupowane zestawami danych, zbiorami danych (stopniując złożoność).
5. Przedstawić wnioski dotyczące eksperymentu. Odnieść się do następujących zagadnień:
   (a) Czy można powiązać parametr k z złożonością zbioru danych (np. z wymiarem czy liczbą kwadratów szachownicy) lub klasyfikatora?
   (b) Czy wybór płaszczyzny przy generowaniu danych ma jakieś znaczenie?
   (c) Czy rozkład punktów (x-ów, np. na tle kwadratu, szachownicy) ma znaczny wpływ na wyniki eksperymentu?

9 Klasyfikacja cyfr pisanych ręcznie na podstawie bazy MNIST

Cel ćwiczenia

Celem ćwiczenia jest zbudowanie klasyfikatora rozpoznającego cyfry pisane odręcznie zapisane na matrycach 28×28 na podstawie bazy MNIST `http://yann.lecun.com/exdb/mnist/`. Cel drugorzędny to porównanie klasyfikatorów liniowych (regresja logistyczna, liniowy SVM) względem bardziej złożonych (k-NN, SVM (wielomianowe, RBF), klasyfikatory zespołowe).

Zadania

1. Wczytać dane, przygotowane dane posiadają część uczącą oraz część testową, cześć testową należy wykorzystać jedynie na końcu eksperymentów do porównania wyników, wybór najlepszego klasyfikatora powinien być wykonany na próbie uczącej - wykorzystując kroswalidację lub procedurę bootstrap. Pracujemy na danych surowych, bez dodatkowych metod ekstrakcji cech.
2. Zwizualiwoać przykładowe obrazy/znaki, wymagana zamiana wektora na matrycę 28×28 (`reshape`).
3. Przygotować skrypt z eksperymentem (dwie pętle iterujące po klasyfikatorach i testowa). Jako miarę jakości można przyjąć błąd klasyfikacji. Do zamiany klasyfikatora binarnego na klasyfikator multiklasowy można wykorzystać strategię $1-1$ lub $1-pozostali$ (w `sklearn`: $OneVsRestClassifier$, OneVsOneClassifier).
4. Prezentacja wyników:
   - tabelka przedstawiająca wyniki eksperymentu (dokładność $\pm$ błąd) wykroswalidowane z próby uczącej oraz błąd wyznaczony na próbie testowej,
   - macierz konfuzji dla klasyfikatora najlepszego na próbie testowej,
   - załączyć rysunki 10 przykładowych błędnych rozpoznań klasyfikatora,
   - załączyć rysunek próbek po przekształceniu PCA różnymi kolorami zaznaczając różne klasy decyzyjne - w tym przypadku cyfry.
5. Sprawozdanie zakończyć wnioskami dotyczącymi.

10 Grupowanie danych hierarchiczne algorytmy klasteryzacji, algorytmy EM i K-środków

Hierarchiczne algorytmy klasteryzacji - łączenie

Napisać funkcje:

`d=distance(X,C, method)` która zdefiniuje metrykę w przestrzeni euklidesowej oraz wyliczy macierz odległości pomiędzy dwoma zbiorami punktów X i C

`D=distclustXXX(A, B, @distance)` , określić kilka podstawowych funkcji do mierzenia odległości pomiędzy skupieniami A i B, $XXX = \{Nearest, Complete, Average, Mean, Ward\}$

`[cl, part]=agglomerative(X, @distclustXXX, n)` która wykona algorytm łączenia do momentu uzyskania n-skupień (dla $n = 1$ algorytm wykona się do końca uzyskując jednoelementową klasę zawierającą cały zbiór X). Argumenty zwracane: `cl` - lista (np. macierz komórkowa) zawierająca indeksy punktów należące do kolejnych skupień, `part` - odpowiednio dobrana struktura pamiętająca historię łączenia skupień w trakcie działania algorytmu, oraz odległości między łączonymi skupieniami.

`dendrogram(part)` funkcja rysująca drzewo (dendrogram) ilustrujące proces łączenia klas

Hierarchiczne algorytmy klasteryzacji - podział

Napisać funkcje

`tree=minspan(X, @distance)` , która wyznaczy minimalne drzewo spinające zbioru X

`cl, part]=partitioning1(X, tree, n)` , która będzie dokonywała podziałów usuwając kolejno z drzewa spinającego krawędzie o maksymalnej długości, do momentu uzyskania n-skupień (dla $n = length(X)$ algorytm wykona się do końca uzyskując jednoelementową klasę zawierającą cały zbiór X). Argumenty zwracane jak poprzednio: `cl` - lista (np. macierz komórkowa) zawierająca indeksy punktów należące do kolejnych skupień, `part` - odpowiednio dobrana struktura pamiętająca historię łączenia skupień w trakcie działania algorytmu, oraz odległości między łączonymi skupieniami.

`[cl, part]=partitioning2(X, tree, n)` , która wykona algorytm podziału do momentu uzyskania n-skupień usuwając w kolejnych krokach krawędzie leżące ńajbardziej wewnątrz"drzewa spinającego. Argumenty zwracane jak poprzednio: `cl` lista (np. macierz komórkowa) zawierająca indeksy punktów należące do kolejnych skupień, `part` - odpowiednio dobrana struktura pamiętająca historię łączenia skupień w trakcie działania algorytmu, oraz odległości między łączonymi skupieniami.

`dendrogram(part)` funkcja rysująca drzewo ilustrujące proces łączenia klas

Algorytm K–środków

Napisać funkcje:

`d=dp(X,C,p)` , która wyliczy odległość pomiędzy dwoma zbiorami punktów X (to mogą być wzorce uczące) i C (to mogą być centra), p jest wykładnikiem (d_2 – to zwykła odległość euklidesowa), zgodnie ze wzorem:

$$d_p(x, c) = \sqrt[p]{\sum_{i=1}^{n} |x_i - c_i|^p}.$$

`d=dm(X,C,A)` , która wyliczy odległość Mahalanobis'a pomiędzy dwoma zbiorami punktów X (to mogą być wzorce uczące) i C (to mogą być centra), dla zadanej macierzy cowariancji A, zgodnie ze wzorem:

$$d_p(x, c) = \sqrt{\sum_{i=1}^{n} (x - c)^T A^{-1} (x - c)}.$$

`[C,CX]=ksrodki(X, k)` która dla zadanej macierzy wzorców X oraz liczby grup k, wyznaczy centra C i sąsiedztwa CX.

`b=bladKwantowania(X,C)` wyznaczy błąd kwantowania zbioru X, przez zbiór C:

$$e(\mathbf{c}) = \sum_{p=1}^{P} \|\mathbf{x}^{(p)} - \mathbf{c}^*(\mathbf{x}^{(p)})\|$$

, gdzie $\mathbf{c}^*(\mathbf{x})$ jest środkiem sąsiedztwa dla punktu.$\mathbf{x}$

Pakiet `netlab`

Zapoznać się z algorytmami: PCA, K–środków i EM z pakietu `netlab`, można go pobrać ze strony: `http://www.ncrg.aston.ac.uk/netlab/book.php`.

- Wczytać do Matlaba zbiór irysów (iris.txt).
- Zwizualizować dane używając funkcji `plot`, `plot3`, `pca` lub `ppca`, różne klasy decyzyjne zaznaczyć używając różnych wzorów
- Wykonać dla danych algorytm `kmeans`
- Wykonać algorytm EM, przydatne funkcje to: `gmm`, `gmminit`, `gmmem`, `gmmprob`, `gmmactiv`, `gmmpost`. Całość zwizualizowac.
- Porównać wyniki klasteryzacji z klasami decyzyjnymi
- Dokonać dyskretyzacji wybranej zmiennej posługując się algorytmami: EM i k–środków.
- dodać obsługę zmiennych ciągłych do naiwnego klasyfikatora Bayes'a (zmienne ciągłe można dyskretyzować lub identyfikować rozkłady zmiennych ciągłych używając algorytmu EM).

Sprawozdanie

1. Pobrać kilka zbiorów danych z repozytorium UCI, oraz wygenerować kilka zbiorów testowych
2. Zilustrować graficznie działanie poszczególnych algorytmów, jeżeli zajdzie taka konieczność wizualizować w przestrzeni wyznaczonej przez dwie, trzy pierwsze składowe główne
3. Ilościowo określić miary jakości grupowania stosując odpowiednie miary BIC, błąd kwantyzacji
4. Określić teoretycznie lub oszacować empirycznie szybkość działania algorytmów w zależności od rozmiaru zbioru wejściowego.
5. Dokonać jakościowej analizy uzyskanych wyników wskazać metody nadające się do zastosowań praktycznych ,wnioski dotyczące eksperymentów

11 Wykrywanie reguł i wzorców

Zbiór zadań obejmuje preprocesing danych (przede wszystkim binaryzację), wykrywanie reguł asocjacyjnych o zadanym wsparciu i zaufaniu, oraz wykrywanie reguł najbardziej interesujących.

Algorytm apriori

Zadanie praktyczne, anpisać następujące funkcje:

1. `binTab=binaryzacja(X)` funkcja ta powinna dokonać binaryzację tablicy z danymi, jako argument można podać dodatkowo macierz komórkową z wartościami atrybutów, które nas interesują (przykładowo dla zmiennej płeć zwykle interesują nas obie wartości zarówno M jak i F; natomiast jeżeli zmienna zawiera przykładowo wynik testu medycznego to raczej interesujący przypadek jest wynik dodatni natomiast wynik ujemny może nie mieć znaczenia dla analizy)
2. `rules=apriori(binTab, minSupp, minConf)` lub `rules=apriori(Xbin, dbin, minSupp, minConf)` funkcja do znajdowania reguł częstych o dany minimalnym wsparciu i zaufaniu, w drugim wywołaniu interesują nas te reguły które w konkluzji maja atrybut d, a nie dowolne reguły. Sam algorytm składa się z dwóch części: w pierwszej przeszukujemy wszystkie układy reguł oraz przycinamy drzewo przeszukiwań korzystając z monotoniczności wsparcia, w drugiej mając zbiory częste tworzymy na ich podstawie reguły o zadanej wartości wsparcia. (uwaga: 1) wyszukiwanie zbiorów częstych ,można zrealizować na dwa sposoby, 2) można ograniczyć głębokość drzewa (liczbę termów w regule) do pewnej ustalonej wartości np.: 4, pełne wykonanie algorytmu zwłaszcza z małym `minSupp` oraz dużą liczbą atrybutów może zająć trochę czasu.)
3. `inds=pareto_(s, c)` Napisać funkcję, która dla zadanych list `s` (wsparcie), `c` (zaufanie) znajdzie indeksy reguł leżących na brzegu Pareto.
4. Poeksperymentować z innymi miarami 'zaufania': jak entropia, dywergencja Kullback'a –Leiber'a

Sprawozdanie:

1. Przygotować zbiór danych do eksperymentów (baza Reuters np.: wybrać same tagi lub same słowa lub niektóre słowa, liczba atrybutów powinna być rzędu setek/tysięcy. Dla wybranego zbioru ustalić eksperymentalnie wsparcie na rozsądnym poziomie.
2. Rozbić dane na część ucząca i testującą.
3. Uruchomić algorytm na danych uczących, następnie wypisać reguły spełniające zadane wsparcie i zaufanie, uwaga: reguły wypisać w formie zrozumiałej dla człowieka posługując się nazwami atrybutów. We wnioskach skomentować czy reguły te są zgodne ze zdrowym rozsądkiem.
4. Wszystkie reguły zaznaczyć jako punkty na wykresie wsparcie – zaufanie. Ze znalezionych reguł wybrać te które są Pareto optymalne – o maksymalnych wartościach par (wsparcie, zaufanie). We wnioskach dodać komentarz jak wyżej.
5. Przetestować znalezione reguły, a następnie wypisać te, które są istotne na zbiorze testującym (np. test dokładny Fishera lub test χ^2, poziom istotności skorygować o liczbę reguł). Porównać ich wsparcie i zaufania z wynikami na zbiorze uczącym.
6. Poeksperymentować z różnymi wartościami wsparcia i obserwować liczbę zbiorów częstych oraz czas działania algorytmu. Wyniki przedstawić na wykresie w układzie: `log(LiczbaCzestych)`, `log(czas)`. Wypowiedzieć się na temat złożoności obliczeniowej wyznaczonej empirycznie.
7. Poeksperymentować z innymi miarami 'zaufania' jak np.: entropia, dywergencja Kullback'a – Leiber'a
8. Przedstawić wnioski dotyczące eksperymentów.

12 Krótkie wprowadzenie do R

1. Wprowadzenie do języka R: `http://cran.r-project.org/doc/contrib/Hiebeler-matlabR.pdf` lub `http://mathesaurus.sourceforge.net/octave-r.html`
2. Wczytać dane do R (polecenia: `read.csv`, (pakiet **lattice**))
3. Zwizualizować dane (polecenia: `plot`,`xyplot` (pakiet **lattice**),`princomp`)
4. Zbudować i przetestować drzewko decyzyjne (polecenia: `rpart`, `tree`, `cv.tree`, `prune`, `predict`, `plot`, `text`, pakiety: **tree**, **rpart**), wyznaczyć macierze konfuzji (table).
5. Zbudować model regresji logistycznej (polecenia: `glm`, `summary`)
6. Wykonać grupowanie danych na używając hierarchicznych metod aglomeracji:
   - metoda najbliższego sąsiedztwa
   - metoda średnich połączeń
   - metoda najdalszych połączeń
   - metoda centroidów
   - metoda Warda

   oraz metody k-środków, a następnie zwizualizować drzewo połączeń oraz wyniki grupowania. Przydatne polecenia (`kmeans`, `hclust`, `cutree` `plot`, `heatmap`)

13 Zadanie praktyczne – klasyfikacja

Zadanie podstawowe:

1. Pobrać dane dowolne dane z repozytoriom Weka lub UCI:
   `http://archive.ics.uci.edu/ml/datasets.html`.
   Powinny dotyczyć one interesującego zagadnienia o którym mamy pewną wiedzę (np. `wine`) oraz zawierać zmienne ciągłe jak i dyskretne, ewentualnie dane brakujące.
2. Wczytać dane do Matlab'a
3. Zwizualizować dane w rzucie na składowe główne, ewentualnie w rzutach na przykładowe zmienne
4. Określić sposób radzenia sobie z brakami
5. Określić sposób radzenia sobie ze zmiennymi ciągłymi
6. Dokonać oceny istotności i współzależności zmiennych. (do oceny współzależności można użyć współczynnika korelacji liniowej, lub przyrostu informacji (por. zadanie 1.5))
7. Podzielić dane na dwie części: część uczącą (P, T) i część testującą (PT, TT)
8. Porównać wybrane algorytmy klasyfikacji z zaimplementowanym przez siebie naiwnym klasyfikatorem Bayesa. (Wziąć pod uwagę regresję logistyczna, drzewko decyzyjne C4.5 lub CART, model SVM, analizę można przeprowadzić w programie Weka lub innym.)
9. Dokonać oszacowania błędu klasyfikacji (odsetek błędnych klasyfikacji na zbiorze testowym lub kroswalidacja)
10. Dokonać grupowania danych bez zmiennej decyzyjnej oraz porównać zgodność wyników grupowania z klasami decyzyjnymi
11. Wypisać kilka najbardziej interesujących reguł dla zmiennej decyzyjnej
12. Analiza i wnioski dotyczące eksperymentu. Wnioski mają:
    (a) być zdroworozsądkowe (przykładowo reguły można zrozumieć — zastanowić się, czy one są zaskakujące?, czy wnoszą coś nowego?)
    (b) ocenić praktyczną przydatność takiego klasyfikatora, w tym konkretnym problemie,
    (c) wskazywać główne przyczyny błędów,
    (d) wskazywać możliwe metody ulepszenia analizy.

14 Warunki rozliczenia laboratorium

Aby rozliczyć się w pełni z laboratorium należy wykonać następujące zadania to:

1. wszystkie zadania postawione na zajęciach,
2. oddać wszystkie sprawozdania.

Aby uzyskać ocenę minimalną można opuścić jedno z zadań programistycznych, sprawozdania muszą być napisane wszystkie. Należy mieć świadomość, że z punku widzenia prowadzącego algorytm Apriori jest zadaniem trudniejszym niż np. algorytm K–środków, jakość wykonanych zadań będzie brana pod uwagę. Istnieje możliwość pracy zespołowej (po konsultacji z prowadzącym) jednak musi być silna podstawa do takiej współpracy: jasno określony podział zadań oraz np. wspólne źródła (np. SVN). W przeciwnym razie praca powinna być samodzielna, niedopuszczalne jest przedstawianie cudzych rozwiązań jako własnych. Próba oszustwa może skutkować brakiem zaliczenia.

Istnieje możliwość zwolnienia z części egzaminacyjnej zaliczenia na podstawie systematycznej pracy w trakcie całego semestru w takiej sytuacji:

1. wszystkie zadania z laboratorium mają być wykonane w pełni i starannie,
2. praca musi być systematyczna w trakcie całego semestru (również na ćwiczeniach).

Literatura