KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Instrukcja współfinansowana przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego
w projekcie

*„Innowacyjna dydaktyka bez ograniczeń
– zintegrowany rozwój Politechniki Łódzkiej – zarządzanie Uczelnią,
nowoczesna oferta edukacyjna i wzmacniania zdolności
do zatrudniania osób niepełnosprawnych”*

Instrukcja jest dystrybuowana bezpłatnie

Instrukcja do laboratorium, część 3

dr inż. Artur Klepaczko

Eksploracja danych

Zadanie nr 13 – Studia podyplomowe „Przetwarzanie i analiza obrazów biomedycznych”

Politechnika Łódzka
Instytut Elektroniki

90-924 Łódź, ul. Żeromskiego 116,
tel. 042 631 28 83
www.kapitalludzki.p.lodz.pl

Spis treści

1 Wstęp

Oprócz poznanego już programu Weka, dostępnych jest wiele innych (także komercyjnych) narzędzi do eksploracji danych. Wśród nich należy przede wszystkim wymienić środowisko Matlab®. Program ten — szczególnie funkcje zawarte w bibliotece *Image Processing Toolbox*™ (zob. pkt 2.1) — będzie wykorzystany do przeprowadzenia analizy tekstury obrazów cyfrowych. Dlatego też ćwiczenia laboratoryjne będą polegać na samodzielnym utworzeniu zbiorów danych, a następnie zastosowaniu do nich wybranych algorytmów klasyfikacji.

Matlab® zawiera własny zestaw narzędzi do klasyfiakacji wektorów danych, m.in. bibliotekę *Neural Network Toolbox*™ (zob. pkt 2.2). Co więcej środowisko Matlab® umożliwia uruchamianie zewnętrznych aplikacji, w tym także napisanych w języku Java™. To z kolei pozwoli wykonać klasyfikację z wykorzystaniem algorytmów należących do pakietu Weka.

Oczekiwane efekty kształcenia:

→ Podstawowe umiejętności pracy z programem Matlab® oraz znajomość wybranych funkcji przetwarzania obrazów dostępnych w tym środowisku.

→ Znajomość zasad uruchamiania programów napisanych w języku Java™ z poziomu środowiska Matlab®.

2 Praca w środowisku Matlab®

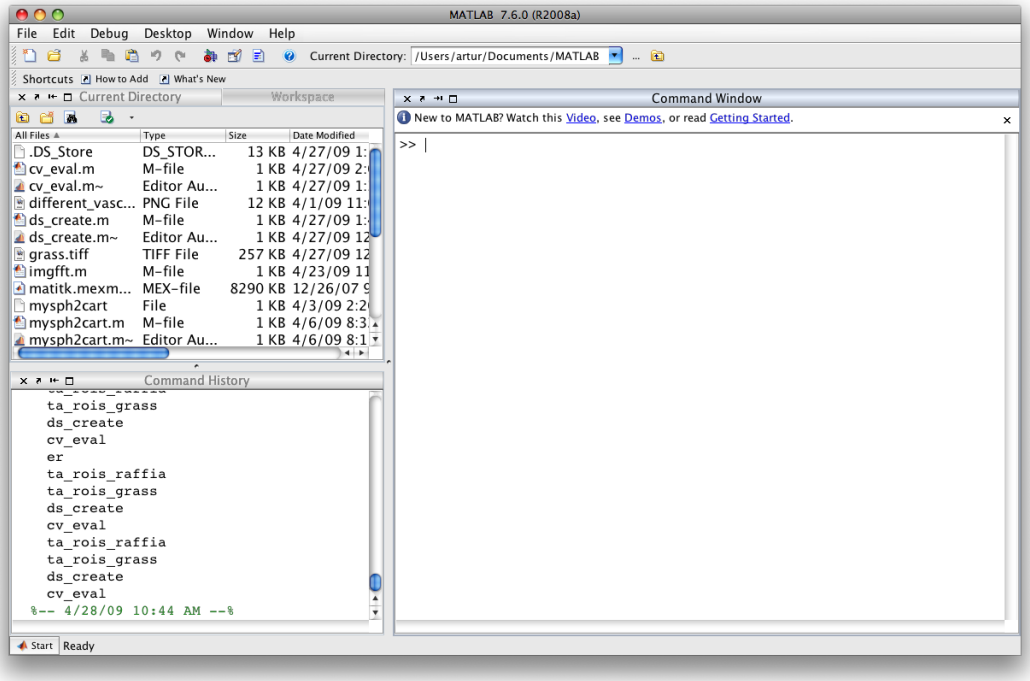
Środowisko Matlab® stanowi rozbudowany zestaw narzędzi służących do rozwiązywania szeregu zagadnień technicznych opisanych za pomocą aparatu matematycznego. Poszczególne funkcje pogrupowane są w tzw. zestawy narządziowe (ang. *toolbox*) odpowiadające wybranym zastosowaniom. Ważnym elementem środowiska jest także dostępny w nim język skryptowy, który pozwala na tworzenie własnych programów przetwarzania danych.

Rysunek 1 przedstawia główne okno programu. Obszar roboczy znajdujący się po prawej stronie zawiera konsolę wraz z linią poleceń (>>). W niej wywołuje się funkcje Matlaba® oraz własne skrypty. Z lewej strony u góry widnieje informacja o zawartości bieżącego katalogu roboczego (zakładka *Current Directory*) lub o utworzonych zmiennych i strukturach danych (zakładka *Workspace*). Poniżej natomiast znajduje się historia poleceń wykonanych do tego momentu (panel *Command History*).

2.1 Biblioteka *Image Processing Toolbox*™

Pierwszym zestawem narzędziowym, który będzie wykorzystywany w ćwiczeniach laboratoryjnych jest biblioteka *Image Processing Toolbox*™ (IPT). Szereg funkcji dostępnych w tej bibliotece umożliwia przeprowadzenie różnorodnych operacji na obrazach (m.in. filtracje, transformacje przestrzenne, operacje morfologiczne, analiza tekstury). Spośród nich wymienić należy:

- `imread` – wczytanie obrazu,



Rysunek 1: Główne okno programu Matlab®

- `imshow` – wyświetlenie obraz,
- `imcrop` – wydzielenie z obrazu prostokątnego obszaru zainteresowania,
- `rangefilt` – obliczenie lokalnych zakresów jasności,
- `stdfilt` – obliczenie lokalnych odchyleń standardowych,
- `entropyfilt` – obliczenie lokalnych wartości entropii,
- `graycomatrix` – utworzenie macierzy zdarzeń dla danego obrazu,
- `graycoprops` – obliczenie parametrów tekstury obrazu na podstawie macierzy zdarzeń.

Szczegółowy opis powyższych funkcji (a także wszystkich innych dostępnych w Matlabie®) znajduje się w systemie pomocy środowiska. Dokumentację dla wybranej funkcji można także wyświetlić wpisując w linii poleceń komendę `help` oraz (po znaku spacji) nazwę funkcji.

Poza wymienionymi funkcjami biblioteki IPT, w przetwarzaniu obrazów zastosowanie znajdują funkcje ogólnego przeznaczenia, takie jak np. `fft2`, która pozwala na wykonanie dwuwymiarowej szybkiej transformaty Fouriera. Na rys. 2 przedstawiono przykładowy skrypt, w którym kolejno wykonywane są operacje wczytania obrazu, wydzielenia w nim obszaru zainteresowania oraz transformacji Fouriera.

```
% Wczytanie obrazu typu TIFF o nazwie raffia.tiff
I = imread('raffia.tiff','tif');
% Wydzielenie ROI o krawędziach równych 64 piksele
I1 = imcrop(I,[0,0,64,64]);
% Wyświetlenie obrazu
figure(1);
imshow(I1);
% Transformata Fouriera w 2D
F = fft2(I1);
% Wyznaczenie amplitudy sygnału
Fabs = abs(F);
% Przeskalowanie
Flog = log(Fabs+1);
% Przesunięcie składowej stałej do środka
Fshift = fftshift(Flog);
% Poprawa kontrastu i zrzutowanie do typu uint8
Fzero = Fshift - min(Fshift(:));
Fuint8 = uint8(round(Fzero*255/max(Fzero(:))));
% Wyświetlenie obrazu transformaty
figure(2);
imshow(Fuint8);
```

Rysunek 2: Przykład analizy obrazu w programie Matlab®

2.2 Biblioteka *Neural Network Toolbox*™

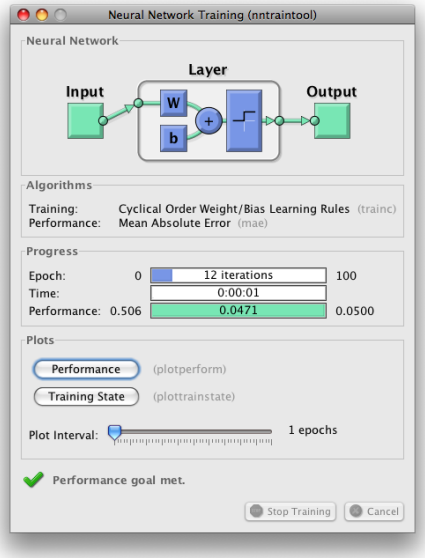
Funkcje należące do biblioteki *Neural Network Toolbox*™ umożliwiają konstruowanie sieci neuronowych o dość dowolnej architekturze, nadających się do rozwiązywania szeregu złożonych zadań. Z punktu widzenia ćwiczeń laboratoryjnych, omówienia wymaga tylko jedna architektura, a mianowicie pojedynczy klasyfikator perceptronowy. Na rys. 3 przedstawiono skrypt, w którym tworzony jest perceptron dla zbioru wektorów danych zawartych w macierzy `data_mat` i odpowiadającym jej wektorze etykiet kategorii `labels`. Należy zauważyć, że w odróżnieniu od konwencji przyjętej w programie Weka, rzędy macierzy danych odpowiadają cechom, zaś kolumny kolejnym wektorom danych.

Procedura tworzenia i uczenia perceptronu wykorzystuje następujące funkcje:

- `newp` – tworzenie percptronu dla określonego zestawu danych,
- `init` – inicjalizacja perceptronu (m.in. ustalenie początkowych wartości wag),
- `train` – uruchomienie procedury uczenia.

```
% Tworzenie perceptronu
perc = newp(data_mat,labels);
% Losowe wartości początkowe wag
perc.inputWeights{1,1}.initFcn = 'rands';
% Inicjalizacja perceptronu
perc = init(perc);
% Ustalanie parametrów uczenia
perc.trainParam.goal = 0.05;
perc.trainParam.epochs = 100;
% Trening perceptronu
perc = train(perc,data_mat,labels);
```

Rysunek 3: Tworzenie i uczenie perceptronu w programie Matlab®



Rysunek 4: Okno uczenia perceptronu

Domyślnie tworzony perceptron przyjmuje skokową funkcję aktywacji. Z kolei procedura uczenia zakłada, że aktualizacja wag — o ile okaże się konieczna — następuje każdorazowo po podaniu wektora danych i wyznaczeniu dla niego odpowiedzi perceptronu. Jeśli nie określono tego inaczej, wagi połączeń między perceptronem, a warstwą wejściową ustawiane są początkowo na wartość 0. Zachowanie to można zmienić poprzez odpowiednie ustawienie zmiennej `initFcn`. Podobnie zmienne `trainParam.goal` oraz `trainParam.epochs` definiują odpowiednio dopuszczalne wartości błędu i liczby iteracji.

Uruchomienie polecenia `train` powoduje wyświetlenie dodatkowego okna (zob. rys. 4), w którym widoczny jest postęp w uczeniu klasyfikatora — zarówno jeśli chodzi o liczbę iteracji (lub *epok*), jak i aktualny błąd. Przycisk *Performance* wyświetla natomiast wykres zmian wartości błędu klasyfikacji poprzez kolejne iteracje.

2.3 Uruchamianie programów Java™

Środowisko Matlab zawiera mechanizm, który pozwala wywoływać w nim zewnętrzne programy napisane w języku C++, Fortran lub Java™. Mechanizm ten pozwala na wydajne tworzenie bardzo zaawansowanych aplikacji integrujących gotowe biblioteki różnych programów. W ten sposób czas niezbędny na przeniesienie kodu pomiędzy środowiskami programistycznymi zostaje znacząco zredukowany.

```
import weka.core.*;
import java.lang.*;
Attributes = weka.core.FastVector(9);
name = java.lang.String('Feature');
for i=1:8
        num = num2str(i);
        f_name = concat(name,num);
        Attr = weka.core.Attribute(f_name);
        addElement(Attributes, Attr);
end
ClassValues = weka.core.FastVector(2);
addElement(ClassValues, 'Raffia');
addElement(ClassValues, 'Grass');
ClassAttr = weka.core.Attribute('Class',ClassValues);
addElement(Attributes, ClassAttr);
DataSet = weka.core.Instances('Raffia_vs_Grass',
                              Attributes, 64);
for i=1:32
        vector = weka.core.Instance(1, rds(i,:));
        add(DataSet, vector);
        vector = weka.core.Instance(1, gds(i,:));
        add(DataSet, vector);
end
setClassIndex(DataSet, 8);
```

Rysunek 5: Tworzenie obiektów i wywoływanie metod Javy w kodzie programu Matlab®. Natywne polecenia Java™ podświetlono kolorem niebieskim, natomiast instrukcje pochodzące z biblikoteki Weka zakolorowano na kawowy

W przypadku bibliotek programów Java™, procedura ich włączenia do środowiska Matlab wymaga następujących kroków. Po pierwsze, samą bibliotekę (zazwyczaj jest to archiwum *.jar) należy dołączyć do tzw. dynamicznej ścieżki klas Javy (ang. *javaclasspath*). Odbywa się to poprzez wywołanie polecenia `javaclasspath('/System/Library/Java/...')`, jako argument przekazując pełną ścieżkę dostępu do katalogu, w którym znajduje się dane archiwum. Operacji tej nie trzeba wykonywać dla standardowych klas Javy, które znajdują się w tzw. statycznej ścieżce klas. W obu przypadkach jednak, aby skorzystać z danej biblioteki, należy wpisać polecenie `import`, które ładuje klasy biblioteki do aktywnej sesji Matlaba.

Tworzenie obiektów Javy odbywa się poprzez wywołanie pełnej kwalifikowanej nazwy klasy obiektu (łącznie z nazwą pakietu) lub za pomocą polecenia `javaObject`. Podobnie, aby wywołać metodę na rzecz utworzonego obiektu wystarczy wpisać w linii poleceń nazwę

metody, podając na liście argumentów najpierw nazwę obiektu, następnie zaś pozostałe parametry wykonania metody. Alternatywnym sposobem wywołania metod jest skorzystanie z polecenia `javaMethod`. Przykładowy skrypt Matlaba®, do którego dołączono standardową biliotekę Javy (java.lang) oraz biliotekę Weka pokazano na rys. 5.

3 Zadania do rozwiązania

Uwaga! Zarówno zbiory danych, jak i pliki programów, których dotyczą kolejne ćwiczenia umieszczone są na serwerze pod adresem `http://eletel.p.lodz.pl/aklepaczko/open/`. Pliki wymienione w treści poszczególnych zadań należy skopiować do katalogu roboczego środowiska Matlab®(np. `$\Moje dokumenty\MATLAB`, gdzie znak `$` należy zastąpić ścieżką dostępu do katalogu domowego użytkownika).

3.1 Klasyfikacja na podstawie cech histogramowych

Zadanie 1

- Pobierz z serwera pliki obrazów `raffia.tiff` oraz `grass.tiff` i zachowaj je na dysku lokalnym komputera. Uruchom środowisko Matlab®.
- Wczytaj i wyświetl pobrane obrazy tekstur (polecenia `imread` oraz `imshow`). Za pomocą jakiej struktury danych reprezentowane są w Matlabie wczytane obrazy?
- Pobierz plik `ta_hist.m`. Otwórz pobrany plik (polecenie *Open...* w menu *File*) i przeanalizuj znajdujący się w nim skrypt. Jakie funkcje zostały użyte do wyznaczenia cech tekstury? Jaki jest rozmiar wykorzystywanego obszaru zainteresowania i ile obszarów jest tu przetwarzanych jednocześnie?
- Wywołaj pobrany skrypt wpisując w linii poleceń tylko jego nazwę (bez rozszerzenia).
- Wyświetl rozkład wyznaczonych wektorów cech w wybranej dwuwymiarowej przestrzeni atrybutów. W tym celu należy posłużyć się poleceniem `plot` w następującej postaci:

  ```
  >> plot(DSg(:,1),DSg(:,2),'or',DSr(:,1),DSr(:,2),'db');
  ```

  W powyższym poleceniu wyrażenia umieszczone w cudzysłowach definiują kształt i kolor znaczników wektorów danych. Jak oceniasz przydatność cech obliczonych na podstawie lokalnej zmienności jasności obrazu do klasyfikacji tekstur?

Zadanie 2

- Zmodyfikuj skrypt `ta_hist` tak, aby oprócz cech już wyznaczanych, obliczane były także wartości atrubutów dla większych rozmiarów sąsiedztwa pikseli.

- Ponownie wyświetl rozkład atrybutów w dowolnej dwuwymiarowej przestrzeni cech. Czy zwiększenie rozmiaru sąsiedztwa spowodowało także większą separowalność klas?

Zadanie 3

- Pobierz plik `ds_create.m`. Otwórz pobrany plik i przeanalizuj znajdujący się w nim skrypt. Jakie pakiety Javy są w nim wykorzystywane? Jakie obiekty Javy są w nim tworzone? Do czego służą funkcje (metody) `addElement` oraz `add`?
- Wywołaj skrypt `ds_create` w konsoli roboczej, a następnie w linii poleceń wpisz nazwę utworzonego zbioru danych (`DataSet`, bez znaku średnika na końcu linii) i naciśnij klawisz Enter. Jaka informacja pojawiła się na ekranie?
- Pobierz plik `cv_eval.m`. Otwórz pobrany plik i przeanalizuj znajdujący się w nim skrypt. Jaki algorytm został użyty do klasyfikacji? Jaka metoda została wybrana do oceny jego błędu? Znajdź opis poszczególnych metod w dokumentacji biblioteki Weka. Wykonaj skrypt `cv_eval`. Ile wynosi oszacowanie błędu klasyfikatora?

3.2 Klasyfikacja na podstawie cechy częstotliowościowych

Zadanie 4

- Pobierz plik `ta_fft.m`. Otwórz pobrany plik i przeanalizuj znajdujący się w nim skrypt. Jakie parametry pobiera funkcja `ta_fft` i co ona zwraca? Jakie funkcje tym razem zostały użyte do wyznaczenia cech tekstury?
- W linii poleceń w konsoli roboczej zadeklaruj macierze `gds2` oraz `rds2`, obie o rozmiarach 32×3, przy czym pierwsza z nich niech będzie wypełniona zerami (polecenie `gds2=zeros(32,3)`), druga zaś jedynkami (`rds2=ones(32,3)`).
- Wykonaj skrypt `ta_fft` dla obu obrazów przy następujących wartościach parametrów wywołania:

  1. r1=8, r2=16, teta1=-22.5, teta2=22.5 oraz
  2. r1=16, r2=32, teta1=-22.5, teta2=22.5,

  przechowując wyniki w odpowiednich kolumnach macierzy `gds2` (dla obrazu *grass*) oraz `rds2` (dla obrazu *raffia*). Przykładowo, polecenie dla pierwszego z obrazów i drugiego zestawu parametrów powinno wyglądać tak:

```
>> gds2(:,2) = ta_fft('grass.tiff', 'tif', 16, 32, -22.5, 22.5);
```

- Wykonaj wykres rozproszenia utworzonych wektorów cech. Jak oceniasz przydatność cech obliczonych na podstawie transformaty Fouriera do klasyfikacji tekstur?

Zadanie 5

- Zmodyfikuj skrypt `ds_create` tak, aby utworzyć zbiór danych na podstawie macierzy `gds2` oraz `rds2` (pamiętaj o zmianie rozmiaru wektora `Attributes` oraz parametru wywołania metody `setClassIndex`).
- Wykonaj skrypt `cv_eval`. Ile wynosi oszacowanie błędu klasyfikatora?

Zadanie 6

- Zadeklaruj macierze `gds` i `rds` o rozmiarach 32×9, wypełnione odpowiednio zerami oraz jedynkami. Wykonaj skrypt `ta_fft` dla obu obrazów i następujących zestawów parametrów:

  1. r1=2, r2=4, teta1=-22.5, teta2=22.5,
  2. r1=4, r2=8, teta1=-22.5, teta2=22.5,
  3. r1=2, r2=4, teta1=22.5, teta2=67.5,
  4. r1=4, r2=8, teta1=22.5, teta2=67.5,
  5. r1=2, r2=4, teta1=67.5, teta2=112.5,
  6. r1=4, r2=8, teta1=67.5, teta2=112.5,
  7. r1=2, r2=4, teta1=-67.5, teta2=-22.5,
  8. r1=4, r2=8, teta1=-67.5, teta2=-22.5.

  Przechowaj wyniki obliczeń w odpowiednich kolumnach macierzy `gds` i `rds`. Zadanie to można zrealizować w oparciu o samodzielne napisany prosty skrypt (polecenie *File→New→M-file*), którego kolejne wiersze zawierałyby odpowiednie wywołania funkcji `ta_fft`.

- Zmodyfikuj skrypt `ds_create` tak, aby odczytywał dane z macierzy `gds` i `rds`.
- Wykonaj klasyfikację dla nowo utworzonego zbioru danych wywołując skrypt `cv_eval`. Jaki jest błąd klasyfikatora?

Zadanie 7

- Zmodyfikuj skrypt `ta_fft` tak, aby cechy tekstury obliczane były dla obszaru zainteresowania o rozmiarach 32×32 piksele.
- Utwórz zbiór danych na podstawie nowych wektorów cech i wykonaj klasyfikację.
- Powtórz obliczenia dla obszaru zainteresowania o rozmiarach 16×16 pikseli.
- Porównaj uzyskane teraz wyniki z wartościami błędów otrzymanymi w zadaniach nr 5 i nr 6. Jaka jest zależność pomiędzy parametrami wywołania funkcji `ta_fft`, rozmiarami obszaru zainteresowania oraz błędem klasyfikacji?

3.3 Klasyfikacja na podstawie macierzy zdarzeń

Zadanie 8

- Pobierz plik `ta_com.m`. Otwórz pobrany plik i przeanalizuj znajdujący się w nim skrypt. Jakie cechy tekstury obliczane są na podstawie macierzy zdarzeń?
- Wykonaj skrypt `ta_com` a następnie wyświetl wykres rozproszenia utworzonych wektorów danych w wybranej trójwymiarowej przestrzeni cech. W tym celu należy wywołać funkcję `plot3`[1]. Przykładowo, aby wykonać wykres w przestrzeni atrybutów *Contrast*, *Correalation* oraz *Energy* wyznaczonych dla kierunku 45° należy w konsoli wpisać polecenie w następującej postaci:

```
>> plot3(DSg(:,13), DSg(:,14), DSg(:,15), 'or',...
   DSr(:,13), DSr(:,14), DSr(:,15), 'db');
```

- Uwtórz zbiór danych za pomocą odpowiednio zmodyfikowanego skryptu `ds_create`.
- Zmodyfikuj skrypt `cv_eval` tak, aby klasyfikacja wykonywana była za pomocą algorytmu *Support Vector Machines* z liniową funkcją jądra. Wykonaj klasyfikację i sprawdź oszacowanie błędu.

Zadanie 9

- W pliku `ta_com.m` zmniejsz rozmiar krawędzi obszaru zainteresowania do 16 pikseli.
- Wykonaj czterokrotnie sekwencję skryptów: `ta_com`, `ds_create` oraz `cv_eval`, za każdym razem pozostawiając do wyznaczenia inną cechę tekstury (tworzone zbiory danych będą zawierać tylko 4 atrybuty oprócz etykiety kategorii).
- Która grupa cech pozwoliła uzyskać najmniejszy błąd klasyfikacji?

Zadanie 10

- Powtórz doświadczenie z poprzedniego zadania tym razem elimując wszystkie poza jednym wybranym kierunkiem sąsiedztwa pikseli przy wyznaczaniu macierzy zdarzeń.
- Który kierunek okazał się najlepszy, a który najgorszy do dyskryminacji klas tekstur?

Zadanie 11

- W pliku `ta_com.m` zwiększ rozmiar sąsiedztwa pikseli branych pod uwagę przy wyznaczaniu macierzy zdarzeń do 3 oraz 5 punktów a następnie powtórz obliczenia z zadania nr 10.
- W jaki sposób zwiększenie analizowanego sąsiedztwa wpływa na wartość błędu klasyfikacji?

[1] Alternatywnie można posłużyć się funkcją `scatter3`.

3.4 Klasyfikacja na podstawie cechy obliczonych w zewnętrznym programie

Zadanie 12

- Pobierz plik `input_ds.m`. Otwórz pobrany plik i przeanalizuj znajdujący się w nim skrypt. Do czego służą funkcje `setSource`, `getDataSet` oraz `useFilter`.
- Pobierz zbiór danych `BR_RA.csv`. Wykonaj skrypt `input_ds`. Co znajduje się w poszczególnych rzędach i kolumnach macierzy `data_mat` oraz w wektorze `labels`?
- Pobierz i przeanalizuj skrypt `perc_train.m`. Jaka jast maksymalna dopuszczalna wartość błędu perceptronu na zbiorze uczącym?
- Wykonaj skrypt `perc_train`. Ile iteracji wystarcza do zakończenia przez perceptron procedury uczenia?

Zadanie 13

- Zmień wartości parametrów `trainParam.epochs` na 2000 oraz `trainParam.goal` na 0. Uruchom procedurę uczenia perceptronu. Jaką minimalną wartość błędu klasyfikacji udało się zaobserwować (wywołaj polecenie `min(tr_rec.perf)`)?
- Powtórz procedurę uczenia dla parametru `trainParam.goal` ustawionym na wartość wyznaczaną w poprzednim kroku.
- Pobierz zbiór danych `BA_RA_test.csv` oraz plik `perc_test.m` i wykonaj znajdujący się w nim skrypt. Jak oceniasz zdolność uogólniającą perceptronu?

Łódź 2009

Zredagowane w LaTeX-u