Home page

Dydaktyka – bieżące

- **AiR – Inf1**
- AiR – PAOM
- AiR – ML
- Inf – AdvML
- Prace mgr 2018/2019

Przydatne

- Jak korzystać z wiki?
- WiFi AGH
- UCI AGH

Zagadnienie klasyfikacji

Wykład

Slajdy

Materiały

Przeanalizuj przykład: Algorytm C4.5
Materiał dodatkowy: Drzewa decyzyjne

Zadanie 1 - kNN - własna implementacja

Proszę przedstawić własną implementację algorytmu “K najbliższych sąsiadów”.

Algorytm:

1. Baza danych Iris - proszę podzielić na zbiór uczący i testowy
2. Wybieramy wartość k
3. Poszukujemy k obserwacji, które są najbliższe do naszego analizowanego przykładu. Do wyznaczania odległości skorzystaj z algorytmu Eukleidesa.
4. Użyj najczęściej pojawiającej się wartości z “k najbliższych sąsiadów” jako wartość dla nieznanego Iris.

Przydatne biblioteki i funkcje:

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from scipy.spatial import distance
from sklearn.metrics import accuracy_score
```

Sposób użycia:

```
db_iris = datasets.load_iris()

#Podziel zbiór na uczący i testowy, test_size - procentowy udział (przykład 50 % uczący i testowy)
features_train, features_test, labels_train, labels_test = train_test_split(iris.data, iris.target, test_size=0.5)

#Przykład użycia odległości euklidesowej
a = (1, 2, 3)
b = (4, 5, 6)
dst = distance.euclidean(a, b)

# Sprawdzanie skuteczności klasyfikatora
output = accuracy_score(labels_test, predictions)
print(output)
```

Zadanie 2 - kNN - Python (sklearn)

Proszę rozwiązać powyższe zadanie korzystając z biblioteki `sklearn.neighbors.KNeighborsClassifier`
Przydatne biblioteki i funkcje:

```
from sklearn import datasets
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

Algorytm:
1. Podziel zbiór na uczący (70 %) i treningowy (30 %).
2. Stworzenie klasyfikatora i jego wyuczenie.
3. Predykcja wartości.
4. Wyznaczenie miary skuteczności dla zbioru testowego (ang. *accuracy*).

Zadanie 3 - drzewa decyzyjne

W bibliotece scikit-learn drzewa decyzyjne implementowane są przez klasę `DecisionTreeClassifier` Ze szczegółami implementacji proszę zapoznać się tutaj:
Decision Trees - Python
Klasa sklearn.tree.DecisionTreeClassifier

Przykład - zbiór danych Iris

```
from sklearn.datasets import load_iris
from sklearn import tree
iris = load_iris()
clf = tree.DecisionTreeClassifier()
clf.fit(iris.data, iris.target)
```

Przewidywanie przynależności przykładów do klas:

```
clf.predict(iris.data[:1, :])
```

lub estymowania prawdopodobieństwa przynależności do klas:

```
clf.predict_proba(iris.data[:1, :])
```

Proszę zilustrować wynik za pomocą narzędzia `Graphiz` oraz doinstalowania do pythona biblioteki `pydot` (zadanie proszę wykonać w domu i wygenerować plik `*.png` lub `*.pdf`)

```
from sklearn.externals.six import StringIO
import pydot
dot_data = StringIO()
tree.export_graphviz(clf, out_file=dot_data)
graph = pydot.graph_from_dot_data(dot_data.getvalue())
graph[0].write_pdf("iris.pdf")
```

Zadanie

Klasyfikacja ręczni pisanych cyfr. Obrazki zostały znormalizowane do rozmiaru 20 x 20 px.
Dane z których będziemy korzystać stanowią obrazek przetworzony do formatu jednowymiarowego przez ułżenie kolejnych wierszy. Dane te pochodzą z bazy danych ręcznie pisanych cyfr MNIST.

Baza danych: Baza MNIST

```
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix, f1_score
from sklearn import tree
from sklearn.cross_validation import train_test_split
from scipy.io import loadmat



# wczytywanie danych
dane = loadmat('baza_mnist.mat.pdf')

#Zad 1. Podziel dane na parametry X oraz odpowiedź y:

X = #TODO
y = #TODO

# Standaryzacja
for i in range(X.shape[0]):
    X[i,:] = X[i,:]/np.std(X[i,:])

# Zamiana cyfry 10 -> 0 (błąd w zbiorze danych)
y[np.where(y==10)]=0

# wysokość i szerokość obrazka z cyfrą
h = 20
w = 20

# Zad 2. Proszę wyświetlić liczbę cyfr oraz liczbę pikseli przypadającą na jeden obraz
#TODO
```

Funkcja pomocnicza `plot_mnist` do wyświetlania obrazków z bazy danych:

```
def plot_mnist(images, titles, h, w, n_row=3, n_col=4):
    plt.figure(figsize=(1.8 * n_col, 2.4 * n_row))
    plt.subplots_adjust(bottom=0, left=.01, right=.99, top=.90, hspace=.05)
    for i in range(n_row * n_col):
        plt.subplot(n_row, n_col, i + 1)
        plt.imshow(images[i].reshape((h, w)).T, cmap=plt.cm.gray)
        plt.title(titles[i], size=12)
        plt.xticks(())
        plt.yticks(())
```

Zad 3. Proszę wyświetlić przykładowe cyfry ze zbioru danych (funkcja `plot_mnist`).
Zad 4. Proszę podzielić zbiór danych na uczący (70 %) i treningowy.
Zad 5. Proszę stworzyć instancję klasyfikatora, następnie uczenie oraz predykcja dla danych testowych.
Parametry drzewa:

```
DEPTH = 10
```

Zad 6. Proszę przedstawić wynik F1, macierz błędów (confusion matrix) oraz raport klasyfikacji.