Wielowarstwowy perceptron - klasyfikacja i regresja

Joanna Kołodziejczyk

1 Wprowadzenie - cel zajęć

Celem zajęć jest wykorzystanie MLPClassifier and MLPRegressor z biblioteki scikit-learn. Wykonane zostaną dwa przykłady demonstrujące zadanie klasyfikacji na zbiorze diabetes i regresji dla sztucznego zbioru generującego funkcję $sin()$ i $cos()$.

1.1 Biblioteki

Konieczne jest załadowanie bibliotek.

Listing 1: Konieczne biblioteki

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
```

2 Klasyfikacja binarna

Do badania wykorzystany zostanie zbiór „Diabetes Data Set", którego opis i źródło można znaleźć na stronie `https://www.kaggle.com/uciml/pima-indians-diabetes-database`.

2.1 Wczytanie i analiza zbioru

Pracując z danymi dokonujemy analizy, liczby wejść, wyjść, typu atrybutu, istnienia brakujących danych i przeprowadzamy analizy statystyczne rozkładów wartości atrybutów w zbiorze. Kolejność wykonanych zadań:

1. wczytać dane;
2. obliczyć statystyki, wykorzystać je do oceny zbioru;
3. sprawdzić czy są puste wartości w rekordach;

4. sporządzić wykres korelacji pomiędzy atrybutami – ocenić siłę zależności z atrybutem wyjściowym lub innym i w tej sytuacji podjąć decyzję o ich wykorzystaniu lub nie;

5. sporządzić histogramy dla każdego atrybutu (lub wykresy pudełkowe) – określić skośność rozkładu – dla sieci neuronowej jest to ważna informacja (aby zmniejszyć skośność rozkładu można zastosować przekształcenie logarytmiczne).

Listing 2: Wczytanie i analiza danych

```
df = pd.read_csv('../diabetes.csv') #Wczytanie zbioru z odpowiedniej ścieżki
print(df.sample(5)) #Wyświetlenie wylosowanych 5 rekordów
stats = df.describe() #Obliczenie statystyk deskrypcyjnych i przechowanie struktury DataFrame w zmiennej stats

#sprawdzić czy są puste rekordy
if df.isnull().sum().sum() <= 0:
    print("Nie ma pustych")

# sporządzić wykres korelacji
plt.figure(figsize = (5,5))
sns.heatmap(df.corr(), annot=True) % annot- zapisz wartość w każdej komórce

# wykres prezentujący histogram i funkcję gęstości oraz mediany dla atrybutu w rozkładzie zmiennej decyzyjnej dla zmiennej pregnencies
plt.figure(figsize = (5,5))
sns.distplot(df[df['Outcome']==0]['Pregnancies'],bins=10,kde_kws={'label':'Zdrowa'})
sns.distplot(df[df['Outcome']==1]['Pregnancies'],bins=10,kde_kws={'label':'Cukrzyca'})
plt.title('Liczba ciąż wśród badanych')
plt.axvline(np.median(df[df['Outcome']==0]['Pregnancies']),color='blue',linestyle='—')
plt.axvline(np.median(df[df['Outcome']==1]['Pregnancies']),color='red',linestyle='—')
```

2.2 ZeroR

ZeroR jest najprostszą metodą klasyfikacji, która przewiduje kategorię (klasę) najbardziej liczną. Chociaż ZeroR nie jest modelem predykcyjnym, jest on przydatny do określenia podstawowej wydajności jako punkt odniesienia dla innych metod klasyfikacji.

Listing 3: ZeroR

```
#Określenie dokładności klasyfikatora ZeroR
accuZeroR =  df['Outcome'].value_counts()/len(df)
if accuZeroR[0] > accuZeroR[1]:
    print("Skuteczność klasyfikatora ZeroR to %1.3f" % (accuZeroR[0]))
else:
    print("Skuteczność klasyfikatora ZeroR to %1.3f" % accuZeroR[1])
```

2.3 Uruchomienie klasyfikacji bez normalizacji

Pierwsza próba uczenia zostanie podjęta dla wejść ze zbioru, bez normalizacji.

Listing 4: MLP z wejściami bez normalizacji

```
#Zmienne X, y bez standaryzacji
y = df.iloc[:,-1] #Ostatnia kolumna jako zmienna decyzyjna
X = df.drop('Outcome', axis=1)  #Pozostałe kolumny jako wejścia

#Podział zbioru na testowy i uczący w proporcjach 30% do 70%
# random_state = parametr do inicjalizacji generatora liczb losowych
x_train, x_test, y_train, y_test = train_test_split(X,y, test_size= 0.30,
    random_state=0)

#Utworzenie modelu neuronowego przy wybranych cechach uczenia:
# Rozmiar sieci 3 warstwy ukryte po 100 neuronów każda
# f. aktywacji logistyczna
# max_liczba_iteracji (epoki) 500
# wartość alpha  regularyzacji L2 (Ridge regression) 0.0001
# algorytm uczenia sgd – stochastic gradient descent
# współczynnik uczenia
# czy pokazywać progres uczenia
# inicjalizacja generatora liczba losowych
# tolerancja – jeżeli funkcja straty nie poprawia się o wartość większą niż
    tol to przerwać uczenie
model = MLPClassifier(hidden_layer_sizes=(100,100,100),
        activation='logistic',
        max_iter=500,
        alpha=0.0001,
        solver='sgd',
        learning_rate_init = 0.1,
        verbose=False,
        random_state=0,
        tol=0.000000001)
```

2.4 Analiza wyników

Ocena jakości klasyfikatora może być wynikiem analizy zmiany w funkcji straty. Można podać macierz pomyłek i ocenić błąd uczenia i błąd testu.

Listing 5: Ocena modelu

```
# Uruchomienie procesu uczenia
model.fit(x_train, y_train)
# Obliczenie wyjść przy zadanych wejściach w zbiorze uczącym
y_pred_train = model.predict(x_train)
# Obliczenie wyjść przy zadanych wejściach w zbiorze testowym
y_pred = model.predict(x_test)

#Porównanie dokładności modelu na zbiorze uczącym i testującym
print("Skuteczność klasyfikatora dla danych uczących to %1.3f" %
    accuracy_score(y_train, y_pred_train))
print("Skuteczność klasyfikatora dla danych testowych to %1.3f" %
    accuracy_score(y_test, y_pred))
```

```
11
12 #wykres błędu w  krokach uczenia
13 plt.figure(figsize = (5,5))
14 plt.plot(model.loss_curve_)
15 plt.title('Zmiana wartości funkcji straty w procesie uczenia')
16 plt.xlabel("liczba korków")
17 plt.ylabel( "wartość funkcji błędu")
18
19 #Macierz pomyłek dla testu
20 cm = confusion_matrix(y_test, y_pred)
21 print(cm)
22
23 plt.figure(figsize = (5,5))
24 sns.heatmap(cm, center=True)
25 plt.show()
```

2.5 Dodanie normalizacji

Sieci neuronowe są bardzo wrażliwe na jakość danych wejściowych oraz ich wartość. Zapewnienie lepszej zbieżność modelu zapewni normalizacja wejść (sprowadzenia wartości do ten samej dziedziny). Należy podmienić odpowiedni fragment kodu:

Listing 6: Dodane normalizacji dla wejść

```
1 std = StandardScaler()                    # wykorzystanie metody standaryzacji
2 X = pd.DataFrame(std.fit_transform(df.iloc[:,:-1]),columns=df.iloc[:,:-1].
      columns)
```

3 Regresja liniowa

Zadaniem jest wykorzystanie wielowarstwowej sieci neuronowej do regresji funkcji. Dane są wygenerowane syntetycznie. Założenie jest takie, że zbiór posiada jedno wejście (współrzędną X), a na wyjściu mamy dwie współrzędne (Y1 i Y2), które odpowiadają odpowiednio wartościom funkcji $sin(X)$ i $cos(X)$.

Listing 7: Wielowyjściowa regresja

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.neural_network import MLPRegressor
4 from sklearn.metrics import r2_score
5
6 # Generowanie wejścia i wyjścia sin i cos z przedzialu 0..2pi
7 X =  np.arange(0,2*np.pi,0.02)
8 Y = np.array([np.sin(X),np.cos(X)])
9
10 plt.figure(figsize=[10,5])
11 plt.subplot(1,3,1)
12 plt.plot(X, Y[0], 'r')
13 plt.plot(X, Y[1], 'b')
14 plt.title('Wykres wygenerowanych punktów')
```

```
15
16
17 #Utworzenie danych wejściowych w odpowiednim formacie
18 x_train = X.reshape(315,1)
19 y_train = Y.transpose()
20 #Utowrzenie modelu z zadanymi parametrami
21
22 model = MLPRegressor(hidden_layer_sizes=(100),
23     activation='logistic',
24     max_iter=10000,
25     alpha=0.00000001,
26     solver='sgd',
27     learning_rate_init=0.05,
28     verbose=False,
29     random_state=0,
30     tol=0.00001)
31
32 # Uczenie modelu
33 model.fit(x_train,y_train)
34
35 #Obliczenie wartości wyjścia z sieci generowanych przez model dla zadanych
       wartości x
36 y_pred = model.predict(x_train)
37
38 # Obliczenie błędu popełnianego przez model regresyjny
39 error = r2_score(y_train, y_pred)
40 print("Błąd r2 policzony pomiędzy wartościami oczekiwanymi i wyjściami z
       sieci %1.3f" % error)
41
42 #wykres błędu w  krokach uczenia
43 plt.subplot(1,3,2)
44 plt.plot(model.loss_curve_)
45 plt.title('Zmiana wartości funkcji straty w procesie uczenia')
46 plt.xlabel("liczba korków")
47 plt.ylabel( "wartość funkcji błędu")
48
49 plt.subplot(1,3,3)
50 plt.plot(X, y_pred[:,0], 'r')
51 plt.plot(X, y_pred[:,1], 'b')
52 plt.title('Wykres punktów generowanych przez sieć neuronową')
53
54 plt.show()
```

4 Sprawozdanie

Wykonać sprawozdanie składające się z dwóch części:

4.1 Klasyfikacja z użyciem MLP

1. Do zadania wykorzystać zbiór załączony do instrukcji.
2. Zbiór przeanalizować i opisać jego charakterystykę.

3. Celem badania jest określenie najlepszego z przebadanych zestawu parametrów MLP:
   (a) architektury sieci neuronowej (hidden_layer_size = $(i_1, i_2, i_3, ...)$ np. (2,1)) domyślnie=(100,). Element i-ty reprezentuje liczbę neuronów w i-ej ukrytej warstwie. Należy zaczynać od małych wartości, by model nie był przeuczony;
   (b) algorytmu uczenia (solver : {'lbfgs', 'sgd', 'adam'}, default='adam'});
   (c) funkcji aktywacji (activation : {'identity', 'logistic', 'tanh', 'relu'}, default='relu' });
   (d) wartości współczynnika uczenia ($learning\_rate\_init \in (0, 1]$) (nie ma zastosowania przy uczeniu metodą lbfgs);
   (e) wartości $alpha \in (0, 1]$ - współczynnika regularyzacji;
   (f) liczba iteracji, default=200, (max_iter sugerowane wartości wielokrotności 100).
4. Jako wynik należy podać przebadane zestawy i uzyskaną dokładność modelu (accuracy) na danych uczących i na danych testowych.
5. Przedstawić wyniki w postaci wykresu/diagramu.
6. Opisać wnioski z eksperymentu.
7. Konstruując wnioski proszę pamiętać o istotności statystycznej w uzyskanych wynikach. Opisać wpływ parametrów na procesu uczenia i jakość modelu.

4.1.1 Komentarz do techniki doboru parametrów sieci w obu zadaniach.

Poniżej zaprezentowano dwie techniki doboru parametrów. W sprawozdaniu trzeba się posłużyć jedną z nich. Opisać w sprawozdaniu, jaka technika została wybrana i dlaczego.

Metoda doboru współczynników może być oparta na systematycznej zmianie parametrów z wybranych arbitralnie przedziałów, bądź zbiorów wartości, np. zmieniać współczynnik uczenia learning_rate_init od 0.1 do 1 co 0.2, w tym czasie zamrażając inne parametry. Dobór liczby neuronów można wykonać metodą addytywną. Zacząć od małej sieci i powiększać ją aż do osiągnięcia dobrego wyniku. Należy pamiętać, że sieć może mieć wiele warstw ukrytych, nie tylko jedną.

Można do tego zadania wykorzystać metodę sklearn.model_selection.GridSearchCV `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV`

Alternatywna technika badania zestawu parametrów polega na losowaniu zestawu z wybranej przestrzeni wszystkich możliwych wartości. Wykonanie powtórek wielokrotnie i wybranie z niego najlepszego zestawu. Można do tego zadania wykorzystać metodę sklearn.model_selection.RandomizedSearchCV `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html#sklearn.model_selection.RandomizedSearchCV`

4.2 Regresja z użyciem MLP

1. Do zadania wykorzystać zbiór załączony do instrukcji
2. Zbiór przeanalizować i opisać jego charakterystykę.
3. Celem badania jest określenie najlepszego z przebadanych zestawu parametrów MLP
   (a) architektury sieci neuronowej (hidden_layer_size = $(i_1, i_2, i_3, ...)$ np. (2,1)) domyślnie=(100,). Element i-ty reprezentuje liczbę neuronów w i-ej ukrytej warstwie;
   (b) algorytmu uczenia (solver : {'lbfgs', 'sgd', 'adam'}, default='adam'});
   (c) funkcji aktywacji (activation : {'identity', 'logistic', 'tanh', 'relu'}, default='relu' });
   (d) wartości współczynnika uczenia ($learning\_rate\_init \in (0, 1]$) (nie ma zastosowania przy uczeniu metodą lbfgs);
   (e) wartości $alpha \in (0, 1]$ - współczynnika regularyzacji;
   (f) liczba iteracji, default=200 (max_iter sugerowane wartości wielokrotności 100).
4. Jako wynik należy podać przebadane zestawy i uzyskaną wartość współczynnika determinacji R^2 (error) na danych uczących i na danych testowych (w przykładzie z zajęć nie ma podziału na train i test trzeba to wykonać samodzielnie). (UWAGA: Najlepszy możliwy wynik to 1.0. Może być ujemny - bardzo źle). Model stały uzyskałby R^2 równy 0.0. Do poczytania: `https://www.statystyka-zadania.pl/wspolczynniki-zbieznosci-i-determina`
5. Przedstawić wyniki w postaci wykresu/diagramu.
6. Opisać wnioski z eksperymentu.
7. Konstruując wnioski proszę pamiętać o istotności statystycznej w uzyskanych wynikach. Opisać wpływ parametrów na procesu uczenia i jakość modelu.

5 Wyniki

Sprawozdanie i kod wykorzystany do badania proszę zamieścić w Teams. Sprawozdanie może być przygotowane w formie notatki w Jupyter Notebook (format .ipynb). Proszę pamiętać o używaniu w nazwach plików swojego nazwiska.