Maquina de soporte vectorial OnevsOne

Plata Salinas Eidan Owen

November 8, 2023

Introducción

El machine learning, o aprendizaje automático, es una rama de la inteligencia artificial que se centra en la construcción de sistemas que pueden aprender de datos. Dentro de este campo.

La regresión logística es un método estadístico utilizado para modelar y analizar conjuntos de datos en los que la variable de respuesta es categórica. Es especialmente útil cuando la variable de respuesta es binaria.

En este reporte, se explora una implementación de maquina de soporte vectorial 1vs1 para abordar un problema de clasificación de tres categorías. Para el data set se descargó uno de la pagina https://archive.ics.uci.edu/dataset/212/vertebral+column sobre columnas vertebrales, que se clasifican en 3 categorias, DH (Disk Hernia), Spondylolisthesis (SL), Normal (NO).

Desarrollo

A continuación, se describe el código Python que implementa la regresión logística en cascada. De modo que va comparando si pertenece a 1 clase o al resto, si pertenece al resto entonces agarra esa y la compara con las demas.

Explicación del Código

El código define una estructura de SVM en cascada y contiene funciones para manipular y procesar datos. Empecemos por partes:

Clase LinearSVM

Esta clase implementa un SVM lineal simple. SVM, o Máquina de Vectores de Soporte, es un algoritmo de aprendizaje supervisado utilizado para clasificación o regresión.

```
class LinearSVM:
def __init__(self, learning_rate=0.001, n_iterations=1000,
 lambda_param=0.1):
...
def fit(self, X, y):
...
def predict(self, X):
...
```

Clase CascadeSVM

La idea detrás de esta estructura es clasificar usando SVMs en cascada. Primero, se utiliza un SVM para separar la clase ‘DH‘ de las demás. Luego, otro SVM se entrena solo con los datos que no pertenecen a la clase ‘DH‘ para separar la clase ‘SL‘ de la clase ‘NO‘.

```
class CascadeSVM:
```

```
  def __init__(self, learning_rate=0.01, n_iterations=1000,
   lambda_param=0.1):
  ...
  def print_weights(self):
  ...
  def fit(self, X, y):
  ...
  def predict(self, X):
  ...
```

Funciones Auxiliares

El código tiene funciones para convertir archivos .dat a .csv, procesar datos y más.

```
  def dat_to_csv(dat_filename, csv_filename):
  ...
  def csv_datos(nombre_archivo):
  ...
  def process_csv(data: str):
  ...
```

Flujo Principal

El flujo principal del programa convierte un archivo .dat a .csv, procesa esos datos y entrena el modelo ‘CascadeSVM‘ para luego predecir las etiquetas en un conjunto de test.

```
  dat_filename = 'datata.dat'
  csv_filename = 'dataset.csv'
  dat_to_csv(dat_filename, csv_filename)
  ...
  clf_cascade = CascadeSVM(...)
  clf_cascade.fit(X_train, y_train)
  ...
  y_pred = clf_cascade.predict(X_test)
  accuracy = ...
```

1 Còdigo completo

```
1
2 import csv
3 import pandas as pd
4 from sklearn.model_selection import train_test_split
5 import numpy as np
6
7 class CascadeSVM:
8 def __init__(self, learning_rate=0.01, n_iterations=1000, lambda_param
    =0.1):
9 self.clf_dh = LinearSVM(learning_rate=learning_rate, n_iterations=
    n_iterations, lambda_param=lambda_param)
10 self.clf_sl = LinearSVM(learning_rate=learning_rate, n_iterations=
    n_iterations, lambda_param=lambda_param)
11
12 def print_weights(self):
13 print("Weights for DH Classifier:")
14 print("------------------------------")
15 for idx, w in enumerate(self.clf_dh.weights):
16 print(f"Feature {idx+1}: {w:.5f}")
17 print(f"Bias (DH Classifier): {self.clf_dh.bias:.5f}\n")
18
19 print("Weights for SL Classifier:")
20 print("------------------------------")
21 for idx, w in enumerate(self.clf_sl.weights):
22 print(f"Feature {idx+1}: {w:.5f}")
23 print(f"Bias (SL Classifier): {self.clf_sl.bias:.5f}")
24
25 def fit(self, X, y):
26 y_dh = np.where(y == 'DH', 1, -1)
27 self.clf_dh.fit(X, y_dh)
28
29 mask = y != 'DH'
30 X_sl = X[mask]
31 y_sl = np.where(y[mask] == 'SL', 1, -1)
32 self.clf_sl.fit(X_sl, y_sl)
33
34 def predict(self, X):
35 preds = []
36
37 dh_predictions = self.clf_dh.predict(X)
38 sl_predictions = self.clf_sl.predict(X)
39
40 for idx, (dh_pred, sl_pred) in enumerate(zip(dh_predictions,
    sl_predictions)):
41 if dh_pred == 1:
42 label = 'DH'
43 elif sl_pred == 1:
44 label = 'SL'
45 else:
46 label = 'NO'
47 preds.append(label)
48 print(f"Data {idx+1}: {X.iloc[idx].values} -> Predicted label: {label}"
    )
49
50 return np.array(preds)
```

```
51
52 class LinearSVM:
53 def __init__(self, learning_rate=0.001, n_iterations=1000, lambda_param
    =0.1):
54 self.learning_rate = learning_rate
55 self.n_iterations = n_iterations
56 self.lambda_param = lambda_param
57 self.weights = None
58 self.bias = None
59
60 def fit(self, X, y):
61 n_samples, n_features = X.shape
62 self.weights = np.zeros(n_features)
63 self.bias = 0
64
65 for _ in range(self.n_iterations):
66 for idx, x_i in enumerate(X.values):
67 condition = y[idx] * (np.dot(x_i, self.weights) - self.bias) >= 1
68 if condition:
69 dw = 2 * self.lambda_param * self.weights
70 db = 0
71 else:
72 dw = 2 * self.lambda_param * self.weights - np.dot(x_i, y[idx])
73 db = y[idx]
74 self.weights -= self.learning_rate * dw
75 self.bias -= self.learning_rate * db
76
77 def predict(self, X):
78 linear_output = np.dot(X, self.weights) - self.bias
79 return np.sign(linear_output)
80
81 def dat_to_csv(dat_filename, csv_filename):
82 with open(dat_filename, 'r') as dat_file:
83 lines = dat_file.readlines()
84 with open(csv_filename, 'w', newline='') as csv_file:
85 writer = csv.writer(csv_file, delimiter=',')
86 for line in lines:
87 data = line.split()[:]
88 writer.writerow(data)
89
90 def csv_datos(nombre_archivo):
91 datos = []
92 with open(nombre_archivo, newline='') as archivo_csv:
93 lector_csv = csv.reader(archivo_csv, delimiter=',')
94 for fila in lector_csv:
95 datos.append(','.join(fila))
96
97 csv_data = '\n'.join(datos)
98 return csv_data
99
100 def process_csv(data: str):
101 from io import StringIO
102 data_io = StringIO(data)
103 df = pd.read_csv(data_io, header=None)
104 X = df.iloc[:, :-1]
105 y = df.iloc[:, -1]
106 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, random_state=42)
```

```
107
108 return X_train, X_test, y_train, y_test
109
110 dat_filename = 'datata.dat'
111 csv_filename = 'dataset.csv'
112 dat_to_csv(dat_filename, csv_filename)
113 X_train, X_test, y_train, y_test = process_csv(csv_datos('dataset.csv')
    )
114 clf_cascade = CascadeSVM(learning_rate=0.01, n_iterations=1000,
    lambda_param=0.1)
115 clf_cascade.fit(X_train, y_train)
116 clf_cascade.print_weights()
117 y_pred = clf_cascade.predict(X_test)
118 accuracy = np.mean(y_pred == y_test)
119 print(f"Accuracy: {accuracy * 100:.2f}%")
```