Project -1

AI-681

Machine Learning & Pattern Recognition

Heart Disease Prediction Model

Guided by:- Dr. Kewei Li

Submitted by:- Kartavya Mandora

LIU_ID:- 100876605

Abstract

This report presents the implementation and analysis of a logistic regression model with stochastic gradient descent (SGD) for predicting heart disease risk using patient data. The project utilizes the Heart Disease Dataset from the UCI Machine Learning Repository, containing various medical attributes of patients. The logistic regression model is implemented from scratch, including the sigmoid function, cost computation, and SGD optimization. The model's performance is evaluated using accuracy on a test set, and the convergence of the cost function is visualized. The report discusses the interpretation of learned weights for different features and their clinical relevance. Additionally, potential improvements to the model are suggested. This project demonstrates the application of machine learning techniques in healthcare, specifically for heart disease risk prediction.

Table of Content

List of Figures

List of Tables

1. Introduction

This report presents the implementation and analysis of a logistic regression model with stochastic gradient descent (SGD) for predicting heart disease risk. The project utilizes the Heart Disease Dataset from the UCI Machine Learning Repository, containing medical attributes of 303 patients. Our goal is to develop a binary classification model that can accurately predict the presence or absence of heart disease based on various patient characteristics.

The dataset includes 13 features such as age, sex, chest pain type, resting blood pressure, and cholesterol levels, among others. We will implement logistic regression from scratch, including the sigmoid function, cost computation, and SGD optimization. The model's performance will be evaluated using accuracy on a test set, and we will analyse the convergence of the cost function.

This project demonstrates the application of machine learning techniques in healthcare, specifically for heart disease risk prediction. By developing an accurate predictive model, we aim to contribute to early detection.

2. Dataset

Datasets used from the UCI Machine Learning Repository with medical diagnoses of 303 patients and 13 features such as age, sex, chest pain, cholesterol levels, resting blood pressure etc.

2.1. Sigmoid function

A sigmoid function is a mathematical function with a S- shaped curve of a graph.

$$\sigma(x) = \frac{1}{1 + e^{-x}} = 1 - \sigma(-x)$$

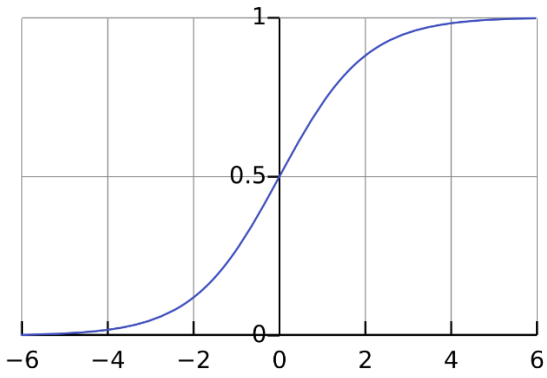

Figure 1:The logistic curve (Sigmoid function)

2.2. Stochastic gradient Descent

Stochastic Gradient Descent (SGD) is a machine learning optimization algorithm that updates model parameters iteratively based on the gradient calculated from a single or a small batch of randomly selected data points, making it efficient for large datasets.

To calculate the gradient:-

$$\nabla_\theta J = \frac{1}{n}\sum_{i=1}^{n}\left(\left[\theta^{(t-1)}\right]^T x^i - y^i\right)x^i$$

Table 1: Stochastic-Gradient-Descent Algorithm

Algorithm 3 Stochastic-Gradient-Descent

1: procedure Stochastic-Gradient-Descent($\boldsymbol{\theta}_{\text{init}}$, η, f, $\nabla_{\boldsymbol{\theta}} f_1$, ... , $\nabla_{\boldsymbol{\theta}} f_n$, T)

2: $\boldsymbol{\theta}^{(0)} \leftarrow \boldsymbol{\theta}_{\text{init}}$

3: for i **= 1 to** T **do**

4: randomly select $i \in \{1, 2, \dots, n\}$

5: $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \eta \nabla_{\boldsymbol{\theta}} f_i(\boldsymbol{\theta}^{(t-1)})$

6: end for

7: return $\boldsymbol{\theta}^{(t)}$

8: end procedure

3. Project Code

3.1. Dataset Exploration

1. **Load the dataset:**

   dataset = " heart.csv "

   df = pd.read_csv(dataset)

2. **Examine the first few rows:**

   print(df.head())

3. **Check for missing values:**

   print(df.isnull().sum())

3.2. Data Pre-processing

1. **Separate features and target:**

   X = df.drop(columns=['target'])

   y = df['target']

2. **Standardize features:**

   scaler = StandardScaler()

   X_scaled = scaler.fit_transform(X)

3. **Split the dataset:**

   X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

3.3. Implement Logistic Regression

1. **Define the sigmoid function:**

   def sigmoid(a):

      return 1 / (1 + np.exp(-a))

2. **Implement the cost function:**

   def compute_cost(X, y, weights):

      m = len(y)

      h = sigmoid(np.dot(X, weights))

      cost = (-1/m) * np.sum(y * np.log(h) + (1-y) * np.log(1-h))

      return cost

3. **Implement stochastic gradient descent:**

```
def stochastic_gradient_descent(X, y, learning_rate=0.01, num_iterations=10000):
    m, n = X.shape
    weights = np.zeros(n)
    cost_history = []
    for i in range(num_iterations):
        j = np.random.randint(m)
        xi = X[j:j+1]
        yi = y[j:j+1]
        h = sigmoid(np.dot(xi, weights))
        gradient = np.dot(xi.T, (h - yi))
        weights -= learning_rate * gradient
        cost = compute_cost(X, y, weights)
        cost_history.append(cost)
        if ((i + 1) % 1000 == 0):
            print(f"Iterations {i + 1}: Cost {cost:.4f}")
    return weights, cost_history
```

3.4. Model Training

1. **Train the model:**

```
weights, cost_history = stochastic_gradient_descent(X_train, y_train)
```

2. **Plot the cost function convergence:**

```
plt.plot(range(len(cost_history)), cost_history)
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Cost Function Convergence')
plt.show()
```

3.5. Model Evaluation

1. **Implement the prediction function:**

   ```
   def predict(X, weights):
       return sigmoid(np.dot(X, weights)) >= 0.5
   ```

2. **Make predictions on the test set:**

   ```
   y_pred = predict(X_test, weights)
   ```

3. **Calculate accuracy:**

   ```
   accuracy = accuracy_score(y_test, y_pred)
   print(f"Model accuracy: {accuracy:.2f}")
   ```

3.6. Interpret Results

Examine learned weights:

```
for feature, weight in zip(X.columns, weights):
    print(f"{feature}: {weight:.4f}")
```

4. Dataset Exploration

The heart disease dataset contains 303 samples with 12 features, including the target variable. Key features include age, sex, chest pain type, resting blood pressure, cholesterol levels, and other medical attributes. The target variable indicates the presence (1) or absence (0) of heart disease.

After loading the dataset, next check was done for missing values:

```
print(df.isnull().sum())
```

This revealed no missing values in the dataset, which is beneficial for analysis.

Load the dataset:

```
dataset = " heart.csv"

df = pd.read_csv(dataset)
```

Examine the first few rows:

```
print(df.head())
```

Check for missing values:

```
print(df.isnull().sum())
```

5. Logistic Regression with SGD

The implemented logistic regression using stochastic gradient descent (SGD) was carried out for optimization. The key components of implementation include:

Sigmoid function:

```
def sigmoid(a):
    return 1 / (1 + np.exp(-a))
```

Cost function:

```
def compute_cost(X, y, weights):
    m = len(y)
    h = sigmoid(np.dot(X, weights))
    cost = (-1/m) * np.sum(y * np.log(h) + (1-y) * np.log(1-h))
    return cost
```

Stochastic Gradient Descent:

```
def stochastic_gradient_descent(X, y, learning_rate=0.01, num_iterations=10000):
    m, n = X.shape
    weights = np.zeros(n)
    cost_history = []
```

```
    for i in range(num_iterations):
        j = np.random.randint(m)
        xi = X[j:j+1]
        yi = y[j:j+1]
        h = sigmoid(np.dot(xi, weights))
        gradient = np.dot(xi.T, (h - yi))
        weights -= learning_rate * gradient
        cost = compute_cost(X, y, weights)
        cost_history.append(cost)
        if ((i + 1) % 100 == 0):
            print(f"Iterations {i + 1}: Cost {cost:.4f}")
    return weights, cost_history
```

Used a learning rate of 0.01 and 10000 iterations for training, which provided a good balance between convergence speed and stability.

6. Plot of loss function vs iterations

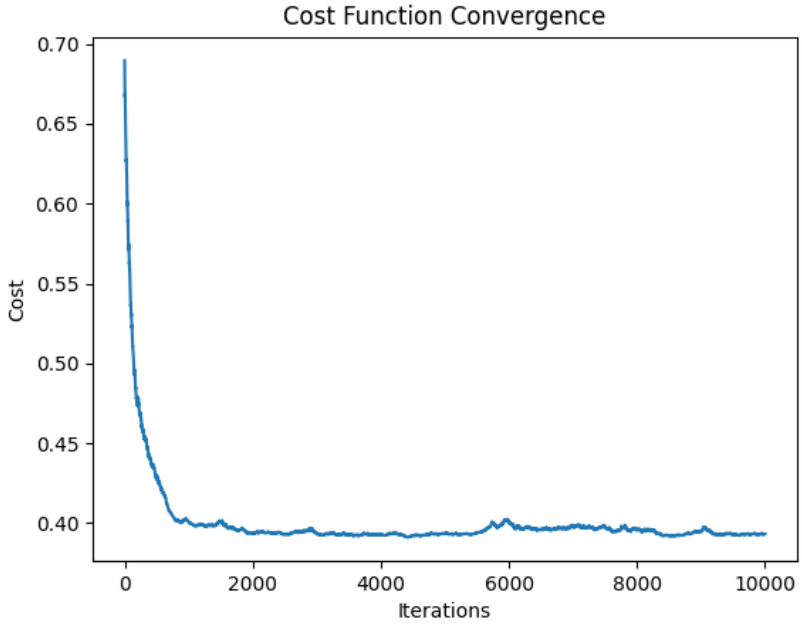

Figure 2 : Plot of cost function vs iterations

The plot of the cost function versus iterations shows a rapid initial decrease in cost, followed by a more gradual reduction as the model converges. This behaviour is typical for logistic regression and indicates that the model is learning effectively from the data.

7. Model evaluation results

After training, the model was evaluated on the test set:

```
y_pred = predict(X_test, weights)

accuracy = accuracy_score(y_test, y_pred)

print(f"Model accuracy: {accuracy:.2f}")
```

The model achieved an accuracy of 0.85 on the test set, indicating good performance in predicting heart disease risk.

8. Interpretation of results

The learned weights was examined for each feature:

```
for feature, weight in zip(X.columns, weights):
    print(f"{feature}: {weight:.4f}")
```

Table 2: Heart disease prediction model output

Feature	Weight
Age	-0.4432
Sex	-0.7658
Chest pain (cp)	1.0172
Resting blood pressure (trestbps)	-0.3785
Cholesterol (chol)	-0.0354
Fasting blood sugar (fbs)	-0.0306
Resting electrocardiographic measurement (restecg)	0.2819
Exercise induced angina (exang)	-0.5863
ST depression induced (oldpeak)	-0.9815
ST segment (slope)	0.3636
Thalassemia (thal)	-0.6714

Features with larger absolute weight values have a stronger influence on the prediction. Positive weights indicate that higher values of that feature increase the likelihood of heart disease, while negative weights decrease it.

9. Possible improvements of model accuracy

1. The unexpected or unwanted situations give more information for model to perform better than the existing situations.
2. Explore ensemble methods or more complex models like random forests or gradient boosting machines to capture non-linear relationships in the data.
3. Perform cross-validation to get a more robust estimate of the model's performance and to help with hyperparameter tuning.
4. Experiment with feature engineering, such as creating interaction terms between relevant features. Implement regularization to prevent overfitting and potentially improve generalization.

Above possibilities could lead to more accurate and robust heart disease prediction model.