Logistic_Regression

November 15, 2022

1 CSCC11 - Introduction to Machine Learning, Fall 2022, Assignment 2

1.1 Authors

Shawn Santhoshgeorge (1006094673)
Anaqi Amir Razif (1005813880)

1.2 Part 1: Written Component

1.2.1 Setup

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
# Import the Data and Setup X and y for Training
df_train = pd.DataFrame({
    "Width": [4, 6, 6, 6 , 6, 8 , 8],
    "Height": [4, 4, 5, 8, 10, 8, 10],
    "Orange": [1, 1, 1, 0, 0, 1, 0]
})

print(df_train)

# Split Data into X and Y
X_train = df_train[['Width', 'Height']].to_numpy()
y_train = df_train['Orange'].to_numpy()

w = np.asarray([0.3,-0.2, 0.7]) # Initial Weights
STEP_SIZE = 0.01
```

	Width	Height	Orange
0	4	4	1
1	6	4	1
2	6	5	1
3	6	8	0
4	6	10	0
5	8	8	1
6	8	10	0

1.2.2 Write the corresponding optimization problem in terms of the data provided above and specify the parameters to be estimated

The optimization problem we are trying to solve is the following

Model: $P(\text{Orange}|\mathbf{X}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{X}}}$, where $\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix}$ and $\mathbf{X} = \begin{bmatrix} x_1 \text{ (Weight)} \\ x_2 \text{ (Height)} \\ 1 \end{bmatrix}$

Given data $\{x_i, y_i\}_{i=1,\cdots,N}$. we minimize the negative log of

$$
\begin{aligned}
p(\{x_i, y_i\}|w) &\propto p(\{y_i\}|\{x_i\}, w) && \text{Assume} \{x_i, y_i\} \text{ are independent of } w \\
&= \prod_i^N p(y_i|x_i, w) && \text{Assume} \{x_i, y_i\} \text{ are independent} \\
&= \prod_{i:y_i=c_1}^N P(c_1|x_i) \prod_{i:y_i=c_2}^N (1 - P(c_1|x_i))
\end{aligned}
$$

Let $c_1 = 1$ (Orange) and $c_2 = 0$ (Not Orange), then the likelihood over N data points can be expressed as

$$p(\{x_i, y_i\} \propto \prod_{i=1}^N P(c_1|x_i)^{y_i}(1 - P(c_1|x_i))^{1-y_i}$$

To find the estimation for the model parameters we would want to minimize the negative log-likelihood as follows

$L(\mathbf{w}) = -\sum_{i=1}^{N} y_i \log(P(c_1|x_i)) + (1-y_i)\log(1-P(c_2|x_i))$

After taking the derivative of the negative likelihood we get the following $\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}) = -\sum_{i=1}^{N}(y_i - p_i)x_i$
where$(p_i \equiv g(w^T x))$ and g is the sigmoid function.

1.2.3 Perform 3 iterations of the steepest descent algorithm to determine the parameters assuming that the initial estimate is $[0.3, 0.2, 0.7]$ and the step size (λ) is 0.01. For each estimate (including the initial one), you are required to report the following:

- The value of the estimate
- The accuracy of the resulting logistic regression model when applied to the training data

[3]:
```python
def sigmoid(values):
    """
    Return the value from the Sigmoid Function

    Args:
        - values (ndarray (Shape: (N, 1))): Result of the Dot Product with Model Parameters and Input (w^Tx)

    Output:
        Values from the Sigmoid Function
    """

    'Checks if values is an array'
    assert isinstance(values, np.ndarray), 'values must be an ndarray of Nx1'

    return 1 / (1 + np.exp(-values))
```

[4]:
```python
def train(x, y, init_w, iters=3):
    """
    Finds the model parameter estimations using Gradient Descent

    Args:
        - x: (ndarray (Shape: (N, 3))): A Nx3 matrix corresponding to the inputs and 1's.
        - y: (ndarray (Shape: (N, 1))): A N-column vector corresponding to the outputs given the inputs.
        - init_w: (ndarray (Shape: (3, 1))): Initial Weights and Bias Term for the model
        - iters (int): Number of iterations for the Gradient Descent Algorithm (Default=3)

    Output:
        - w: (ndarray (Shape: (3, 1))): Estimated Weights and Bias Term for the model
    """

    # Add Column of 1's for the bias term
    X = np.hstack((x, np.ones((x.shape[0], 1))))

    # Creates a copy of the initial weights
    w = np.copy(init_w)

    # Calculates the gradient and moves the weight closer to the estimate
    for i in range(iters):
        deltaW = np.dot(X.T, (sigmoid(np.dot(X, w)) - y))
        w -= STEP_SIZE * deltaW
        print(f'Iteration {i + 1}:', w)
    return w

def predict(x, w) -> np.ndarray:
    """
    Returns predictions for new values

    Args:
        - x: (ndarray (Shape: (N, 3))): A Nx3 matrix corresponding to the inputs and 1's.
        - init_w: (ndarray (Shape: (3, 1))): Estimated Weights and Bias Term for the model

    Output:
        (ndarray (Shape: (3, 1))): Predictions either 1 or 0
    """

    # Add Column of 1's for the bias term
    X = np.hstack((x, np.ones((x.shape[0], 1))))

    return 1 * (sigmoid(np.dot(X, w)) >= 1/2)
```

```
# Model Parameter Optimization
print("Initial Model Parameters: ", w)
w = train(X_train, y_train, w)
print("\nAfter Optimization: ", w)

# Model Testing on Training Data
y_train_pred = predict(X_train, w)
print("\nTrain Data Result: ", y_train_pred)
print(f"Accuracy on Training Data: {100 * np.mean(y_train == y_train_pred)} %")
```

```
Initial Model Parameters:  [ 0.3 -0.2  0.7]
Iteration 1: [ 0.20477175 -0.35426438  0.68685387]
Iteration 2: [ 0.27913091 -0.3089443   0.69951893]
Iteration 3: [ 0.27208574 -0.35574855  0.69978802]

After Optimization:  [ 0.27208574 -0.35574855  0.69978802]

Train Data Result:  [1 1 1 0 0 1 0]
Accuracy on Training Data: 100.0 %
```

1.2.4 Classify the following data points using the model you obtained in part b:

- (3,3), (4, 10), (9, 8), and (9, 10).

[5]:
```
# Import the Data and Setup X for Testing
X = np.array([(3,3), (4, 10), (9, 8), (9, 10)])
y_pred = predict(X, w)
print("Test Data Result: ", y_pred)
```

```
Test Data Result:  [1 0 1 0]
```

Therefore, the new points fit into the following class

Width	Height	Orange
3	3	Yes
4	10	No
9	8	Yes
9	10	No

1.2.5 Discuss one advantage of Logistic Regression

One advantage of Logistic Regression is that it has fewer model parameters compared to another classifier like Gaussian Class Conditionals and Naive Bayes, this means the training phase will be relatively quick to compute and also for prediction.

1.2.6 Briefly explain whether Logistic Regression is discriminative or generative ?

Logistic Regression is a discriminative model since it does not attempt to model the complete probability of the training data instead it only attempts to model the conditional probability of the target output given the input, for in this case $P(\text{Orange}|X)$.