Weekly Report 1 - Logistic Regression

Ganji Varshitha
AI20BTECH11009

Introduction

Logistic regression is a misnomer since it is a classification algorithm. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

Algorithm

We try to fit the data into a curved 'S' like graph unlike the straight line graph in linear regression.
The name of the algorithm comes from the function which is used to transform its output to return a probability value which can then be mapped to two or more discrete classes.

We assume functional form of $P(Y|X)$ and estimate parameters of $P(Y|X)$ directly from training data.

Let $p_y(x;w)$ be our estimate of $P(Y|X)$, where $\mathbf{w}$ is a vector of adjustable parameters.

$$P(Y=1|X,\mathbf{w}) = \frac{1}{1+\exp\left(-\mathbf{w}^\top X\right)} \tag{1}$$

In case of binary class problem, log odds of y or logit transformation of y is a linear function of x.
Inverse of logit is the Logistic function which is sigmoid function for binary class problem.
We estimate $\mathbf{w}$ using maximum likelihood estimation.

$$\mathbf{w} = \arg\max_{\mathbf{w}} \sum_{l} \ln P(y^l|\mathbf{x}^l,\mathbf{w}) \tag{2}$$

This involves minimising negative likelihood which is convex.
We use gradient descent for the minimisation problem by updating the weights in the direction of gradient.

Key points

- It does not assume on $P(X|Y)$ and learns the parameters that maximises the conditional probability $P(Y|X)$.

- It is a discriminative model which draws boundaries in the data space and focusses on predicting the labels.
- After predicting the probabilities of the output belong to a class, we need to map the probabilities to classes by using the decision threshold.
- It is a linear classifier as the decision rule is hyperplane.
- Softmax activation is used for multi class logistic regression.

Figure 1: Algorithm for binary class gradient descent(Logistic Regression)

For $j = 0, \ldots, d$
    $w_j \leftarrow \text{rand}(-0.01, 0.01)$
Repeat
    For $j = 0, \ldots, d$
        $\Delta w_j \leftarrow 0$
    For $t = 1, \ldots, N$
        $o \leftarrow 0$
        For $j = 0, \ldots, d$
            $o \leftarrow o + w_j x_j^t$
        $y \leftarrow \text{sigmoid}(o)$
        For $j = 0, \ldots, d$
            $\Delta w_j \leftarrow \Delta w_j + (r^t - y) x_j^t$
    For $j = 0, \ldots, d$
        $w_j \leftarrow w_j + \eta \Delta w_j$
Until convergence

Figure 2: Algorithm for multi class gradient descent(Logistic Regression)

For $i = 1, \ldots, K$
  For $j = 0, \ldots, d$
    $w_{ij} \leftarrow \text{rand}(-0.01, 0.01)$
Repeat
  For $i = 1, \ldots, K$
    For $j = 0, \ldots, d$
      $\Delta w_{ij} \leftarrow 0$
  For $t = 1, \ldots, N$
    For $i = 1, \ldots, K$
      $o_i \leftarrow 0$
      For $j = 0, \ldots, d$
        $o_i \leftarrow o_i + w_{ij} x_j^t$
    For $i = 1, \ldots, K$
      $y_i \leftarrow \exp(o_i) / \sum_k \exp(o_k)$
    For $i = 1, \ldots, K$
      For $j = 0, \ldots, d$
        $\Delta w_{ij} \leftarrow \Delta w_{ij} + (r_i^t - y_i) x_j^t$
  For $i = 1, \ldots, K$
    For $j = 0, \ldots, d$
      $w_{ij} \leftarrow w_{ij} + \eta \Delta w_{ij}$
Until convergence

```
1
2  class LogisticRegression():
3    def __init__(self,w, learning_rate,num_epochs):
4      self.w = w
5      self.learning_rate = learning_rate
6      self.num_epochs = num_epochs
7
8    def sigmoid(self,x):
9      return 1/(1 + np.exp(-x))
10
11   def train(self,X, y):
12     n, m = X.shape
13
14
15     for epoch in range(self.num_epochs):
16       y_hat = [self.sigmoid(w@X[i]) for i in range(n)]
17       error = -(np.sum([y[i]*np.log(y_hat[i]) + (1-y[i])*np.log(1-y_hat
     [i]) for i in range(n)]))
18       dw = X.T@(y-y_hat)
19       self.w = self.w - self.learning_rate*dw
20     return y_hat, error, self.w
21
22   def predict(self,X, w):
23     y_pred = self.sigmoid(X@w)
24     y_class = [1 if i > 0.5 else 0 for i in y_pred]
25     return y_class
26
27   def accuracy(self,y_true, y_pred):
```

```
28      return np.mean([1 if (y_true[i]==y_pred[i]) else 0 for i in range(
    len(y_pred))])
```

Listing 1: Logistic Regression Code

Questions

1. Parameters of Logistic regression model are estimated using maximum likelihood estimation(MLE). State true or false.
   Solution: True

2. Logit function gives _____ of output variable.
   Solution: Probability

3. Does the probability sum equal to 1 for sigmoid function in Logistic regression?

   A. Yes, always.

   B. Not always.

   Solution: B. Not always

4. What can be done to avoid over-fitting in the model?
   Solution:
   Complex models often overfit the data. It can arise when there are many parameters relative to the dimensions of the input or if the input has many dimensions and is sparse.
   Regularization is one of the best ways to prevent overfitting. Since we use L2 norm in the bias term, we also call it L2 regularization. We penalize large values of $\mathbf{w}$ by modifying the log likelihood as follows:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \sum_{l} \ln P(y^l|\mathbf{x}^l, \mathbf{w}) - \frac{\lambda}{2}||\mathbf{w}||^2 \tag{3}$$

5. Which metric should not be used to compare the logistic regression model output to the target variable?
   A. AUC-ROC curve B. Logloss C. MSE
   Solution: C. MSE