Logistic Regression

Fuhao Zou(邹复好)

Intelligent and Embedded Computing Lab,
Huazhong University of Science & Technology

fuhao_zou@hust.edu.cn

2019年04月14日

Table of contents

Table of Contents

Basic idea:

- function

$$f(x) = \sum_{m=1}^{p} w_m x_{im} + w_0 = w^T x_i \tag{1}$$

- w_i are what the model learn through the samples

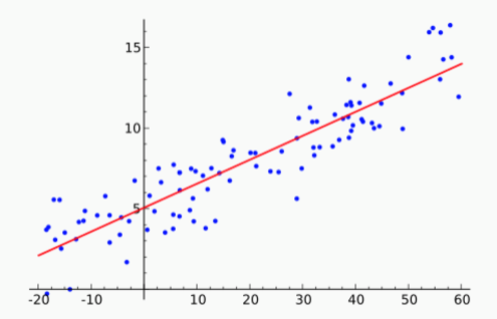


图: Linear regression is a linear approach to modelling the relationship between input X and output Y.

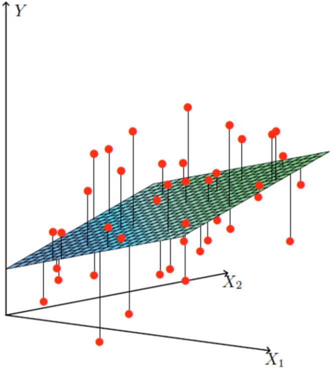
Loss Function:

Basic idea

- Loss function of Linear Regression-Least Square:

$$J(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 = \frac{1}{n}||y - X_w||^2 \tag{2}$$

- We seek the linear function of X that minimizes the sum of squared residuals from Y



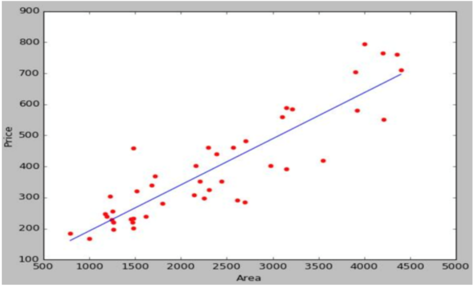
Python Code

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model
```

```
datasets_X = []
datasets_Y = []
fr = open('prices.txt','r')
lines = fr.readlines()
for line in lines:
    items = line.strip().split(',')
    datasets_X.append(int(items[0]))
    datasets_Y.append(int(items[1]))

length = len(datasets_X)
datasets_X = np.array(datasets_X).reshape([length,1])
datasets_Y = np.array(datasets_Y)

minX = min(datasets_X)
maxX = max(datasets_X)
X = np.arange(minX,maxX).reshape([-1,1])

linear = linear_model.LinearRegression()
linear.fit(datasets_X, datasets_Y)
```

```
plt.scatter(datasets_X, datasets_Y, color = 'red')
plt.plot(X, linear.predict(X), color = 'blue')
plt.xlabel('Area')
plt.ylabel('Price')
plt.show()
```

visualization



Outcome

Table of Contents

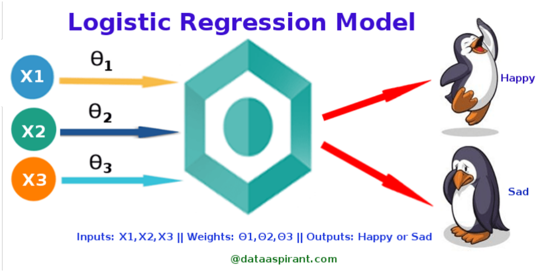


图: Logistic regression is a classic machine learning algorithom used for calssification task. As shown in the picture, we first feed the data, the inputs to the model, and the model gives us its classification result.

Basic idea:

- $f(Z) = h_\theta(x) = sigmoid(Z) = \frac{1}{1+e^{-Z}}$
- Output $=$ 0 or 1
- $Hypothesis \longrightarrow Z = W_x + B$

We usually attach the sigmod function to the end of the model, it gives us the probaility of the label being 1

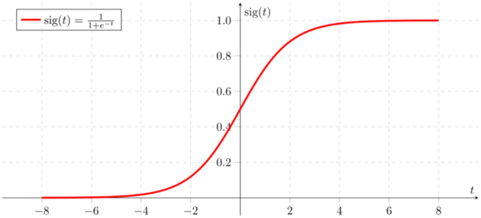


图: If ‘Z’ goes to infinity, Y(predicted) will become 1 and if ‘Z’ goes to negative infinity, Y(predicted) will become 0.

Loss Function

Predicted Probability:

$$P(y=1|x) = h\theta(x) = \frac{1}{1+e^(-\theta^T x)} = \delta(\theta^T x) \tag{3}$$

$$P(y=0|x) = 1 - P(y=1|x) = 1 - h_\theta(x) \tag{4}$$

Combined (3).(4):

$$P(y|x;0) = (h_\theta(x))^y (1-h_\theta(x))^{1-y} \tag{5}$$

y stands for the labels, being 0 or 1

Using maximum likelihood estimation(MLE) according to the m given samples:

$$L(\theta) = \prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta) = \prod_{i=1}^{m} (h_\theta(x^{(i)}))y^{(i)}(1-h_\theta(x^{(i)}))^{1-y^{(i)}} \tag{6}$$

$$\longrightarrow l(\theta) = \log L(\theta) = \sum_{i=1}^{m} (y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(y^{(i)}))) \tag{7}$$

$$J(\theta) = -\frac{1}{m} l(\theta)$$

$J(\theta)$ is the desired loss function

Basic idea

- L2

$$J(\theta) = \frac{1}{2m}[\sum_{i=1}^{m}(h_\theta(x^{(i)}))^2 + \lambda \sum_{j=1}^{n} \theta_j^2] \tag{8}$$

- Regularization : prevent the weights from getting too large
- Regularization can avoid overfitting to some extent through the restraint of weights

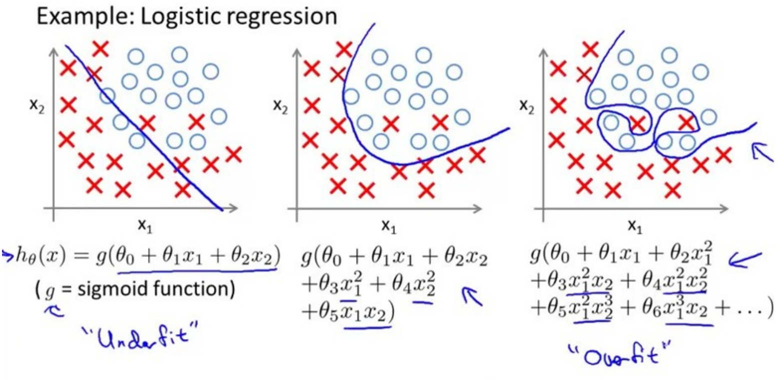
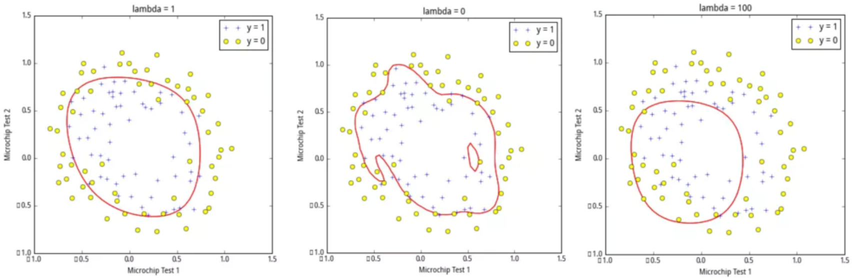


图: Underfitting/overfitting

Regularization

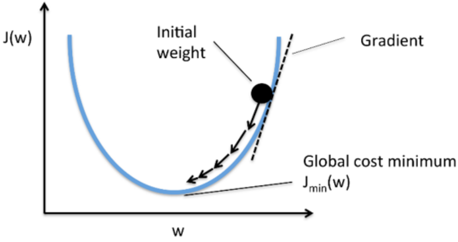
Classification task through Logistic Regression

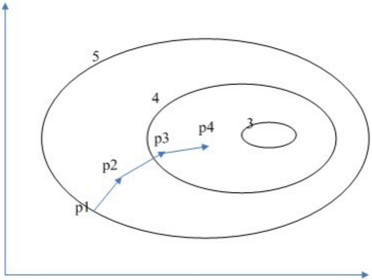


λ = 1 Good ! **λ = 0 Overfiting** **λ = 100 Underfiting**

图: Two-class classification when λ has different values

Gradient Descent





$$J(\theta) = -\sum_{i=1}^{m}(y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(y^{(i)}))) \tag{9}$$

$$\downarrow$$

Partial derivative:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_i x_j^{(i)}(h_\theta(x^{(i)}) - y^{(i)})$$

$$\downarrow$$

Weights update:

$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial J(\theta_j)}$$

Table of Contents

Basic idea

- Machine learning algorithms can be (roughly) categorized into two categories:
- Generative algorithms, that estimate $P(x_i, y)$ (often they model $P(x_i|y)$ and $P(y)$ separately).
- Discriminative algorithms, that model $P(y|x_i)$

1. The Naive Bayes algorithm is generative. $(p(y), p(x|y) \rightarrow p(y|x))$

$$P(y|x) = \frac{p(x,y)}{\prod_y p(x,y)} = \frac{p(y)p(x|y)}{\prod_y p(y)p(x|y)}$$

2. Logistic Regression is discriminative. (Gradient descent$\rightarrow w$)

$$P(y|x_i) = \frac{1}{1 + e^{y(w^T x_i + b)}}$$

The End