Chapter 1 Classification

1.1 Classification and Representation

1.1.1 Classification

Classification is to predict the varible y into discrete values, in other workds, an assignment to classify features into classes. For example, divide eamils by spam or not.

We can set a list of threshold of some parameters of hypothesis to devide the dataset into different classes.

But if we apply linear regression, some extreme data will have a nonnegligible effect on the hypothesis.

As we all know, the hypothesis usually gives a continuous value, while the classification problem gives discrete values. But apply hypothesis and map values into discrete values sometimes cannot work well.

Here we will focus on the binary classification which has a postive class and a negetive class.

1.1.2 Hypothesis Representation

A logistic regression model want it's $h_\theta(x) \in [0, 1]$. We could apply **Sigmoid Function**:

$$g(z) = \frac{1}{1 + e^{-z}}$$

It's like

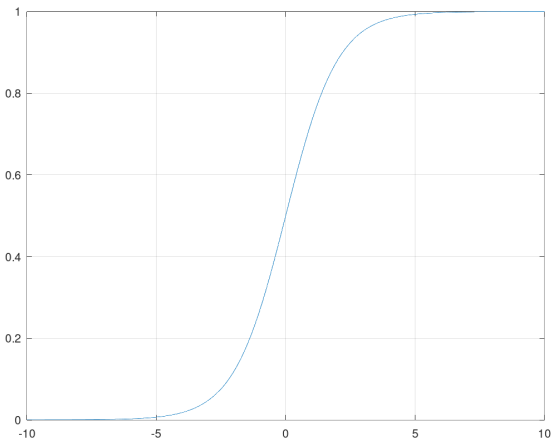

Figure 1.1: Sigmoid Function

Then, we get:

$$h_\theta(x) = g(\theta^T x)$$

Let's interprete the hypothesis' output: $h_\theta(x)$ is the estimated probability that $y = 1$ on input x. That is **probability that $y = 1$, given a x parameterized by θ**:

$$h_\theta(x) = P(y = 1|x; \theta)$$

1.1.3 Decision Boundary

Why Sigmoid Function makes sense? When $h_\theta(x) > 0.5$ we predict $y = 1$ and predict $y = 0$ in else condition. And it means that $\theta^T x > 0$ or not.

$$h_\theta(x) = g(\theta^T x)$$

By Sigmoid Function we can classify the hypothesis by it's values.

When we get 2 varibles, like $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ and we still make binary classification, wo we can still predict $y = 1$ if $h_\theta(x) \geq 0$ and $y = 0$ if $h_\theta(x) \leq 0$.

Further more, how about we need a non-linear decision boundaries, like a circle or a a ellipse? Just use non-linear functions! For a ellipse:

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

And we define the boundary like:

$$\begin{cases} y = 1, \text{ if } x_1^2 + x_2^2 \geq 1 \\ y = 0, \text{ else} \end{cases}$$

1.2 Logistic Regression Model

1.2.1 Cost Function

For a classification problem, we need the cost function, too. This is related to how to choose the parameters.

In chapters before, we defined as a sum of squared error:

$$\text{Cost}\left(h_\theta(x^{(i)}, y^{(i)})\right) = \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}, y^{(i)}))$$

For a simple cost function, it's non-convex so it does not guarantee to converge.

In logistic regression cost function, we define:

$$\text{Cost}\left(h_\theta(x), y\right) = \begin{cases} -\log(h_\theta(x)) \text{ if } y = 1 \\ -\log(1 - h_\theta(x)) \text{ if } y = 0 \end{cases}$$

If the hypothesis gives an approximate output as y, the cost is rather small; but when it goes to the other end, the cost will grow in a very fast speed.

And this cost function could be convex in our problem.

1.2.2 Simplified Cost Function and Gradient Descent

Let's re-write the cost function in a more compact way as:

$$\text{Cost}\left(h_\theta(x), y\right) = -y\log(h_\theta(x), y) - (1-y)\log(1-h_\theta(x))$$

$$\begin{aligned} J(\theta) &= \frac{1}{m}\sum_{i=1}^{m}\text{Cost}(h_\theta(x^{(i)}, y^{(i)}) \\ &= -\frac{1}{m}\left[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] \end{aligned}$$

To fit parameters θ, we need to minimize the $J(\theta)$. Then we can apply the gradient descent as in previous chapters.

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=0}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}, \text{where } \theta = 0, 1, 2, \cdots, n$$

Note: the $h_\theta(x)$ is different! Now we have $h_\theta(x) = 1/(1+e^{-\theta x})$.

1.2.3 Advanced Optimization

With this section, we can get logistic regression run more quickly. Like, gradient descent, conjugate gradient, BFGS[1], L-BFGS[2]

We can just simply call `fminunc` to get gradient descent.

1.3 Multi-class Classification

For multiclass problem, we can encode each class with some num like 1, 2, 3, 4 and so on. One-verse-all algorithm is to seperate different class to a special postive class in turn. Or: train a logistic regersss classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$. Finally, pick the $\max_i h_\theta^{(i)}(x)$.

1.4 Solve Overfitting: Regularization

1.4.1 The Problem of Overfitting

What's overfitting? If a little bit more complicate dataset cannot be fitted with a line, the linear fit will have a high preconception or bias(underfitting). If we apply a polynomial regression with too high order, hte learned hypothesis will fit the dataset very well but fall to have the ablity to generalize the problem(overfitting).

If we want to address the overfitting, we can

- reduce number of features
  - manually select features to keep
  - model selecttion algorithm
- aplly regularization

[1]Broyden-Fletcher-Goldfarb-Shanno algorithm,

[2]Limited-memory BFGS

- keep all features, but reduce magnitude or value of some parameters θ_j
- works well when get lots of features

1.4.2 Cost Function for Regularization

If we want to make some parameters really small, we can add some terms like:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^(i)) - y^{(i)})^2 + 1000\theta_3^2 + 1000\theta_4^2$$

Small parameters will turn to simpler hypothesis and get more smoth and less prone to overfit. If we want to shrink all parameters, the cost could be implemented as[3]:

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^(i)) - y^{(i)})^2 + \lambda\sum_{i=1}^{m}\theta_i^2\right]$$

The latter term is called regularization term. The regularization will put off a more general hypothesis.

1.4.3 Regularized Linear Regression

If we do gradient descent, the gradient will be different for θ_0 because the extra term does nothing with it.

$$\theta_j := \begin{cases} \theta_j - \alpha\frac{1}{m}\sum_{i=0}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j, \text{ where } \theta = 1, 2, \cdots, n \\ \theta_j - \alpha\frac{1}{m}\sum_{i=0}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}, \text{ where } \theta = 0 \end{cases}$$

Similarly, we can still apply the normal equation in conditions:

$$\theta = \left(X^TX + \lambda M\right)^{-1}X^Ty, \text{ where } M = \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$$

1.4.4 Regularized Logistic Regression

$$J(\theta) = -\frac{1}{m}\left[y^{(i)}\log(h_\theta(x^{(i)}), y^{(i)}) + (1 - y^{(i)})\log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\theta_i^2$$

[3]remeber that: θ starts with 0, but we just ignore the first term

$$\theta_j := \begin{cases} \theta_j - \alpha \dfrac{1}{m} \displaystyle\sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}, \text{ where } \theta = 1, 2, \cdots, n \\ \theta_j - \alpha \left[\dfrac{1}{m} \displaystyle\sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \dfrac{\lambda}{m} \theta_j \right], \text{ where } \theta = 0 \end{cases}$$

Words

ameliorate

preconception

contort

penalize

prone