Week 3

Arjun Sarao

March 17, 2021

1 Classification and Representation

Sometimes we want to predict a small number of discrete values as opposed to continuous. For example a problem that has only 2 possible values (e.g. True or False) is called a *Binary Classification Problem*. The range of values would be $y\epsilon\{0,1\}$. We would call the 0 value the *Negative Class* and the 1 the *Positive Class*. We want to find a function such that: $0 \leq h_\theta(x) \leq 1$.

$$h_\theta(x) = \theta^T x \tag{1}$$

This is the linear regression function, to turn this into the hypothesis function for Classification we can:

$$h_\theta(x) = g(\theta^T x) \tag{2}$$

where,

$$g(z) = \frac{1}{1+e^{-z}} \tag{3}$$

giving us:

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \tag{4}$$

This is called the *Logistic Function* or the *Sigmoid Function*. Note that this new hypothesis functions gives us the **Probability** of whether a value is the positive or negative class. We can use the Simple formula $P(A) + P(A') = 1$ to determine the probability for or against a certain value.

2 Logistic Regression Model

Because of the non-linearity of the Logistic Regression function, we must create a new cost function. The new Cost function is:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) \tag{5}$$

where,

$$\begin{cases} Cost(h_\theta(x^{(i)}), y^{(i)}) = -\log(h_\theta(x^{(i)})), y = 1 \\ Cost(h_\theta(x^{(i)}), y^{(i)}) = -\log(1 - h_\theta(x^{(i)})), y = 0 \end{cases} \tag{6}$$

We can also write a more compact cost function that is not a piece-wise:

$$Cost(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x)) \tag{7}$$

This works because if $y = 0$, the first term becomes 0 because of $-y$, and if $y = 1$ the second term becomes 0 because of $(1 - y)$.
Plugging the new cost function gives us:

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] \tag{8}$$

We will also use *Gradient Descent* to minimize θ.

$$\theta_J := \theta_J - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \tag{9}$$

Note that $h_\theta(x^{(i)})$ has changed from $\theta^T x$ to $\frac{1}{1+e^{\theta^T x}}$.
We can also use more complex optimization strategies than Gradient Descent. Some include

- Conjugate Gradient
- Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS)
- Limited-memory BFGS (LBFGS)

These offer many advantages such as the ability to not need to pick a learning rate (α) and are faster than Gradient Descent. However, these algorithms are quite advanced and require a high level of math skills to understand.

3 Multiclass Classification: One vs. All

When we have a problem with more than one class, we cant use only one equation to get an answer. To solve these multiclass classification problems, we use a technique called *One vs. All*. This method essentially assigns one class as the positive class and all other classes as the negative class. It then computes the θ_J for one, then switches to another class and repeats for all classes. Mathematically:

$$h_\theta^{(i)}(x) = P(y = i|x; 0) \tag{10}$$

for all values of i. When we have to use these equations to find a real number, we just run all of them and pick whichever one gives us the largest value.

4 Solving the problem of Overfitting

Underfitting: when the model does not fit the training data well or has a high bias. *Overfitting*: when the model fits the training data too well or has a high variance. Both these models perform poorly in the real world.To solve overfitting we can:

1. Reduce the number of features
    - Manually ignore unimportant features
    - Model Selection Algorithm
2. Regularization
    - Keeps all features, but reduces magnitudes/ values of parameters θ_j
    - Works well with lots of features

We can implement Regularization using a modified cost function. For example lets say we have the parameters $\theta_0, \theta_n, \ldots, \theta_j$:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left[(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} (\theta_j)^2 \right] \tag{11}$$

Where λ is the *regularization parameter*. Keep in mind that if the regularization parameter is too large, then the algorithm will result in an underfitted model. Regularization also has uses in linear regresssion models. We must first alter the gradient descent function as follows:

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \tag{12}$$

$$\theta_j = \theta_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right] \tag{13}$$

The term with the lambda performs the regularization. With some manipulation it looks like:

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \tag{14}$$

For the above equation the term: $1 - \alpha \frac{\lambda}{m}$ will always be less than 1. You can see this as reducing the value of θ_j by some amount.The addition of regularizaion also changes the normal equation.

$$\theta = (X^T X + \lambda L)^{-} 1 X^T y \tag{15}$$

where:

$$L = \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \tag{16}$$

with dimensions $(n+1) \times (n+1)$. Our last section for week 3 will be the defining the cost function for regularized logistic regression:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 \tag{17}$$