

# MACHINE LEARNING

Ludovico Bessi

July 2018

## Contents

<b>1</b>	<b>Week 1</b>	<b>2</b>
1.1	Linear regression . . . . .	2
<b>2</b>	<b>Week 2</b>	<b>3</b>
2.1	Multivariate linear regression . . . . .	3
2.2	Alternative to Gradient descent: Normal equation . . . . .	3
<b>3</b>	<b>Week 3</b>	<b>3</b>
3.1	Linear classification . . . . .	3
3.2	Logistic regression model . . . . .	4
3.3	Multiclass classification . . . . .	4
3.4	Overfitting . . . . .	4
3.5	Regularized linear regression . . . . .	4
3.6	Regularized logistic regression . . . . .	5
<b>4</b>	<b>Week 4</b>	<b>5</b>
4.1	Introduction . . . . .	5
4.2	Model representation . . . . .	5
4.3	Examples and intuitions . . . . .	5
4.3.1	AND operator, OR operator . . . . .	5
4.4	XNOR operator . . . . .	5
4.5	Multiclass classification . . . . .	6
<b>5</b>	<b>Week 5</b>	<b>6</b>
5.1	Cost function . . . . .	6
5.2	Back propagation algorithm . . . . .	6
5.3	Error in back propagation . . . . .	6
5.4	Gradient checking . . . . .	6
5.5	random initialization . . . . .	7
5.6	Wrapping it up . . . . .	7

# 1 Week 1

## 1.1 Linear regression

Suppose we have this data: feet and price of some houses. We want to be able to predict the price of the next house knowing only the feet. The best way to do that is using linear regression:

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

We want to find  $\theta_i$  such that the line behaves well with the data. This is the same as minimizing the following function, called cost function:

$$J(\theta_0, \theta_1) = \frac{1}{2n} \sum_{i=1}^n (h_{\theta}(x_i) - y_i)^2$$

We can minimize that function using "Gradient descent". We need to repeat the following operation until convergence:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

This is the same as saying:

$$\theta_j := \theta_j - \alpha * \frac{1}{n} \sum_{i=1}^n (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

The intuition behind this is that at the minimum the second term goes to 0 as the derivative goes to 0.  $\alpha$  is called the learning rate, and it is decided by the user.

## 2 Week 2

### 2.1 Multivariate linear regression

Suppose now that we have multiple information about houses. Then, we want to find a function like this:

$$h_{\theta}(x) = \sum_{i=1}^n \theta_i x_i$$

We can vectorize that function, obtaining just:

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

The gradient descent method does not change. However, there is a way to speed things up. We can normalize the  $\theta_i$  to have them all in the same value range. It is helpful because we minimize the risk of oscillations between values. We just need to do the following:

$$x_i := \frac{x_i - \mu_i}{s_i}$$

Where  $\mu_i$  is the average and  $s_i$  is the standard deviation.

### 2.2 Alternative to Gradient descent: Normal equation

Gradient descent is an iterative method, it is good even when we have many features. However it is still iterative, meaning that in principle we may spend a lot of time computing if we choose the wrong  $\alpha$ .

We can then use the normal equation if  $n$  is not too big:

$$\theta = (X^T X)^{-1} X^T y$$

We need to be careful for two reasons: it may be the case that  $X^T X$  is not invertible, because there are linearly dependent features, or too many of them. Moreover, calculating  $\theta$  in this way costs  $O(n^3)$ .

## 3 Week 3

### 3.1 Linear classification

Suppose we get an e-mail. We want to understand whether or not it is spam. We need a function that takes as input all the features of said email, and outputs either 0 or 1. We use the sigmoid function to achieve this. Let  $g : \mathbf{R} \rightarrow (0, 1)$  be the function  $g(z) = \frac{1}{1+e^{-z}}$ . Then we can write  $h_{\theta}(x) = g(\theta^T x)$ . For example, if we have  $h_{\theta}(x) = 0.7$ , this means that we 70% of times the output will be 1. We define  $y = 1$  if  $h_{\theta}(x) \geq 0.5$  and  $y = 0$  otherwise.

### 3.2 Logistic regression model

It is evident that we cannot use the same cost function as the regression model. Instead, we need to use the following:

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

, with  $\text{Cost}(h_{\theta}(x^{(i)}), y^{(i)}) = -y \log(h_{\theta}(x)) - (1 - y) \log(1 - h_{\theta}(x))$ . We then have the vectorized equation, with  $h = g(X\theta)$ :

$$J(\theta) = \frac{1}{m} (-y^T \log(h) - (1 - y)^T \log(1 - h))$$

### 3.3 Multiclass classification

We now treat the case with  $y \in \{0, 1, 2, \dots, n\}$ . We divide the problem in  $n+1$  sub problems of binary classification, the prediction will be the maximum value of  $h_{\theta}(x)$ .

### 3.4 Overfitting

If we try to fit too many features, we end up with a function that cannot predict well new results. To avoid it, we need to choose which features to analyze or to reduce the magnitude of the parameters. Let's see this last point in more detail: we need to change the cost function to avoid overfitting. Suppose we have:

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

We do not want to ditch the last terms, but we want to reduce their influence. The new cost function will then be like this:

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

When the cost function goes to 0, inevitably the terms  $\theta_3$  and  $\theta_4$  must go to 0 as well. We can rewrite it with  $\lambda$  as regularization parameter:

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2$$

We must be careful with the choice of  $\lambda$ , if it is too large we could risk underfitting, moreover the Gradient descent method won't converge. If it is too little, we do not remove overfitting.

### 3.5 Regularized linear regression

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

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

### 3.6 Regularized logistic regression

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

## 4 Week 4

### 4.1 Introduction

In this section the main topic is neural networks. The focus is on understanding how to build them, with the help of some examples to build intuition.

### 4.2 Model representation

The input is a vector  $x_i$  of features.  $x_0$  is sometimes called the bias feature. We can still use the sigmoid activation function  $\frac{1}{1+e^{-x}}$ . Between the input and the output, there may be more layers, called hidden layers. The layers are "connected" using weights. Suppose we have just one hidden layer. We can find the first value of that vector as following:

$$a_1 = g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3)$$

Let's explain it a bit. To get the **first** value of the layer, we just multiplied the **first** row of the matrices of weights by the features. Observation: if layer 1 has 2 input nodes and layer 2 has 4 activation nodes, then our matrix  $\theta_{ij}$  has dimension 4x3.

### 4.3 Examples and intuitions

#### 4.3.1 AND operator, OR operator

Recall that  $a \text{ AND } b == \text{TRUE}$  if and only if both  $a$  and  $b$  are true. Then we construct a neural network as follows:

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} \rightarrow (g(z^{(2)})) \rightarrow h_{\theta}(x)$$

With  $\theta_{1,3} = [-30, 20, 20]$ , then our output will be positive if both  $x_1$  and  $x_2$  are 1. Remember that  $x_0$  is always fixed at 1.

By setting  $\theta_{1,3} = [-10, 20, 20]$  we get the OR operator.

### 4.4 XNOR operator

By combining OR, AND and XOR we can get the XNOR operator. XNOR will give 1 if  $x_1$  and  $x_2$  are both 0 or 1. To do so, we need to add another hidden layer:

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} a_1^{(2)} \\ a_2^{(2)} \end{pmatrix} \rightarrow \begin{pmatrix} a_1^{(2)} \\ a^{(3)} \end{pmatrix} \rightarrow h_\theta(x)$$

For the transition from the input layer to the first hidden layer, we use a matrix that combines AND and NOR:

$$\theta^{(1)} = \begin{pmatrix} -30 & 20 & 20 \\ 10 & -20 & -20 \end{pmatrix}$$

For the transition between second and third layer, we use the values for OR:

$$\theta^{(2)} = [-10, 20, 20]$$

## 4.5 Multiclass classification

The output of the neural network may as well be a vector and not just a single number. Suppose we have a photo of a vehicle and we need to tell if it is a car, a truck or a scooter. The output vector will have three numbers ranging from 0 to 1, that indicate the level of confidence of our claim.

## 5 Week 5

The main topic will be: How to train a Neural network?

### 5.1 Cost function

The cost function is similar to the logistic regression cost function, we just need to add a summation in the first term, because it needs to loop through all the output nodes. In the regularization term, we need to add two summations, because we must take into account multiple  $\theta$  matrices.

### 5.2 Back propagation algorithm

Backpropagation means: "minimizing cost function".

### 5.3 Error in back propagation

We call  $\delta$  the error in the backpropagation algorithm. It is calculated by multiplying the following  $a_i$  nodes by the corresponding  $\theta_{i,j}$ .

### 5.4 Gradient checking

The following formula is useful when it is needed to check whether or not the backpropagation algorithm is working fine:

$$\frac{\partial}{\partial \theta} J(\theta) \approx \frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon}$$

## 5.5 random initialization

Setting all theta's to 0 at the start will not work. It is needed to set those in a random range  $[-\epsilon, +\epsilon]$ .

## 5.6 Wrapping it up

The following steps are required to train a neural network:

- adawd
- asd