MACHINE LEARNING by Andrew NG

Ludovico Bessi

July 2018

Contents

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 knowking 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 θ_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. α is called the learnig 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 θ_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 μ_i is the average and s_i is the standard deviation.

2.2 Alternative to Gradient descent: Normal equation

Gradiend 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 α.

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 θ 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 $g : \mathbf{R} \to (0, 1)$ 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:

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

, with $Cost(h_\theta(x^{(i)}), y^{(i)}) = -ylog(h_\theta(x)) - (1-y)log(1-h_\theta(x))$ 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 $y \in \{0, 1, 2, ..., n\}$. 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 θ_3 and θ_4 must go to 0 as well. We can rewrite it with λ 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 choiche of λ, 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 $\frac{1}{1-e^{\theta^T x}}$. 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 weigths by the features. Osservaton: if layer 1 has 2 input nodes and layer 2 has 4 activation nodes, then our matrix θ_{ij} has dimension 4x3.

4.3 Examples and intuitions

4.3.1 AND operator, OR operator

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

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

With $\theta_{1,3} = [-302020]$, 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 θ matrices.

5.2 Back propagation algorithm

Backpropagation means: "minimizing cost function".

5.3 Error in back propagation

We call δ 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:

- Randomly initialize the weights
- Implement forward propagation to get $h_\theta(x)$ for all x_i
- Implement Cost function
- Using back propagation to compute partial derivatives
- Use gradient checking one time to be sure that back propagation is working properly.
- Use gradient descent or other built-in functions to minimize the cost function with the weights.

6 Week 6

The main topic is about understanding if a given neural network works properly. Moreover, How can we speed up the algorithm?

6.1 Evaluating hypothesis

An idea is to split our data: 70% will be training set and the remaining 30% will be the set test. Then we introduce a test error function:

$$err(h_\theta(x), y) = 1$$

if $h_\theta(x) \geq 0.5$ and $y = 0$ or the opposite.

6.2 Bias vs Varince

High bias = underfitting while High variance means overfitting.

High bias: J_{train} and J_{CV} very high.
High variance: $J_{CV} >> J_{train}$.

Having more data helps with high variance, while it does not help if we have high bias.

More training examples, smaller set of features and increasing λ helps with high variance.
Adding features and decreasing λ helps with high bias.

6.3 Week 7

Support vector machine is one of the most useful black box algorithm for machine learning. It uses different functions you can choose (kernels) that are handy when tracting similarity. Useful to perform feature scaling when using Gaussian kernel. n = # of featues, m = # of training examples.

if n *geqm*, use logistic regression or SVM without kernel. If n is small and m intermediate, use SVM with Gaussian kernel. If n is small and m is large, add more features then use logistic regression or SVM without kernel.

6.4 Week 8

The main topics will be clustering and principal component analysis, in the setting of unsupervised learning.

6.4.1 Clustering and K-mean algorithm

We have some data, and we need to classify it. To do so, we pick K centroids and we run K-mean algorithm to optimise their position. That is, to minimize the total distance between the K_i centroid and the cluster points. At first, we need to randomize the centroids. It is higly inneficient to just choose random points. Instead, we should pick as starting centroid a point of the cluster.

6.4.2 Data compression and Principal component analysis

Data compression means moving your data set from nD to mD, with m $<$ n. (Example: a line in $\mathbf{R}^2$ becomes point on the real line). Principal components analysis is deeply interconnected with data compression: if we have a scatter plot, then going to 1D may be tricky. We neeed to project in such a way to reduce the error. We do that by computing the covariance matrix, computing the eigenvectors and we multiply U by x, where U is [U,S,V] = svd(covariance). You pick k such that 99% of covariance is retained. PCA is not useful if you want to avoid overfitting, in that case it is best to use normalisation.

6.5 Week 9

The main topic will be anomaly detection and recommendation systems.

6.5.1 Anomaly detection

Self explanatory, makes use of Gaussian distribution.

6.5.2 recommendation systems

How a user will rate a new film given the previous ratings? How is it connected with other users ratings? We answer these questions by using gradient descent algorithm to minimize different cost functions.

6.6 Week 10

The main topic will be big data in the context of machine learning.

6.6.1 Batch gradient descent and stochastic gradient descent

Batch gradient descent computes the gradient using the whole dataset. This is great for convex, or relatively smooth error manifolds. In this case, we move somewhat directly towards an optimum solution, either local or global. Additionally, batch gradient descent, given an annealed learning rate, will eventually find the minimum located in it's basin of attraction.

The way I like to think of how SGD works is to imagine that I have one point that represents my input distribution. My model is attempting to learn that input distribution. Surrounding the input distribution is a shaded area that represents the input distributions of all of the possible minibatches I could sample. It's usually a fair assumption that the minibatch input distributions are close in proximity to the true input distribution. Batch gradient descent, at all steps, takes the steepest route to reach the true input distribution. SGD, on the other hand, chooses a random point within the shaded area, and takes the steepest route towards this point. At each iteration, though, it chooses a new point. The average of all of these steps will approximate the true input distribution, usually quite well.

6.7 Week 11

6.7.1 Application example: Photo OCR

Self-explanatory. Get image -¿ Text detection -¿ Character segmentation -¿ Character classification.