Deep Learning

Cheng Guan

VISION@OUC

July 15, 2018

1 Logistic Regression

During the learning of last week, we have learned $\hat{y} = \sigma\left(w^{\mathrm{T}} + b\right)$, where $\sigma(z) = \frac{1}{1+e^{-z}}$. So to learn parameters for model, we're given a training set of m training examples,given $\left\{\left(x^{(1)}, y^{(1)}\right), \cdots, \left(x^{(m)}, y^{(m)}\right)\right\}$, want $\hat{y}^{(i)} = y^{(i)}$. We want to find parameters w and b and get the outputs on the training set. Our prediction on training sample(i) is $\hat{y}^{(i)}$. We introduces a loss(error) function which measures the discrepancy between the prediction$(\hat{y}^{(i)})$ and the desired output $(y^{(i)}]$).Usually we can define loss function euqals to one half squared error of $\hat{y}$ and y as Eq.1 and it turns out that we can do like this.

$$L\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right) \tag{1}$$

However, in logistic regression people don't usually do this. Because when we come to learn the parameters, we will find that optimization problem which we talk about becomes non-convex.So we end up with optimization problem with multiple local optima and gradient descent may not find the global optimum.So in the logistic regression, we will actually define a different loss function that plays a similar role as squared error that will give us an optimization problem that is convex. What we use in logistic regression is Eq.2.

$$L\left(\hat{y}^{(i)}, y^{(i)}\right) = -\left(y^{(i)} \log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)\right) \tag{2}$$

We define cost function as shown in Eq.3 which is the average of the loss function of the entire training set. We are going to find the parameters w and b that minimize the overall cost function.

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)\right] \tag{3}$$

The loss function is applied to a single training example and the cost function is the the cost of the parameters. So in training logistic regression model, we should find the parameters w and b to minimize the overall cost function J.

2 Gradient Descent

2.1 Single Example on Logistic Regression

In this section, we introduce gradient descent algorithms which is used to train or learn the parameters w and b on training set. As shown in Fig. 1, the cost function $J(w, b)$ is the surface above these horizontal axes w and b, so the height of the surface represents the value of $J(w, b)$ at a certain point. What we want to do is to find the value of w and b that minimize the cost function. From the Fig. 1,

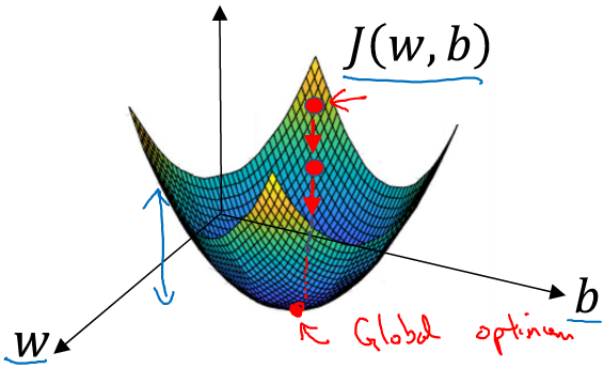

Figure 1: The cost function $J(w, b)$

we know that the cost function J is a convex function,which is not look like the non-convex function that has lots of different local optimum. And it is one of the reasons why we use this particular cost function J for logistic regression. What gradient descent does is that it starts at the initial point and takes step in the steepest downhill direction. So, after one step of gradient descent, that's one iteration of gradient descent. After two iterations of gradient descent, we might get another lower point. After three or more iterations of gradient descent, the curve eventually converge to the global optimum. So the Fig. 1 illustrates the gradient descent algorithm.

In the Fig. 2, we ignore b and mike this a one-dimensional plot instead of a high-dimensional plot. Gradient descent does like this. Repeat $w := w - \alpha \frac{dJ}{dw}$ and the α is the learning rate which can controls

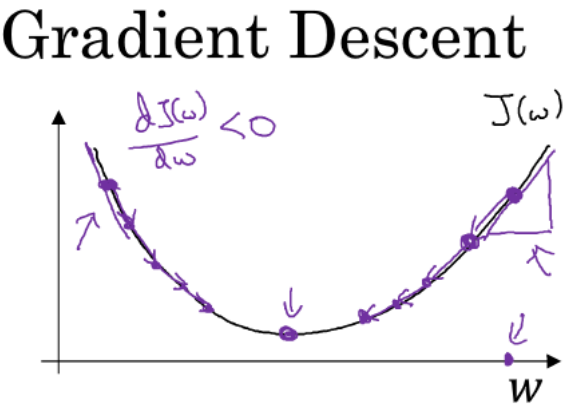

Figure 2: Gradient descent

how big a step we take on each iteration or gradient descent. We use := to represent updating w. We should make sure that gradient makes sense. Supposing that certain point in the right half of Fig. 2, the derivative of the point is positive. w gets new value which equals to $w - \alpha \frac{dJ}{dw}$. The derivative is positive and we slowly decrease the parameter w. Supposing that certain point in the left half, the derivative is positive and we slowly increase the parameter w.

So whether we initialize on the left or on the right, gradient descent will move toward to the global minimum. $J(w, b)$ is a function of both parameter w and b. So in this case, the inner loop of gradient decent should continue to update w and b as shown in Eq.4.

$$\begin{aligned} w &:= w - \alpha \frac{dJ(w,b)}{dw} \\ b &:= b - \alpha \frac{dJ(w,b)}{db} \end{aligned} \tag{4}$$

According to the above formulas we have learned. Next we dive into gradient descent for logistic regression. The predictions $\hat{y}$ is defined as Eq.5.

$$\hat{y} = a = \sigma(z) \tag{5}$$

Now we just focus on just one example and the loss function is defined as Eq.6.

$$L(a, y) = -(y \log(a) + (1 - y) \log(1 - a)) \tag{6}$$

a is the output of the logistic regression and y is truth label.Supposing that this example has two features x_1 and x_2. In order to compute z, we need input w_1, w_2 and b. These parameters in a computation graph are used to compute z which is equals to $w_1 \times x_1 + w_2 \times x_2 + b$ as shown in Fig. 3. What we want to is to modify parameters w and b in order to reduce the loss. In the back propagation, we should compute $\frac{dL}{da}$, $\frac{dL}{z}$ and $\frac{dL}{w}$. In the final step of back propagation, we need to compute how much we change parameters w and b. Compute $\frac{\partial L}{\partial w_1} = x_1 dz$, $\frac{\partial L}{\partial w_2} = x_2 dz$ and $\frac{dL}{db} = dz$ in turn and update w_1, w_2 with Eq.4.

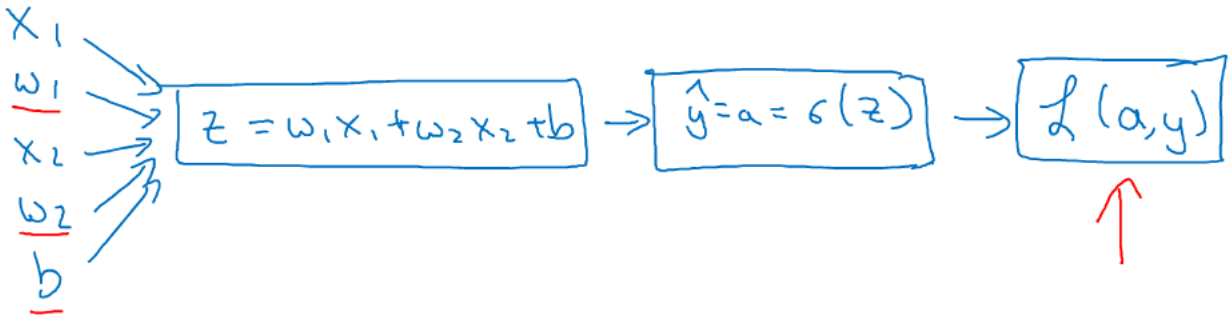

Figure 3: The computation graph

2.2 m Examples on Logistic Regression

Next step, we discuss m examples on logistic regression. According to the Eq.3, we know that overall cost functions with the sum was really the average of the 1 over m term of the individual losses. It turns out that the derivative of overall cost function to w is also the average of derivatives of the individual loss term to w. We have learned how to compute single training example and next what we need to do is really compute these own derivatives. And then, we write code to implement the updating of w_1 and w_2 and b with multiple steps of gradient descent.

3 Vectorization

Vectorization is the art of getting rid of explicit for loop in our code. Andrew Ng show a demo comparing the run time of for loop and vectorization. From the result, we know that vectorization can significantly speed up our code. So the rule of thumb is whenever possible, avoid using explicit for loop.

3.1 Vectorizing Logistic Regression

In order to carry out the forward propagation step, it need to compute these predictions on m training examples. We can use vectorization to compute instead of explicit for loop. We define X is a matrix of $n_x \times m$ and $Z = \left[z^{(1)} \cdots z^{(m)}\right] = w^{\mathrm{T}}X + [b \cdots b]$. According to vectorization, the numpy command is $Z = np.dot\,(w.T, X) + b$, b is a real number, but Python automatically takes this real number and expands a $1 \times m$ row vector. So, we just use one line code to calculate Z, which is a $1 \times m$ matrix that contains all of the $z^{(i)}, i \in (1, m)$. We define the Eq.7,

$$a^{(i)} = \sigma\left(z^{(i)}\right), i \in (1, m) \tag{7}$$

and define all of $a^{(i)}$ as a $1 \times m$ matrix A as shown in Eq.8, m training example for dZ, whose matrix is $1 \times m$, as shown in Eq.9, db is the Eq.10.

$$A = \left[a^{(1)}a^{(2)} \cdots a^{(m)}\right] = \sigma\,(Z) \tag{8}$$

$$dZ = A - Y \tag{9}$$

$$db = \frac{1}{m}\sum_{i=1}^{m} dz^{(i)} \tag{10}$$

The process of algorithm [1] of one step of gradient descent as shown in Table.1.

4 Broadcasting in Python

Broadcasting is another technique that we can use to make our Python code faster.

Prcoess of Algirithm with Python
import numpy as np
Z = np.dot(w.T,X) + b
A = sigmoid(Z)
dZ = A-Y
dw = 1/m*np.dot(X,dZ.T)
db = 1/m*np.sum(dZ)
w = w - alpha*dw
b = b - alpha*db

Table 1: Process of algorithm

References

[1] A.Ng. Neutral network and deep learning. http://mooc.study.163.com/. 3