Weekly Work Report

Zhang, Liqiang

VISION@OUC

July 21, 2018

1 Neural Networks Basics

1.1 Binary Classification

Binary classification is that its result is a discrete value output. Professor Wu give us a example, if we have an input of an image (Fig. 1) and want to output a label to recognize this image as either being a cat. This is an example of a classic binary classification classification.

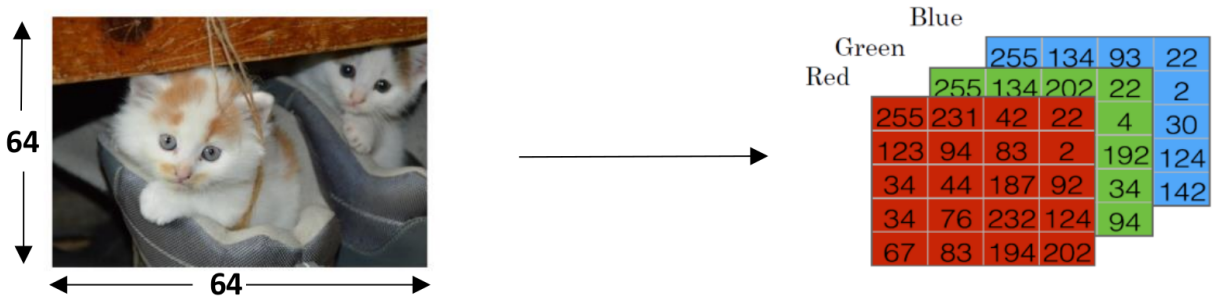

Figure 1: An example of binary classification

In the problem of binary classification, the goal is that we could get a classifier which can help us to predict the output of the input image is 0 or 1. In this class, Professor Wu show us an command: x.shape, which can gets the shape of the matrix.

1.2 Logistic Regression

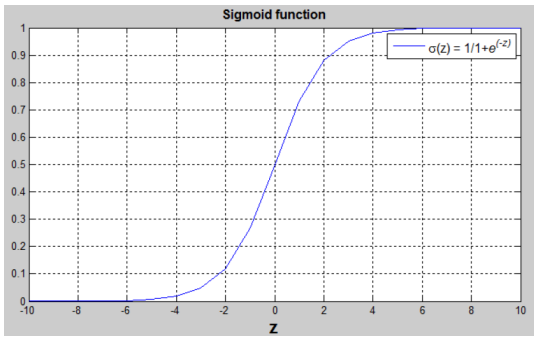

Figure 2: The image of sigmoid function

In this subsection, Professor Wu introduce the logistic regression. In the problem of binary classification, we want some algorithm which can give us a output called $\hat{y}$. The input x is a n_x dimensional vector and given that the parameters of logistic regression will be W which is also a n_x dimensional vector, together with b which is just a real number. If use the linear regression, the function is showed in Eq. 1:

$$\hat{y} = w^T x + b. \tag{1}$$

In fact, we want to let the value of $\hat{y}$ between 0 and 1. But it is hard to meet this requirement for this function because the value of $w_T x + b$ may be much bigger than 1 or a negative. Professor Wu use the sigmoid function as Eq. 2 to solve this problem. The image of $\sigma(z) = \frac{1}{1+e^{-z}}$ is showed as Fig. 2.

$$\hat{y} = \sigma(w^T x + b), where\ \sigma(z) = \frac{1}{1+e^{-z}}. \tag{2}$$

The advantage of this function is that the value of $\sigma(z)$ will be close to 1 if the value of $z = w^T x + b$ is very large, similarly, the value of $\sigma(z)$ will be close to 0 if the value of $z = w^T x + b$ is very small. So the value of $\hat{y}$ comply with our requirements and the goal of realizing logical regression is to find a appropriate value of w and b, which can make the value of $\hat{y}$ close to 1.

1.3 Logistic Regression Cost Function

To train the parameters W and B of the logistic regression model, a cost function will be used. Before knowing the cost function, Professor Wu first introduced the loss function to us. Loss funtion as Eq. 3 measures the discrepancy between the prediction $\hat{y}^{(i)}$ and the desired output $y^{(i)}$. In other words, the loss function computes the error for a single training example.

$$L(\hat{y}^{(i)}, y^{(i)}) = -(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}). \tag{3}$$

In Eq. 3 if $y^{(i)} = 1 : L(\hat{y}^{(i)}, y^{(i)}) = -\log(\hat{y}^{(i)})$ where $\log(\hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 1. If $y^{(i)} = 1 : L(\hat{y}^{(i)}, y^{(i)}) = -\log(1 - \hat{y}^{(i)})$ where $\log(1 - \hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 0. The loss function is a measure of the performance on a single sample and cost function measures how well we are doing an entire training set. The cost function as Eq. 4 is the average of the loss function of the entire training set.

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}[(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)})]. \tag{4}$$

1.4 Gradient Descent

In this subsection, I learn how to use gradient descent to adjust the w and b in the model, gradient descent can be expressed in Fig. 3.

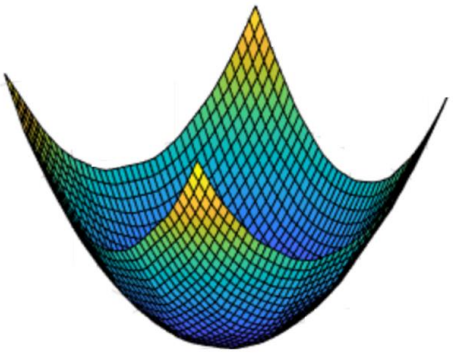

Figure 3: Gradient Descent

What gradient descent does is it starts at that initial point and then takes a step in the steepest downhill direction. The way to update the values of W and B is to repeat the Eq. 5.

$$\begin{aligned} w &:= w - \alpha\frac{dJ(w, b)}{dw} \\ b &:= b - \alpha\frac{dJ(w, b)}{db} \end{aligned} \tag{5}$$

where $:=$ represent iteration of w, α represent learning rate which can control the step size in the gradient descent method.

1.5 Vectorization

In the practical application of deep learning, we often train on relatively large data, we will need to take a long time if using For loop to realize it. So in this subsection, Professor Wu introduce the knowledge

about vectorization which is basically the art of getting rid of explicit folders in code. For example, in the function $z = wx + b$, where W was this column vector and X is also this vector. It maybe there are very large vectors if we have a lot of features. A vectorized implementation would just compute W transpose X directly. In Python, the command we use for that is $z = np.dot(w, x) + b$.

1.6 Vectorization Logistic Regression

Vectorization could let us speed up our code. Therefore, if we use the quantization to achieve the logical regression and let it be used to process the whole training set, the training time will be greatly reduced. In the forward propagation of logical regression, we should compute z used the function is that $y = \sigma(wX + b)$. If we have M training examples, then to make a prediction on the first example, you need to compute that, compute Z. In order to implement this, we could use the numpy command which is $np.dot(w.T, x) + b$, where $w.T$ is the transposition of the w and b is a real number. That is a very wonderful place in Python, if a number plus a vectorization Python will automatically extend this number to a 1*m dimensional vector, which is called broadcasting in Python.

1.7 Vectorization Logistic Regression's Gradient Output

In previous methods, we need the following methods as Eq. 6 to achieve gradient descent.

$$
\begin{aligned}
&J = 0, dw_1 = 0, dw_2 = 0, db = 0 \\
&for\ i = 1\ to\ m: \\
&\quad z^{(i)} = w^T \\
&\quad a^{(i)} = \sigma(z^{(i)}) \\
&\quad J+ = -[y^{(i)} log a^{(i)} + (1 - y^{(i)}) log(1 - a^{(i)})] \\
&\quad dz^{(i)} = a^{(i)} - y^{(i)} \\
&\quad dw_1+ = x_1^{(i)} dz^{(i)} \\
&\quad dw_2+ = x_2^{(i)} dz^{(i)} \\
&\quad db+ = dz^{(i)} \\
&\quad J = \frac{J}{m}, dw_1 = \frac{dw_1}{m}, dw_2 = \frac{dw_1}{m} \\
&\quad db = \frac{db}{m}
\end{aligned}
\tag{6}
$$

But if using the vectorization , We only need a few lines of code as Eq. 7 to complete the above tasks.

$$
\begin{aligned}
&z = w^T x + b = np.dot(w.T, x) + b \\
&A = \sigma(z) \\
&dz = A - Y \\
&dw = \frac{1}{m} x dz^T \\
&db = \frac{1}{m} np.sum(dz) \\
&w := w - \alpha dw \\
&b := b - \alpha db
\end{aligned}
\tag{7}
$$

But if we want to have a thousand deliberations of gradient descent, we might still need a full loop over the iteration number. However, vectorization has help us to reduce more computing time.

1.8 Broadcasting in Python

In this subsection, Professor Wu tell us the mechanism of broadcasting. For example, if we have a 4 by 1 vector and want to add it to a number, what Python will do is take this number and auto-expand it

into a four by one vector as well, as follows. And so the vector [1, 2, 3, 4] plus the number 100 ends up with that every element will plus 100. This is broadcasting in Python.

In fact, The parameter b in $z = w^T x + b$ is similar to the real number here, and this type of broadcasting works with both column vectors and row vectors. Another example is that if we have a m by matrix and we add it to a 1 by n as Fig .4. What Python will do is copy the matrix m times to turn this into m by n matrix, so instead of this one by three matrix it'll copy it twice in this example to turn it into this.

In summary, the more general principle of broadcasting in Python is that if we have an m by n matrix and you add or subtract or multiply or divide with a 1 by n matrix, then this will copy it n times into an m by n matrix. And then apply the addition, subtraction, and multiplication of division element wise.

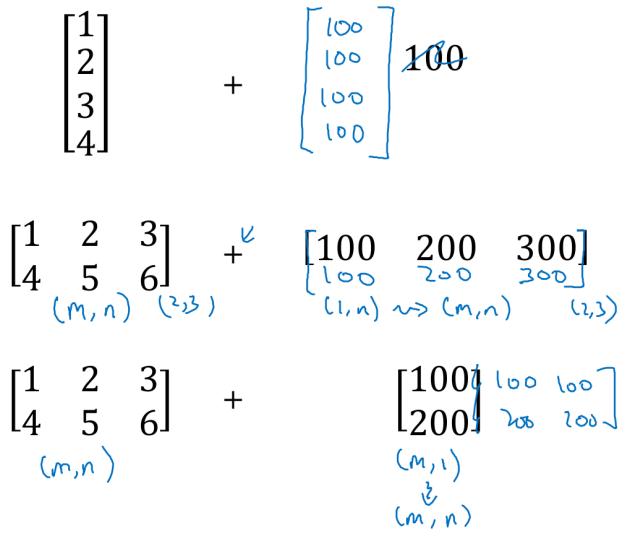

Figure 4: Some examples of the Python broadcasting

1.9 A Note on Python/Numpy Vectors

In this subsection, Professor Wu teach us some numpy command about broadcasting. The function of $np.random.randn()$ is producing random Gauss variables. If we use the command of $a = np.random.randn(5)$, which will produce 5 random Gauss variables stored in vector a. And so this is called a rank 1 array in Python and it's neither a row vector nor a column vector. Instead, if you set a to be $np.random.randn(5, 1)$, then this commits a to be (5,1) column vector.

Instead, every time we create an array, we should commit to making it either a column vector, so this creates a (5,1) vector, or commit to making it a row vector, then the behavior of our vectors may be easier to understand. One more important code is $a = a.reshape((5, 1))$ which can help us transform the dimension of the array.

2 Summarize

In this week, I get some knowledge about logistic regression and vectorization, vectorization does greatly increase the speed of algorithm. And I learned some simple commands in Python such as $np.dot(), np.sum(), np.random.randn()$ and so on, which are really effective compared with the display of the For loop. In the homework of this course, I have consolidated the knowledge I learned.