The Fourth Week Report

Lu Guorui

2018.7.22

Contents

[1] The source of package "pythonhighlight":https://github.com/chenfeng1234
latex-highlighting

Questions Remained

1 What is Forward and Back propagation?

In short, forward propagation is a way we used to count the value of compound functions. And back propagation is used to count the derivative of compound functions. We needn't complicate them.

2 How can we express the derivative of $\sigma(z)$?

By taking the derivative of $\sigma(z)$, we can easily get the result:$\sigma'(z) = \sigma(z)(1 - \sigma)$. Further, we can gain the consequence:

$$
\begin{aligned}
dz = \frac{\partial L}{\partial z} &= \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} && (1)\\
&= (-\frac{y}{a} + \frac{1-y}{1-1} \cdot a(1-a) && (2)\\
&= a - y && (3)
\end{aligned}
$$

$$
\begin{aligned}
dw = \frac{\partial L(w,b)}{\partial w} &= \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial w} && (4)\\
&= dz \cdot x && (5)\\
&= x(a-y) && (6)
\end{aligned}
$$

$$
\begin{aligned}
db = \frac{\partial L}{\partial b} &= \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial b} && (7)\\
&= 1 \cdot dz && (8)\\
&= a - y && (9)
\end{aligned}
$$

3 Some uses of numpy. [2]

1. Creat a vector:

```
x = np.array([...])
```

2. Call some frequently-used functions

```
np.exp(x)
np.log(x)
```

3. Get the shape of matrices

```
x.shape[0] # the number of rows
x.shape[1] # the number of column
```

4. Change the dimension

```
x.reshape((row,col))
```

5. Count the length of each row

```
x_norm = np.linalg.norm(x, axis=1, keepdims = True)
```

[2] The source of package "pythonhighlight":https://github.com/chenfeng123
latex-highlighting

6. Matrix-matrix or matrix-vector multiplication

```
# matrix multiplication:
# x1.shape = (a, m), x2.shape = (m, b)
np.dot(x1,x2)
# multiply the elements in corresponding locations
# x1.shape = (a, m), x2.shape = (a, m)
np.multiply(x1, x2)
```

7. Gain random numbers

- Get one number

```
n = numpy.random.random()
```

- Get a matrix

```
n = numpy.random.random(size=(3, 2))
```

- Generate random numbers between 0 and 1

```
np.random.rand(2,3) # (2,3) show the dimension
```

- Generate the same random numbers

```
'''
Set the same seed every time,
and we can get the same random numbers
'''
numpy.random.seed(num)
numpy.random.rand(the_length_of_the_array)
```

4 What do high bias and high variance mean?

High bias means that your model can't fit your train set well and high variance means your model overfit the train set so that it can't fit other data well.

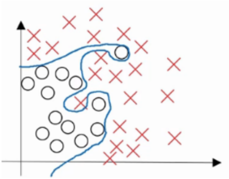

Figure 1: Overfitting

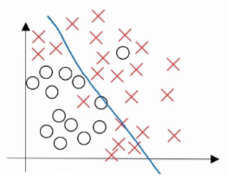

Figure 2: Underfitting

5 How does regularization prevent overfitting?

Regularization is able to reduce overfitting by adding the regularization term, $\frac{\lambda}{2m}||w||_2^2$. So the derivative $\frac{\partial L}{\partial W}$ turns into: $dW^{[l]} = (form\_backprop) + \frac{\lambda}{m}W^{[l]}$. Then we can get the update of grads as follows:

$$W^{[l]} := W^{[l]} - \alpha[(form\_backprop) + \frac{\lambda}{m}W^{[l]}] \quad (10)$$

$$= W^{[l]} - \alpha\frac{\lambda}{m}W^{[l]} - \alpha(form\_backprop) \quad (11)$$

$$= (1 - \frac{\alpha\lambda}{m})W^{[l]} - \alpha(form\_backprop) \quad (12)$$

If λ is big enough, the weight matrix W can be very small, and so does the Z. As the Figure 3 shows, when Z is relatively small, $tanh(Z)$ could be almost linear. We have known that when $g(x)$ is linear, no matter how deep the neural network is, what it counts is alaways a linear function which can't fit very complicate data.

Of course we need't keep Z so small all the time, but we can find a reasonable range that makes our model "just right".

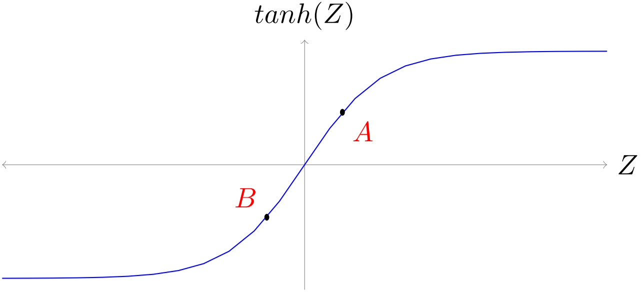

Figure 3: $tanh(Z)$ function