# 

[**3.6: Sigmoid Neuron and Cross Entropy**](#_oohhegi49883) **1**

[3.6.1: Sigmoid Neuron and Cross Entropy](#_h0e7h4t6jv8j) 1

[3.6.2: Using Cross Entropy With Sigmoid Neuron](#_9q2ljm0ecsc) 2

[3.6.3: Learning Algorithm for Cross Entropy Function](#_s1v994mucnuh) 3

[3.6.4: Computing Partial Derivatives With Cross Entropy Loss](#_3ku1pj59tit1) 4

[3.6.5: Code for Cross Entropy Loss Function](#_hihzy4aj3jdu) 5

# 

# Sigmoid Neuron and Cross Entropy

## Sigmoid Neuron and Cross Entropy

How does it all tie up to the Sigmoid Neuron

1. Consider the Example:
   1. A signboard with the text **Mumbai**
   2. A random variable X which maps the signboard to: Text, No-Text
   3. The distributions are as follows

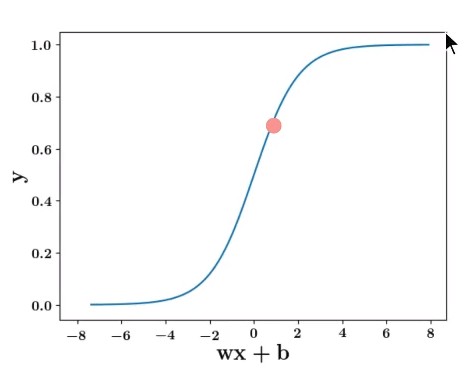
|  |  |  |
| --- | --- | --- |
| **X** | **y (We don’t know initially)** | **ŷ (Predicted using sigmoid)** |
| T | 1 | 0.7 |
| NT | 0 | 0.3 |

* 1. Previously, we were using **Squared-error Loss** =
  2. Now, we have a better metric, one that is grounded in probability theory (**KL- Divergence**)
  3. KLD(y||ŷ) =
  4. We aim to minimize loss by KLD with respect to the parameters w, b
  5. From KLD equation, we can see that yi doesn’t depend on w, b. So therefore, we are really only trying to minimize the first term, i.e. the cross-entropy
  6. So in practice, we can treat the second term as a constant, and the equation would really be
  7. The second terms cancels out and we are left with which is the same as
  8. It can be called where c can take the value 0 or 1 which correspond to NT and T

## 

## Using Cross Entropy With Sigmoid Neuron

What does the cross entropy loss function look like

1. Consider an example in the scope of our final project
2. Look at the following signboard 
3. (True distribution, where 1 corresponds to Text)
4. This corresponds to =0.7 
5. Thus, the predicted distribution is (where 0.7 corresponds to Text)
6. The Loss function is where
7. (from probability axioms, y0 = 1 - y1)
8. Consider two examples side by side

|  |  |  |  |
| --- | --- | --- | --- |
| Training Data | Image |  | Loss function |
| y = [0, 1]  ŷ = 0.7  ỹ = [0.3, 0.7]  (Text) |  |  | When true output is 1 |
| y = [1, 0]  ŷ = 0.2  ỹ = [0.8, 0.2]  (No-Text) |  |  | When true output is 0 |

1. The Loss function can be expressed as follows
   1. if y = 1
   2. if y = 0
   3. Combining them and removing the if conditions:
      1. When y = 1, the first term becomes 0
      2. When y = 0, the second term becomes 0

## Learning Algorithm for Cross Entropy Function

What is a more simplified way of writing the cross entropy loss function

1. From the previous step, we have
2. Cross entropy loss only makes sense for classification problems
3. The rest of the procedure is the same as the sigmoid neuron, except we use Cross-Entropy to minimize the loss and choose the best parameters w & b
4. **Initialise:** w, b randomly
5. **Iterate over data**
   1. Compute ŷ
   2. Compute L(w,b) (Where L is the cross-entropy loss function)
   3. wt+1 = wt - η𝚫wt
   4. bt+1 = bt + η𝚫bt
   5. Pytorch/Tensorflow have functions to compute
6. **Till satisfied**
   1. Number of epochs is reached ( ie 1000 passes/epochs)
   2. Continue till Loss < ε (some defined value)
   3. Continue till Loss(w,b)t+1 ≈ Loss(w,b)t

## 

## Computing Partial Derivatives With Cross Entropy Loss

How do we compute 𝚫w and 𝚫b

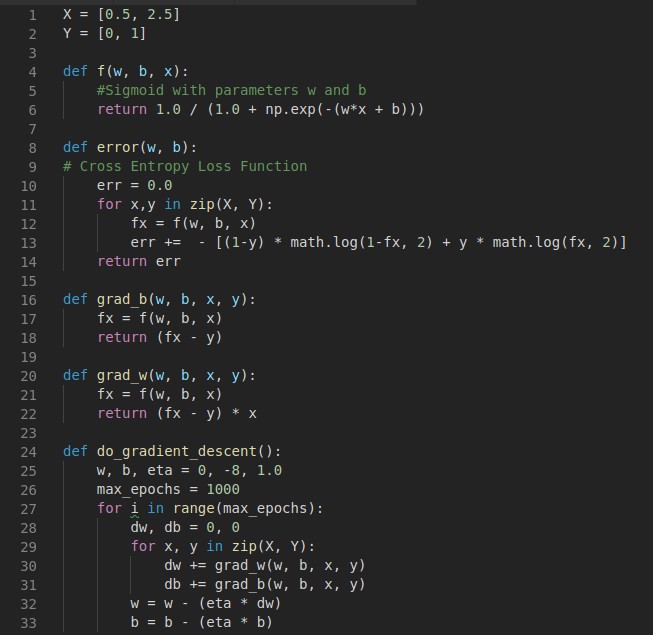
1. Loss Function
2. Consider 𝚫w for 1 training example

   2. The first part:
   3. The second part:
      1. This is the exact same as from the Squared Error Loss
   4. The final derivative is the first part multiplied with the second part
3. We then plug the values of 𝚫w and 𝚫b into the learning algorithm to optimise the parameters w and b.

## 

## Code for Cross Entropy Loss Function

What are the changes we need to make in the code

1. Here is the Python code for Gradient Descent with Cross Entropy Loss function
2. Here, the functions f(w, b, x), grad\_b(w,b,x,y) and grad\_w(w,b,x,y) have been changed to suit the Cross entropy loss function.

Fin