1. Truth table for XOR function is :

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Overallmatrix can be written as :

matrix can be written as :

Let weights and bias be:

Prediction:

MSE:

Gradient:

Computed weights and bias are:

Based on these weights, and bias the predictions are:

As shown above, the linear regression predicts output as 0.5 always.

1. Lets’ take an example of predicting if a person is sick or not based on their body temperature. And guidelines for this is given by: if the body temperature of a person is between 97-to-99-degree Fahrenheit, then the person is healthy (1), else the person is sick (0). Here 1, 0 are the binary classification.

Assume the following input data:

|  |  |
| --- | --- |
| Temperature | Health Status |
| 97.7 | 1 |
| 99.3 | 0 |
| 98.1 | 1 |
| 98.4 | 1 |
| 97.3 | 1 |
| 99.7 | 0 |
| 96.4 | 0 |
| 97.1 | 1 |
| 98.1 | 1 |
| 97.5 | 1 |

Training Set consists only positive examples:

|  |  |
| --- | --- |
| Temperature | Health Status |
| 97.7 | 1 |
| 98.1 | 1 |
| 98.4 | 1 |
| 97.3 | 1 |
| 97.1 | 1 |

Testing Data consists of both positive and negative examples

|  |  |
| --- | --- |
| Temperature | Health Status |
| 99.3 | 0 |
| 99.7 | 0 |
| 96.4 | 0 |
| 98.1 | 1 |
| 97.5 | 1 |

Assume we are initializing the weight and bias as (0,0).

Training first iteration:

Gradient:

Let’s update the weight and bias (let be 0.01) :

Let’s repeat this process for certain number of iterations.

2nd iteration:

Gradient:

Update weight and bias:

As seen from these two iterations, the weight and bias are going to stay same for all the upcoming iterations. This is because the gradient is going to remain zero as we have reached the minima.

But can this be the same if we initialize other weight and bias?

For example, start with weight as zero but bias as -100.

we are initializing the weight and bias as (0, -100).

Gradient:

Update weights and bias:

Second iteration:

Gradient:

Updates:

In the further iterations, as computed from python:

Graphical user interface

Description automatically generated with low confidence

A picture containing chart

Description automatically generated

As seen here, the weights and bias converged quickly.

What if we start with other set of weight, bias?

we are initializing the weight and bias as (-100, -100).

A picture containing graphical user interface

Description automatically generated

A picture containing graphical user interface

Description automatically generated

As seen here, the weights and bias converged quickly.

For w,b = (-100,0)

A picture containing shape

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Converged again!

What if we give different learning rate?

Graphical user interface

Description automatically generated with medium confidence

A picture containing graphical user interface

Description automatically generated

No problem again. This is the plot for alpha= 1 and (w,b) starting at (0,0)

1. **Convergence of b:**

If we initialize b as 0, gradient descent will adjust the value of b such that the logistic function produces probabilities close to 1 for all positive examples in the training set.

If we initialize b as -100, gradient descent will have to make much larger adjustments to reach the same solution as in the previous case.

1. **Convergence of w:**

If we initialize w as [0, 0, ..., 0], gradient descent will adjust the values of w such that the logistic function produces probabilities close to 1 for all positive examples in the training set.

If we initialize w as [1, 1, ..., 1], gradient descent will have to make much larger adjustments to reach the same solution as in the previous case.

1. **Training loss:**

Training loss will always converge zero. This is because both weight and bias are converging

(d) Testing loss:

The testing loss may not converge even if the training loss converges, as the model may not perform well on the test set, especially if the test set contains negative examples. If the model overfits to the training set, the testing loss may be high and may not converge. On the other hand, if the model generalizes well to the test set and makes accurate predictions on both positive and negative examples, the testing loss may converge to a minimum value.

1. Weights are given by the following plot:

A picture containing text, sofa, seat, blurry

Description automatically generated

Testing accuracy: 72.24 %

Unregularized Training Error: 0.40803586897774474

Unregularized Testing Error: 0.45120853186927046

Batch Size: 300.0

Learning Rate: 0.1

Epochs: 300.0

Regularization: 0.001