Training your neural network requires specifying an initial value of the weights. A well chosen initialization method will help learning.

If you completed the previous course of this specialization, you probably followed our instructions for weight initialization, and it has worked out so far. But how do you choose the initialization for a new neural network? In this notebook, you will see how different initializations lead to different results.

A well chosen initialization can:

* Speed up the convergence of gradient descent
* Increase the odds of gradient descent converging to a lower training (and generalization) error

0 Initialization

The model is predicting 0 for every example.

In general, initializing all the weights to zero results in the network failing to break symmetry. This means that every neuron in each layer will learn the same thing, and you might as well be training a neural network with n[l]=1n[l]=1 for every layer, and the network is no more powerful than a linear classifier such as logistic regression.

**What you should remember**:

* The weights W[l]W[l] should be initialized randomly to break symmetry.
* It is however okay to initialize the biases b[l]b[l] to zeros. Symmetry is still broken so long as W[l]W[l] is initialized randomly.

Random Big initialization

**Observations**:

* The cost starts very high. This is because with large random-valued weights, the last activation (sigmoid) outputs results that are very close to 0 or 1 for some examples, and when it gets that example wrong it incurs a very high loss for that example. Indeed, when log(a[3])=log(0)log⁡(a[3])=log⁡(0), the loss goes to infinity.
* Poor initialization can lead to vanishing/exploding gradients, which also slows down the optimization algorithm.
* If you train this network longer you will see better results, but initializing with overly large random numbers slows down the optimization.

**In summary**:

* Initializing weights to very large random values does not work well.
* Hopefully intializing with small random values does better. The important question is: how small should be these random values be? Lets find out in the next part!

You have seen three different types of initializations. For the same number of iterations and same hyperparameters the comparison is:

|  |  |  |
| --- | --- | --- |
| **Model** | **Train accuracy** | **Problem/Comment** |
| 3-layer NN with zeros initialization | 50% | fails to break symmetry |
| 3-layer NN with large random initialization | 83% | too large weights |
| 3-layer NN with He initialization | 99% | recommended method |

**What you should remember from this notebook**:

* Different initializations lead to different results
* Random initialization is used to break symmetry and make sure different hidden units can learn different things
* Don't intialize to values that are too large
* He initialization works well for networks with ReLU activations.