**Week 3 – Notes**

**Hyperparameter Tuning**

**Tuning Process**

The most important hyperparameters to tune are:

1. learning rate

2. mini-batch size, beta and the number of hidden units

3. number of layers and learning rate decay

Rarely you optimize beta 1, beta 2 or epsilon (of the optimizer)

Do not use a grid, instead use random values because some hyperparameter may not be that important and by using a random sampling technique, we test more hyperparameters values

The second technique is to go from coarse to fine, so for some hp values to zoom in the hp space

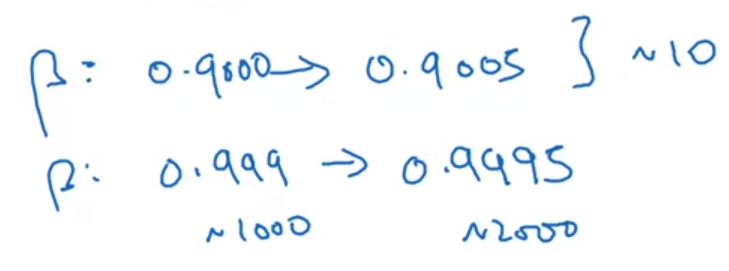
**Using an Appropriate Scale to pick Hyperparameters**

For some hp, the best sampling technique is to pick them by using a uniform distribution; e.g.: the number of neurons in a layer

For the learning rate, is better to pick the values by using a logarithm scale => explore the log space uniformly

Alpha: 0.0001 -> 1 => sample values of the form 10^r, where r is a random variable (picked uniformly) between -4 and 0

For exponentially weighted averages again pick beta = 1 – 10^r, where r is uniformly samples between a negative value and -1 => in this way we use the same number of resources to sample between 0.9 and 0.99 as we use to sample between 0.99 and 0.999

 -> in this way, these 2 intervals are samples uniformly, without letting out a lot of values between 0.999 and 0.9995

**Hyperparameters Tuning in Practice: Pandas vs. Caviar**

Intuition do get stale, so over re-evaluate the picked hyperparameters

There are two different approaches:

1. Babysitting one model => you deal with a huge data set / large model and do not have the resources to train several models => babysit one model: train it and over time to store various versions of the model and change the hyperparameters manually
2. Caviar => you have a lot of power / deal with small data sets or models => train lots of models and then pick the best one

**Batch Normalization**

**Normalizing Activations in a Network**

Makes the network much more robust to a larger spectrum of hp

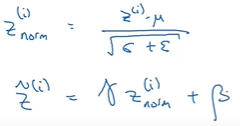
Enables the training of larger neural network

As you normalize the inputs, you normalize the inputs or the output of the activation functions

In practice is more common to normalize the inputs of the activation functions

For example, if you apply batch normalization for the layer 2, then the layer 3 will benefit from a faster training

Considering that we work with one layer [l], we take the z value of that layer for each training example i and compute the mean and the standard deviation; we normalize z for each example (i)



Because we don’t want for each normalized distribution to have the mean 1 and the standard deviation 0 (for example this won’t fit a sigmoid activation function), we use 2 learnable parameters gamma and beta to modify the mean and the standard deviation

