Hyperparameter tuning process

Usually the most important hyper parameter is α the learning rate. Second in importance are the number of hidden units, the mini-batch size and the β momentum term. Third in importance are the number of layers and the learning rate decay.

Which hyperparemeters to try? Try random values; don't use a grid search. Also it's common to use the "coarse to fine" sampling schema where you start by sampling randomly in a region, then pick a good subregion and continue sampling in smaller and smaller regions.

Use an appropiate scale to pick hyperparameters: Don't necessarily sample the hyperparameters randomly. For example if you are searching hyperparameters for the learning rate $\alpha \in [.001, 1]$ uniformly then 90% of the values you're trying lie between .1 and 1, it would be more appropiate to use a log scale and sample uniformly from using that scale so that you explore more than 10% between $[.001, .1]$.

Batch Normalization

Batch normalization makes the hyperparameter search problem much easier and the neural network trained much more robust. The choice of hyperparameters is a much bigger range of hyperparameters that work well, and will also enable you to much more easily train even very deep networks.

For any hidden layer, can we normalize the value $a^{[l]}$ to train $W^{[l+1]}, b^{[l+]}$ faster? That's the main idea of batch normalization, in the practice $z^{[l]}$ is the one that's normalized.

Given $z^{[l]} = (z^{(1)}, \ldots, z^{(m)})$:

$$\begin{aligned}
\mu_i &= \frac{1}{m}\sum_i z^{(i)} \\
\sigma_i^2 &= \frac{1}{m}\sum_i (z^{(i)} - \mu_i)^2 \\
z^{(i)}_{\text{norm}} &= \frac{z^{(i)} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \\
\hat{z}^{(i)} &= \gamma z^{(i)}_{\text{norm}} + \beta
\end{aligned}$$

Where γ and β are learnable parameters of the model that can modify the distribution of $z^{(i)}$ so that it lies in the range of values you need.

Batch normalization reduces the problem of the input values (of each layer) changing, it causes these values to become more stable, so that the later layers of the neural network has more firm ground to stand on.

Batch norm as Regularization Each mini-batch is scaled by the mean and variance computed on just that mini-batch, this adds some noice the values $z^{[l]}$ because it's just computed on one mini-batch. This noisy procedure is similar to dropout having a slight regularization effect.

Batch norm at test time During test time the estimations of μ and σ^2 are obtained using exponentially weighted averages (across mini-batches)

Multi-class classification

Think of the problem of classificating across multiple (C) classes, instead of just two as we've seen so far. This problem is called *multi-class classficiation*, for it we'll start to introduce a generalization of the logistic regression called softmax regression, softmax regression generalizes logistic regression to C classes.

The softmax activativation function is given by:

$$t = e^{(z^{[l]})}$$
$$a_i^{[l]} = \frac{t_i}{\sum_i t_i}$$

Training a softmax classifier, the loss function and cost functions are given by:

$$L(\hat{y}, y) = -\sum_{j=1}^{C} y_j \log \hat{y}_j$$
$$J(W, b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}, y)$$