Hyperparameter Tuning

1. α (Learning Rate)
2. Mini-batch Size, Beta (momentum), #hidden units
3. #layers, learning rate decay
4. Adam parameters

- When sampling hyperparameters to select the best don't use a grid system. As it might seem intuitive in the end it reduces the search space intuition of which parameter affects the metrics mre or less. Randomly selecting values in a range can provide much better variation and perception on which one is more important and how that one (or event the combination) varies on the selected range.
- Coarse to fine scheme: given an initial coarse sampling and a focused area of interest we can increase the sampling in that specific area.

Using an appropriate scale to pick hyperparameters

- number of layers, number of neurons in a given layer -> Good examples of randomly linear sampling working well
- learning rate, exponentially weighted averages betas -> better with logscale since in the linear scale much more resources would be spent on bigger variations than smaller variations. $r = -4 * np.random.rand(), alpha = 10^r$

Hyperparameters tuning in practice: Pandas vs. Caviar

- There might be cross information in other application domains
- As everything else changes (data, model, etc) the hyperparameters might change as well
- Babysitting model (Panda) x Parallel models (aCviar) -> It's about resources. If you don't have resources you focus everything you have on that model; if you have many resources available simply train them all.

Normalizing activations in a network

- Can we normalize $a^{[l]}$ to train $W^{[l+1]}, b^{[l+1]}$ faster? Instead of using $a^{[l]}$ is more common to use $Z^{[l]}$

Given some intermediate values in NN $z^{(1)}...z^{(n)} = Z^{[l](i)}$

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i z_i - \mu^2$$

$$z^{(i)}_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$\tilde{z}^{(i)} = \gamma z^{(i)} + \beta$ where γ and β are learnable parameters

- The γ and β values allow the batch to model any kind of distribution necessary to model and redistribute the data

Fitting Batch Norm into a neural network

- The Batch Norm (BN) step takes place between the computation of Z and a
- As we're changing the distribution adding any bias value has no meaning. The BN turn the mean to 0 and therefore has no reason to add any bias

for $t = 1 \ldots num_{mini-batches}$:

Compute forward-propagation on $X^{\{t\}}$

In each hidden layer use BN to replace $z^{(l)}$ to $\tilde{z}^{(i)}$

Use backpropagation to compute the gradients

Update the weights

Why does Batch Norm work?

The idea of your data distribution changing is called **covariate shift**. If you've learned some mapping X to Y and the distribution of X changes then you might need to retrain your algorithm. This is true even if the mapping remains unchanged.

As for an intermediary layer it receives its inputs trough the previous layer but it could also be isolated as a single layer receiving a overchanging input mapped to the same output always, therefore, the covariate shift explained above.

The batch norm helps the layers because it makes the intermediary values to become more stable so that the later layers of the network can do a better job learning how separate the data.

Since the parameters of the batch norm are also learned through the network it's easy for the network to decide by itself which normalization it should use (if any) on the given intermediary data representation.

Batch norm also has a normalization effect. As it's computed in the mini-batch the small sampled data it adds some noise to the $z^{[l]}$ values. Because of that it adds a slightly regularization effect. The noise added is proportional to the size of the mini-batch used, in other words, as the size of the mini batch increases the noise added is reduced and so is the normalization effect.

As the mean and variance is calculated on the mini batch there are some changes that must be done to ensure that on the inference everything will go accordingly.

Batch Norm at test time

A proposed solution is to use a exponentially weighted average on the mean and variation on the whole training set, one mini batch at time. And as the test is executed these estimated values are used.

Softmax Regression

- Generalization of logistic regression.
- Used to predict many outputs classes
- Softmax

$$t = e^{z^{[l]}}$$

$$a^{[l]} = \frac{e^{z^{[l]}}}{\sum_{j=1}^{\#classes} t_i}$$

$$a^{[l]} = \frac{t_i}{\sum_{j=1}^{\#classes} t_i}$$

- As the e^x value always output positive values the sum is always positive and can be used without a problem to compute \sum_{j=1}^{#classes}t_i
- The unusual part of this “activation” is that its input and output always have the same shapes, usually with a single row value input and a single row output representing the probabilities of each respective node over their previous values
- In a perfectly linear separable space with n classes a single softmax layer can separate them all given the correct weights on the Z calculation

Training a softmax classifier

- Softmax came from the “hard max” variant which simply puts a 1 on the highest probability value and 0 on all the others.

Quiz

1. If searching among a large number of hyperparameters, you should try values in a grid rather than random values, so that you can carry out the search more systematically and not rely on chance. True or False?
   - ☐ True
   - ☑ False
2. Every hyperparameter, if set poorly, can have a huge negative impact on training, and so all hyperparameters are about equally important to tune well. True or False?
   - ☐ True
   - ☑ False
3. During hyperparameter search, whether you try to babysit one model (“Panda” strategy) or train a lot of models in parallel (“Caviar”) is largely determined by:
   - ☐ Whether you use batch or mini-batch optimization
   - ☐ The presence of local minima (and saddle points) in your neural network
   - ☑ The amount of computational power you can access
   - ☐ The number of hyperparameters you have to tune
4. If you think \betaβ (hyperparameter for momentum) is between on 0.9 and 0.99, which of the following is the recommended way to sample a value for beta?
   - ☐

```
r = np.random.rand()
beta = r*0.09 + 0.9
```

- ☑
```
r = np.random.rand()
beta = 1-10**(- r - 1)
```

- ☐
```
r = np.random.rand()
beta = 1-10**(- r + 1)
```

- ☐
```
r = np.random.rand()
beta = r*0.9 + 0.09
```

5. Finding good hyperparameter values is very time-consuming. So typically you should do it once at the start of the project, and try to find very good hyperparameters so that you don't ever have to revisit tuning them again. True or false?
   - ☐ True
   - ☑ False
6. In batch normalization as presented in the videos, if you apply it on the llth layer of your neural network, what are you normalizing?
   - ☑ $z^{[l]}$
   - ☐ $a^{[l]}$
   - ☐ $W^{[l]}$
   - ☐ $b^{[l]}$
7. In the normalization formula $z^{(i)}_{norm} = \frac{z^{(i)}-\mu}{\sqrt{\sigma^2+\varepsilon}}$, why do we use epsilon?
   - ☐ To have a more accurate normalization
   - ☑ To avoid division by zero
   - ☐ To speed up convergence
   - ☐ In case μ is too small
8. Which of the following statements about γ and β in Batch Norm are true?
   - ☐ There is one global value of $\gamma \in R$ and one global value of $\beta \in R$ for each layer, and applies to all the hidden units in that layer.
   - ☑ They can be learned using Adam, Gradient descent with momentum, or RMSprop, not just with gradient descent.
   - ☑ They set the mean and variance of the linear variable $z^{[l]}$ of a given layer.
   - ☐ The optimal values are $\gamma = \sqrt{sigma^2 + \varepsilon}$ and $\beta = \mu$.
   - ☐ β and γ are hyperparameters of the algorithm, which we tune via random sampling. point
9. After training a neural network with Batch Norm, at test time, to evaluate the neural network on a new example you should:
   - ☐ If you implemented Batch Norm on mini-batches of (say) 256 examples, then to evaluate on one test example, duplicate that example 256 times so that you're working with a mini-batch the same size as during training.
   - ☐ Skip the step where you normalize using μ and σ^2 since a single test example cannot be normalized.
   - ☑ Perform the needed normalizations, use μ and σ^2 estimated using an exponentially weighted average across mini-batches seen during training.
   - ☐ Use the most recent mini-batch's value of μ and σ^2 to perform the needed normalizations.
10. Which of these statements about deep learning programming frameworks are true? (Check all that apply)

- ☑ Even if a project is currently open source, good governance of the project helps ensure that the it remains open even in the long term, rather than become closed or modified to benefit only one company.
- ☐ Deep learning programming frameworks require cloud-based machines to run.
- ☑ A programming framework allows you to code up deep learning algorithms with typically fewer lines of code than a lower-level language such as Python.