Hyperparameters optimization

We **want** and **can** optimize:

- # of layers (2 to 6)
- # of neurons in each layer (from 32 to 512, step = 128)
- dropout in each layer (from 0 to 0.3, step = 0.1)
- optimizer (*RMSprop* or *Adam*)
- learning rate and learning rate decay

We **don't want** to optimize:

- activation function (*relu* in hidden layers, *sigmoid* in output layer)
- loss (*binary cross entropy*)

We **want** but **can't** optimize:

- batch size

Learning rate decay

- Training neural networks with constant learning rates usually converges towards minima in a noisy manner and end up oscillating far away from actual minima
- Solution: decay the learning rate over time helps the network converge to a local minimum and avoid oscillation

1. Exponential Decay : (one of several possible methods)

$$\text{“ } \alpha = (\text{ decayRate}\text{^}(\text{epochNumber}))^{*}\alpha 0 \text{ “}$$

This function applies an exponential decay function to a provided initial learning rate so that learning rate decay over time , exponentially.

The decayRate of this method is always less then 1 , 0.95 is most commonly used among practitioners.

Learning rate decay in Keras

```
initial_learning_rate = 0.1
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate,
    decay_steps=100000,
    decay_rate=0.96,
    staircase=True)
```

Should we change only the initial learning rate?

Optimization logic

- We can define a maximum number of hyperparameters combination to test
- We can define the number of models that should be built and fit for each combination

> Note: the purpose of having multiple executions per trial is to reduce results variance and therefore be able to more accurately assess the performance of a model. If you want to get results faster, you could set executions_per_trial=1 (single round of training for each model configuration).

- We can define the number of epochs for each trial
- **How do we choose the best set of hyperparameters? What do we look at?**