CME 216, ME 343 - Spring 2020

Eric Darve, ICME

We have seen in the previous lecture how the optimal learning rate could be computed using the Hessian.

In practice, the quadratic approximate we used is not exact and the Hessian is very difficult to estimate.

Instead, we are going to find a good learning rate empirically.

For this, we need to be able to vary the learning rate in a specific manner.

We use a formula where the rate is in increased geometrically at every epoch:

```
rate = initial_rate * growth_rate^(epoch / growth_step)
```

This is implemented in TF using a callback .

We first implement our formula for the learning rate.

```
def lr_exponential_decay(epoch, lr):
  growth_rate = final_learning_rate / initial_learning_rate
  lr_ = initial_learning_rate * growth_rate**(epoch / (n_epochs-1.))
  metric_lr[epoch] = lr_
  return lr_
```

`metric_lr[epoch]` is used to store the value of the learning rate for analysis later on.

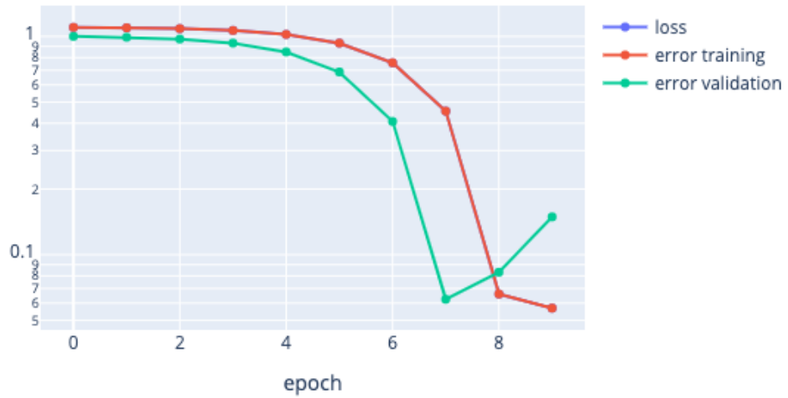
Then we create a learning rate callback in Keras:

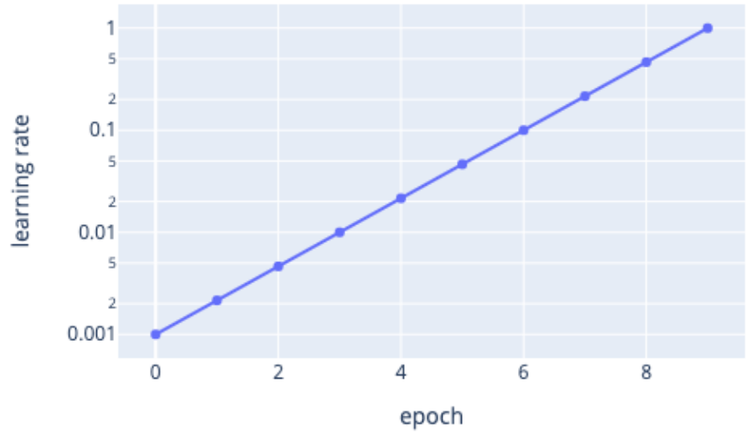
```
lr_callback = keras.callbacks.LearningRateScheduler(lr_exponential_decay)
```

Finally, we register the callback function:

```
history = dnn.fit(x_t, y_t,
        validation_data=(x_v,y_v),
        epochs=n_epochs,
        batch_size=int(x_t.size),
        callbacks = [lr_callback])
```

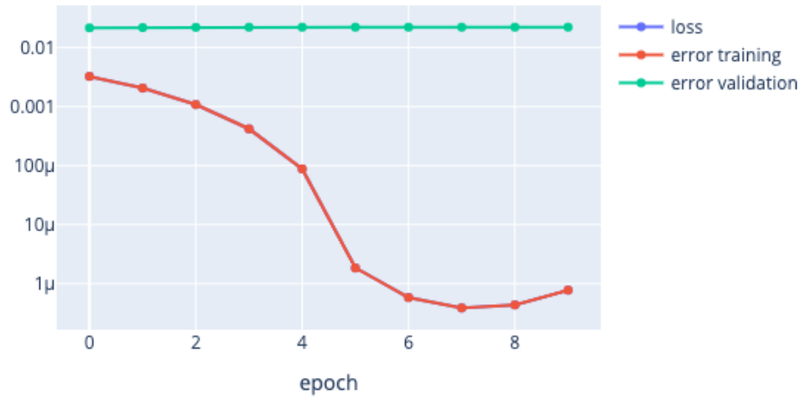
We plot the decay of the loss and the learning rate

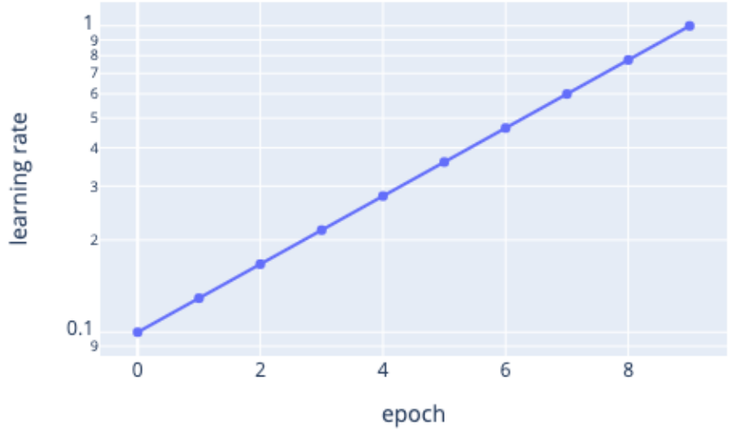




We see that the loss decays rapidly around epoch 6 which corresponds to a learning rate of 0.1.

We can restart this analysis with a smaller range for the learning rate interval.





Again the loss decays rapidly around epoch 4, which corresponds to a learning rate of 0.3.

So the final learning rate we choose is around 0.3.