Fundamentals of Deep Learning

The Learning Rate (α)

What is the Learning Rate ?

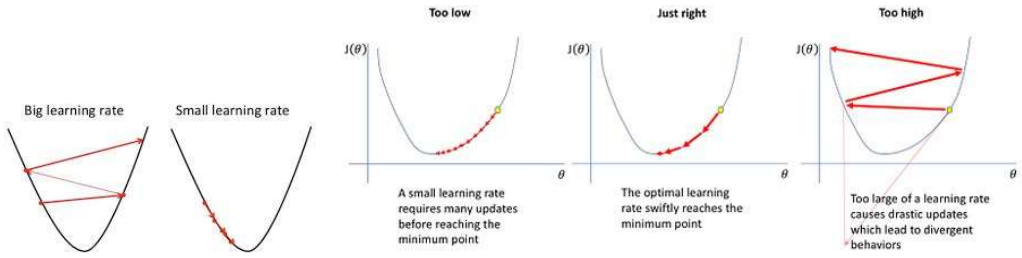
- Learning rate (α) is the magnitude of *change made to model weights* during backpropagation when the neural network is trained.

How is the Learning Rate set ?

- Learning rate (α) is specified as a hyperparameter with the optimizer (SGD,Adam. etc.)

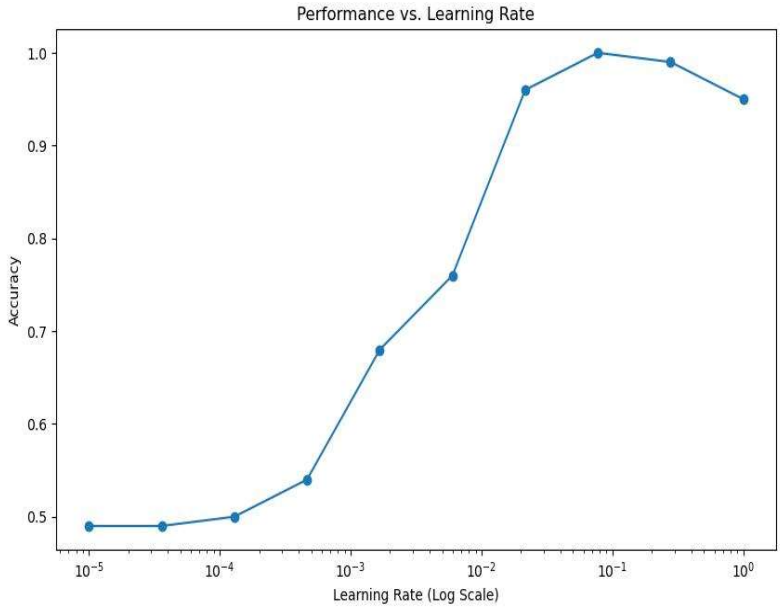
```
> model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
loss=tf.keras.losses.CategoricalCrossentropy())
```

How do we decide the Learning Rate value ?



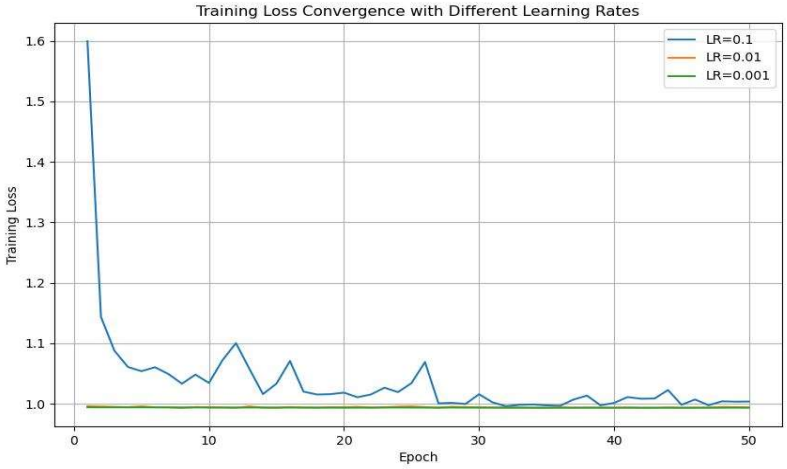
A commonly used default learning rate value is 0.001 (or 1e-3).

- If learning rate is too **small**, training will be very slow, and if it is too **high**, we might overshoot the minima.
- We can use learning rate range test, grid search, random search etc. to identify the optimal learning rate.
- Learning rate can also be dynamically adjusted.



How can learning rate be adjusted dynamically ?

We use a learning rate scheduler which makes pre-specified adjustments to the learning rate at specified intervals during the training procedure.



Adjusting the learning rate is done by decaying (reducing) the learning rate.

- We can find the ideal value of learning rate which gives the better training loss convergence.

What are the learning rate schedules in Keras ?

Keras has the following schedules :

1) Constant Learning Rate

- The momentum and decay rate is zero.
- LR remains same.
- This is used as a baseline.

2) Time based decay Learning Rate Scheduler

- The scheduler takes the decay rate (k) as a parameter.
- Keras updates the learning rate after every batch update.
- The learning rate is updated considering the decay rate (k), and total number of steps per epoch (t) :

$$LR = \frac{LR_{initial}}{1+k*t}$$

3) Step based decay learning rate scheduler

- The scheduler decreases the learning rate by a factor every few epochs.
- The factor, similar to decay rate, and the step size i.e. how often to update the LR, are provided by the user.
- The learning rate is updated as :

$$LR = LR_{initial} * factor^{\left\lfloor \frac{current-epoch}{step-size} \right\rfloor}$$

4) Exponential decay learning rate scheduler

- We define an exponential decay function and pass it to LR Scheduler :

$$LR = LR_{initial} * e^{(-1*k*t)}$$

5) Custom learning rate

- You can define your own learning rate function by subclassing tf.keras.callbacks.Callback and overriding the on_epoch_begin or on_batch_begin method.

```python
import tensorflow as tf

class CustomLearningRateScheduler(tf.keras.optimizers.schedules.LearningRateSchedule):

    def __init__(self, initial_learning_rate, decay_steps, decay_rate):

        super(CustomLearningRateScheduler, self).__init__()

        self.initial_learning_rate = initial_learning_rate

        self.decay_steps = decay_steps

        self.decay_rate = decay_rate

    def __call__(self, step):

        return self.initial_learning_rate * tf.math.pow(self.decay_rate, (step / self.decay_steps))

# Usage in a model
initial_learning_rate = 0.1

decay_steps = 1000

decay_rate = 0.96

lr_schedule = CustomLearningRateScheduler(initial_learning_rate, decay_steps, decay_rate)

optimizer = tf.keras.optimizers.Adam(learning_rate=lr_schedule)

# Build and compile your model
model = tf.keras.models.Sequential([...])

model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```