Learning Rate Scheduler

Learning rate in neural networks

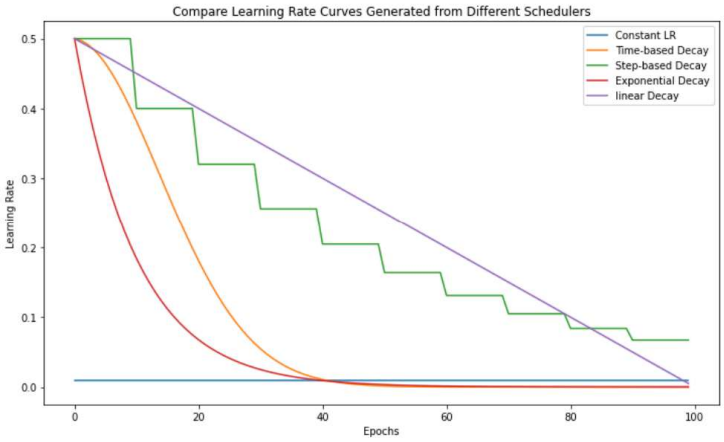
Learning rate (LR) is the magnitude of change made to model weights during backpropagation in the model training procedure. It is specified as a hyper-parameter when building neural network models in Keras.

Effect of LR on training

When training a neural network model, we typically use a default schedule with a constant LR to update network weights for each training epoch. With a small LR, the training can progress very slowly, and with a larger LR, we often observe overshooting, that is, an undesirable divergent behavior in our loss function. An optimal LR is a trade-off between the two. To help with this, we use the LR scheduler, which is a framework that makes pre-specified adjustments to the LR at set intervals during the training procedure.

Learning rate schedule

Keras provides a LR schedule base class that can be used to adapt the LR of our optimizer during training. This can enable our model to learn good weights early on, and be fine-tuned later. Keras supports LR schedules through callbacks, and the API includes various adaptive LR techniques (found here). Some common techniques that we will be discussing in this reading include time-based decay, exponential decay and step decay.



We will demonstrate the effectiveness of a few LR schedulers on a model trained using the stochastic gradient descent (SGD) optimization method, on the CIFAR-10 dataset for an binary image classification task.

Task and model definition

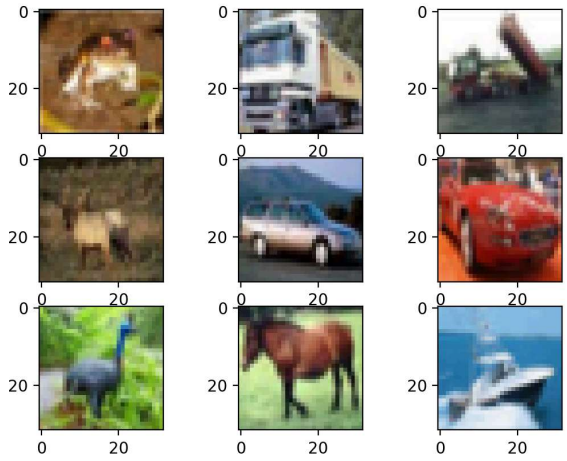
CIFAR-10 is a popular image classification dataset used in computer vision and deep learning. It is comprised of 60,000 32×32 pixel colored images of objects belonging to 10 different classes, that is, airplane, automobile, bird, and so on. In this example, we will be using class 3 (cats) and class 5 (dogs) only.

In the LR scheduler implementations for this reading, we will be using the following settings. As seen in the table below, the LR changes at a different rate for each LR scheduler. Each model will be trained for 50 epochs using the SGD algorithm.

epochs	0	10	25	50
constant	0.01	0.01	0.01	0.01
time-based decay (LR_0=0.1, decayRate=LR/#epochs)	0.1	0.099	0.097	0.095
step-based decay (LR_0=0.1, drop=0.5, epochsDrop=10)	0.1	0.05	0.025	0.00625
exponential decay (LR_0=0.1, k=0.1, momentum=0.8)	0.1	0.037	0.008	0.0006

Using the code below, we can load in the CIFAR-10 dataset using the Keras API.

▶ Click here for the code used for loading the dataset.



Pixel values for each image in the dataset are unsigned integers in the 0-255 range. We can normalize the pixel values such that the new range is 0-1.

▶ Click here for the code used to normalize the images.

Below we will define and return a base model architecture, which we will use with all LR schedulers. This will allow for direct comparison between all models.

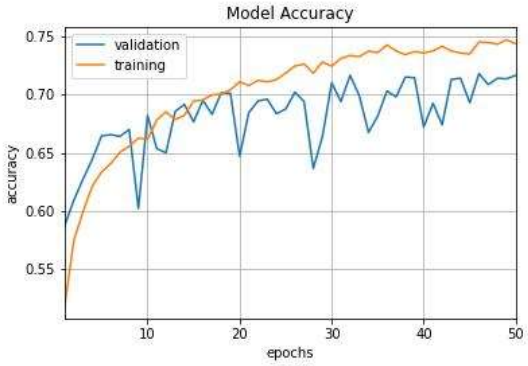
▶ Click here for the code used to train the model with a constant LR.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 30, 30, 4)         112
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 28, 28, 8)         296
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 8)         0
_________________________________________________________________
dropout_5 (Dropout)          (None, 14, 14, 8)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1568)              0
_________________________________________________________________
dense_5 (Dense)              (None, 16)                25104
_________________________________________________________________
dropout_6 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_6 (Dense)              (None, 2)                 34
=================================================================
Total params: 25,546
Trainable params: 25,546
Non-trainable params: 0
```

I. Constant

The default LR schedule for the stochastic gradient descent optimizer in Keras uses a momentum and decay rate of zero and a constant LR. The results for this model will serve as a baseline for us to experiment with different LR techniques. We use the following function to train our model:

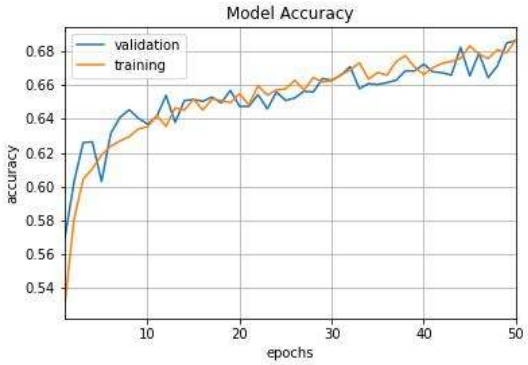
▶ Click here for the code used to train the model with a constant LR.



II. Time-based decay LR Scheduler:

This LR scheduler technique takes in a learning decay rate. We initialize our SGD optimizer with a LR of 0.1 and then set our decay to be the LR divided by the total number of epochs. Keras updates the LR after every batch update. The update formula used by Keras is $LR = LR_0/(1 + k * t)$LR = LR0 /(1 + k ∗ t) where LR_0 is the initial LR, k is the decay rate, and t is the total number of steps per epoch (or weight updates). In CIFAR-10, we have 50,000 training images. We set our batch size to 64. Therefore a total of 50,000/64 weight updates need to be applied per epoch. Our initial LR is 0.1, and decay rate is 0.1/50.

▶ Click here for the code used to train the model with a time-based decay LR scheduler.



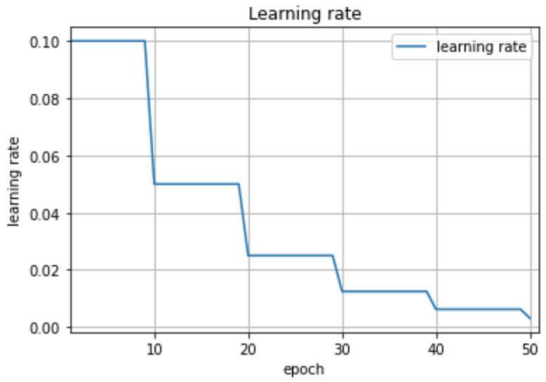
III. Step based-decay LR Scheduler:

This technique decreases the learning rate by a factor every few epochs. For example, we could halve the LR every 10 epochs. This can be defined by the following formula:

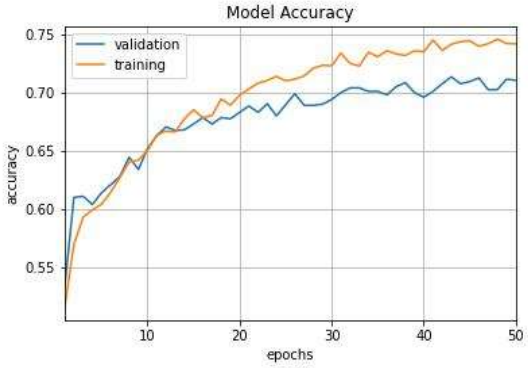
$$LR = LR_0 * drop^{floor(epoch/epochs_{drop})}$$LR = LR0 ∗ dropfloor(epoch/epochsdrop)

Here, $drop$ is 50%, $epoch_{drop}$ is the number of epochs after which we update the LR, which in this case is 10.

We use the LR scheduler callback which takes in the step decay function and return the updated LR to the optimizer.



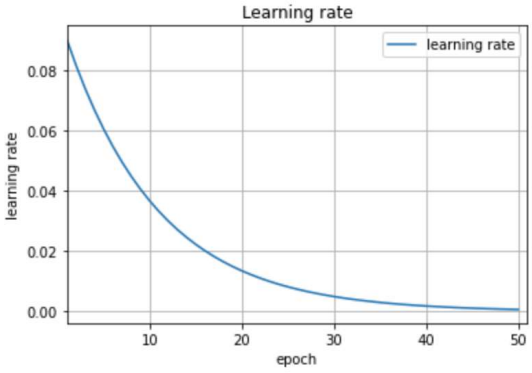
▶ Click here for the code used to train the model with a step decay LR scheduler.

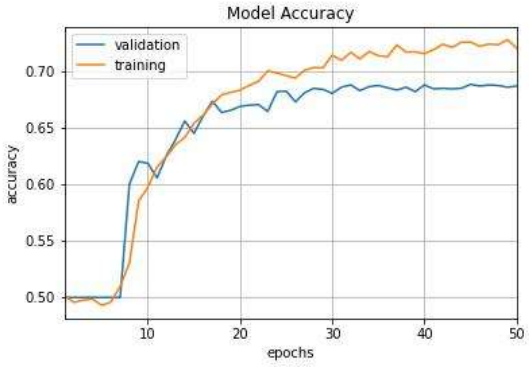


IV. Exponential decay LR Scheduler:

This is another commonly used LR scheduler technique. It is based on the following formula: $LR = LR_0 * e^{(-1*kt)}$LR = LR0 ∗ e(−1∗kt). As we did in the case of step decay, here we define an exponential decay function and pass it to LearningRateScheduler callback.

▶ Click here for the code used to train the model with an exponential decay LR scheduler.





In this reading, we used the LearningRateScheduler in Keras to define custom LR schedules. In most cases, we see that using some form of a LR scheduling technique improves our model's performance, in comparison to the constant LR schedule which is used in Keras, by default.

::page{title="Authors"}
Kopal Garg

Kopal Garg is a Masters student in Computer Science at the University of Toronto.

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-07-10	0.1	Kopal Garg	Create Reading