In TensorFlow 2.0, the **'decay'** parameter of the selected optimizer from the **'tf.keras.optimizers'** module allows time inverse decay of learning rate.

Adaptive Learning Rates

Adaptive learning rate, on the other hand, re-adjusts the learning rate in accordance with the training data. It basically uses a different learning rate for each parameter and adapts it during training. These techniques are based on the observation that each parameter results in a different type of gradient. The following list outlines types of adaptive learning rates in use and their method calls in TensorFlow 2.0:

- AdaGrad: **tf.keras.optimizers.Adagrad()**
- AdaDelta: **tf.keras.optimizers.Adadelta()**
- RMSProp: **tf.keras.optimizers.RMSprop()**
- Adaptive Moments, (Adam): **tf.keras.optimizers.Adam()**

However, AdaGrad performs poorly when used for training deep learning models due to its monotonic learning rate which could be too aggressive, and learning may stop early during training. As of now, there is no proven best optimization technique, so the choice of the optimization technique is down to the preference of the model designer.

This chapter surveys some other techniques for optimizing the weights of a deep neural network. These techniques have implementations in deep learning libraries such as Tensorflow and Keras and can be explored as hyper-parameters when designing a neural network solution for a particular learning use case.

In the next chapter, we will discuss techniques for applying regularization to a deep neural network to prevent overfitting.

CHAPTER 34

Regularization for Deep Learning

Regularization is a technique for reducing the variance in the validation set, thus preventing the model from overfitting during training. In doing so, the model can better generalize to new examples. When training deep neural networks, a couple of strategies exist for use as a regularizer.

Dropout

Dropout is a regularization technique that prevents a deep neural network from overfitting by randomly discarding a number of neurons at every layer during training. In doing so, the neural network is not overly dominated by any one feature as it only makes use of a subset of neurons in each layer during training. In doing so, Dropout resembles an ensemble of neural networks as a similar but distinct neural network is trained at each layer. Dropout works by designating a probability that a neuron will be dropped in a layer. This probability value is called the Dropout rate. Figure 34-1 shows an example of a network with and without Dropout.

E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform*,
https://doi.org/10.1007/978-1-4842-4470-8_34