Algorithm 8.6 RMSProp algorithm with Nesterov momentum

Require: Global learning rate ϵ, decay rate ρ, momentum coefficient α.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
  Initialize accumulation variable $\boldsymbol{r} = \mathbf{0}$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute interim update: $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{v}$
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)}; \tilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$
    Accumulate gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho) \boldsymbol{g} \odot \boldsymbol{g}$
    Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \frac{\epsilon}{\sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$.   ($\frac{1}{\sqrt{\boldsymbol{r}}}$ applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
  end while

largely on the user's familiarity with the algorithm (for ease of hyperparameter tuning).

8.6 Approximate Second-Order Methods

In this section we discuss the application of second-order methods to the training of deep networks. See LeCun *et al.* (1998a) for an earlier treatment of this subject. For simplicity of exposition, the only objective function we examine is the empirical risk:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}, \mathrm{y} \sim \hat{p}_{\text{data}}(\boldsymbol{x}, y)}[L(f(\boldsymbol{x}; \boldsymbol{\theta}), y)] = \frac{1}{m} \sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), y^{(i)}). \tag{8.25}$$

However the methods we discuss here extend readily to more general objective functions that, for instance, include parameter regularization terms such as those discussed in chapter 7.

8.6.1 Newton's Method

In section 4.3, we introduced second-order gradient methods. In contrast to first-order methods, second-order methods make use of second derivatives to improve optimization. The most widely used second-order method is Newton's method. We now describe Newton's method in more detail, with emphasis on its application to neural network training.