AdaDelta 算法

1. 产生背景

对于解决 AdaGrad 算法中在迭代后期可能较难找到有用解的问题，除了 RMSProp 算法外，还可以使用 AdaDelta 算法实现。特别地，AdaDelta 算法并没有学习率这一超参数。

2. 算法内容

AdaDelta 算法使用了小批量随机梯度g_t按元素平方的指数加权移动平均变量S_t。在时间步 0，他的所有元素被初始化为 0。给定超参数$0 \leq \rho < 1$(与 RMSProp 中的γ对应)，在时间步$t > 0$,与 RMSProp 一样:

$$\boldsymbol{s}_t \leftarrow \rho \boldsymbol{s}_{t-1} + (1-\rho)\boldsymbol{g}_t \odot \boldsymbol{g}_t$$

此处，AdaDelta 算法还维护一个额外的状态变量Δx_t,其元素同样在时间步 0 时被初始化为 0，使用Δx_{t-1}来计算自变量的变化量:

$$\boldsymbol{g}_t' \leftarrow \sqrt{\frac{\Delta \boldsymbol{x}_{t-1} + \epsilon}{\boldsymbol{s}_t + \epsilon}} \odot \boldsymbol{g}_t$$

其中ε是维持数值稳定性添加的常数。
然后更新变量:

$$\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} - \boldsymbol{g}_t'$$

最后，使用Δx_t来记录自变量变化量g_t'按元素平方的指数加权移动平均:

$$\Delta \boldsymbol{x}_t \leftarrow \rho \Delta \boldsymbol{x}_{t-1} + (1-\rho)\boldsymbol{g}_t' \odot \boldsymbol{g}_t'$$

AdaDelta 算法与 RMSProp 算法的不同之处在于使用$\sqrt{\Delta x_{t-1}}$来带起超参数η

3. 代码实现

AdaDelta 算法需要对每个自变量维护两个状态变量，即s_t和Δx_t。

In [1]:
```
%matplotlib inline
import d2lzh as d2l
from mxnet import nd

features, labels = d2l.get_data_ch7()

def init_adadelta_states():
    s_w, s_b = nd.zeros((features.shape[1], 1)), nd.zeros(1)
    delta_w, delta_b = nd.zeros((features.shape[1], 1)), nd.zeros(1)
    return ((s_w, delta_w), (s_b, delta_b))

def adadelta(params, states, hyperparams):
    rho, eps = hyperparams['rho'], 1e-5
    for p, (s, delta) in zip(params, states):
        s[:] = rho * s + (1 - rho) * p.grad.square()
        g = ((delta + eps).sqrt() / (s + eps).sqrt()) * p.grad
        p[:] -= g
        delta[:] = rho * delta + (1 - rho) * g * g
```

4. 相关文献

[1] Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.