Adam 算法

1. 产生背景

Adam 算法在 RMSProp 算法的基础上，对小批量随机梯度也做了指数加权移动平均。

2. 算法内容

Adam 算法使用动量变量v_t和 RMSProp 中的小批量随机梯度按元素平方的指数加权移动平均变量s_t，并在时间步 0 将它们中每个元素初始化为 0。给定超参数$0 \leq \beta_1 < 1$(文献中作者建议设为 0.9)，时间步 t 的动量变量v_t（即小批量随机梯度g_t的指数加权移动平平均)：

$$\boldsymbol{v}_t \leftarrow \beta_1 \boldsymbol{v}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$$

和 RMSProp 算法中一样，给定超参数$0 \leq \beta_2 < 1$(文献中作者建议设为 0.999)，将小批量随机梯度按元素平方后的项$\boldsymbol{g}_t \odot \boldsymbol{g}_t$做指数加权平均得到$s_t$：

$$\boldsymbol{s}_t \leftarrow \beta_2 \boldsymbol{s}_{t-1} + (1-\beta_2)\boldsymbol{g}_t \odot \boldsymbol{g}_t$$

由于我们将v_0和s_0中的元素都初始化为 0，在时间步 t 我们得到$\boldsymbol{v}_t = (1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i}\boldsymbol{g}_i$。将过去各时间步小批量随机梯度的权重相加，得到$(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} = 1-\beta_1^t$。注意，当 t 较小时，过去各时间步小批量随机梯度权值之和会较小。为了消除这种影响，对于任意时间步 t，可以将v_t除以$1-\beta_1^t$，从而使过去各时间步小批量随机的梯度权值之和为 1，即偏差修正。

$$\boldsymbol{v}_t \leftarrow \frac{\boldsymbol{v}_t}{1-\beta_1^t}$$

$$\boldsymbol{s}_t \leftarrow \frac{\boldsymbol{s}_t}{1-\beta_2^t}$$

然后，Adam 算法使用以上偏差修正后的变量$\hat{\boldsymbol{v}}_t$与$\hat{\boldsymbol{s}}_t$，将模型参数中每个元素的学习率按元素运算重新调整：

$$\boldsymbol{g}_t' \leftarrow \frac{\eta \hat{\boldsymbol{v}}_t}{\sqrt{\hat{\boldsymbol{s}}_t} + \epsilon}$$

其中η是学习率，ϵ是为了维持数值稳定添加的常数。和 AdaGrad 算法、RMSProp 算法以及 AdaDelta 算法一样，目标函数自变量中每个元素都分别拥有自己的学习率。最后，使用g_t'迭代自变量：

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

3. 代码实现

In [1]:

```
%matplotlib inline
import d2lzh as d2l
from mxnet import nd

features, labels = d2l.get_data_ch7()

def init_adam_states():
    v_w, v_b = nd.zeros((features.shape[1], 1)), nd.zeros(1)
    s_w, s_b = nd.zeros((features.shape[1], 1)), nd.zeros(1)
    return ((v_w, s_w), (v_b, s_b))

def adam(params, states, hyperparams):
    beta1, beta2, eps = 0.9, 0.999, 1e-6
    for p, (v, s) in zip(params, states):
        v[:] = beta1 * v + (1 - beta1) * p.grad
        s[:] = beta2 * s + (1 - beta2) * p.grad.square()
        v_bias_corr = v / (1 - beta1 ** hyperparams['t'])
        s_bias_corr = s / (1 - beta2 ** hyperparams['t'])
        p[:] -= hyperparams['lr'] * v_bias_corr / (s_bias_corr.sqrt() + eps)
    hyperparams['t'] += 1
```

4. 相关文献

[1] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.