再次考虑拉长的碗状问题：梯度下降从快速沿最陡的坡度下降开始，该坡度没有直接指向全局最优解，然后非常缓慢地下降到谷底。如果算法可以更早地纠正其方向，使它更多地指向全局最优解，那将是很好的。AdaGrad算法[3]通过沿最陡峭的维度按比例缩小梯度向量（见公式11-6）来实现此校正。

公式11-6：AdaGrad算法

$$1.\quad \boldsymbol{s} \leftarrow \boldsymbol{s} + \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \otimes \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$2.\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \oslash \sqrt{\boldsymbol{s} + \varepsilon}$$

第一步将梯度的平方累加到向量s中（请记住，⊗符号表示逐元素相乘）。此向量化形式等效于针对向量s中的每个元素si计算$s_i \leftarrow s_i + (\partial J(\theta)/\partial \theta_i)^2$。换句话说，每个$s_i$累加关于参数$\theta_i$的成本函数偏导数的平方。如果成本函数沿第i个维度陡峭，则s_i将在每次迭代中变得越来越大。

第二步几乎与“梯度下降”相同，但有一个很大的区别：梯度向量按比例因子$\sqrt{\boldsymbol{s}+\varepsilon}$缩小了（⊘符号代表逐元素相除，而ε是避免除以零的平滑项，通常设置为10^{-10}）。此向量化形式等效于对所有参数θi同时计算$\theta_i \leftarrow \theta_i - \eta \, \partial J(\boldsymbol{\theta}) / \partial \theta_i / \sqrt{s_i + \varepsilon}$。

简而言之，该算法会降低学习率，但是对于陡峭的维度，它的执行速度要比对缓慢下降的维度的执行速度要快。这称为自适应学习率。它有助于将结果更新更直接地指向全局最优解（见图11-7）。另一个好处是，它几乎不需要调整学习率超参数η。