

## 7.4 动量法

在7.2节（梯度下降和随机梯度下降）中我们提到，目标函数有关自变量的梯度代表了目标函数在自变量当前位置下降最快的方向。因此，梯度下降也叫作最陡下降（steepest descent）。在每次迭代中，梯度下降根据自变量当前位置，沿着当前位置的梯度更新自变量。然而，如果自变量的迭代方向仅仅取决于自变量当前位置，这可能会带来一些问题。

### 7.4.1 梯度下降的问题

让我们考虑一个输入和输出分别为二维向量 $\mathbf{x} = [x_1, x_2]^T$ 和标量的目标函数 $f(\mathbf{x}) = 0.1x_1^2 + 2x_2^2$ 。与7.2节中不同，这里将 $x_1^2$ 系数从1减小到了0.1。下面实现基于这个目标函数的梯度下降，并演示使用学习率为0.4时自变量的迭代轨迹。

```
%matplotlib inline
import sys
sys.path.append("..")
import d2lzh_pytorch as d2l
import torch

eta = 0.4 # 学习率

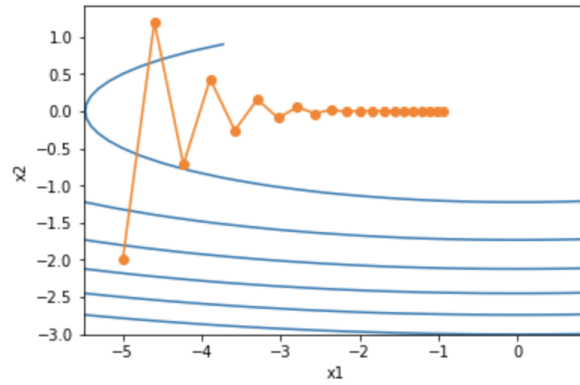
def f_2d(x1, x2):
    return 0.1 * x1 ** 2 + 2 * x2 ** 2

def gd_2d(x1, x2, s1, s2):
    return (x1 - eta * 0.2 * x1, x2 - eta * 4 * x2, 0, 0)

d2l.show_trace_2d(f_2d, d2l.train_2d(gd_2d))
```

输出：

```
epoch 20, x1 -0.943467, x2 -0.000073
```



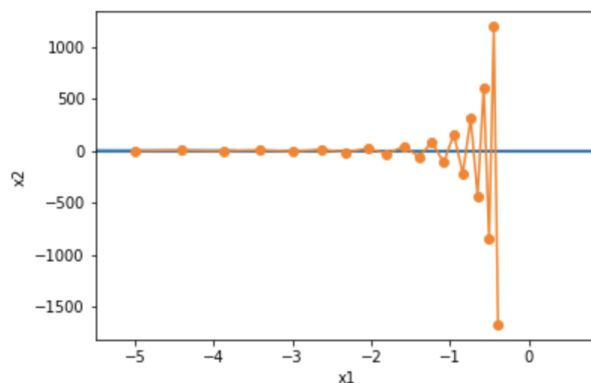
可以看到，同一位置上，目标函数在竖直方向（ $x_2$  轴方向）比在水平方向（ $x_1$  轴方向）的斜率的绝对值更大。因此，给定学习率，梯度下降迭代自变量时会使得自变量在竖直方向比在水平方向移动幅度更大。那么，我们需要一个较小的学习率从而避免自变量在竖直方向上越过目标函数最优解。然而，这会造成自变量在水平方向上朝最优解移动变慢。

下面我们试着将学习率调得稍大一点，此时自变量在竖直方向不断越过最优解并逐渐发散。

```
eta = 0.6
d2l.show_trace_2d(f_2d, d2l.train_2d(gd_2d))
```

输出：

```
epoch 20, x1 -0.387814, x2 -1673.365109
```



## 7.4.2 动量法

动量法的提出是为了解决梯度下降的上述问题。由于小批量随机梯度下降比梯度下降更为广义，本章后续讨论将沿用7.3节（小批量随机梯度下降）中时间步  $t$  的小批量随机梯度  $\mathbf{g}_t$  的定义。设时间步  $t$  的自变量为  $\mathbf{x}_t$ ，学习率为  $\eta_t$ 。在时间步0，动量法创建速度变量  $\mathbf{v}_0$ ，并将其元素初始化成0。在时间步  $t > 0$ ，动量法对每次迭代的步骤做如下修改：

$$\begin{aligned}\mathbf{v}_t &\leftarrow \gamma \mathbf{v}_{t-1} + \eta_t \mathbf{g}_t \\ \mathbf{x}_t &\leftarrow \mathbf{x}_{t-1} - \mathbf{v}_t,\end{aligned}$$

其中，动量超参数 $\gamma$ 满足 $0 \leq \gamma < 1$ 。当 $\gamma = 0$ 时，动量法等价于小批量随机梯度下降。

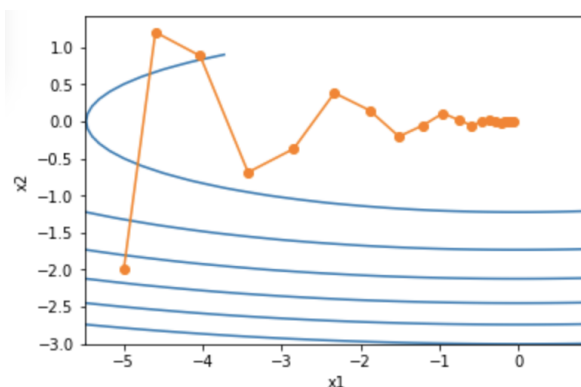
在解释动量法的数学原理前，让我们先从实验中观察梯度下降在使用动量法后的迭代轨迹。

```
def momentum_2d(x1, x2, v1, v2):
    v1 = gamma * v1 + eta * 0.2 * x1
    v2 = gamma * v2 + eta * 4 * x2
    return x1 - v1, x2 - v2, v1, v2

eta, gamma = 0.4, 0.5
d2l.show_trace_2d(f_2d, d2l.train_2d(momentum_2d))
```

输出：

```
epoch 20, x1 -0.062843, x2 0.001202
```

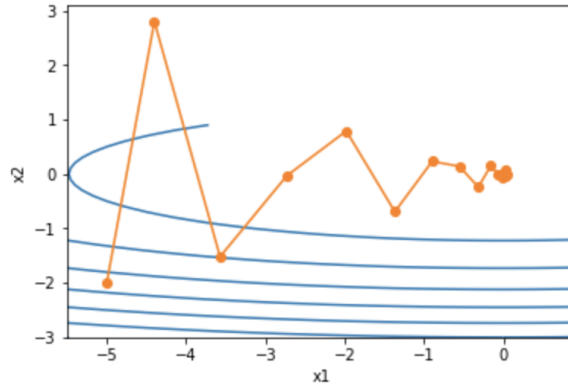


可以看到使用较小的学习率 $\eta = 0.4$ 和动量超参数 $\gamma = 0.5$ 时，动量法在竖直方向上的移动更加平滑，且在水平方向上更快逼近最优解。下面使用较大的学习率 $\eta = 0.6$ ，此时自变量也不再发散。

```
eta = 0.6
d2l.show_trace_2d(f_2d, d2l.train_2d(momentum_2d))
```

输出：

epoch 20, x1 0.007188, x2 0.002553



### 7.4.2.1 指数加权移动平均

为了从数学上理解动量法，让我们先解释一下指数加权移动平均（exponentially weighted moving average）。给定超参数  $0 \leq \gamma < 1$ ，当前时间步  $t$  的变量  $y_t$  是上一时间步  $t-1$  的变量  $y_{t-1}$  和当前时间步另一变量  $x_t$  的线性组合：

$$y_t = \gamma y_{t-1} + (1 - \gamma) x_t.$$

我们可以对  $y_t$  展开：

$$\begin{aligned} y_t &= (1 - \gamma) x_t + \gamma y_{t-1} \\ &= (1 - \gamma) x_t + (1 - \gamma) \cdot \gamma x_{t-1} + \gamma^2 y_{t-2} \\ &= (1 - \gamma) x_t + (1 - \gamma) \cdot \gamma x_{t-1} + (1 - \gamma) \cdot \gamma^2 x_{t-2} + \gamma^3 y_{t-3} \\ &\dots \end{aligned}$$

令  $n = 1/(1 - \gamma)$ ，那么  $(1 - 1/n)^n = \gamma^{1/(1-\gamma)}$ 。因为

$$\lim_{n \rightarrow \infty} (1 - \frac{1}{n})^n = \exp(-1) \approx 0.3679,$$

所以当  $\gamma \rightarrow 1$  时， $\gamma^{1/(1-\gamma)} = \exp(-1)$ ，如  $0.95^{20} \approx \exp(-1)$ 。如果把  $\exp(-1)$  当作一个比较小的数，我们可以在近似中忽略所有含  $\gamma^{1/(1-\gamma)}$  和比  $\gamma^{1/(1-\gamma)}$  更高阶的系数的项。例如，当  $\gamma = 0.95$  时，

$$y_t \approx 0.05 \sum_{i=0}^{19} 0.95^i x_{t-i}.$$

因此，在实际中，我们常常将  $y_t$  看作是对最近  $1/(1 - \gamma)$  个时间步的  $x_t$  值的加权平均。例如，当  $\gamma = 0.95$  时， $y_t$  可以被看作对最近 20 个时间步的  $x_t$  值的加权平均；当  $\gamma = 0.9$  时， $y_t$  可以看作是对最近 10 个时间步的  $x_t$  值的加权平均。而且，离当前时间步  $t$  越近的  $x_t$  值获得的权重越大（越接近 1）。

## 7.4.2.2 由指数加权移动平均理解动量法

现在，我们对动量法的速度变量做变形：

$$\mathbf{v}_t \leftarrow \gamma \mathbf{v}_{t-1} + (1 - \gamma) \left( \frac{\eta_t}{1 - \gamma} \mathbf{g}_t \right).$$

由指数加权移动平均的形式可得，速度变量  $\mathbf{v}_t$  实际上对序列  $\{\eta_{t-i} \mathbf{g}_{t-i} / (1 - \gamma) : i = 0, \dots, 1/(1 - \gamma) - 1\}$  做了指数加权移动平均。换句话说，相比于小批量随机梯度下降，**动量法在每个时间步的自变量更新量近似于将最近  $1/(1 - \gamma)$  个时间步的普通更新量（即学习率乘以梯度）做了指数加权移动平均后再除以  $1 - \gamma$** 。所以，在动量法中，自变量在各个方向上的移动幅度不仅取决当前梯度，还取决于过去的各个梯度在各个方向上是否一致。在本节之前示例的优化问题中，所有梯度在水平方向上为正（向右），而在竖直方向上时正（向上）时负（向下）。这样，我们就可以使用较大的学习率，从而使自变量向最优解更快移动。

## 7.4.3 从零开始实现

相对于小批量随机梯度下降，动量法需要对每一个自变量维护一个同它一样形状的速度变量，且超参数里多了动量超参数。实现中，我们将速度变量用更广义的状态变量 `states` 表示。

```
features, labels = d2l.get_data_ch7()

def init_momentum_states():
    v_w = torch.zeros((features.shape[1], 1), dtype=torch.float32)
    v_b = torch.zeros(1, dtype=torch.float32)
    return (v_w, v_b)

def sgd_momentum(params, states, hyperparams):
    for p, v in zip(params, states):
        v.data = hyperparams['momentum'] * v.data + hyperparams['lr'] * p.grad.data
        p.data -= v.data
```

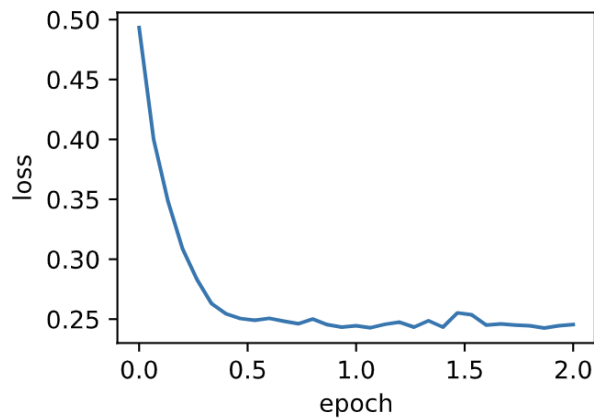
我们先将动量超参数 `momentum` 设0.5，这时可以看成是特殊的小批量随机梯度下降：其小批量随机梯度为最近2个时间步的2倍小批量梯度的加权平均。

**注：**个人认为这里不应该是“加权平均”而应该是“加权和”，因为根据7.4.2.2节分析，加权平均最后除了  $1 - \gamma$ ，所以就相当于没有进行平均。

```
d2l.train_ch7(sgd_momentum, init_momentum_states(),
              {'lr': 0.02, 'momentum': 0.5}, features, labels)
```

输出:

```
loss: 0.245518, 0.042304 sec per epoch
```



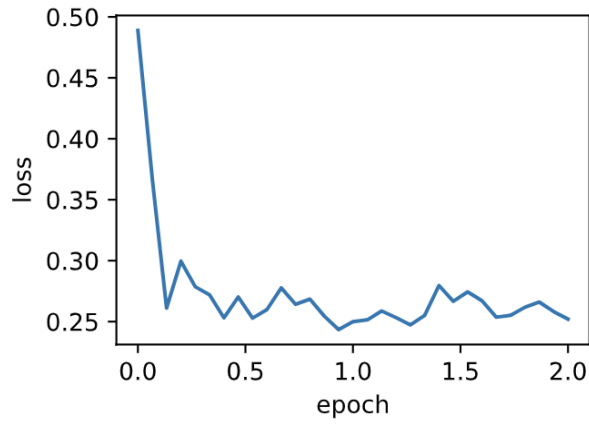
将动量超参数 `momentum` 增大到0.9, 这时依然可以看成是特殊的小批量随机梯度下降: 其小批量随机梯度为最近10个时间步的10倍小批量梯度的加权平均。我们先保持学习率0.02不变。

**同理, 这里不应该是“加权平均”而应该是“加权和”。**

```
d2l.train_ch7(sgd_momentum, init_momentum_states(),
              {'lr': 0.02, 'momentum': 0.9}, features, labels)
```

输出:

```
loss: 0.252046, 0.095708 sec per epoch
```



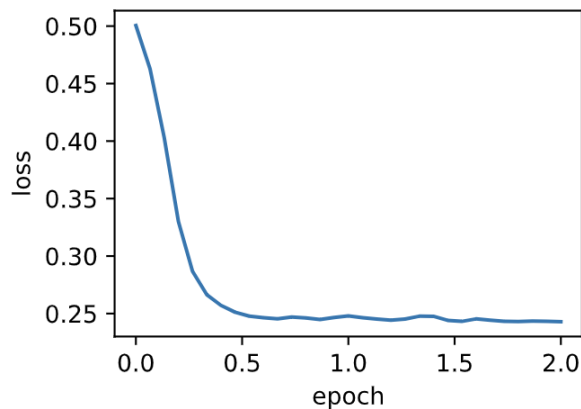
可见目标函数值在后期迭代过程中的变化不够平滑。直觉上，10倍小批量梯度比2倍小批量梯度大了5倍，我们可以试着将学习率减小到原来的1/5。此时目标函数值在下降了一段时间后变化更加平滑。

这也印证了刚刚的观点。

```
d2l.train_ch7(sgd_momentum, init_momentum_states(),
              {'lr': 0.004, 'momentum': 0.9}, features, labels)
```

输出：

```
loss: 0.242905, 0.073496 sec per epoch
```



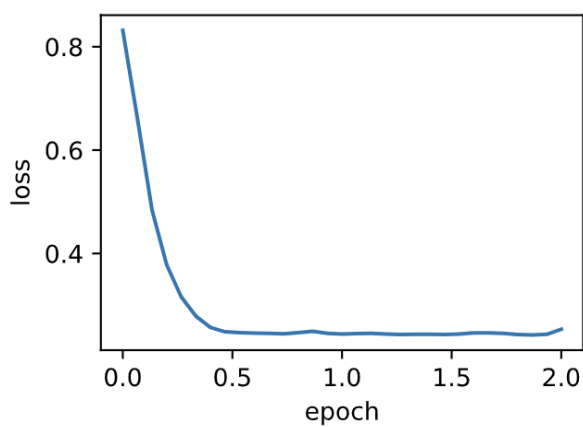
## 7.4.4 简洁实现

在PyTorch中，只需要通过参数 `momentum` 来指定动量超参数即可使用动量法。

```
d2l.train_pytorch_ch7(torch.optim.SGD, {'lr': 0.004, 'momentum': 0.9},  
                        features, labels)
```

输出:

```
loss: 0.253280, 0.060247 sec per epoch
```



## 小结

- 动量法使用了指数加权移动平均的思想。它将过去时间步的梯度做了加权平均，且权重按时间步指数衰减。
- 动量法使得相邻时间步的自变量更新在方向上更加一致。