前一阵子在看吴恩达教授在Coursera上的机器学习的课程，在做到多元梯度下降（Multivariate Gradient Descent）的时候，一下子不知道怎么矢量化这个操作，后来写了写才算出来，稍微记录一下过程和想法。

首先，多元梯度下降的算法一般被定义为迭代以下更新操作直到梯度收敛

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \quad j = \{0 \dots n\} \text{ simultaneously}$$

在这里，n 是特征的数量，α 是学习率，$\frac{\partial}{\partial \theta_j} J(\theta)$ 则是代价函数 J 关于 θ_j 的偏导数，将其求出，我们得到

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(X^{(i)}) - y^{(i)}) X_j^{(i)}$$

在这里，m 是训练集的数量，h_θ 是关于特定 θ 的假设函数，$x^{(i)}$ 和 $y^{(i)}$ 分别表示第 i 个训练样本中的输入值和目标值。在这个例子中，训练集的输入矩阵 $X \in \mathbb{R}^{m \times (x+1)}$ 的每一列是一个参数（特征），每一行是一组输入数据，因此这个矩阵可以被定义为

$$X = \begin{bmatrix} X_0^{(1)} & X_1^{(1)} & \dots & X_n^{(1)} \\ X_0^{(2)} & X_1^{(2)} & \dots & X_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ X_0^{(m)} & X_1^{(m)} & \dots & X_n^{(m)} \end{bmatrix}$$

而训练集的目标向量 $y \in \mathbb{R}^m$ 则包含了所有 m 个训练样本的目标值。

很显然，利用 `for` 循环来迭代计算并更新所有 θ_j 的值非常低效，因此矢量化（Vectorization）在这里将显得尤为重要。

首先，令 θ 成为一个 $(n+1)$ 维的向量，

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

这可以让我们之后的矢量化变得容易许多，同时也保持了访问某个特定 θ_j 的方式不变。

接着，我们定义梯度下降的每一次迭代如下

$$\theta := \theta - \alpha\delta$$

在这里，$\delta \in \mathbb{R}^{n+1}$ 包含了关于所有 θ_j 的偏导数值，即 $\delta_j = \frac{\partial}{\partial\theta_j} J(\theta)$ 对任意的 $j = (0 \dots n)$，经过简化，我们得到

$$\begin{aligned}
\delta_j &= \frac{\partial}{\partial\theta_j} J(\theta) \\
&= \frac{1}{m} \sum_{i=1}^{m} (h_\theta(X^{(i)}) - y^{(i)}) X_j^{(i)} \\
&= \frac{1}{m} \sum_{i=1}^{m} (X^{(i)}\theta - y^{(i)}) X_j^{(i)}
\end{aligned}$$

注意到该式第二部分是 m 个积的和，因此想到可以将其写作两个向量之积（一个行向量一个列向量），$a \in \mathbb{R}^{1\times m}$ 和 $b \in \mathbb{R}^m$，分别定义为

$$\begin{aligned}
a_{1,i} &= X^{(i)}\theta - y^{(i)} \\
b_i &= X_j^{(i)}
\end{aligned}$$

因此，δ_j 的式子可以被继续简化为

$$\delta_j = \frac{1}{m} (X\theta - y)^\top X_j$$

注意到 $\frac{1}{m}(X\theta - y)^\top$ 关于变量 j 是不变的，因此我们可以把它提取出来并定义整个 δ 为

$$\begin{aligned}
\delta &= \frac{1}{m} (X\theta - y)^\top \begin{bmatrix} X_0 & X_1 & \dots & X_n \end{bmatrix} \\
&= \frac{1}{m} (X\theta - y)^\top X
\end{aligned}$$

或者，利用矩阵转置的特性，$\delta = \frac{1}{m} X^\top (X\theta - y)$.

最后，我们得到的矢量化的多元梯度下降更新公式如下

$$\theta := \theta - \frac{\alpha}{m} X^\top (X\theta - y)$$

以下是实现这个的 MATLAB 代码（仅供参考）：

```
theta = theta - alpha / m * X' * (X * theta - y);
```

最后补一句题外话，感谢知乎终于把公式改成矢量图了！新版公式编辑器也很好用！感动！

来源： https://zhuanlan.zhihu.com/p/28674008