矢量化计算

emacsun

目录

1 简介

吴恩达cosera上机器学习前三次课程作业本身难度不大，很顺利完成，每次作业submit之后都是100points。值得注意的是作业特别强调计算的矢量化。这本身是一件很有意义的事情，因为以 `for` 或者 `while` 实现的矢量计算，效率远远不及以矢量本身为操作对象的计算。毕竟 `for` 或者 `while` 每次循环执行的是一个矢量元素的计算，而以矢量为操作对象的计算一次就执行了对所有元素的计算。另外，以矢量为操作对象的计算实现起来代码更简短。

这里以第三次作业为例，记录matlab实现矢量化操作的过程。

2 multi-class classification

由于本文着重强调计算的矢量化，关于什么是 `multi-class classification` 在这里就不在重述，请参考 这里 在作业中我们以5000个手写数字为训练样本，得到一个多类分类器。这5000个样本都是20×20的灰度图像，每一个像素都用一个浮点数表示其灰度。这样一个图像可以用长为400的矢量来表示。在本例中每一个样本是X的一行。

```
% Load saved matrices from file
load('ex3data1.mat');
% The matrices X and y will now be in your Octave environment
```

通过 `load` 导入训练样本X

$$X = \begin{bmatrix} - & (x^{(1)})^T & - \\ - & (x^{(2)})^T & - \\ & \vdots & - \\ - & (x^{(m)})^T & - \end{bmatrix} \tag{2.1}$$

其中X的每一行都是一个样本，存储着一个数字灰度图像的所有像素构成的矢量，这里每一行都是长为400的矢量。一共m行，代表着m 个样本，这里$m = 5000$。另外导入的数据中还有y，包含了这5000个样本的正确映射结果。

2.1 矢量化损失函数

logistic回归的损失函数是：

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))] \tag{2.2}$$

计算损失函数过程中，m是样本个数，就是说上面的函数把所有的样本都考虑在内了。为了计算求和项的每一个元素，我们需要计算$h_\theta(x^{(i)})$，其中$h_\theta(x^{(i)}) = g(\theta^T x^{(i)})$ $g(z) = \frac{1}{1+e^{-z}}$。对于$h_\theta(x^{(i)})$的计算我们可以利用

$$X = \begin{bmatrix} - & (x^{(1)})^T & - \\ - & (x^{(2)})^T & - \\ & \vdots & - \\ - & (x^{(m)})^T & - \end{bmatrix} \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \tag{2.3}$$

其中X的维度是5000×400, θ的维度是400×1，通过计算：

$$X\theta = \begin{bmatrix} - & (x^{(1)})^T\theta & - \\ - & (x^{(2)})^T\theta & - \\ & \vdots & - \\ - & (x^{(m)})^T\theta & - \end{bmatrix} \tag{2.4}$$

$X\theta$的维度是5000×1，然后根据~(2.2) 计算损失函数。

为了防止出现overfitting现象，我们通常需要对损失函数进行正则化：

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log(h_\theta(x^{(i)}))-(1-y^{(i)})\log(1-h_\theta(x^{(i)}))]+\frac{\alpha}{2m}\sum_{j=1}^{n}\theta_j^2 \quad (2.5)$$

注意这个正则项不包括θ_0，也就是说我们不对偏移项进行正则化。

2.2 矢量化梯度

未正则化的logistic regression的梯度函数是：

$$\frac{\partial J}{\partial \theta_j}=\frac{1}{m}\sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}) \quad (2.6)$$

我们写出所有$\frac{\partial J}{\partial \theta_j}$:

$$\begin{bmatrix}\frac{\partial J}{\partial \theta_0}\\ \frac{\partial J}{\partial \theta_1}\\ \frac{\partial J}{\partial \theta_2}\\ \vdots\\ \frac{\partial J}{\partial \theta_n}\end{bmatrix}=\frac{1}{m}\begin{bmatrix}\sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})x_0^{(i)})\\ \sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})x_1^{(i)})\\ \sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})x_2^{(i)})\\ \vdots\\ \sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})x_n^{(i)})\end{bmatrix}$$

(2.7)

上式右端可以矢量化为$\frac{1}{m}\sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})x^{\Phi i})=\frac{1}{m}X^T(h_\theta(x)-y)$:

因为这个过程有两个地方体现了矢量化，所以稍微难理解一些。首先从式˜(2.2)到$\frac{1}{m}\sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})x^{\Phi i})$，我们可以看到$h_\theta(x^{(i)})-y^{(i)}$是一个标量，$\sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})x_0^{(i)})$ 实现了 m个样本与其对应的第0个feature的点积。依次类推，我们可以得到$\frac{1}{m}X^T(h_\theta(x)-y)$

对梯度函数的正则化如下：

$$\begin{aligned}\frac{\partial J}{\partial \theta_0}&=\frac{1}{m}\sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})x_0^{\Phi i})\\ \frac{\partial J}{\partial \theta_j}&=\frac{1}{m}\sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})x_j^{\Phi i})+\frac{\lambda}{m}\theta_j \qquad \text{for} \quad j\geq 1\end{aligned}$$

正则化的矢量化操作比较简单，直接加上$\frac{\lambda}{m}\theta$，把θ的第一项θ_0赋值为0即可。