Mutinomial logistic 回归实现文本分类项目报告

蒋旗志

1. 二值分类问题与 logistic 回归

二值分类问题 y 值只能取 0 或 1。（在有的情况下取 y 为+1 和-1 更方便处理）例如判断一个邮件是不是垃圾邮件就是一个二值分类问题。通常我们把某件事情的出现表示为 1，例如一个邮件是垃圾邮件，称之为正例。某件事情若没有发生，则用 0 表示，叫做反例，如一封邮件为正常邮件。

一个 logistic 回归的假设函数 $h_\theta(x)$具有如下形式：

$$h_\theta(\mathrm{x}) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}},$$

这里 g(x)为：

$$g(s) = \frac{1}{1+e^{-s}},$$

称 g(x)为 logistic 函数或 sigmoid 函数。它是一个单调增函数，在负无穷到正无穷上取值为 0 到 1 之间。它可导，且有如下性质：

$$\begin{aligned} g'(s) &= \frac{d}{ds}\frac{1}{1+e^{-s}} = \frac{-1}{(1+e^{-s})^2}\frac{d}{ds}(1+e^{-s}) \\ &= \frac{e^{-s}}{(1+e^{-s})^2} = \frac{1}{1+e^{-s}}\left(1-\frac{1}{1+e^{-s}}\right) \\ &= g(s)(1-g(s)), \end{aligned}$$

我们假设：

$$\begin{aligned} P\{y=1 \mid x;\theta\} &= h_\theta(x), \\ P\{y=0 \mid x;\theta\} &= 1-h_\theta(x), \end{aligned}$$

在给定 x 与θ的情况下，y 服从伯努利分布，可以更紧凑的表示为：

$$p(y \mid x;\theta) = h_\theta(x)^y (1-h_\theta(x))^{1-y}$$

假设 m 个样本是独立产生的，则它的对数似然函数为：

$$\begin{aligned}\ell(\theta) &= \log p(\vec{\mathrm{y}} \mid \vec{x};\theta) \\ &= \sum_{i=1}^{m} \log p(\mathrm{y}^i \mid x^i;\theta) \\ &= \sum_{i=1}^{m} \log h_\theta(x^i)^{y^i}(1-h_\theta(x^i))^{1-y^i} \\ &= \sum_{i=1}^{m} \mathrm{y}^i \log h_\theta(x^i) + (1-\mathrm{y}^i)\log(1-h_\theta(x^i))\end{aligned}$$

为了求出极大似然值，对对数似然函数关于参数求导：

$$\begin{aligned}\frac{\partial \ell(\theta)}{\partial \theta} &= \log p(\vec{\mathrm{y}} \mid \vec{x};\theta) \\ &= \sum_{i=1}^{m} \mathrm{y}^i \frac{\partial \log h_\theta(x^i)}{\partial \theta} + (1-\mathrm{y}^i)\frac{\partial(1-\log h_\theta(x^i))}{\partial \theta}\log(1-h_\theta(x^i))\end{aligned}$$

为了求出这个和式的关于参数的导数，我们先求它的通项关于参数的导数：

$$\begin{aligned}&\mathrm{y}^i \frac{\partial \log h_\theta(x^i)}{\partial \theta} + (1-\mathrm{y}^i)\frac{\partial \log(1-\log h_\theta(x^i))}{\partial \theta} \\ &= \mathrm{y}^i \frac{1}{h_\theta(x^i)}\frac{\partial h_\theta(x^i)}{\partial \theta} - (1-\mathrm{y}^i)\frac{1}{(1-\log h_\theta(x^i))}\frac{\partial h_\theta(x^i)}{\partial \theta} \\ &= \mathrm{y}^i \frac{1}{h_\theta(x^i)} h_\theta(x^i)(1-h_\theta(x^i))x^i - (1-\mathrm{y}^i)\frac{1}{(1-\log h_\theta(x^i))} h_\theta(x^i)(1-h_\theta(x^i))x^i \\ &= \mathrm{y}^i(1-h_\theta(x^i))x^i - (1-\mathrm{y}^i)h_\theta(x^i)x^i \\ &= (\mathrm{y}^i - h_\theta(x^i))x^i\end{aligned}$$

其中第二个等式用到了 logistic 函数的导数。需要注意的是这是一个函数对一个参数

向量求偏导（一个函数关于一个向量求偏导的结果就是这个函数对每个分量求偏导所组成的

向量），所以它的结果也是一个向量。

为了求出似然函数最大时所对应的参数向量值，我们用随机梯度上升（stochastic gradient ascent）算法更新参数向量：

$$\theta := \theta + \alpha(\mathrm{y}^i - h_\theta(x^i))x^i$$

我们想要使似然函数尽可能大所以用梯度上升算法。我们还注意到每新来一个样本 x^i 参数向量θ就会更新一次，这也是为什么称它为随机梯度的原因。

此外，我们还可以用梯度上升算法：

$$\theta := \theta + \alpha\sum_{i=1}^{m}(\mathrm{y}^i - h_\theta(x^i))x^i$$

它与随机梯度上升算法的区别在于梯度上升算法的梯度是在全体样本上计算的梯度，参数的每次更新都需要计算参数向量在全体样本上的梯度。

2. 程序向量化

在实际更新参数向量的过程中，我们还需要注意尽量用矩阵运算代替 for 循环，尽量使用 numpy 中的内置函数。下面我们以梯度上升算法为例说明这个原则。我们有数据矩阵 X 与标签向量 y：

$$X=\begin{pmatrix}(x^1)^T\\(x^2)^T\\\vdots\\(x^m)^T\end{pmatrix},y=\begin{pmatrix}y^1\\y^2\\\vdots\\y^m\end{pmatrix}$$

每个行向量的维数为 n，参数向量θ的维数也为 n。

首先，我们需要计算全体梯度的和，而计算这个量时需要先计算向量 h（也就是 h 在每个样本上的值组成 m 维向量）具体地，我们可以先计算-Xθ得到一个 m 维列向量，然后对每个分量求指数（用 np.exp()）再算 1/（1+np.exp(-Xθ)）.

梯度上升算法后面的和式实质上就是 X 行向量的加权后求和，其中权值向量 z 就是 y-h

向量，故梯度上升算法的向量化版本为：

$$\theta := \theta + \alpha(\mathrm{z}^T\mathrm{X})^T$$

3. Softmax 回归与多分类问题

Logistic 回归适合于二值分类问题，在处理多值分类问题时我们用 softmax 分类器（Softmax Classifier ）。Softmax 回归（Softmax Regression），又叫做多元 logistic 回归(Multinomial Logistic Regression)可以看做 logistic 回归的推广。

Softmax 回归假设函数 $h_\theta(x)$具有如下形式：

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)}=1|x^{(i)};\theta) \\ p(y^{(i)}=2|x^{(i)};\theta) \\ \vdots \\ p(y^{(i)}=k|x^{(i)};\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$$

与 logistic 回归类似，它的代价函数（代价函数是对数似然函数的相反数，上面我们最大化似然函数事实上就是在最小化代价函数）为：

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)}=j\right\}\log\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right]$$

与 logistic 回归的推导过程类似，用梯度下降法最小化代价函数，我们得到权值更新的公式：

$$\theta_k := \theta_k + \alpha \sum_{i=1}^{m} (\mathrm{y}^i - h_\theta(x^i))_k \, x^i, \mathrm{k} = 1, 2, \cdots, \mathrm{c}.$$

这里每个θ_k都是是一个参数向量，和式里面下标 k 为类别向量第 k 个分量。类似地，向量化更新公式为：

$$\theta_k := \theta_k + \alpha(\mathrm{z}^T \mathrm{X})^T$$

这里的 z 为 y-h 这个 m×c 矩阵的第 k 列。事实上我们还可以对权值更新公式进一步向量化：

$$\theta := \theta + \alpha((\mathrm{y} - \mathrm{h})^T \mathrm{X})^T$$

此时θ为 n×c 矩阵，每一列表示某一类的权值向量。

4. 程序思路

程序主要由 readDataIntoList(largefile)，

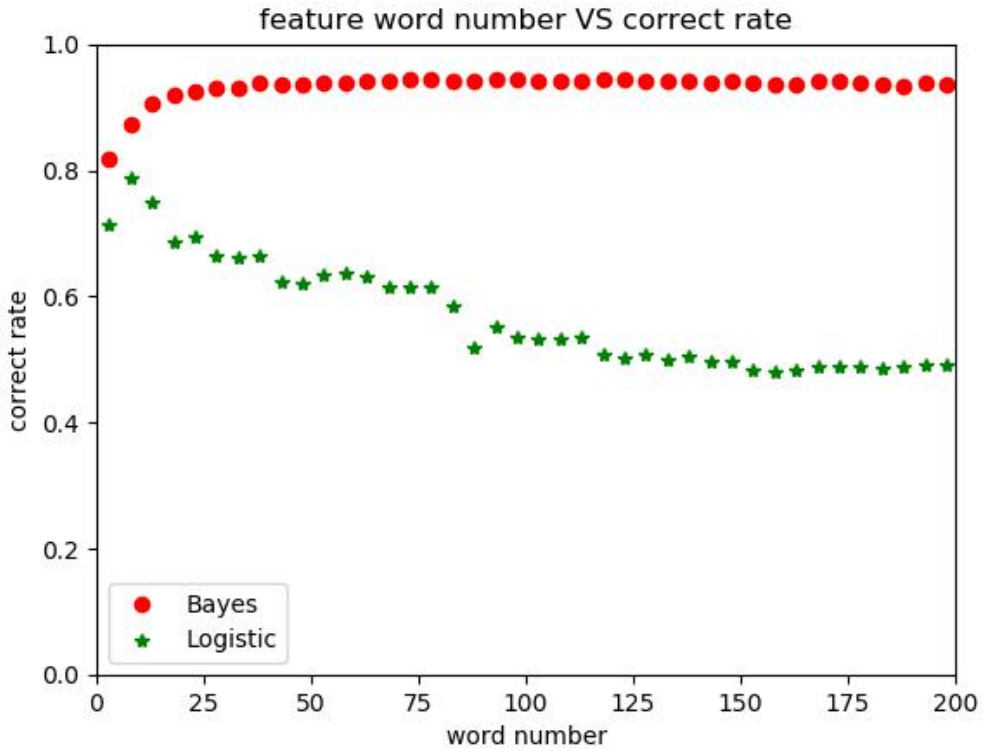
readFeaturewordtoset(filename, num=200)，listToNumpy(featurewords, datalist)三个函数负责数据预处理。需要注意的是，我们直接采用了项目一合并所有小文件后的大文件big.txt 与计算出的特征词文件 featureword.txt.

训练与预测过程主要由 softmaxtrain(X, Y, iternum= 10000, rate = 0.1)，softmaxpredict(W, X, Y)，crossValidate(X, Y, turns_num)三个函数实现，其中训练函数用了矩阵运算尽量避免 for 循环以提高程序执行效率。

为了比较特征词数量对模型准确率的影响，又写了个 wordnumTocorrectRate()函数，在这个函数中我们调用了 softmax 和朴素贝叶斯算法，并以散点图的形式展现了两个算法的区别。

5. 程序结果

每个类别选 100 特征词时，sofmax 平均准确率为 52%，尝试去除截距项，增加迭代次数，增大学习率均无效果。当每个类别特征词为 200 时，平均准确率为 48.6%，贝叶斯 93.4%.减少每个类别的特征词，当每个类别特征词为 50 时，平均准确率为 63%，当每个类别特征词为 25 时，平均准确率为 68%，当每个类别特征词为 12 时，平均准确率为 74.8%，贝叶斯 90.4%，当每个类别特征词为 6 时，平均准确率为 77.9%，贝叶斯 87.3%.



图中每个点代表经过 4 折交叉验证后的平均正确率，我们从每个类选三个词开始，以 5

为步长画出两类分类器在不同特征词数量下准确率的散点图。从上图我们可以看出，朴素贝叶斯模型对特征词的稳定性相当好，只需要很少的特征词就可以获得相当高的准确率，且几乎不受不相关特征词的干扰，而 logistic 回归模型对特征词个数十分敏感，且在整体上低于贝叶斯分类器正确率。

程序运行结果表明，当从每个类中选取特征词个数为 9 时，也就是总共选取 90 个特征词时，多元 logistic 回归正确率达到最高，为 80.0%.

2017.10.24

参考资料：

1. http://ufldl.stanford.edu/wiki/index.php/Softmax%E5%9B%9E%E5%BD%92

2.http://cs229.stanford.edu/section/vec_demo/Vectorization_Section.pdf