高级机器学习
作业一

MG1833067, 汪浩港, whg19961229@gmail.com

2018 年 12 月 11 日

1 [25pts] Multi-Class Logistic Regression

教材的章节 3.3 介绍了对数几率回归解决二分类问题的具体做法。假定现在的任务不再是二分类问题，而是多分类问题，其中 $y \in \{1, 2 \dots, K\}$。请将对数几率回归算法拓展到该多分类问题。

(1) **[15pts]** 给出该对率回归模型的"对数似然"(log-likelihood);

(2) **[5pts]** 计算出该"对数似然"的梯度。

提示 1：假设该多分类问题满足如下 $K-1$ 个对数几率，

$$\begin{aligned}
\ln \frac{p(y=1|\mathbf{x})}{p(y=K|\mathbf{x})} &= \mathbf{w}_1^{\mathrm{T}}\mathbf{x} + b_1 \\
\ln \frac{p(y=2|\mathbf{x})}{p(y=K|\mathbf{x})} &= \mathbf{w}_2^{\mathrm{T}}\mathbf{x} + b_2 \\
&\cdots \\
\ln \frac{p(y=K-1|\mathbf{x})}{p(y=K|\mathbf{x})} &= \mathbf{w}_{K-1}^{\mathrm{T}}\mathbf{x} + b_{K-1}
\end{aligned}$$

提示 2：定义指示函数 $\mathbb{I}(\cdot)$，

$$\mathbb{I}(y=j) = \begin{cases} 1 & \text{若 } y \text{ 等于 } j \\ 0 & \text{若 } y \text{ 不等于 } j \end{cases}$$

Solution. 此处用于写解答 *(中英文均可)*

(1) 假设该多分类问题满足如下对数几率，

$$\ln \frac{p(y=i|\boldsymbol{x})}{p(y=K|\boldsymbol{x})} = \boldsymbol{w}_i^{\top}\boldsymbol{x} + b_i, i = 1, 2, \dots, K-1$$

可得

$$p(y=i|\boldsymbol{x}) = e^{\boldsymbol{w}_i^{\top}\boldsymbol{x}+b_i} p(y=K|\boldsymbol{x}), i = 1, 2, \dots, K-1$$

$$
\begin{aligned}
\sum_{i=1}^{K} p(y=i|\boldsymbol{x}) = (\sum_{i=1}^{K-1} e^{\boldsymbol{w}_i^\top \boldsymbol{x}+b_i}+1)p(y=K|\boldsymbol{x}) = 1 \\
p(y=K|\boldsymbol{x}) = \frac{1}{1+\sum_{i=1}^{K-1} e^{\boldsymbol{w}_i^\top \boldsymbol{x}+b_i}} \\
p(y=i|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_i^\top \boldsymbol{x}+b_i}}{1+\sum_{i=1}^{K-1} e^{\boldsymbol{w}_i^\top \boldsymbol{x}+b_i}}, i=1,2,\ldots,K-1\,.
\end{aligned}
$$

定义指示函数 $\mathbb{I}(\cdot)$

$$
\mathbb{I}(y=j) = \begin{cases} 1 & \text{若 } y \text{ 等于 } j \\ 0 & \text{若 } y \text{ 不等于 } j \end{cases}
$$

,

设数据集为 $\{(x_i, y_i)\}_{i=1}^m$，对于任意 y_i，有

$$
\sum_{j=1}^{K} \mathbb{I}(y_i=j) = 1
$$

则对数似然如下，

$$
\begin{aligned}
\ell(w,b) &= \sum_{i=1}^{m}\sum_{j=1}^{K} \mathbb{I}(y_i=j)\ln p(y_i=j|\boldsymbol{x}_i) \\
&= \sum_{i=1}^{m}\left(\sum_{j=1}^{K-1} \mathbb{I}(y_i=j)(\boldsymbol{w}_j^\top \boldsymbol{x}_i + b_j + \ln p(y_i=K|\boldsymbol{x}_i)) + \mathbb{I}(y_i=K)\ln p(y_i=K|\boldsymbol{x}_i)\right) \\
&= \sum_{i=1}^{m}\left(\sum_{j=1}^{K-1} \mathbb{I}(y_i=j)(\boldsymbol{w}_j^\top \boldsymbol{x}_i + b_j) + \sum_{j=1}^{K} \mathbb{I}(y_i=j)\ln p(y_i=K|\boldsymbol{x}_i)\right) \\
&= \sum_{i=1}^{m}\left(\sum_{j=1}^{K-1} \mathbb{I}(y_i=j)(\boldsymbol{w}_j^\top \boldsymbol{x}_i + b_j) - \ln(1+\sum_{k=1}^{K-1} e^{\boldsymbol{w_k}^\top \boldsymbol{x_i}+\boldsymbol{b_k}})\right)
\end{aligned}
\tag{1.1}
$$

(2) 由上一问可知对数似然如 *Eq.1.1*形式。

令 $\boldsymbol{\beta}=(\boldsymbol{w},b)$,$\hat{\boldsymbol{x}}=(\boldsymbol{x},1)$,$\boldsymbol{\beta_j}=(\boldsymbol{w_j},b_j)$,$\hat{\boldsymbol{x_j}}=(\boldsymbol{x_j},1)$，则该对数似然的梯度为：

$$
\begin{aligned}
\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta_j}} &= \sum_{i=1}^{m}\left(\mathbb{I}(y_i=j)\hat{\boldsymbol{x}}_i - \frac{e^{\boldsymbol{\beta_j}^\top \hat{\boldsymbol{x}}_i}}{1+\sum_{k=1}^{K-1} e^{\boldsymbol{\beta_k}^\top \hat{\boldsymbol{x}}_i}}\hat{\boldsymbol{x}}_i\right) \\
&= \sum_{i=1}^{m}(\mathbb{I}(y_i=j) - p(y_i=j|\hat{\boldsymbol{x}}_i))\hat{\boldsymbol{x}}_i\,.
\end{aligned}
\tag{1.2}
$$

2 [15pts] Semi-Supervised Learning

我们希望使用半监督学习的方法来对文本文档进行分类。假设我们使用二进制指示符的词袋模型描述各个文档，在这里，我们的词库有 10000 个单词，因此每个文档由长度为 10000 的二进制向量表示。

对于以下提出的分类器，说明其是否可以用于改进学习性能并提供简要说明。

1. **[5pts]** 使用 EM 的朴素贝叶斯；
2. **[5pts]** 使用协同训练的朴素贝叶斯；
3. **[5pts]** 使用特征选择的朴素贝叶斯；

Solution. 此处用于写解答 *(中英文均可)*

1. 使用 *EM* 的朴素贝叶斯：

EM 算法使用两个步骤交替计算，第一步是期望步，利用当前估计的参数值来计算对数似然的期望值；第二步是最大化，寻找能使第一步产生的似然期望最大化的参数值。然后，新得到的参数值重新被用于第一步。EM 算法其实就是一个迭代的过程。可以更好地帮助其收敛。最大化先确认最大期望，相当于确认了下界，然后用最大化来提高这个界，这样一步一步就可以优化。
使用 EM 的朴素贝叶斯可以适用于文本分类。文本分类问题有三个维度来描述——类别、特征、样本。首先，仅从标记文档估计朴素贝叶斯参数 。然后，分类器用于通过计算缺失类标签 $p(c_j \mid d_i;\theta)$ 的期望来将概率加权的类标签分配给每个未标记的文档。接下来，使用原始和新标记的所有文档来估计新的分类器参数 。迭代这最后两步直到 不变。

它的优点也很显然，朴素贝叶斯假设特征之间是相互独立的，假设太强，可以结合 EM 学习大量未标记样本，减少因特征相关性造成的分类误差。

2. 使用协同训练的朴素贝叶斯：

协同训练是一种多视角学习方法。

1. 首先分别在每个视图上利用有标记样本训练一个分类器；

2. 然后，每个分类器从未标记样本中挑选若干标记置信度 (即对样本赋予正确标记的置信度) 高的样本标记赋予“伪标记”，并把这些“伪标记”样本 (即其标记是由学习器给出的) 加入另一个分类器的训练集中，以便对方利用这些新增的有标记样本进行更新。

这个“互相学习、共同进步”的过程不断迭代进行下去，直到两个分类器都不再发生变化，或达到预先设定的学习轮数为止。这样可以通过一群低泛化性的贝叶斯分类器来逼近一个泛化性高的贝叶斯。

3. 使用特征选择的朴素贝叶斯：

对于贝叶斯分类器，如果估计的参数过多，必然需要很大的样例，但是在半监督学习中，训练的规模总是有限的，这样就会导致数据稀疏性问题的出现，特征选择从训练集中选出一部分子集，减小特征空间，去除噪声特征来提高分类器训练的效率和精度。可以使用词袋模型预处理，词袋模型的主要思想，是构建各类文本的词典，然后针对每一个文本，

计算该文本每个词在词典中对应位置出现的次数。可以在构建词袋前对文本进行预处理，选择一批特征词进行构建。使用这些特征词构建词典，可以防止构建的词典过于庞大，即不利于存储，也不利于后续词频统计运算等。

3 [60pts] Dimensionality Reduction

请实现三种降维方法：PCA，SVD 和 ISOMAP，并在降维后的空间上用 1-NN 方法分类。

1. 数据：我们给出了两个数据集，都是二分类的数据。可以从https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html找到，同时也可以在提交作业的目录文件夹中找名为“two datasets”的压缩文件下载使用。每个数据集都由训练集和测试集组成。

2. 格式：再每个数据集中，每一行表示一个带标记的样例，即每行最后一列表示对应样例的标记，其余列表示对应样例的特征。

具体任务描述如下：

1. **[20pts]** 请实现 PCA 完成降维(方法可在参考书http://www.charuaggarwal.net/Data-Mining.htm 中 Section 2.4.3.1 中找到)

   首先，仅使用训练数据学习投影矩阵；

   其次，用学得投影矩阵将训练数据与测试数据投影到 k-维空间 ($k = 10, 20, 30$)；

   最后，在降维后空间上用 1-NN 预测降维后 k 维数据对应的标记 ($k = 10, 20, 30$)，并汇报准确率。注意，测试数据集中的真实标记仅用来计算准确率。

2. **[20pts]** 请实现 SVD 完成降维（方法在上述参考书 Section 2.4.3.2 中找到）

   首先，仅使用训练数据学习投影矩阵；

   其次，用学得投影矩阵将训练数据与测试数据投影到 k-维空间 ($k = 10, 20, 30$)；

   最后，在降维后空间上用 1-NN 预测降维后 k 维数据对应的标记 ($k = 10, 20, 30$)，并汇报准确率。注意，测试数据集中的真实标记仅用来计算准确率。

3. **[20pts]** 请实现 ISOMAP 完成降维（方法在参考书 Section 3.2.1.7 中找到）

   首先，使用训练数据与测试数据学习投影矩阵。在这一步中，请用 4-NN 来构建权重图。（请注意此处 4 仅仅是用来举例的，可以使用其他 k-NN, $k \geq 4$ 并给出你选择的 k。如果发现构建的权重图不连通，请查找可以解决该问题的方法并汇报你使用的方法）

   其次，用学得投影矩阵将训练数据与测试数据投影到 k-维空间 ($k = 10, 20, 30$)。

   最后，在降维后空间上用 1-NN 预测降维后 k 维数据对应的标记 ($k = 10, 20, 30$)，并汇报准确率。注意，测试数据集中的真实标记仅用来计算准确率。

可以使用已有的工具、库、函数等直接计算特征值和特征向量，执行矩阵的 SVD 分解，计算 graph 上两个节点之间的最短路。PCA/SVD/ISOMAP 和 1-NN 中的其他步骤必须由自己实现。

报告中需要包含三个方法的伪代码和最终的结果。最终结果请以表格形式呈现，表中包含三种方法在两个数据集中，不同 $k = 10, 20, 30$ 下的准确率。

Solution. *此处用于写解答 (中英文均可)*

1. *PCA:*

使用训练数据学习投影矩阵，对测试集使用训练数据的均值去中心化，与投影矩阵进行矩阵相乘得到降维后的数据，用 $1NN$ 预测样本标签，计算准确率，结果如表 *1*。

表 *1*

Acc	$k = 10$	$k = 20$	$k = 30$
sonar	*58.2524%*	*56.3107%*	*56.3107%*
splice	*75.8161%*	*76.2759%*	*73.5632%*

算法 1 用 PCA 降维

输入: 数据集 $D = x_1, x_2, \ldots, x_m$；降维后的维数 d'

输出: 投影矩阵 $W = (w_1, w_2, \ldots, w_{d'})$

```
function PCA(D, d')
    对数据集进行中心化: D ← D − mean(D)
    计算协方差矩阵: convD ← DD^T
    对协方差矩阵作特征值分解: eigenValues ← λ(convD)
    取最大的 d' 个特征值所对应的特征向量: w_1, w_2, ..., w_d'
    return W = (w_1, w_2, ..., w_d')
end function
```

2. *SVD:*

使用训练数据学习投影矩阵，将测试集与投影矩阵进行矩阵相乘得到降维后的数据，用 $1NN$ 预测样本标签，计算准确率，结果如表 *2*。

算法 2 用 SVD 降维

输入: 数据集 $D = x_1, x_2, \ldots, x_m$；降维后的维数 d'

输出: 投影矩阵 $W = (w_1, w_2, \ldots, w_{d'})$

```
function SVD(D, d')
    计算右奇异矩阵 P
    取右奇异矩阵的左边的 d' × d' 列: W ← P(: d', :)
    return W
end function
```

表 *2*

Acc	$k = 10$	$k = 20$	$k = 30$
sonar	*59.2233%*	*58.2524%*	*56.3107%*
splice	*75.8621%*	*76.4138%*	*74.8046%*

3. *ISOMAP:*

使用训练数据与测试数据拼接后作为样本集作为 *isomp* 的输入得到降维后的数据集，取出测试集对应的部分，用 $1NN$ 预测样本标签，计算准确率，结果如表 *3*。对于 *sonar*

算法 3 用 ISOMAP 降维

输入: 数据集 $D = x_1, x_2, \ldots, x_m$；降维后的维数 d'；近邻参数 k

输出: 降维后的数据集 D'

function ISOMAP($D, d'\ k$)
    for $i = 1, 2, \ldots, m$ **do**
        确定 x_i 的 k 近邻集合 KNN；
        for $j = 1, 2, \ldots, m$ **do**
            if $x_j \in KNN$ **then**
                $Dis_{ij} \leftarrow x_i$ 与 x_j 之间的欧式距离
            else
                $Dis_{ij} \leftarrow \infty$
            end if
        end for
    end for 判断当前带权图是否联通，如果不联通，k=k+1，回到上面循环的开头重新生成带权无向图
    调用最短路径算法获取任意两样本之间最短路径长度 $dist$，输入是 Dis
    return MDS$\{dist\}$
end function

算法 4 MDS

输入: 距离矩阵 $D \in \mathcal{R}^{m\times m}$, 其元素 $dist_{ij}$ 为样本 x_i 到样本 y_i 的距离；降维后的维数 d'

输出: 降维后的数据集 D'

function MDS(D)
    $dist^2_{i\cdot} = \frac{1}{m}\sum_{j=1}^{m} dist^2_{ij}$
    $dist^2_{\cdot j} = \frac{1}{m}\sum_{i=1}^{m} dist^2_{ij}$
    $dist^2_{\cdot\cdot} = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{m} dist^2_{ij}$
    $b_{ij} = -\frac{1}{2}(dist^2_{ij} - dist^2_{i\cdot} - dist^2_{\cdot j} + dist^{)}_{\cdot\cdot}$
    $B = b_{ij}, i, j = 1, 2, \ldots, m$
    对矩阵 B 做特征值分解
    取 Λ 为最大的 d' 个特征值所构成的对角矩阵，$\boldsymbol{V}$ 为对应的特征向量的矩阵
    return $\Lambda\boldsymbol{V}^{1/2} \in \mathcal{R}^{m\times d'}$
end function

数据集，当 KNN 中的 $k=6$ 时可以构造出全连通的带权无向图；而对于 *splice* 数据集，当 KNN 中的 $k=4$ 时可以构造出全连通的带权无向图，

表 *3*

$4-NN$	$k=10$	$k=20$	$k=30$
sonar	*41.7476%*	*41.7476%*	*43.6893%*
splice	*68.0920%*	*69.0115%*	*69.1954%*