"""

Created on 2017/7/27

machine-learning-course

@author: DSG
"""

● 梯度

某一点的所有方向导数中数值最大的方向导数
梯度的本意是一个向量（矢量），表示某一函数在该点处的方向导数沿着该方向取得最大值，即函数在该点处沿着该方向（此梯度的方向）变化最快，变化率最大（为该梯度的模）。

● Logistic 回归原理

Logistic 回归是一种广义线性回归，常用的分类器函数是 Sigmoid 函数，其公式如下：

$$\sigma(z)=\frac{1}{1+e^{-z}}$$

其中 z，可由下面公式得出：

$$z=w_0x_0+w_1x_1+w_2x_2+\cdots+w_nx_n$$

其中 x 表示样本数据，在这里 x 为已知参数，w 为未知变量，如果采用向量的写法，上面的公式可以写成：

$$z=w^Tx$$

我们的主要任务是找到最佳参数 w 使得分类器尽可能准确，怎么来求这组参数呢？

● 梯度上升算法

这组样本既然已经被抽到，则就认为这组样本出现的概率最大，而这组样本服从 $X^{(i)}\sim\sigma(w)$ 分布，现在只要求的参数 w ,那么就可以得到一个完整的估计函数，而似然函数的最大值就用来描述已经抽出的这组样本就是全部样本空间中抽出这么多样本最应该出现的结果。似然用来计算已知随机变量输出结果时，求未知参数的可能取值，实质上他是一个关于参数 w 的函数，因此用求出似然函数在最大值附近的参数 w 的值就可构建完整的预测函数，所以这个似然函数必须有 w 来构建，至于求似然函数的最大值，这里采用梯度上升的方法来求似然函数的最大值，在这里没有采用解方程的方法式来求，因为梯度上升算法的公式是收敛的，所以这里只是用梯度函数迭代 500 次用近似的方法来估计最大值。在计算 2000 次左右基本可以达到极值。

1. 梯度

函数 $f(x,y)$ 的梯度的公式如下：

$$\nabla f(x,y)=\begin{bmatrix}\frac{\partial f(x,y)}{\partial x}\\ \frac{\partial f(x,y)}{\partial y}\end{bmatrix}$$

梯度上升算法的原理

在这里的梯度是一个关于 w 的函数

$$w=w+\alpha\nabla_w f(w)$$

2. 目标函数

假定样本与样本之间相互独立，那么整个样本集生成的概率即为所有样本生成概率的乘积
在实测试的例子中，只有两组类别，所以就考虑两二分类的问题，那么就可以得到他的预测公式

$$h_w(x)=\sigma(w^Tx)=\frac{1}{1+e^{-w^Tx}}$$

$h_w(x)$ 的值表示 $y=1$的概率，因此分类结果为类别 1 和类别 0 的概率分别为：

$$P(y=1\mid x;w)=h_w(x)$$
$$P(y=0\mid x;w)=1-h_w(x)$$

合起来写就是：

$$P(y\mid x;w)=(h_w(x))^y(1-h_w(x))^{1-y}$$

这个就是 logistics 密度函数。

那么他的似然函数为密度函数的乘积：

$$L(w)=\prod_{i=1}^{m}P(y^{(i)}\mid x^{(i)};w)=\prod_{i=1}^{m}(h_w(x^{(i)}))^{y^{(i)}}(1-h_w(x^{(i)}))^{1-y^{(i)}}$$

m 为样本总数。

对数似然函数为：

$$l(w)=\log L(w)=\sum_{i=1}^{m}(y^{(i)}\log h_w(x^{(i)})+(1-y^{(i)})\log(1-h_w(x^{(i)})))$$

最大似然估计就是要求使得 $l(w)$ 取值最大时的 w，所以我们的目标函数就是 $l(w)$。

梯度求法：

$$\begin{aligned}
\frac{\partial l(w)}{\partial w_j}&=\sum_{i=1}^{m}(y^{(i)}\frac{1}{h_w(x^{(i)})}\frac{\partial h_w(x^{(i)})}{\partial w_j}-(1-y^{(i)})\frac{1}{1-h_w(x^{(i)})}\frac{\partial h_w(x^{(i)})}{\partial w_j})\\
&=\sum_{i=1}^{m}(y^{(i)}\frac{1}{\sigma(w^Tx^{(i)})}-(1-y^{(i)})\frac{1}{1-\sigma(w^Tx^{(i)})})\frac{\partial\sigma(w^T(x^{(i)}))}{\partial w_j}\\
&=\sum_{i=1}^{m}(y^{(i)}\frac{1}{\sigma(w^Tx^{(i)})}-(1-y^{(i)})\frac{1}{1-\sigma(w^Tx^{(i)})})\frac{\partial(1+e^{-w^Tx})^{-1}}{\partial w_j}\\
&=\sum_{i=1}^{m}(y^{(i)}\frac{1}{\sigma(w^Tx^{(i)})}-(1-y^{(i)})\frac{1}{1-\sigma(w^Tx^{(i)})})\frac{-(1+e^{-w^Tx})^{-2}\cdot e^{-w^Tx}\cdot-\partial(w^Tx^{(i)})}{\partial w_j}\\
&=\sum_{i=1}^{m}(y^{(i)}\frac{1}{\sigma(w^Tx^{(i)})}-(1-y^{(i)})\frac{1}{1-\sigma(w^Tx^{(i)})})(1+e^{-w^Tx})^{-2}\cdot e^{-w^Tx}\cdot\frac{\partial(w^Tx^{(i)})}{\partial w_j}\\
&=\sum_{i=1}^{m}(y^{(i)}\frac{1}{\sigma(w^Tx^{(i)})}-(1-y^{(i)})\frac{1}{1-\sigma(w^Tx^{(i)})})\sigma(w^Tx^{(i)})(1-\sigma(w^Tx^{(i)}))\frac{\partial w^Tx^{(i)}}{\partial w_j}\\
&=\sum_{i=1}^{m}(y^{(i)}(1-\sigma(w^Tx^{(i)}))-(1-y^{(i)})\sigma(w^Tx^{(i)}))x^{(i)}\\
&=\sum_{i=1}^{m}(y^{(i)}-\sigma(w^Tx^{(i)}))x^{(i)}\\
&=\sum_{i=1}^{m}(y^{(i)}-h_w(x^{(i)}))x^{(i)}
\end{aligned}$$

其中，m 为样本的总数，$y^{(i)}$ 表示第 i 个样本的类别，$x^{(i)}$ 表示第 i 个样本，需要注意的是 w 是多维向量，$x^{(i)}$ 也是多维向量。j 代表第 i 样本的第 j 个分量。

那么最后的梯度迭代公式就是：

$$w=w+\alpha\sum_{i=1}^{m}(y^{(i)}-h_w(x^{(i)}))x^{(i)}$$
$$\vec{w}=\vec{w}+\alpha\sum_{i=1}^{m}\left(y^{(i)}-\frac{1}{1+e^{-\vec{w}^Tx}}\right)\cdot x^{(i)}$$

有趣的是 $y^{(i)} - h_w(x^{(i)})$ 刚好就是估计误差，是数据原本的类别减去估计的类别。
代码里面只要将这个式子迭代无数次，或者直到误差在接受的范围以内。

● 梯度下降

构建损失函数，然后梯度下降算法，结果一样。