逻辑回归

Leon

2018 年 12 月 21 日

1 机器学习的概念

1.1 有监督学习

对训练集来说X对应着是确定的f(x)，然后通过构建模型，进行超参的学习

1.2 无监督学习

大部分无监督学习都是没有确定的f(x)的，通过一些规则，让机器自己去判断，比如knn算法，用距离来做聚类。

1.3 泛化能力

在机器学习方法中，泛化能力通俗来讲就是指学习到的模型对未知数据的预测能力。在实际情况中，我们通常通过测试误差来评价学习方法的泛化能力。

1.4 过拟合

1.4.1 概念

先谈谈过拟合，所谓过拟合，指的是模型在训练集上表现的很好，但是在交叉验证集合测试集上表现一般，也就是说模型对未知样本的预测表现一般，泛化（generalization）能力较差。

1.4.2 解决办法

一般的方法有early stopping、数据集扩增（Data augmentation）、正则化（Regularization）、Dropout等。在机器学习算法中，我们常常将原

始数据集分为三部分：training data、validation data，testing data。这个validation data是什么？它其实就是用来避免过拟合的，在训练过程中，我们通常用它来确定一些超参数（比如根据validation data上的accuracy来确定early stopping的epoch大小、根据validation data确定learning rate等等）。那为啥不直接在testing data上做这些呢？因为如果在testing data做这些，那么随着训练的进行，我们的网络实际上就是在一点一点地overfitting我们的testing data，导致最后得到的testing accuracy没有任何参考意义。

Early stopping: Early stopping便是一种迭代次数截断的方法来防止过拟合的方法，即在模型对训练数据集迭代收敛之前停止迭代来防止过拟合。对模型进行训练的过程即是对模型的参数进行学习更新的过程，这个参数学习的过程往往会用到一些迭代方法，如梯度下降（Gradient descent）学习算法。这样可以有效阻止过拟合的发生，因为过拟合本质上就是对自身特点过度地学习。

正则化: 指的是在目标函数后面添加一个正则化项，一般有L1正则化与L2正则化。L1正则是基于L1范数，即在目标函数后面加上参数的L1范数和项，即参数绝对值和与参数的积项

$$C = C_0 + \frac{\lambda}{n}\sum_w |w|$$

L2正则是基于L2范数，即在目标函数后面加上参数的L2范数和项，即参数的平方和与参数的积项：

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2$$

1.5 交叉验证(cross-validation)

交叉验证，是重复的使用数据，把得到的样本数据进行切分，组合为不同的训练集和测试集，用训练集来训练模型，用测试集来评估模型预测的好坏。在此基础上可以得到多组不同的训练集和测试集，某次训练集中的某样本在下次可能成为测试集中的样本，即所谓“交叉”。有简单交叉验证、S折交叉验证、留一交叉验证。

1.6 线性回归的原理

1:函数模型(Model):

$$h_w(x^i) = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + ... + \omega_n x_n = \sum \omega^T x_i = W^T X$$

$$X = \begin{bmatrix} 1 \\ x_1 \\ ... \\ x_n \end{bmatrix}, W = \begin{bmatrix} \omega_0 \\ \omega_2 \\ ... \\ \omega_n \end{bmatrix} \tag{1}$$

假设有训练数据

$$D = (X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)$$

那么方便我们写成矩阵的形式

$$X = \begin{bmatrix} 1, x_n^1, x_2^1, ..., x_n^1 \\ 1, x_1^2, x_2^2, ..., x_n^2 \\ ..... \\ 1, x_1^n, x_2^n, ...x_n^n \end{bmatrix}, XW = h_\omega(x^i)$$

2.损失代价函数:

$$J(W) = \frac{1}{2M}\sum_{i=0}^{M}(h_\omega(x^i) - y^i)^2 = \frac{1}{2M}(XW - y)^T(XW - Y)$$

3.算法(algorithm): 求解使得损失函数最小。

1.7 优化方法

1.7.1 梯度下降法

梯度下降沿损失函数的导数方向下降，下降的步幅自己设置。

1.7.2 牛顿法

二阶下降，比梯度下降法更快，而且是求全局最优解，不是局部最优

1.7.3 拟牛顿法

没看懂，但知道适合非线性

1.8 sklearn参数

Ordinary least squares Linear Regression.

1.8.1 fit_intercept:boolean, optional, default True

whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (e.g. data is expected to be already centered).

1.8.2 normalize : boolean, optional, default False

This parameter is ignored when “fit_intercept“ is set to False. If True, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm. If you wish to standardize, please use :class:‘sklearn.preprocessing.StandardScaler‘ before calling “fit“ on an estimator with “normalize=False“.

1.8.3 copy_X : boolean, optional, default True

If True, X will be copied; else, it may be overwritten.

1.8.4 n_jobs : int or None, optional (default=None)

The number of jobs to use for the computation. This will only provide speedup for n_targets ¿ 1 and sufficient large problems.“None“ means 1.

2 逻辑回归

2.1 逻辑回归和线性回归的联系与区别

2.1.1 联系

线性回归和逻辑回归都是广义线性模型，具体的说，都是从指数分布族导出的线性模型，线性回归假设Y—X服从高斯分布，逻辑回归假设Y—X服从伯努利分布，这两种分布都是属于指数分布族，我们可以通过指数分布族求解广义线性模型（GLM）的一般形式，导出这两种模型。

2.1.2 区别

区别是线性回归是用来做预测任务，逻辑回归是用来做分类任务，把每一个点映射到0-1之间，都是用最大似然概率去求解参数值。

2.2 逻辑回归的原理

线性回归的模型是求出输出特征向量Y和输入样本矩阵X之间的线性关系系数 θ，满足$Y = X\theta^T$。我们的Y是连续的，所以是回归模型。如果Y是离散的话，可以想到的办法是，我们对于这个Y再做一次函数转换，变为g(Y)。如果我们令g(Y)的值在某个实数区间的时候是类别A，在另一个实数区间的时候是类别B，以此类推，就得到了一个分类模型。如果结果的类别只有两种，那么就是一个二元分类模型了。逻辑回归的出发点就

是从这来的。下面我们开始引入二元逻辑回归。也就是输入X，输出的Y，只有1,0两种情况,而线性回归的X*系数矩阵得到的值如果直接通过比较得出Y为1或0，是可以做，但是得到的结果无法进行再次学习，或者说很麻烦，优化方法跟导数有关，所以如果要优化，就得打造一个完美的连续函数，比如sigmoid，方便求导，而且两个极限值是0和1，但是有一个不好的地方，有可能求解到局部最优。

2.3 逻辑回归损失函数推导及优化

对线性回归的结果做一个函数g上的转换，可以变为逻辑回归，一般取为sigmoid函数,形式如下:

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

它有一个非常好的导数性质:

$$g'(z) = g(z)(1-g(z))$$

这个通过函数对g(z)求导很容易得到如果我们令g(z)中的z为:$z = x\theta$,这样就得到了二元逻辑回归模型的一般形式:

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

其中x为样本输入，$h_\theta(x)$为模型输出，可以理解为某一分类的概率大小。而θ为分类模型的要求出的模型参数。对于模型输出$h_\theta(x)$,我们让它和我们的二元样本输出y(假设0和1)有这样的对应关系,如果$h_\theta(x)$¿0.5,即$x\theta$¿0，则y为1，反之亦然，y=0.5是临界情况，此时无法确定分类,$h_\theta(x)$值越小，而分类为0的概率越高，反之，值越大的话分类为1的概率越高。如果靠近临界点，则分类准确率会下降。

此处将模型写成矩阵模式: $h_\theta(X) = \frac{1}{1+e^{-X\theta}}$

假设我们的样本输出是0或者1两类，那么我们有：

$$P(y=1|x,\theta) = h_\theta(x)$$

$$P(y=0|x,\theta) = 1-h_\theta(x)$$

把这两个式子写成一个式子，就是：

$$P(y|x,\theta) = h_\theta(x)^y(1-h_\theta(x))^{1-y}$$

其中y的取值只能是0或者1。用矩阵法表示，即为

$$P(Y|X,\theta) = h_\theta(X)^Y(E - h_\theta(X))^{1-Y}, v{-}E{:}UM_{\rm J}\Theta$$

得到了y的概率分布函数表达式，我们就可以用似然函数最大化来求解我们需要的模型系数 θ 。为了方便求解，这里用对数似然函数最大化，对数似然函数取反即为我们的损失函数J(θ)。其中：

$$L(\theta) = \prod_{i=1}^{m}(h_\theta(x^{(i)}))^{y(i)}(1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

其中m为样本的个数对似然函数对数化取反的表达式，即损失函数表达式为：

$$J(\theta) = -lnL(\theta) = -\sum_{i=1}^{m}(y^{(i)}log(h_\theta(x^{(i)})) + (1 - y^{(i)})log(1 - h_\theta(x^{(i)})))$$

损失函数用矩阵法表达更加简洁：
J(θ)= Y⊙logh θ (X) (E Y)⊙log(E h θ (X))
其中E为全1向量,⊙为哈达马乘积(对应位置相乘)。
对于J(θ)= Y⊙logh θ (X) (E Y)⊙log(E h θ (X))，我们用J(θ)对 θ 向量求导可得：

$$\frac{\partial J(\theta)}{\partial\theta} = \text{`}Y \bigodot X^T\frac{1}{h_\theta(X)} \bigodot h_\theta(X) \bigodot(E\text{`}h_\theta(X))+(E\text{`}Y) \bigodot X^T\frac{1}{1\text{`}h_\theta(X)} \bigodot h_\theta(X) \bigodot(E\text{`}h_\theta(X))$$

简化得到
$\frac{\partial}{\partial\theta}J(\theta) = X^T(h_\theta(X) - Y)$
从而在梯度下降法中每一步向量θ的迭代公式如下：
$\theta = \theta - \alpha X^T(h_\theta(X) - Y)$
其中，α为梯度下降法的步长。

2.4 正则化与模型评估指标

在最小化残差平方和的基础上加上L1范数或者L2范数的惩罚项，如果L2正则正则化就是-岭回归 Ridge Regression,L1正则化是lasso回归。回归模型评估指标有
1.解释方差
$Explained_variance(y, y_hat) = 1 - Var(y - y_hat)/Var(y)$ 2.绝对平均误差
3.均方误差
4.决定系数(R^2 score) 5.AIC 6.BIC

2.5 逻辑回归的优缺点

优点：

1）预测结果是界于0和1之间的概率；

2）可以适用于连续性和类别性自变量；

3）容易使用和解释；

缺点：

1）对模型中自变量多重共线性较为敏感，例如两个高度相关自变量同时放入模型，可能导致较弱的一个自变量回归符号不符合预期，符号被扭转。 需要利用因子分析或者变量聚类分析等手段来选择代表性的自变量，以减少候选变量之间的相关性；

2）预测结果呈“S”型，因此从log(odds)向概率转化的过程是非线性的，在两端随着 log(odds)值的变化，概率变化很小，边际值太小，slope太小，而中间概率的变化很大，很敏感。导致很多区间的变量变化对目标概率的影响没有区分度，无法确定阀值。

2.6 样本不均衡问题的解决办法

分类时，由于训练集合中各样本数量不均衡，导致模型训偏在测试集合上的泛化性不好。解决样本不均衡的方法主要包括两类：（1）数据层面，修改各类别的分布；（2）分类器层面，修改训练算法或目标函数进行改进。还有方法是将上述两类进行融合。

2.6.1 数据层面

过采样

1.基础版本的过采样：随机过采样训练样本中数量比较少的数据；缺点，容易过拟合；

2.改进版本的过采样：SMOTE，通过插值的方式加入近邻的数据点；

3.基于聚类的过采样：先对数据进行聚类，然后对聚类后的数据分别进行过采样。这种方法能够降低类间和类内的不平衡。

4.神经网络中的过采样：SGD训练时，保证每个batch内部样本均衡。

欠采样

与过采样方法相对立的是欠采样方法，主要是移除数据量较多类别中的部分数据。这个方法的问题在于，丢失数据带来的信息缺失。为克服这一缺点，可以丢掉一些类别边界部分的数据。

2.6.2 分类器层面

过采样，欠采样，都存在相应的问题。

过采样：可能会存在过拟合问题。（可以使用SMOTE算法，增加随机的噪声的方式来改善这个问题）

欠采样：可能会存在信息减少的问题。因为只是利用了一部分数据，所以模型只是学习到了一部分模型。

有以下两种方法可以解决欠采样所带来的问题。

方法一：模型融合（bagging的思想）

思路：从丰富类样本中随机的选取（有放回的选取）和稀有类等量样本的数据。和稀有类样本组合成新的训练集。这样我们就产生了多个训练集，并且是互相独立的，然后训练得到多个分类器。

若是分类问题，就把多个分类器投票的结果（少数服从多数）作为分类结果。

若是回归问题，就将均值作为最后结果。

方法二：增量模型（boosting的思想）

思路：使用全部的样本作为训练集，得到分类器L1

从L1正确分类的样本中和错误分类的样本中各抽取50

从L1和L2分类结果中，选取结果不一致的样本作为训练集得到分类器L3.

最后投票L1,L2,L3结果得到最后的分类结果。