## [scikit-learn模块学习笔记(数据预处理模块preprocessing)](https://www.cnblogs.com/MrFiona/p/7900355.html)

本篇文章主要简单介绍sklearn中的数据预处理preprocessing模块，它可以对数据进行标准化。preprocessing 模块提供了数据预处理函数和预处理类，预处理类主要是为了方便添加到pipeline 过程中。

以下内容包含了一些个人观点和理解，如有疏漏或错误，欢迎补充和指出。

# **数据标准化**

数据标准化：当单个特征的样本取值相差甚大或明显不遵从高斯正态分布时，标准化表现的效果较差。实际操作中，经常忽略特征数据的分布形状，移除每个特征均值，划分离散特征的标准差，从而等级化，进而实现数据中心化。

公式为：(X-X\_mean)/X\_std 计算时对每个属性/每列分别进行.

将数据按其属性(按列进行)减去其均值，然后除以其方差。最后得到的结果是，对每个属性/每列来说所有数据都聚集在0附近，方差值为1。

标准化预处理函数：

* **preprocessing.scale**(X,axis=0, with\_mean=True, with\_std=True, copy=True)：  
  将数据转化为标准正态分布（均值为0，方差为1）
* **preprocessing.minmax\_scale**(X,feature\_range=(0, 1), axis=0, copy=True)：  
  将数据在缩放在固定区间，默认缩放到区间 [0, 1]
* **preprocessing.maxabs\_scale**(X,axis=0, copy=True)：  
  数据的缩放比例为绝对值最大值，并保留正负号，即在区间 [-1.0, 1.0] 内。唯一可用于稀疏数据 scipy.sparse的标准化
* **preprocessing.robust\_scale**(X,axis=0, with\_centering=True, with\_scaling=True,copy=True)：  
  通过 Interquartile Range (IQR) 标准化数据，即四分之一和四分之三分位点之间

它们对应的标准化预处理类：

* **classpreprocessing.StandardScaler**(copy=True, with\_mean=True,with\_std=True)：  
  标准正态分布化的类  
  属性：
  + **scale\_**：ndarray，缩放比例
  + **mean\_**：ndarray，均值
  + **var\_**：ndarray，方差
  + **n\_samples\_seen\_**：int，已处理的样本个数，调用partial\_fit()时会累加，调用fit()会重设
* **classpreprocessing.MinMaxScaler**(feature\_range=(0, 1),copy=True)：  
  将数据在缩放在固定区间的类，默认缩放到区间 [0, 1]，对于方差非常小的属性可以增强其稳定性，维持稀疏矩阵中为0的条目

属性：

* + **min\_**：ndarray，缩放后的最小值偏移量
  + **scale\_**：ndarray，缩放比例
  + **data\_min\_**：ndarray，数据最小值
  + **data\_max\_**：ndarray，数据最大值
  + **data\_range\_**：ndarray，数据最大最小范围的长度
* **classpreprocessing.MaxAbsScaler**(copy=True)：  
  数据的缩放比例为绝对值最大值，并保留正负号，即在区间 [-1.0, 1.0] 内。可以用于稀疏数据scipy.sparse  
  属性：
  + **scale\_**：ndarray，缩放比例
  + **max\_abs\_**：ndarray，绝对值最大值
  + **n\_samples\_seen\_**：int，已处理的样本个数
* **classpreprocessing.RobustScaler**(with\_centering=True,with\_scaling=True, copy=True)：  
  通过 Interquartile Range (IQR) 标准化数据，即四分之一和四分之三分位点之间  
  属性：
  + **center\_**：ndarray，中心点
  + **scale\_**：ndarray，缩放比例
* **classpreprocessing.KernelCenterer**：  
  生成 kernel 矩阵，用于将 svm kernel 的数据标准化（参考资料不全）

以上几个标准化类的方法：

* + **fit**(X[,y])：根据数据 X 的值，设置标准化缩放的比例
  + **transform**(X[,y, copy])：用之前设置的比例标准化 X
  + **fit\_transform**(X[, y])：根据 X设置标准化缩放比例并标准化
  + **partial\_fit**(X[,y])：累加性的计算缩放比例
  + **inverse\_transform**(X[,copy])：将标准化后的数据转换成原数据比例
  + **get\_params**([deep])：获取参数
  + **set\_params**(\*\*params)：设置参数

# **数据归一化**

* **preprocessing.normalize**(X,norm='l2', axis=1, copy=True)：  
  将数据归一化到区间 [0, 1]，norm 可取值 'l1'、'l2'、'max'。可用于稀疏数据 scipy.sparse
* **classpreprocessing.Normalizer**(norm='l2', copy=True)：  
  数据归一化的类。可用于稀疏数据 scipy.sparse  
  方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**get\_params**([deep])、**set\_params**(\*\*params)

# **数据二值化**

* **preprocessing.binarize**(X,threshold=0.0, copy=True)：  
  将数据转化为 0 和 1，其中小于等于 threshold 为 0，可用于稀疏数据 scipy.sparse
* **classpreprocessing.Binarizer**(threshold=0.0,copy=True)：  
  二值化处理的类，可用于稀疏数据 scipy.sparse  
  方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**get\_params**([deep])、**set\_params**(\*\*params)，其中fit 函数不会做任何操作

# **类别数据编码**

数据的某些特征是文本，特征是无序的，比如国籍，但数字是有序的，所以不能直接用数字编码

* **classpreprocessing.OneHotEncoder**(n\_values='auto',categorical\_features='all', dtype='float', sparse=True,handle\_unknown='error')：  
  将具有多个类别的特征转换为多维二元特征，所有二元特征互斥，当某个二元特征为 1 时，表示取某个类别  
  参数：
  + **n\_values**：处理的类别个数，可以为‘auto’，int或者 int数组
  + **categorical\_features**：被当作类别来处理的特征，可以为“all”或者下标数组指定或者mask数组指定

属性：

* + **active\_features\_**：ndarray，实际处理的类别数
  + **feature\_indices\_**：ndarray，第 i个原特征在转换后的特征中的下标在 feature\_indices\_[i] 和 feature\_indices\_[i+1]之间
  + **n\_values\_**：ndarray，每维的类别数

方法：**fit**(X[, y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**get\_params**([deep])、**set\_params**(\*\*params)

* **classpreprocessing.LabelBinarizer**(neg\_label=0, pos\_label=1,sparse\_output=False)：  
  和 OneHotEncoder 类似，将类别特征转换为多维二元特征，并将每个特征扩展成用一维表示  
  属性：
  + **classes**：ndarry，所有类别的值
  + **y\_type\_**：str
  + **multilabel\_**：bool
  + **sparse\_input\_**：bool
  + **indicator\_matrix\_**：str

方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**inverse\_transform**(y)、**get\_params**([deep])、**set\_params**(\*\*params)

* **preprocessing.label\_binarize**(y,classes, neg\_label=0, pos\_label=1, sparse\_output=False)：  
  LabelBinarizer 类对应的处理函数
* **classpreprocessing.LabelEncoder**：  
  将类别特征标记为 0 到 n\_classes - 1 的数  
  方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**inverse\_transform**(y)、**get\_params**([deep])、**set\_params**(\*\*params)

**classpreprocessing.MultiLabelBinarizer**(classes=None,sparse\_output=False)：  
和 LabelBinarizer 类似

feature\_extraction.DictVectorizer类  
patsy包

# **数据缺失**

* **classpreprocessing.Imputer**(missing\_values='NaN',strategy='mean', axis=0, verbose=0, copy=True)：  
  参数：
  + **missing\_values**：int 或者“NaN”，对np.nan的值用 "NaN"
  + **strategy**："mean"、"median"、"most\_frequent"

属性：

* + **statistics\_**：ndarray，当axis==0时，取每列填补时用的值

方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**get\_params**([deep])、**set\_params**(\*\*params)

# **生成多项式数据**

可以将数据多项式结合生成多维特征，比如 [a,b] 的二次多项式特征为 [1, a, b, a^2, ab, b^2]

* **classpreprocessing.PolynomialFeatures**(degree=2,interaction\_only=False, include\_bias=True)：  
  参数：
  + **degree**：int，多项式次数
  + **interaction\_only**：boolean，是否只产生交叉相乘的特征
  + **include\_bias**：boolean，是否包含偏移列，即全为1 的列

属性：

* + **powers\_**：ndarray，二维数组。powers\_[i,j] 表示第 i 维输出中包含的第 j 维输入的次数
  + **n\_input\_features\_**：int，输入维数
  + **n\_output\_features\_**：int，输出维数

方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**get\_params**([deep])、**set\_params**(\*\*params)

# **增加伪特征**

* **preprocessing.add\_dummy\_feature**(X,value=1.0)：  
  在 X 的第一列插入值为 value 的列

# **自定义数据转换**

可以使用自定义的 python函数来转换数据

* + **classpreprocessing.FunctionTransformer**(func=None,validate=True, accept\_sparse=False, pass\_y=False)：  
    方法：**fit**(X[,y])、**transform**(X[, y,copy])、**fit\_transform**(X[,y])、**get\_params**([deep])、**set\_params**(\*\*params)

分类: [python机器学习](https://www.cnblogs.com/MrFiona/category/1121337.html" \t "https://www.cnblogs.com/MrFiona/p/_blank)