[机器学习 数据预处理之独热编码（One-Hot Encoding）](http://www.renchengbin.com/post/1c4723_af96c8)

问题由来

在很多[**机器学习**](http://lib.csdn.net/base/2)任务中，特征并不总是连续值，而有可能是分类值。

例如，考虑一下的三个特征：

["male", "female"]

["from Europe", "from US", "from Asia"]

["uses Firefox", "uses Chrome", "uses Safari", "uses Internet Explorer"]

如果将上述特征用数字表示，效率会高很多。例如：

["male", "from US", "uses Internet Explorer"] 表示为[0, 1, 3]

["female", "from Asia", "uses Chrome"]表示为[1, 2, 1]

但是，即使转化为数字表示后，上述数据也不能直接用在我们的分类器中。因为，分类器往往默认数据数据是连续的，并且是有序的。但是，按照我们上述的表示，数字并不是有序的，而是随机分配的。

独热编码

为了解决上述问题，其中一种可能的解决方法是采用独热编码（One-Hot Encoding）。

独热编码即 One-Hot 编码，又称一位有效编码，其方法是使用N位状态寄存器来对N个状态进行编码，每个状态都由他独立的寄存器位，并且在任意时候，其中只有一位有效。

例如：

自然状态码为：000,001,010,011,100,101

独热编码为：000001,000010,000100,001000,010000,100000

可以这样理解，对于每一个特征，如果它有m个可能值，那么经过独热编码后，就变成了m个二元特征。并且，这些特征互斥，每次只有一个激活。因此，数据会变成稀疏的。

这样做的好处主要有：

1. 解决了分类器不好处理属性数据的问题
2. 在一定程度上也起到了扩充特征的作用

举例

我们基于[**Python**](http://lib.csdn.net/base/11)和Scikit-learn写一个简单的例子：

from sklearn import preprocessing

enc = preprocessing.OneHotEncoder()

enc.fit([[0, 0, 3], [1, 1, 0], [0, 2, 1], [1, 0, 2]])

enc.transform([[0, 1, 3]]).toarray()

输出结果：

array([[ 1.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  1.]])

处理离散型特征和连续型特征并存的情况，如何做归一化。  
参考博客进行了总结：  
https://www.quora.com/What-are-good-ways-to-handle-discrete-and-continuous-inputs-together  
总结如下：  
1、拿到获取的原始特征，必须对每一特征分别进行归一化，比如，特征A的取值范围是[-1000,1000]，特征B的取值范围是[-1,1].  
如果使用logistic回归，w1\*x1+w2\*x2，因为x1的取值太大了，所以x2基本起不了作用。  
所以，必须进行特征的归一化，每个特征都单独进行归一化。  
2、连续型特征归一化的常用方法：  
 **2.1：Rescale bounded continuous features**: All continuous input that are bounded, rescale them to [-1, 1] through x = (2x - max - min)/(max - min).线性放缩到[-1,1]  
  **2.2:Standardize all continuous features**: All continuous input should be standardized and by this I mean, for every continuous feature, compute its mean (u) and standard deviation (s) and do x = (x - u)/s.放缩到均值为0，方差为1  
**1、离散型特征的处理方法：**

**a) Binarize categorical/discrete features**: For all categorical features, represent them as multiple boolean features. For example, instead of having one feature called marriage\_status, have 3 boolean features - married\_status\_single, married\_status\_married, married\_status\_divorced and appropriately set these features to 1 or -1. As you can see, for every categorical feature, you are adding k binary feature where k is the number of values that the categorical feature takes.对于离散的特征基本就是按照one-hot编码，该离散特征有多少取值，就用多少维来表示该特征。

为什么使用one-hot编码来处理离散型特征，这是有理由的，不是随便拍脑袋想出来的！！！具体原因，分下面几点来阐述：   
**1、Why do we binarize categorical features?**  
We binarize the categorical input so that they can be thought of as a vector from the Euclidean space (we call this as embedding the vector in the Euclidean space).使用one-hot编码，将离散特征的取值扩展到了欧式空间，离散特征的某个取值就对应欧式空间的某个点。  
   
**2、Why do we embed the feature vectors in the Euclidean space?**  
Because many algorithms for classification/regression/clustering etc. requires computing distances between features or similarities between features. And many definitions of distances and similarities are defined over features in Euclidean space. So, we would like our features to lie in the Euclidean space as well.将离散特征通过one-hot编码映射到欧式空间，是因为，在回归，分类，聚类等机器学习算法中，特征之间距离的计算或相似度的计算是非常重要的，而我们常用的距离或相似度的计算都是在欧式空间的相似度计算，计算余弦相似性，基于的就是欧式空间。  
  
  
**3、Why does embedding the feature vector in Euclidean space require us to binarize categorical features?**  
Let us take an example of a dataset with just one feature (say job\_type as per your example) and let us say it takes three values 1,2,3.  
Now, let us take three feature vectors x\_1 = (1), x\_2 = (2), x\_3 = (3). What is the euclidean distance between x\_1 and x\_2, x\_2 and x\_3 & x\_1 and x\_3? d(x\_1, x\_2) = 1, d(x\_2, x\_3) = 1, d(x\_1, x\_3) = 2. This shows that distance between job type 1 and job type 2 is smaller than job type 1 and job type 3. Does this make sense? Can we even rationally define a proper distance between different job types? In many cases of categorical features, we can properly define distance between different values that the categorical feature takes. In such cases, isn't it fair to assume that all categorical features are equally far away from each other?  
Now, let us see what happens when we binary the same feature vectors. Then, x\_1 = (1, 0, 0), x\_2 = (0, 1, 0), x\_3 = (0, 0, 1). Now, what are the distances between them? They are sqrt(2). So, essentially, when we binarize the input, we implicitly state that all values of the categorical features are equally away from each other.  
将离散型特征使用one-hot编码，确实会让特征之间的距离计算更加合理。比如，有一个离散型特征，代表工作类型，该离散型特征，共有三个取值，不使用one-hot编码，其表示分别是x\_1 = (1), x\_2 = (2), x\_3 = (3)。两个工作之间的距离是，(x\_1, x\_2) = 1, d(x\_2, x\_3) = 1, d(x\_1, x\_3) = 2。那么x\_1和x\_3工作之间就越不相似吗？显然这样的表示，计算出来的特征的距离是不合理。那如果使用one-hot编码，则得到x\_1 = (1, 0, 0), x\_2 = (0, 1, 0), x\_3 = (0, 0, 1)，那么两个工作之间的距离就都是sqrt(2).即每两个工作之间的距离是一样的，显得更合理。  
**4、About the original question?**  
Note that our reason for why binarize the categorical features is independent of the number of the values the categorical features take, so yes, even if the categorical feature takes 1000 values, we still would prefer to do binarization.  
对离散型特征进行one-hot编码是为了让距离的计算显得更加合理。  
**5、Are there cases when we can avoid doing binarization?**  
Yes. As we figured out earlier, the reason we binarize is because we want some meaningful distance relationship between the different values. As long as there is some meaningful distance relationship, we can avoid binarizing the categorical feature. For example, if you are building a classifier to classify a webpage as important entity page (a page important to a particular entity) or not and let us say that you have the rank of the webpage in the search result for that entity as a feature, then 1] note that the rank feature is categorical, 2] rank 1 and rank 2 are clearly closer to each other than rank 1 and rank 3, so the rank feature defines a meaningful distance relationship and so, in this case, we don't have to binarize the categorical rank feature.  
  
More generally, if you can cluster the categorical values into disjoint subsets such that the subsets have meaningful distance relationship amongst them, then you don't have binarize fully, instead you can split them only over these clusters. For example, if there is a categorical feature with 1000 values, but you can split these 1000 values into 2 groups of 400 and 600 (say) and within each group, the values have meaningful distance relationship, then instead of fully binarizing, you can just add 2 features, one for each cluster and that should be fine.  
将离散型特征进行one-hot编码的作用，是为了让距离计算更合理，但如果特征是离散的，并且不用one-hot编码就可以很合理的计算出距离，那么就没必要进行one-hot编码，比如，该离散特征共有1000个取值，我们分成两组，分别是400和600,两个小组之间的距离有合适的定义，组内的距离也有合适的定义，那就没必要用one-hot 编码  
   
离散特征进行one-hot编码后，编码后的特征，其实每一维度的特征都可以看做是连续的特征。就可以跟对连续型特征的归一化方法一样，对每一维特征进行归一化。比如归一化到[-1,1]或归一化到均值为0,方差为1  
   
**有些情况不需要进行特征的归一化：**  
     It depends on your ML algorithms, some methods requires almost no efforts to normalize features or handle both continuous and discrete features, like tree based methods: c4.5, Cart, random Forrest, bagging or boosting. But most of parametric models (generalized linear models, neural network, SVM,etc) or methods using distance metrics (KNN, kernels, etc) will require careful work to achieve good results. Standard approaches including binary all features, 0 mean unit variance all continuous features, etc。  
      **基于树的方法是不需要进行特征的归一化，例如随机森林，bagging 和 boosting等。基于参数的模型或基于距离的模型，都是要进行特征的归一化。**

**one-hot编码为什么可以解决类别型数据的离散值问题   
首先，one-hot编码是N位状态寄存器为N个状态进行编码的方式   
eg：高、中、低不可分，→ 用0 0 0 三位编码之后变得可分了，并且成为互相独立的事件   
→ 类似 SVM中，原本线性不可分的特征，经过project之后到高维之后变得可分了   
GBDT处理高维稀疏矩阵的时候效果并不好，即使是低维的稀疏矩阵也未必比SVM好   
Tree Model不太需要one-hot编码：   
对于决策树来说，one-hot的本质是增加树的深度   
tree-model是在动态的过程中生成类似 One-Hot + Feature Crossing 的机制   
1. 一个特征或者多个特征最终转换成一个叶子节点作为编码 ，one-hot可以理解成三个独立事件   
2. 决策树是没有特征大小的概念的，只有特征处于他分布的哪一部分的概念   
one-hot可以解决线性可分问题 但是比不上label econding   
one-hot降维后的缺点：   
降维前可以交叉的降维后可能变得不能交叉   
树模型的训练过程：   
从根节点到叶子节点整条路中有多少个节点相当于交叉了多少次，所以树的模型是自行交叉   
eg：是否是长的 { 否（是→ 柚子，否 → 苹果） ，是 → 香蕉 } 园 cross 黄 → 形状 （圆，长） 颜色 （黄，红） one-hot度为4的样本   
使用树模型的叶子节点作为特征集交叉结果可以减少不必要的特征交叉的操作 或者减少维度和degree候选集   
eg 2 degree → 8的特征向量 树 → 3个叶子节点   
树模型：Ont-Hot + 高degree笛卡尔积 + lasso 要消耗更少的计算量和计算资源   
这就是为什么树模型之后可以stack线性模型   
n\*m的输入样本 → 决策树训练之后可以知道在哪一个叶子节点上 → 输出叶子节点的index → 变成一个n\*1的矩阵 → one-hot编码 → 可以得到一个n\*o的矩阵（o是叶子节点的个数） → 训练一个线性模型   
典型的使用： GBDT +　ＲＦ   
优点 ： 节省做特征交叉的时间和空间   
如果只使用one-hot训练模型，特征之间是独立的   
对于现有模型的理解：（G（l（张量）））：   
其中：l（·）为节点的模型   
G（·）为节点的拓扑方式   
神经网络：l（·）取逻辑回归模型   
G（·）取全连接的方式   
决策树： l（·）取LR   
G（·）取树形链接方式   
创新点： l（·）取 NB，SVM 单层NN ，等   
G（·）取怎样的信息传递方式**