# 数据预处理与交叉熵在机器学习中的应用

我们学习《深度学习》课程前，需要先了解一些应用数学和机器学习基础知识。在经过小组成员的学习讨论和分享交流后，我们针对其中数据预处理和交叉熵在机器学习中的应用作了简单总结，形成这份小组报告。

## 一. 数据预处理

在实际工程中，我们拿到的数据可能包含了大量的缺失值和冗余值，也可能因为人工录入错误导致有异常点存在，非常不利于算法模型的训练。数据预处理就是对各种脏数据进行对应方式的处理，得到标准的、干净的、规范的数据，提供给机器学习使用。数据预处理的主要步骤分为：数据清理、数据集成、数据规约和数据变换，下面将从这四个方面详细的介绍具体的方法。

### 1.1 数据清理

数据清理的主要思想是通过填补缺失值、光滑噪声数据，平滑或删除离群点，并解决数据的不一致性来“清理”数据。

#### 1.1.1 缺失值处理

由于现实世界中，获取信息和数据的过程中，会存在各类的原因导致数据丢失和空缺。针对这些缺失值的处理方法，主要是基于变量的分布特性和变量的重要性（信息量和预测能力）采用不同的方法。主要分为以下几种：

- **删除变量**：若变量的缺失率较高（例如大于80%）且重要性较低，可以直接将变量删除。
- **统计量填充**：若缺失率较低（例如小于95%）且重要性较低，则根据数据分布的情况进行填充。
  - 对于数据符合均匀分布，可采用均值（期望）进行填补。
  - 对于数据存在倾斜分布的情况，课采用中位数进行填补。
- **哑变量填充**：若变量是离散型，且不同值较少，可转换成哑变量。例如性别SEX变量，一般来说只有[“男”,“女”]两个不同的值，此时可以将缺失值记为“未知（NA）”，于是该列将转换成 *IS_SEX_MALE*, *IS_SEX_FEMALE*, *IS_SEX_NA*。

> 从经验来看，先用`pandas.isnull.sum()`检测出变量的缺失比例，再考虑删除或者填充。若需要填充的变量是连续型，一般采用均值法和随机差值进行填充；若变量是离散型，通常采用中位数或哑变量进行填充。

#### 1.1.2 离群点处理

离群点是数据分布的常态，处于特定分布区域或范围之外的数据通常被定义为离群点。主要有以下检测离群点的方法：

- **简单统计分析**：根据箱线图、各分位点判断是否存在异常，例如pandas的describe函数可以快速发现异常值。
- $3\sigma$**原则**：若数据存在正态分布，偏离均值的$3\sigma$之外. 通常定义$P(|x-\mu|>3\sigma)\leq 0.003$范围内的点为离群点。

- **基于绝对离差中位数（MAD）**：这是一种稳健对抗离群数据的距离值方法，采用计算各观测值与平均值的距离总和的方法。放大了离群值的影响。

具体的处理手段：

- 根据异常点的数量和影响，考虑是否将该条记录删除，信息损失多
- 对数据做了log-scale 对数变换后消除离群点
- 平均值或中位数替代异常点，简单高效，信息的损失较少

#### 1.1.3 噪声处理

噪声是变量的随机误差和方差，是观测值和真实值之间的误差。通常的处理办法是对数据进行分箱操作：等频或等宽分箱，然后用每个箱的平均数，中位数或者边界值（不同数据分布，处理方法不同）代替箱中所有的数，起到平滑数据的作用。

## 1.2 数据集成

数据集的准备过程中有时涉及数据集成。数据集成是指将多个数据源中的数据结合、集成到一个单一的数据集中。这些源可能包括多个数据库、数据方或一般文件。

- **实体识别问题**。由于现实生活中语义的多样性以及数据命名的不规范，可能导致：
  - 两个数据源中都有一个字段名字叫“Payment”，但其实一个数据源中记录的是税前的薪水，另一个数据源中是税后的薪水。
  - 两个数据源都有字段记录税前的薪水，但是一个数据源中字段名称为“Payment”，另一个数据源中字段名称为“Salary”。
- **冗余问题**。字段的冗余一般源自于字段之间存在强相关性或者几个字段间可以相互推导得到。通过检测字段的相关性，可以侦察到数据冗余。具体来说，方法有如下几个：
  - 数值型变量可计算相关系数，协方差矩阵。
  - 分类型变量可使用卡方检验。
- **数据冲突问题**。不同数据源，在统一合并时取值记录不一样。对待这种问题，就需要对实际的业务知识有一定的理解。同时，对数据进行调研，尽量明确造成冲突的原因。这样才能考虑冲突数据是否都要保留、是否要进行取舍，以及如何进行取舍等问题。

## 1.3 数据规约

用于机器学习的原始数据集可能包含各种各样的属性，其中大部分属性与学习任务不相关，是冗余的。维度归约通过删除不相关的属性，来减少数据维度，并保证信息的损失最小。一般有如下策略：

- **逐步向前选择**：每次把一个最好的维度加入到集合。
- **逐步向后删除**：每次把一个最差的维度从数据集删除。

具体的实现方法有：

- **主成分分析（PCA）**：PCA通过空间映射的方式，将当前维度映射到更低的维度，使得每个变量在新空间的方差最大。PCA 本质上是将方差最大的方向作为主要特征，并且在各个正交方向上将数据“离相关”，也就是让它们在不同正交方向上没有相关性。因此，PCA 也存在一些限制，例如它可以很好的解除线性相关，但是对于高阶相关性就没有办法了，对于存在高阶相关性的数据，可以考虑Kernel PCA，通过 Kernel 函数将非线性相关转为线性相关。另外，PCA 假设数据各主特征是分布在正交方向上，如果在非正交方向上存在几个方差较大的方向，PCA 的效果就大打折扣了。下面是基于numpy库的PCA算法主要步骤：

```
import numpy as np
import pandas as pd


def pca(dataSet, topNfeat=999999):

    # 1.对所有样本进行中心化（所有样本属性减去属性的平均值）
    meanVals = np.mean(dataSet, axis=0)
    meanRemoved = dataSet - meanVals

    # 2.计算样本的协方差矩阵 XXT
    covSet = np.cov(meanRemoved, rowvar=0)

    # 3.对协方差矩阵做特征值分解，求得其特征值和特征向量，并将特征值从大到小排序，筛选出前topNfeat个
    eigVals, eigVects = np.linalg.eig(np.mat(covmat))
    eigValInd = np.argsort(eigVals)
    eigValInd = eigValInd[:-(topNfeat+1):-1]     # 取前topNfeat大的特征值的索引
    redEigVects = eigVects[:, eigValInd]         # 取前topNfeat大的特征值所对应的特征向量

    # 4.将数据转换到新的低维空间中
    resultSet = meanRemoved * redEigVects      # 降维之后的数据
    compareSet = (lowDdataSet * redEigVects.T) + meanVals # 重构数据，可在原数据维度下进行对比查看
    return np.array(resultSet), np.array(compareSet)
```

- **奇异值分解（SVD）**：SVD的降维可解释性较低，且计算量比PCA大，一般用在稀疏矩阵上降维，例如图片压缩，推荐系统。
- **聚类**：将某一类具有相似性的特征聚到单个变量，从而大大降低维度。
- **线性组合**：将多个变量做线性回归，根据每个变量的表决系数，赋予变量权重，可将该类变量根据权重组合成一个变量。

## 1.4 数据变换

数据变换包括对数据进行规范化，离散化，稀疏化处理，达到适用于挖掘的目的。

### 1.4.1 规范化处理

数据中不同特征的量纲可能不一致，数值间的差别可能很大，不进行处理可能会影响到机器学习的结果。因此，需要对数据按照一定比例进行缩放，使之落在一个特定的区域，便于进行综合分析。

- **max-min规范化**：将数据映射到[0,1]区间。

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- **Z-Score标准化**：处理后的数据均值为0，方差为1

$$x_{new} = \frac{x - \bar{x}}{\sigma}$$

- **Log变换**：在时间序列数据中，对于数据量级相差较大的变量，通常做Log函数的变换

$$x_{new} = log\, x$$

#### 1.4.2离散化处理

数据离散化是指将连续的数据进行分段，使其变为一段段离散化的区间。分段的原则有基于等距离、等频率或优化的方法。

- **等频法**：使得每个箱中的样本数量相等，例如总样本n=100，分成k=5个箱，则分箱原则是保证落入每个箱的样本量=20。
- **等宽法**：使得属性的箱宽度相等，例如年龄变量（0-100之间），可分成 [0,20]，[20,40]，[40,60]，[60,80]，[80,100]五个等宽的箱。
- **聚类法**：根据聚类出来的簇，每个簇中的数据为一个箱，簇的数量模型给定。

#### 1.4.3 稀疏化处理

针对离散型的标称变量，无法进行有序的LabelEncoder时，通常考虑将变量做$[0, 1]$哑变量的稀疏化处理，例如动物类型变量中含有猫，狗，猪，羊四个不同值，将该变量转换成is*猪，is*猫，is*狗，is*羊四个哑变量。若是变量的不同值较多，则根据频数，将出现次数较少的值统一归为一类'other'。稀疏化处理既有利于模型快速收敛，又能提升模型的抗噪能力。

## 1.5 小结

以上介绍了数据预处理中会用到的大部分方法和技术。数据的质量和包含的有用信息量是决定一个机器学习算法能够学多好的关键因素。因此，我们在训练模型前评估和预处理数据显得至关重要。不过在实践中进行数据预处理时，仍然有非常多需要考虑的细节。准备各种原始数据集亲自动手试一试是最有效的学习方式。

# 二. 交叉熵和信息论

交叉熵（cross entropy）是深度学习中常用的一个概念，一般用来求目标与预测值之间的差距。在学习交叉熵之前，我们先引入几个信息论的概念，然后由熵(Entropy) -> KL散度(Kullback-Leibler Divergence) -> 交叉熵这个顺序来逐步理解交叉熵的意义。

## 2.1 什么是熵(Entropy)?

通用的说，熵(Entropy)被用于描述一个系统中的不确定性(the uncertainty of a system)。在不同领域熵有不同的解释，比如热力学的定义和信息论也不大相同。放在信息论的语境里面来说，就是一个事件所包含的信息量，也叫**香农熵(Shannon Entropy)**。我们常常听到“这句话信息量好大”，比如“五一同济大学不放假，学校要给20级非全研究生发博士毕业证”。这句话为什么信息量大？因为它的内容出乎意料，违反常理。信息论的基本想法时一个不太可能发生的事件居然发生了，要比一个非常可能的事件发生，能提供更多的信息。

我们按照这种基本想法来量化信息，由此引出：

- 越不可能发生的事件信息量越大，比如“中国队拿到了世界杯冠军”这句话信息量就很大。
- 确定事件的信息量就很低，比如“巴西队拿到了世界杯冠军”，信息量就很低。
- 独立事件的信息量可叠加。比如“投掷的硬币两次正面朝上”的信息量就应该是“投掷一次硬币正面朝上“的信息量的两倍。

因此我们可以定义一个事件$\mathrm{x} = x$的**自信息(Self-Information)**为

$$I(x) = -logP(x) \tag{2.1.1}$$

> 自信息的单位为**比特**或者**奈特**。1**奈特**是以自然对数`e`为底得到的自信息。1**比特**是以`2`为底得到的自信息

对于一个一定会发生的事件$x$，其发生概率为1，$I(x) = -log(1) \times 1 = -0 \times 1 = 0$，自信息为0。

自信息只描述单个事件的信息量。对于某个事件，有$n$种可能性，每一种可能性都有一个概率$P(x_i)$。我们用**熵(Entropy)**来表示所有信息量的期望，即：

$$H(x) = -\sum_{i}^{n} P(x_i)logP(x_i) \tag{2.1.2}$$

熵可以用来量化对某个事件概率分布的不确定性总量。接近确定性的分布（几乎会发生或者几乎不会发生）具有较低的熵；接近均匀分布的概率分布具有较高的熵。例如，对于下面两个事件，分别计算它们的熵：

| 事件分布$X$ | $x1$ | $x2$ | $H(x)$ |
| --- | --- | --- | --- |
| 投掷一枚硬币哪一面朝上 | $P(x=_{正面})=0.5$ | $P(x=_{反面})=0.5$ | $-(0.5\times log(0.5)+0.5\times log(0.5))=0.3466+0.3466=0.6932$ |
| 投掷一枚硬币会不会碎 | $P(x=_{碎了})=0.1$ | $P(x=_{没碎})=0.9$ | $-(0.1\times log(0.1)+0.9\times log(0.9))=0.2303+0.0948=0.3251$ |

## 2.2 KL散度 —— 衡量两个事件/分布之间的不同

我们上面说的是对于一个随机变量$x$的事件的概率A的概率分布$P(x)$的熵，如果我们有另一个独立的随机变量$x$相关的事件B，其概率分布为$Q(x)$，该怎么量化计算它们之间的区别?

这个时候我们就需要引入**KL散度（Kullback-Leibler Divergence）**，也叫**相对熵**，用于计算两个分布之间的不同。

KL散度的数学定义：

- 对于**离散事件**我们可以定义事件A和B的差别为：

$$D_{KL}(P||Q)=\sum_{i=1}^{n}p(x_i)log\left(\frac{p(x_i)}{q(x_i)}\right) \tag{2.2.1}$$

- 对于**连续事件**，那么我们只是把求和改为求积分而已(2.3.2)。

$$D_{KL}(P||Q)=\int p(x)log\left(\frac{p(x)}{q(x)}\right) \tag{2.2.2}$$

> $D_{KL}$的值越小，表示$Q(x)$分布和$P(x)$分布越接近。如果两个事件分布完全相同，那么$D_{KL}$散度等于0

在机器学习中，$P(x)$往往用来表示样本的真实分布，$Q(x)$用来表示模型所预测的分布。如果用$P(x)$来描述样本，那么就非常完美。然而我们只能通过机器学习得到$Q(x)$来近似描述样本。$D_{KL}$散度刚好可以描述$Q(x)$和$P(x)$的近似程度。

因为KL散度时非负的，并且衡量的是两个分布之间的差异，有时被当作是分布之间的某种距离。距离是具有对称的，A到B的距离等于B到A的距离。然而**KL散度不是真的距离，因为它不是对称的**：从上面的公式可以看出，$D_{KL}(P||Q)\neq D_{KL}(Q||P)$。

## 2.3 从KL散度到交叉熵

对式2.2.1变形可以得到：

$$
\begin{aligned}
D_{KL}(P||Q) &= \sum_{i=1}^{n} p(x_i)log(p(x_i)) - \sum_{i=1}^{n} p(x_i)log(q(x_i)) \\
&= H(p(x)) + [-\sum_{i=1}^{n} p(x_i)log(q(x_i))]
\end{aligned} \tag{2.3.1}
$$

等式的前一部分恰巧就是P的熵，等式的后一部分，就是**交叉熵（Cross Entropy）**：

$$
H(P,Q) = -\sum_{i=1}^{n} p(x_i)log(q(x_i)) \tag{2.3.2}
$$

**此处最重要的观察是，对于确定的事件P，它的熵$H(P)$是一个常量。那么，最小化KL散度$D_{KL}(P||Q)$等价于最小化交叉熵$H(P,Q)$。**在机器学习中，我们需要评估**真实标签（label）**和**预测值（predicts）**之间的差距。在优化过程中，最小化KL散度和最小化交叉熵是等价的，所以一般在机器学习中直接用用更简单的交叉熵做**损失函数（loss）**评估模型。

## 2.4 交叉熵在分类中的应用

假设有下面2个神经网络模型

模型a

| 预测值 | 标签（one-hot） | 正确性 |
| --- | --- | --- |
| [0.3 0.3 0.4] | [0 0 1] (山鸢尾) | yes |
| [0.3 0.4 0.3] | [0 1 0] (变色鸢尾) | yes |
| [0.1 0.2 0.7] | [1 0 0] (维吉尼亚鸢尾) | no |

模型b

| 预测值 | 标签（one-hot） | 正确性 |
| --- | --- | --- |
| [0.1 0.2 0.7] | [0 0 1] (山鸢尾) | yes |
| [0.1 0.7 0.2] | [0 1 0] (变色鸢尾) | yes |
| [0.3 0.4 0.3] | [1 0 0] (维吉尼亚鸢尾) | no |

两个模型的正确率都是0.67。通过计算平均交叉熵，有

$$
\begin{aligned}
H(Q_a) &= -(ln(0.4) + ln(0.4) + ln(0.1))/3 = 1.38 \\
H(Q_b) &= -(ln(0.7) + ln(0.7) + ln(0.3))/3 = 0.64
\end{aligned}
$$

可以看出，模型b要比模型a好一些。

## 2.5 小结

对于上述概念，可以通俗但不严谨的表述为：

- 自信息：可以表示一个某个事件A发生所携带的信息量。
- 熵：可以表示一个事件A的不确定性。
- KL散度：可以用来表示从事件A的角度来看，事件B有多大不同。
- 交叉熵：可以用来表示从事件A的角度来看，如何描述事件B。

KL散度可以被用于计算**损失函数（loss）**，而在特定情况下最小化KL散度等价于最小化交叉熵。而交叉熵的运算更简单，所以用交叉熵来当做损失函数。

# 三. 报告总结

在整理这份小作业的过程中，我们对《深度学习》有了进一步了解，同时熟悉了组内同学，为后续的学习和研究打下了良好的基础。在此感谢老师悉心的指引，感谢同学们无私的分享。