机器学习过程

1、抽象成数学问题

明确问题是进行机器学习的第一步。机器学习的训练过程通常都是一件非常耗时的事情，胡乱尝试时间成本是非常高的。这里的抽象成数学问题，指的是我们明确我们可以获得什么样的数据，目标是一个分类、回归或者是聚类的问题，如果都不是的话，划归为其中的某类问题。

2、数据的收集

我们可以按以下方法收集数据：

开放数据集

一般针对行业的数据库，比如我们国家的统计局开放的关于经济和社会发展情况的资料性年刊。系统收录了全国和各省、自治区、直辖市上年经济、社会各方面的统计数据，以及多个重要历史年份和近年全国主要统计数据。

各种比赛的数据集，比如kaggle，天池，百度在比赛中，都会开放一些数据集。

高校、研究机构开放的数据集，例如清华开放的关于对话机器人的CrossWOZ数据集。

爬虫抓取

爬虫工具是可视化、易操作的工具。爬虫代码，可以自己编写，抓取特定网站或App。

传感器

比如无人驾驶的传感器数据，新零售的传感器数据，采集的是物理信息（图像、视频、或者某个物体的速度、热度、压强等）。

日志采集

用于统计用户的操作。可以在前端进行埋点，或在后端进行脚本收集、统计，来分析网站的访问情况，使用瓶颈等。

3、数据预处理

业界有一句非常著名的话：“数据的质量决定了机器学习的上界，而模型和算法只是逼近这个上界。”由此可见，高质量的数据对于整个机器学习项目至关重要。

探索性数据分析

得到数据集后应首先进行探索性数据分析，了解数据的分布情况（以下代码中的train为用pandas读取后的数据）。

```
train.head(5) #显示前5行数据
train.tail(5) #显示后5行
train.columns #查看列名
train.info() #查看各字段的信息
train.shape #查看数据集行列分布，几行几列
train.describe() #查看数据的大体情况
```

对于分类问题，数据偏斜不能过于严重，不同类别的数据数量不要有很大的差距。 而且还要对数据的量级有一个评估，多少个样本，多少个特征，可以估算出其对内存的消耗程度，判断训练过程中内存是否能够放得下。如果放不下就得考虑改进算法或者使用一些降维的技巧。如果数据量实在太大，那就要考虑分布式训练了。

通过各种方法得到的数据或多或少都会存在数据缺失、分布不均衡、存在异常数据、混有无关紧要的数据等诸多数据不规范的问题。这就需要我们对收集到的数据进行进一步的处理，包括处理缺失值、异常值检验、数据规范化、数据的转换等，这样的步骤叫做“数据预处理”。

缺失值处理

处理缺失值常用的方法主要有三大类：删除元组、数据填充、不处理。

删除元组：也就是将存在遗漏信息属性值的对象（元组，记录）删除，从而得到一个完备的信息表。它的优点是简单易行，在对象有多个属性缺失值、被删除的含缺失值的对象与初始数据集的数据量相比非常小的情况下非常有效。它的不足是当缺失数据所占比例较大，特别当遗漏数据非随机分布时，这种方法可能导致数据发生偏离，从而引出错误的结论。

数据填充：用一定的值去填充空值，常用的方法，人工填充、特殊值填充、均值、众数、中位数填充、就近补齐、预测填充等.

不处理：不处理缺失值，直接将包含空值的数据上放入模型。

异常值检验

常用的是箱形图检验、均方差检验、DBScan 聚类、孤立森林等。

数据规范化

数据规范化一般指0-1规范化，Z-Score规范化。

0-1规范化(MinMax normalization)：将原始数据缩放到[0,1]区间内，用公式表示为：

$$x_i' = \frac{x_i - min(x)}{max(x) - min(x)}$$

它的缺点是当有新数据加入时，可能会导致最大值最小值发生变化，需要重新计算。

Z-Score(Standardzation)：将原始数据转换为标准正态分布，即原始数据离均值有几个标准差，用公式表示为：

$$x_i' = \frac{x_i - \mu}{\sigma}$$

它的缺点是对原始数据的分布有要求，要求原始数据数据分布为正态分布计算；在真实世界中，总体的均值和标准差很难得到，只能用样本的均值和标准差求得。在Python的sklearn包中提供了StandardScaler 和MinMaxScaler ，代码示例如下：

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler, StandardScaler

path = '/data/course_data/qincc/dating.txt'
data = pd.read_csv(path)
print(data)

# 归一化
transfer = MinMaxScaler(feature_range=(3,5))
# 进行转换，调用fit_transform
ret_data = transfer.fit_transform(data[["milage", "Liters",
"Consumtime"]])
print("归一化之后的数据为:\n",ret_data)

# 标准化
transfer = StandardScaler()
# 进行转换，调用fit_transform
ret_data = transfer.fit_transform(data[["milage", "Liters",
"Consumtime"]])
print("标准化之后的数据为:\n",ret_data)
print("每一列的方差为:\n", transfer.var_)
```

```
print("每一列的平均值为:\n", transfer.mean_)
```

数据的转换

数据转换主要指将非数值变量转换为可以放入模型的数值变量，常用的方法有，one-hot编码，label_encoder编码。在sklearn包中提供了OneHotEncoder和LabelEncoder函数。

4、特征工程

特征工程主要有特征衍生和特征选择。特征衍生是指利用现有的特征进行某种组合生成新的特征，主要是从业务数据和纯技术生产特征方面着手。

业务衍生是从特征的业务意义出发，生成具有不同层面业务含义的新特征。方法：逻辑关联、增量、频率分析、相对强度水平等。这一衍生手段，主要是原始数据方面利用业务思维和统计方法进行衍生，提取一些次级特征。比如逻辑关联主要是从业务逻辑思路里提取新的可用特征；从变化幅度大的特征提取出其增量特征；从覆盖面较大且类别较多的分类特征取出其频率特征；从特征值分布有差异的数值型特征中提取其相对整体平均值的强度水平或相对某一群体平均值的强度水平等等。

技术衍生是在特征计算层次上对特征进行大范围加工衍生，比如相关特征的加减乘除方、二值化、离散化、交叉组合、多项式融合、算法衍生等。特征选择是指衍生出的所有特征都不一定都对模型有效，所以对特征进行有效的选择。筛选出显著特征、摒弃非显著特征，需要机器学习工程师反复理解业务。这对很多结果有决定性的影响。特征选择好了，非常简单的算法也能得出良好、稳定的结果。这需要运用特征有效性分析的相关技术，如相关系数、卡方检验、平均互信息、条件熵、后验概率、逻辑回归权重等方法。

5、数据集分割

一般需要将样本分成独立的三部分：训练集(train set)，验证集(validation set)和测试集(test set)。其中训练集用来估计模型，验证集用来调整模型参数从而得到最优模型，而测试集则检验最优的模型的性能如何。一个典型的划分是训练集占总样本的50%，而其它各占25%，三部分都是从样本中随机抽取，并且不能有交集。

样本少的时候，这种划分就不合适了。常用的是留少部分做测试集。然后对其余N个样本采用K折交叉验证法。就是将样本打乱，然后均匀分成K份，轮流选择其中K-1份训练，剩余的一份做验证，计算预测误差平方和，最后把K次的预测误差平方和再做平均作为选择最优模型结构的依据。

那么，如何选择训练集、验证集、测试集的划分比例呢？在传统的机器学习中，这三者一般的比例为training/validation/test = 50/25/25，但是有些时候如果模型不需要很多调整只要拟合就可时，或者training本身就是training+validation (比如cross validation)时，也可以training/test =7/3。但是在深度学习中，由于数据量本身很大，而且训练神经网络需要的数据很多，可以把更多的数据分给training，而相应减少validation和test。

6、模型的选择与训练

当我们处理好数据之后，就可以选择合适的机器学习模型进行数据的训练了。可供选择的机器学习模型有很多，每个模型都有自己的适用场景，那么如何选择合适的模型呢?

首先我们要对处理好的数据进行分析，判断训练数据有没有类标，若是有类标则应该考虑监督学习的模型，否则可以划分为非监督学习问题。其次分析问题的类型是属于分类问题还是回归问题，当我们确定好问题的类型之后再去选择具体的模型。在模型的实际选择时，通常会考虑尝试不同的模型对数据进行训练，然后比较输出的结果，选择最佳的那个。再次，我们还会考虑到数据集的大小。若是数据集样本较少，训练的时间较短，通常考虑朴素贝叶斯等一些轻量级的算法，否则的话就要考虑SVM等一些重量级算法。最后，调优问题，可以采用交差验证，观察损失曲线，测试结果曲线等分析原因，调节参数：优化器、学习率、batchsize等。此外还可以尝试多模型融合，来提高效果。

7、模型的评价

在模型评价阶段，我们可以根据分类、回归、排序等不同问题关心的问题选择不同的评价指标，多与模型选择时的损失不同。

用途	指标	介绍
分类模型	准确率	Accuracy=N（correct）/N(total) 准确率评价没有对不同类别进行区分。 此外，数据分布不平衡，即有的类别下样本过多，有的类别下样本少，两个类个数相差较大，这样样本占大部分的类别主导了准确率的计算。
	平均准确率	举例，类别 0 的准确率为 80%，类别 1 下的准确率为 97.5%，那么平均准确率为(80%+97.5%)/2=88.75%
	对数损失函数（log-loss）	
	精确率-召回率（Precision-Recall）	精确率(Precision)：分类正确的正样本个数占分类器所有预测为正样本的个数的比例； 召回率(Recall)：分类正确的正样本个数占全部实际正样本个数的比例。
	AUC（ROC 曲线下面积）	ROC 曲线的 x 轴便是 FPR，y 轴便是 TPR。
	混淆矩阵（Confusion Matrix）	由混淆矩阵可以计算以下评价指标: 1.准确率（Accuracy）：分类正确的样本数占所有样本数的比例 2. 平均准确率(Average per-class accuracy)：每个类别下的准确率的算术平均 3.精确率(Precision)：分类正确的正样本个数占分类器所有预测为正样本的个数的比例 4.召回率(Recall)：分类正确的正样本个数占全部实际正样本个数的比例 5.F1-Score：精确率与召回率的**调和平均值**，它的值更接近于 Precision 与 Recall 中较小的值 6.ROC 曲线：ROC 曲线的 x 轴便是 FPR，y 轴便是 TPR。
回归模型	平方根误差（RMSE）	RMSE 虽然广为使用，但是其存在一些缺点，因为它是使用平均误差，而平均值对异常点（outliers）较敏感，如果回归器对某个点的回归值很不理性，那么它的误差则较大，从而会对 RMSE 的值有较大影响，即平均值是**非鲁棒**的。
	MAPE	为了改进 RMSE 的缺点，提高评价指标的鲁棒性，使用**误差的分位**数来代替，如中位数来代替平均数。 假设 100 个数，最大的数再怎么改变，中位数也不会变，因此其对异常点具有鲁棒性。在现实数据中，往往会存在异常点，并且模型可能对异常点拟合得并不好，因此提高评价指标的鲁棒性至关重要，于是可以使用中位数来替代平均数，如 MAPE。

对数损失函数也逻辑斯谛回归损失(Logistic Loss)或交叉熵损失(cross-entropy Loss),是在概率估计上定义的。如果只有两类 {0, 1}, 则对数损失函数的公式简化为

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i logp_i + (1-y_i)log(1-p_i))$$

根据具体业务，实际的评价指标有很多种，最好的方式当然是模型选择时即设计其损失函数即为评价指标，但是通常而言这些指标包含了某些非线性变化，优化起来难度颇大，因此实际模型选择仍是选用经典的那些损失函数，而模型评价则会与其略有不同。

在模型评估的过程中，我们可以判断模型的“过拟合”和“欠拟合”。若是存在数据过度拟合的现象，说明我们可能在训练过程中把噪声也当作了数据的一般特征，可以通过增大训练集的比例或是正则化的方法来解决过拟合的问题；若是存在数据拟合不到位的情况，说明我们数据训练的不到位，未能提取出数据的一般特征，要通过增加多项式维度、减少正则化参数等方法来解决欠拟合问题。此外，模型评估还应考虑时间、空间复杂度，稳定性、迁移性等。

8、 模型融合

一般来说，模型融合后都能使得效果有一定提升。而且效果很好。工程上，主要提升算法准确度的方法是分别在模型的前端（特征清洗和预处理，不同的采样模式）与后端（模型融合）上下功夫。因为他们比较标准可复制，效果比较稳定。而直接调参的工作不会很多，毕竟大量数据训练起来太慢了，而且效果难以保证。

9、 上线运行

这一部分内容主要跟工程实现的相关性比较大。工程上是结果导向，模型在线上运行的效果直接决定模型的成败。 不单纯包括其准确程度、误差等情况，还包括其运行的速度(时间复杂度)、资源消耗程度（空间复杂度）、稳定性是否可接受。这些工作流程主要是工程实践上总结出的一些经验。并不是每个项目都包含完整的一个流程。这里的部分只是一个指导性的说明，只有大家自己多实践，多积累项目经验，才会有自己更深刻的认识。