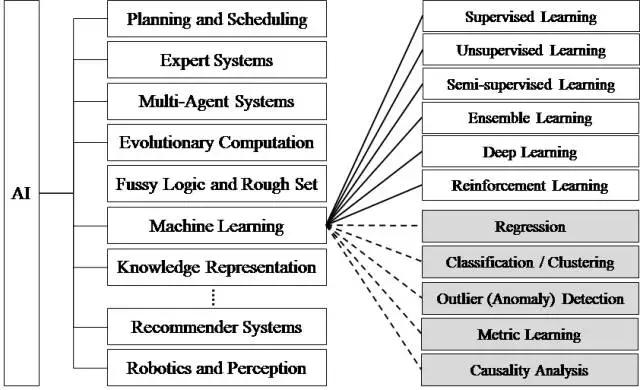
**人工智能课期末总结报告**

**刘笑 2015201925**

在本学期的这门人工智能课上，一开始我是比较疑惑的，因为使用的教材《Hands-On Machine Learning with Scikit-Learn and TensorFlow》是一本讲述机器学习和深度学习的教材，感觉和人工智能好像有些区别。

后来了解到人工智能，机器学习，和深度学习的关系如下：



**机器学习是实现人工智能的一种方法，而深度学习是一种实现机器学习的技术。**

在这门课上以制作小车的形式来让我们初步入门人工智能，这个项目具备可操作性和趣味性，我和小组其他的成员在制作的过程中学习到了很多东西。

接下来我将分别就课堂学习和课下AI-car的项目制作部分分别做课程总结。

1. **课堂学习：机器学习部分**

在附录中作者提到的关于机器学习项目的一般步骤，从总体上来说，一个Machine Learning Project Checklist可以分为下面8个大步骤。

1. Frame the problem and look at the big picture

2. Get the data.

3. Explore the data to gain insights

4. Prepare the data to better expose the underlying data patterns to Machine Learning algorithms

5. Explore many different models and short-list the best ones

6. Fine-tune your models and combine them into a great solution

7. Present your solution

8. Launch, monitor, and maintain your system

这也是第二章的开头内容。

这几大步骤非常简洁明了地概括了一个机器学习项目应该做的事情。

同时也需要注意到需要问你自己三个问题：“问题属于 监督学习 无监督学习 还是强化学习？”、“这个任务属于 分类问题 还是 回归问题 或者其他？”、“是否需要使用离线学习或者在线学习？”

**以课本中提到的手写分类的任务为例。**

我在课下的实验中采用的是基于Python和NumPy函数库，KNN算法实现的手写识别。

KNN（k最近邻）算法，是比较简单的机器学习算法的一个例子，算法核心在于从训练样本集中取出k个与测试样本欧式距离（或者其他类型的“距离”），将这k个样本中最高频率的类别作为测试样本。

kNN算法主要被应用于文本分类、相似推荐。

**1.1步骤一：问题描述**

目标：分类未知类别的手写数字0~9

输入：待分类的未知手写数字图片。已知数字类别集合D，其中包含j个已知类别的案例。

输出：该图可能属于的数字类别。

**问题定位**

问题类型：监督学习，测量指标：最终的输出结果是否正确

**1.2步骤二：**获取数据

MNIST提供了很全面的手写数字数据集，但是因为我是Windows系统，解压不了MNIST网站提供的压缩包，就自己在GitHub上找了另一组处理过的数据。0~9每个数字都有10个训练样本和5个测试样本。

**1.3步骤三：备份原始数据**

**1.4步骤四：整理数据**

将数据处理成可以给算法跑的标准数据，首先需要进行数据清洗，把一些缺失的地方删掉或者填充（比如0，平均数，中位数等等），把偏离很远的点进行删除或者修正。

接着进行特征选择，将一些很明显对分析没有帮助的数据删掉。然后就是极为重要的特征工程，为了数据更好的被利用，或者让算法执行效率更高，我们常常需要对数据做很多分析和变换，在这里我不能展开了说，一方面自己也只是一个初学者，很多不懂，还需要继续学习

**1.5 使用不同的模型跑并分析性能优劣**

knn算法可以直接调用sktlearn的库函数，这里我的目的很明确，就不需要调用不同的算法了。

**1.6调节参数**

这里KNN最重要的地方在于k的取值，但是一般来说k不能大于20

（注：代码及数据集展示在GitHub提交的目录中）

1. **课堂学习：深度学习部分**

深度学习是一个交叉学科，涉及到神经网络、人工智能、图建模、最优化理论、模式识别和信号处理等多学科领域知识。近年来硬件计算能力的升级与发展，使得深度学习在人们的日常生活中有了用武之地。

深度学习的应用领域包括计算机视觉、语音识别、图像识别，自动驾驶，自然语言理解、手写识别、音频处理、信息检索、机器人学等。它是机器学习的众多方法之一，其它方法包括决策树学习，归纳逻辑程序设计

教材的作者在自己的个人GitHub上提供了书中涉及到的各种代码，我跟着跑了deep\_learning.ipynb对应的代码。

解决神经网络问题的基本步骤也与上文机器学习的步骤差不多，我在Coursera上面也报了Andrew Wu教授的DeepLearning.ai课程，以其第二周的课后作业为例：

（注：课程地址和参考链接在GitHub提交的目录中）

**2.1确定问题是否适用于神经网络，是否优于传统机器学习方法**

问题：构建一个图像识别算法，建立一个猫分类器。

——这是一个典型的适用于神经网络的模型

**2.2调查是否已有合适的神经网络结构已经适用于已有的问题**

在这里，题目要求适用逻辑回归（Logistic Regression）

**2.3确定通过哪种框架实现拟建立的神经网络结构**（tensorflow/theano/caffe…）

题目要求使用numpy,matplotlib,PIL和scipy等库函数

**2.4转换数据为合适的进入结构，确定batches，进行数据清洗**

**2.5重复整理数据，以获得更多数据样本拟合模型**

**2.6训练模型，验证模型并进行参数优化**

**2.7测试模型进行比较和应用**

1. **课下项目制作**

本学期的课下项目分为三个阶段，报告截止前我们完成了两个阶段。

**3.1 Random Walker**

在该阶段中我们主要做的有两件事情：安装组件小车，以及控制小车。

在安装阶段我们遇到了各种各样的问题，比如说没有面包板，电池没电，后来被短路，接线错误等等，最终装好了小车。

初始时，在烧代码的时候，loop部分我们默认小车如果距离感应在某个范围内时就自动右转，实现了比较固定的小车行走。

**3.2 Sensored Walker**

在这个阶段中，经过小组讨论一致决定，我们实现了图像识别与判断功能，为了分类方便，仅限于小车识别墙壁，即小车根据图像中显示的墙壁的夹角情况判断是否需要左右拐弯。

期间我们又遇到了一些硬件上的问题，以及蓝牙识别等等问题，并逐一解决。小组成员在课下时间进行了两次实验，收集了超过两千张图片，由我进行筛选和判断，再制作成合适的输入，提供测试集与训练集的原始素材。

查找多个相关图像识别算法后选取了较为经典的图像识别算法，分类套用了SVM，再将预处理过后的图像对应成01标识（分别代表左拐和右拐），采用交叉验证降低过拟合的可能性。

多次调参后正确率大概在70%左右