人工智能课程报告

廖钰蕾　2015201953

1　项目实践

1.1　课程作业

1.1.1　小车随机游走

第一个阶段实现小车随机游走。实验中我主要负责代码调试与拷入，协助完成小车拼装、蓝牙接口调试等问题。由于小车质量问题，代码需稍作调整才能实现直线行走，实验中我学会了单片机的使用方法与基本工作原理。

1.1.2　机器学习模型

第二个阶段实现融入机器学习模型。我们小组希望实现红绿灯检测与根据图片自动转向两个功能。本实验中我主要负责算法设计和代码实现两部分，第二次实验中我学到了很多东西。

红绿灯检测的实现并不复杂，通过仔细阅读 OpenCV 的开发文档，我发现可以把图片从常用的 RGB 格式转化为 HSV 格式，根据遍历色调的角度值可以直接估计出其所属颜色。具体代码如下：

```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
hsv = np.array(hsv[:, :, 0])
for i in range(hsv.shape[0]):
    for j in range(hsv.shape[1]):
        if hsv[i][j] > 0 and hsv[i][j] < 10:
            ans = 5
        if hsv[i][j] > 156 and hsv[i][j] < 180:
            ans = 5
        if hsv[i][j] > 35 and hsv[i][j] < 77:
            ans = 1
```

其中 ans=5 为红灯色系，ans=1 为绿灯色系。

机器学习部分我们组打算攻克墙角识别问题，根据图片中墙角的形状判断出小车应该左转还是右转。实验过程却不太顺利。一开始队友带着小车拍照积累数据，我则开始查找文献研究算法。第一批数据大约有 300 张图片，我直接将像素点拉直通过 SVM 算法交叉验证分类后，发现过拟合严重，测试集正确率较低，大约在 50%左右。于是队友继续收集数据，我

则开始思考如何改进算法。

查阅资料过程中我发现，SIFT 特征是计算机视觉的常用特征之一。于是我考虑提取 SIFT 特征后再进行分类。对每一张图片提取特征后，我发现每一张图片对应的向量数是不同的，但向量维度相同。这就意味着我将向量拉直后每张图片对应的向量维度不同，而简单的补零填充法对准确率提升并没有太大用处。

在查阅各类机器学习问题过程中，我从自然语言处理领域得到启发，开始尝试使用 Bag of Words 词袋模型解决问题。BOW 模型最初用于文本分析领域，将每个文档看成袋子，将袋子中的词语划分成具有不同情感的几类词，对于每篇文档，根据袋中词语所属类别，即可估计出文档的情感类别。在计算机视觉中，可以对 SIFT 向量进行聚类，每一簇向量看成一类“词”，每一幅图片看成一个“袋”，就可得到每一幅图片所属类。

参考自然语言处理实验步骤，我在模型中增加了图片预处理（通过 openCV 自带函数实现）、归一化等步骤，最终我提出如下训练模型并完成代码，在测试过程中达到 70%的准确率：

1. 图片预处理：对图片灰度处理，使用高斯平滑处理图片降噪，再使用 Canny 边检测器检测边，能凸显物体边，且降噪效果相对拉普拉斯边检测器等要好；
2. 特征提取：通过 openCV 对每张图片提取 m*128 的特征向量，m 为特征点数；
3. 特征聚类：提取所有图片的 SIFT 特征并放依次排在一个矩阵中，总共有 n 个向量，将这 n 个向量通过 k-means 算法聚类划分成 k 类，这样 m 个 SIFT 特征均可映射到 k 维向量上；
4. 归一化：由于每幅图片对应多个向量，且个数不定，需计算每一类出现的频率，做归一化处理；
5. 模型调优：经过上述步骤，每幅图片均映射到 k 维向量上。采用 SVM 模型，通过 GridSearchCV 函数在参数范围内自动调优，寻找最优超参数，范围如下：

```
parameter_grid = [
        {'kernel': ['linear'], 'class_weight': ['balanced'], 'gamma': [0.01, 0.001], 'C': [0.5, 1, 10, 50]},
        {'kernel': ['poly'], 'class_weight': ['balanced'], 'gamma': [0.01, 0.001], 'degree': [2, 3]},
        {'kernel': ['rbf'], 'class_weight': ['balanced'], 'gamma': [0.01, 0.001], 'C': [0.5, 1, 10, 50]},
    ]
```

6. 模型存储：将得到的 kmeans、svm 模型保存，方便使用。

1.1.3 深度学习模型

第三次实验尚在进行中，基本思想是采用深度学习模型进行图像识别。深度学习模型通常在大数据集下表现较好，在实际情况中我们很难获取大量数据集。后续我个人的一些实验也给第三阶段工作完成提供参考。

1.2 图象识别

mnist 数据集囊括了 6 万个训练集和 1 万个测试集。每个样本都是 28*28 的像素值，并且是黑白的，没有 RGB 三层。我们要做的是把图片分到 0-9 类别中。注意到输出层需要 10 个结点，所以我们最好把数字 0-9 做成 One Hot 编码的形式。

1.2.1 mnist 数字识别——C++纯手工实现 DNN

本学期我参加了格灵深瞳公司 AI Playground 项目，项目的第一个任务是用 C++实现一个简单的 DNN，仅包含全连接层和激活层，并以在线训练测试的方式输出结果，训练样本数最多为 10000，整个流程（数据读取+训练+测试）限制时长 4s。

对我而言，这个任务收获最大。这个任务仅允许调用 C++的标准库函数，因而全连接层及激活层的核心代码必须手动实现，且网络层数太浅时精度不佳，太深时会造成超时。平时写神经网络时都是在配置文件中简单调参，且会提供默认参数，这次却需要在我自己的代码中修改参数。

通过这次实验，我充分了解了神经网络背后的数学基础，以及各层次之间的关联和作用，而不仅仅是当作一个黑盒去使用它。最终我实现了一个包含 2 层全连接层的神经网络模型，并在最终的在线测评中达到了 91%的准确率。

代码实现：由于代码太长，就不在这里贴出了，仅给出 GitHub 地址：
https://github.com/liaoyulei/Mnist

1.2.2 mnist 数字识别——keras 实现 CNN

CNN 在图片处理中被广泛使用，因为它具有局部连接、全局共享的特点，考虑到了一个图片中每个像素点与其周围像素点的关联性。另一方面卷积操作复杂度较大，运行速度慢，对计算机性能要求较高。在自然语言处理中也具有一定应用。

由于自身机器性能限制，我仅在 CPU 上运行神经网络模型，在搭建较深层神经网络时，训练次数较少，准确率并没能达到大多数论文中的水平。经过 5 个 epoches 训练，准确率可达到 92%，训练代码如下：

```
model = Sequential()
model.add(Conv2D(filters = 64, kernel_size = (3, 3), strides = (1, 1), padding = 'same', input_shape = (28, 28, 1), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.5))
model.add(Conv2D(128, kernel_size = (3, 3), strides = (1, 1), padding = 'same', activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.5))
model.add(Conv2D(256, kernel_size = (3, 3), strides = (1, 1), padding = 'same', activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.5))
```

```
model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(32, activation = 'relu'))
model.add(Dense(10, activation = 'softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer = 'adagrad', metrics = ['accuracy'])
```

1.3 自然语言处理

Imdb 是一个包含大量影评的数据集，在 keras 中，imdb 已经对每个词标注了索引，创建了字典，每段文字的每个词对应了一个数字，得到的数据集均由数字构成。使用时只需读入数据集，不需要进行构建词典等处理。

1.3.1 词嵌入技术——Word2Vec 处理 text8 语料库

词嵌入技术是给每个词赋一个向量，向量代表空间里的点，含义接近的词，其向量也接近。得到的词向量可以直接当作特征向量使用。Python 的 gensim 包中封装了 Word2Vec 的基本功能。基本使用步骤如下：

```
sentences = word2vec.Text8Corpus('text8')
model = word2vec.Word2Vec(sentences, size = 200)
print(model['me'])
```

1.3.2 Imdb 情感分析——keras 实现 DNN

由于文本长短不一，首先需要对文本进行补齐，再利用词嵌入技术（可使用之前介绍的 Word2Vec，也可使用 keras 自带的嵌入层 Embedding）转为向量用于训练。

经过 3 词 epoches 后，准确率可达到 88%，核心训练代码如下：

```
model = Sequential()
model.add(Embedding(vocab_size, 64, input_length = maxword))
model.add(Flatten())
model.add(Dense(2000, activation = 'relu'))
model.add(Dense(500, activation = 'relu'))
model.add(Dense(200, activation = 'relu'))
model.add(Dense(50, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

1.3.3 Imdb 情感分析——keras 实现 CNN

之前在图像识别中提到过，CNN 也可应用在自然语言处理领域中。因为文字也具有局部特征，一个词的前后几个词和这个词本身有关。思想上与之前提到的词袋模型也有想通之

处。核心代码如下：

```
model = Sequential()
model.add(Embedding(vocab_size, 64, input_length = maxword))
model.add(Conv1D(filters = 64, kernel_size = 3, padding = 'same', activation = 'relu'))
model.add(MaxPooling1D(pool_size = 2))
model.add(Dropout(0.25))
model.add(Conv1D(filters = 128, kernel_size = 3, padding = 'same', activation = 'relu'))
model.add(MaxPooling1D(pool_size = 2))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(64, activation = 'relu'))
model.add(Dense(32, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))
model.compile(loss = 'binary_crossentropy', optimizer = 'rmsprop', metrics = ['accuracy'])
```

1.3.4 Imdb 情感分析——keras 实现 LSTM

LSTM 是 RNN 的一种，本质上，它按照时间顺序，把信息进行有效的整合和筛选。这样在每一个时刻，此前到来的信息都被保留一部分，依然会对这一步的训练带来影响。该模型应该是目前自然语言处理中应用最广泛的模型之一。核心代码如下：

```
model = Sequential()
model.add(Embedding(vocab_size, 64, input_length = maxword))
model.add(LSTM(128, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(64, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(32))
model.add(Dropout(0.2))
model.add(Dense(1, activation = 'sigmoid'))
model.compile(loss = 'binary_crossentropy', optimizer = 'rmsprop', metrics = ['accuracy'])
```

1.4 语音识别

语音识别中的深度学习模型训练过程与计算机视觉和自然语言处理基本是想通的，不再重复介绍。简要介绍下语音识别的提取工具和统计模型。

1.4.1 特征提取

语音识别常用的特征是 MFCC 特征、超音段特征等。常用的提取工具有 openSmile，主要通过命令行完成特征提取；kaldi，需要较大存储空间，功能更全面更专业，主要通过 shell 脚本完成；python_speech_features，python 中封装的语音特征提取包，主要用于提取 MFCC

特征。

1.4.2 统计模型——HMM

HMM 模型建立在马尔可夫链和混合高斯模型的基础上，通常情况下，有 n 个类就需要训练 n 个 HMM 模型，相对于其它模型需要考虑语音长度不定需要填充的问题，HMM 模型可直接使用 MFCC 特征进行训练。只需要将同一类的 MFCC 特征依次排列成一个矩阵即可，训练代码如下：

```
model = hmm.GaussianHMM(n_components = 4, covariance_type = 'diag', n_iter = 1000)
model.fit(X)
```

2 理论知识

2.1 梯度下降法

梯度下降法是最基本的最优化模型，通常用在机器学习、深度学习模型最底层，以此为基础改进得到的算法有牛顿法（理论上精度更高，实际上更难实现）等，具体算法步骤如下（来自百度百科）：

举一个非常简单的例子，如求函数 $f(x)=x^2$ 的最小值。

利用梯度下降的方法解题步骤如下：

1、求梯度，$\nabla = 2x$

2、向梯度相反的方向移动 x，如下

$x \leftarrow x - \gamma \cdot \nabla$，其中，$\gamma$ 为步长。如果步长足够小，则可以保证每一次迭代都在减小，但可能导致收敛太慢，如果步长太大，则不能保证每一次迭代都减少，也不能保证收敛。

3、循环迭代步骤2，直到 x 的值变化到使得 $f(x)$ 在两次迭代之间的差值足够小，比如0.00000001，也就是说，直到两次迭代计算出来的 $f(x)$ 基本没有变化，则说明此时 $f(x)$ 已经达到局部最小值了。

4、此时，输出 x，这个 x 就是使得函数 $f(x)$ 最小时的 x 的取值 。

2.2 最大期望算法

EM 算法主要用于在概率模型中寻找最大后验估计，其概率模型依赖于无法观测的隐藏变量。是 HMM 算法的基础。算法流程如下（来自百度百科）：

最大期望算法经过两个步骤交替进行计算：

第一步是计算期望（E），利用概率模型参数的现有估计值，计算隐藏变量的期望；

第二步是最大化（M），利用E 步上求得的隐藏变量的期望，对参数模型进行最大似然估计。

M 步上找到的参数估计值被用于下一个 E 步计算中，这个过程不断交替进行。

总体来说，EM的算法流程如下：

1.初始化分布参数

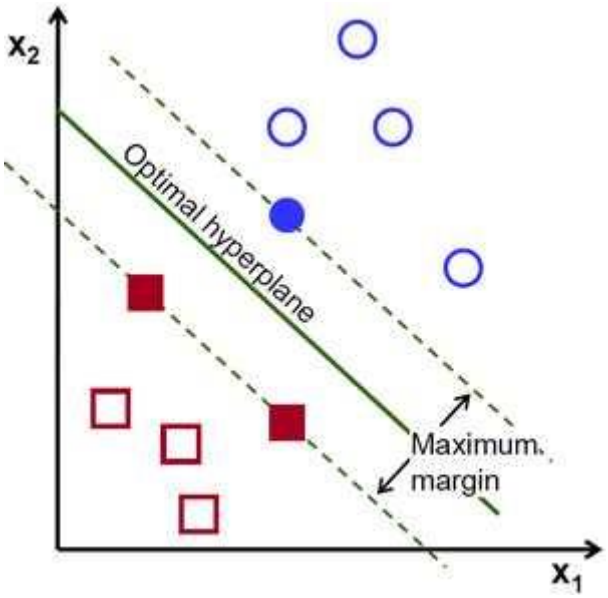
2.重复直到收敛：

E步骤：估计未知参数的期望值，给出当前的参数估计。

M步骤：重新估计分布参数，以使得数据的似然性最大，给出未知变量的期望估计。

2.3 支持向量机

SVM 可以理解为通过一个面（或线）将已有数据点划分为两个区域，且使得分类间隔最大。二维平面如图所示（来自知乎）：



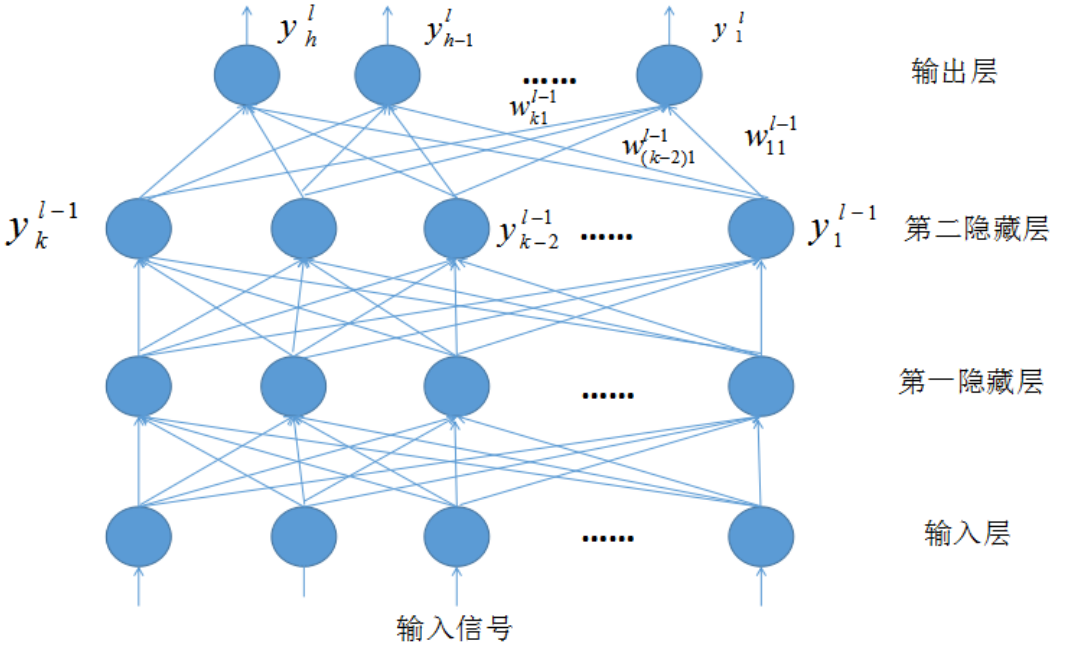
知乎上有很多直观理解 SVM 的问题，在实际中我们将二维平面扩展为多维情况，最底层的优化过程通常通过梯度下降法实现。

2.4 神经网络

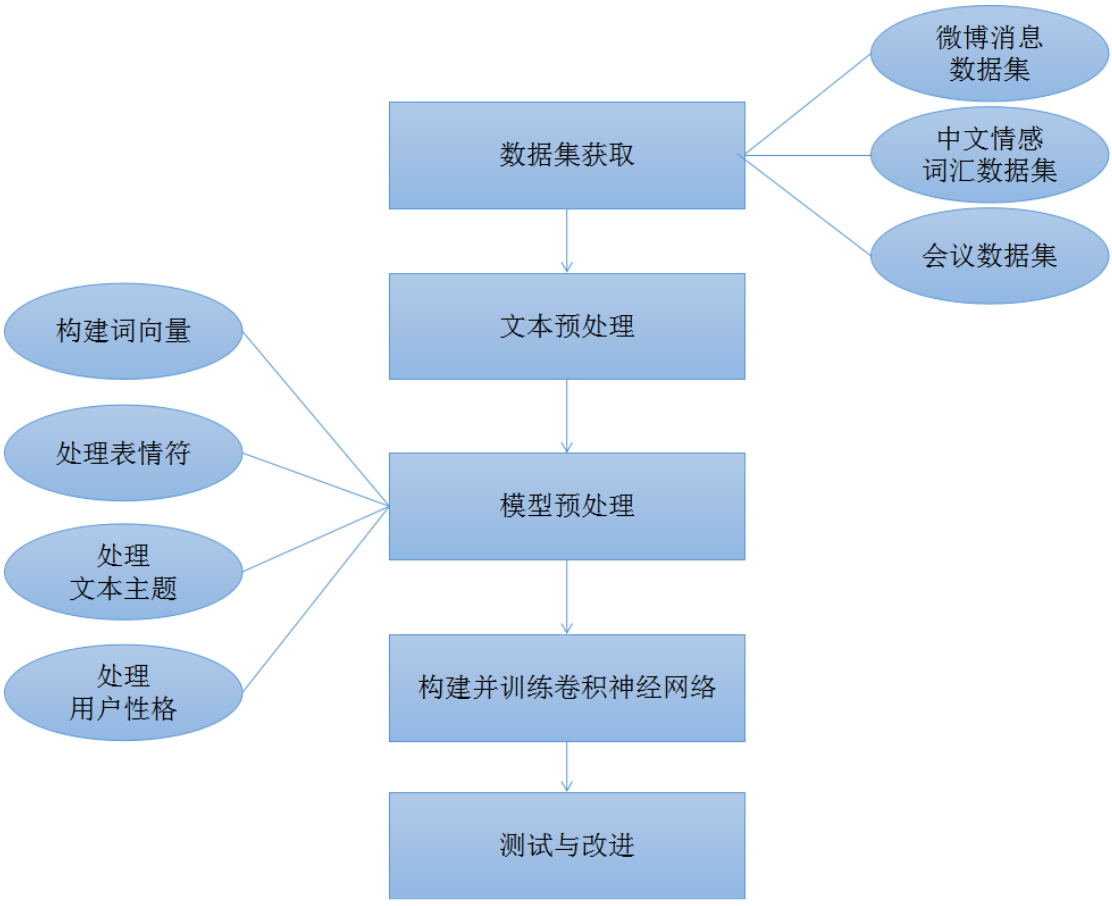
目前而言，各个神经网络模型划分不太明确，各模型训练效果差异通常不明显。且更改

数据格式后往往也可以使用另一种神经网络模型训练成功。

各神经网络均采用增加隐藏层的方式提高模型复杂度。通常情况下，越复杂的模型越难以训练好，但它可达到的效果往往更高。



通常来讲，神经网络训练过程遵从以下几个步骤（以自然语言处理为例）：



3 个人思考

项目实践部分的完整代码我已整理在 GitHub 中：https://github.com/liaoyulei/AI

通过本学期人工智能课程以及其他相关课程学习，我从一个机器学习小白逐渐蜕变成为一个能够调包能简单实现核心代码的初学者。从最基础的统计学模型，到后期演变出的机器学习模型，到现在一统天下的深度学习模型均有涉猎。但这些工作并没有坚定我对这个领域的信心，相反，我逐渐产生了一些困惑。

就我个人而言，学习概率论、随机过程等数学课程的过程中，我对概率、测度、高斯分布、马尔可夫链等知识具有更透彻的了解，再来学习这些机器学习模型时也相对容易理解。但在学习神经网络的过程中，我往往不能理解这个模型，似乎只是论文里说它好，实验发现它好，却不知道它为什么好。

一直以来，计算机学科就是建立在数学的基础上发展至今的。理论计算机作为计算机领域最核心的部分，也是由抽象的数学语言建立起来的。如今一统天下的神经网络，虽然发展的很火，很多计算机方向的研究生都投身此领域，却至今依然没有找寻到它的理论基础，而大多数研究者则陷入了纯应用、唯结果论的研究方向中。一个没有理论基础的研究领域，很容易遇到瓶颈吧。

这门课学时较少，加上涉及到的数学、统计基础知识较多等原因，很难对机器学习底层的数学知识有很透彻的讲解。老师在本学期的授课过程中，既有基础的公式、论文作支撑，又有代码演示、课设实践等环节，让大多数同学都具备了基本的实践能力，作为一门导论课已达到了其应有效果。但作为一名计算机专业学生来说，希望后续有更理论的课程，或者指明一条基础知识的学习方向，只有知其根本才能去研究它改变它，而不是仅仅具备调用黑盒的能力。