人工智能课程论文

人工智能在雷达数据处理中的应用分析与实践

学　　号：	31956004
姓　　名：	高　　亮
授课教师：	赵 国 亮
专业年级：	信息与通信工程(2019 级)
学　　院：	电子信息工程
学　　校：	内蒙古大学
时　　间：	2019 年 6 月

摘　要

近年来，人工智能在雷达数据处理中应用越来越多。如利用人工智能对雷达探测得出的距离多普勒二维谱进行目标识别[1]；利用人工智能方法对海杂波进行高精度预测[2]；利用 BP 神经网络、遗传算法、卷积神经网络和循环卷积网络，尤其是长短时记忆网络(LSTM)，对目标 AIS 点迹进行预测以辅助海上交通管理，从而避免拥堵碰撞事件发生[3-13]；还可利用 BP 神经网络对雷达探测到的目标数据进行修正补偿[14]；此外，也有利用深度递归神经网络对雷达目标进行轨迹跟踪的研究[15]。

本文主要对人工智能在雷达数据处理方向的应用进行了分析与研究。本文的目的是利用所学的人工智能知识并结合个人研究方向做一些理论与实践相结合的总结和探索。现将本文的脉络结构组织如下。

第一章在宏观概念上介绍了人工智能在雷达领域的应用放向以及编程所用的系统与软件环境；第二章介绍了如何用神经网络处理分类任务，并给出了部分最基础的传统分类数据集处理代码；第三章介绍了如何利用循环卷积神经网络处理时序数据并结合模拟的雷达 AIS 点迹数据预测目标轨迹；第四章对全文进行了总结。

关键词：人工智能；神经网络；分类；时间序列；LSTM

Abstract

In recent years, artificial intelligence has been increasingly used in radar data processing. For example, artificial intelligence is used to identify the target of the range Doppler two-dimensional spectrum detected by radar [1].Using artificial intelligence to predict sea clutter with high accuracy [2];BP neural network, genetic algorithm, convolutional neural network and circular convolutional network, especially the long and short time memory network (LSTM), are used to predict the target AIS point trace to assist Marine traffic management, so as to avoid the occurrence of congestion collision events [3-13].BP neural network can also be used to modify and compensate the target data detected by the radar [14].In addition, there are also studies on the trajectory tracking of radar targets using deep recursive neural networks [15].

This paper mainly analyzes and studies the application of artificial intelligence in radar data processing.The purpose of this paper is to summarize and explore the combination of theory and practice with the artificial intelligence knowledge and personal research direction.The contextual structure of this paper is organized as follows.

The first chapter introduces the application of Artificial intelligence in radar field and the programming system and software environment. The second chapter introduces how to use neural network to deal with classification task and gives some basic processing codes of traditional classification data set. The third chapter introduces how to process the timing sequence data by using cyclic convolutional neural network and predict the target trajectory by combining the simulated radar AIS point trace data. Chapter four summarizes the whole thesis.

Key Words: Artificial Intelligence; Neural Network; Classification; Time Sequence; LSTM

目 录

第一章 概述

本章将对本次人工智能课程报告的整体内容进行概述。着重介绍本次报告的研究背景、研究内容与方法、系统及软件环境等，下面将依次介绍。

1.1 研究背景

由于个人的研究方向为高频地波雷达信号处理，本节将介绍一些个人所掌握的人工智能在雷达领域内的应用。

雷达(radar)是英文无线电测距的音译。其基本原理是利用电磁波探测某个区域目标的距离、速度、方向、大小、类型等。

人工智能的优势在于其强大的运算和分析能力，但是这个公认的优点在雷达的信号处理前端并不实用。这是因为人工智能是基于数据的处理分析模式，但雷达前端恰恰是数据的源头。典型的雷达前端框图如图 1.1 所示。

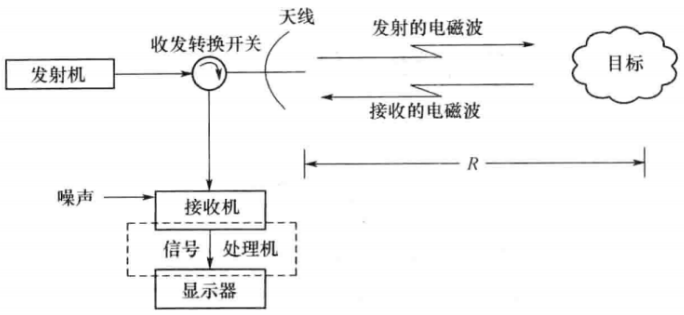


图 1-1 雷达前端信号处理框图

通俗地讲，雷达是一种传感器，它是一种获取某个区域目标信息的工具。这种工具的优劣是用所得数据的准确性来衡量的，而且数据的获得大多是实时性的。

相比于这种准确性和实时性的需求，人工智能则需要大量的历史数据作来训练处理分析模型，这就耗费一定的时间。因此，人工智能用于雷达后端的数据处理与分析比较合适。雷达探测到的数据存储下来之后，才可以进行进一步地处理与分析。

雷达数据处理要用到人工智能方法的内容主要有：

- 分类问题：根据目标回波数据辨别目标类型(舰船、飞机、海冰等)
- 预测问题：根据目标回波数据预测目标轨迹等
- 其他问题

在分类问题中，回波数据主要是目标的距离多普勒三维成像图和一些特征数据(距离、速度、方位角、回波功率等)。距离多普勒三维图是目标的距离与速度信息的载体，图中目标峰值的大小与目标的雷达截面积(RCS)成正比，当然也与背景噪声等因素有关。一种典型的雷达距离多普勒图像如图 1.2 所示。

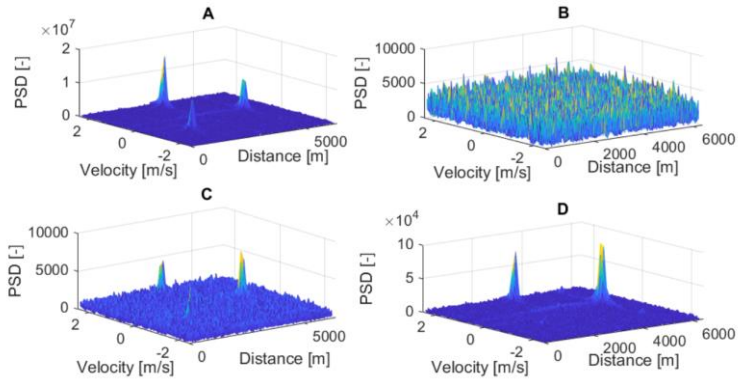


图 1-2 雷达距离多普勒三维图

从上图中可以看到，这种距离多普勒三维图类似于一种图片，因此可以与人工智能图片分类问题进行类比。

在预测问题中，回波数据主要指的是目标的坐标、经纬度、速度、距离等信息。这些数据通常含于合作船只的 AIS(Automatic Identification System)数据中。利用 AIS 数据进行目标运动轨迹预测属于机器学习领域中的回归问题。下面简单地介绍 AIS。

维基百科有关 AIS 的描述。

AIS 是安装在船舶上的一套自动追踪系统，借由与邻近船舶、AIS 岸台、以及卫星等设备交换电子资料，并且供船舶交通管理系统辨识及定位。当卫星侦测到 AIS 讯号，则会显示 S-AIS。AIS 资料可与海事雷达进行数据对接，以优先避免在海上发生交通碰撞事故。

由 AIS 所发出的信息包括独特的识别码、船名、位置、航向、船速等会显示在 AIS 的屏幕或电子海图上。AIS 可协助当值船员以及海事主管单位追踪及监视船舶动向。AIS 整合了标准的 VHF 传送器以及由 GPS 或 LORAN-C 接收器所提供的位置信息。船舶装有 AIS 收发机和问答机时，可以被 AIS 岸台所追踪。当远离海岸过远时，可利用 AIS 接收器，经由相当数量的卫星以便从庞大数量的信号中辨识船位。

国际海事组织中《国际海上人命安全公约》（SOLAS）要求航行于国际水域，总吨位在 300 以上的船舶以及所有不论吨位大小的客船，均应安装 AIS。

将历史记录的 AIS 数据输入到人工智能模型中训练。如果训练后，模型可以以很高的准确率预测出船只位置、速度等信息，那么其结果便可以指导实际的对应海域上的资源调度安排。故此选题具有一定的现实意义。

1.2 研究内容与方法

结合首节所述，人工智在雷达领域的应用主要是数据处理。本文主要从分类和回归预测两个角度进行理论与实践上的总结分析。

由于雷达数据集很庞大且暂时没有收集到可用于训练的高频地波雷达数据集，故本文将着重在原理上进行分析与总结，用机器学习领域中经典的数据集来辅助分析分类问题原理。在目标航行轨迹预测中将使用正弦函数函数生成模拟目标 AIS 坐标数据，再用循环卷积神经网络进行预测。

在软件实践方面，本文将用一些小巧简易的分类数据集来代替雷达数据集进行分类处理。但这种类比分析，对人工智能在雷达领域上的应用也是有积极意义的。

本文将用神经网络、卷积神经网络、循环卷积神经网络建立相应的理论模型。在实践分析环节，本文将结合谷歌人工智能开发框架 Tensorflow2.0 分别实现对应的理论模型。

1.3 系统与软件环境

本文实践部分所用的系统和软件环境如下所示。

系 统 及 配 置：Windows 10 ； 16G 内存； i9 标压处理器； 显卡 GTX1650

编程语言及版本：Python-3.7.7

人 工 智 能 库：Tensorflow-2.0

其 他 功 能 库：Numpy； Matplotlib； PIL； math； os

第二章 分类问题

在雷达的目标识别中要用到分类方法。比如根据探测到的目标速度、距离、方向、回波功率强弱等信息来判断目标是舰船、飞机还是海冰等。本章将用人工智能的方法来辅助雷达对探测到的目标进行的分类。

2.1 数据类比

由于高频地波雷达数据格式比较复杂且暂时缺少相关数据集，本章将用**鸢尾花分类数据集**和 **Mnist 手写数字识别数据集**来分别来类比雷达探测到的目标特征与距离多普勒二维图。

2.1.1 目标数据特征

假设回波数据中有目标的距离、速度、方位角、回波功率大小这四种信息。目标类型有舰船、飞机、海冰这三种类型，例如下表。

表 2.1 雷达探测到的目标特征数据示例

距离(km)	速度(m/s)	方位角(度)	回波功率(dBW)	目标类型
100	50	10	-10	飞机-0
150	20	20	-5	舰船-1
200	5	30	-6	海冰-2

为了将这些数据送入神经网络，还需要考虑格式问题。我们将目标类型分别编号为 0、1、2，这样有利于人工智能模型运算处理。

模型需要大量数据进行训练来得到最优权重参数。雷达的目标识别数据可用鸢尾花数分类据集进行类比。鸢尾花数据分类集的格式如下所示。通过对比表格数据，可知鸢尾花数据集与雷达探测到的目标数据具有结构相似性。

表 2.2 鸢尾花特征数据示例

花萼长(cm)	花萼宽(cm)	花瓣长(cm)	花瓣宽(cm)	鸢尾花类型
5.1	3.5	1.4	0.2	狗尾草-0
4.9	3.0	1.4	0.2	杂色-1
4.7	3.2	1.3	0.2	弗吉尼亚-2

鸢尾花数据集共 150 组，每组包括花萼长、花萼宽、花瓣长、花瓣宽 4 个输

入特征。根据输入特征的差异，共分为三类鸢尾花，分别为狗尾草鸢尾花、杂色鸢尾花、弗吉尼亚鸢尾花，分别用编码 0、1、2、表示。

0-狗尾草鸢尾

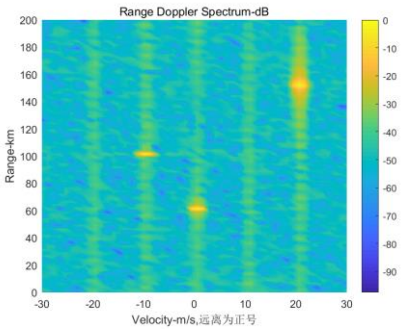
1-杂色鸢尾

2-弗吉尼亚鸢尾

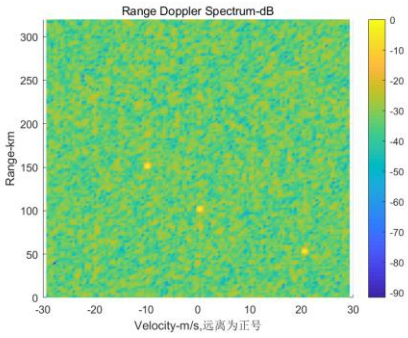
图 2-1　三种鸢尾花

2.1.2 雷达距离多普勒二维成像图

雷达的距离多普勒(RD)二维成像图是雷达领域的经典数据结构。实际上，RD 二维图就是 RD 三维图的俯视图。下面两个二维图分别是基于线性调频连续波(FMCW)和线性调频断续波(FMICW)原理及参数用 matlab 仿真生成的 RD 二维图。



(a) FMCW 距离多普勒二维图



(b) FMICW 距离多普勒二维图

图 2-2 高频地波雷达的距离多普勒二维图

从图中可以看到，横轴为速度，纵轴为距离。图中的亮斑表示目标。在仿真中设置了三个目标，预设的目标参数如图 2.2 所示。

```
16 —    R = [60, 100, 150]*1e3;    % 多目标目标位置向量，相对于雷达，单位千米
17 —    v = [0, -10, 20];          % 多目标目标相对雷达径向速度向量
18 —    RCS = [1, 1, 1];           % 多目标相对有效反射雷达截面积
```

```
35 —    R = [150 100 50]*1e3;    %目标1与雷达的初始距离200km
36 —    V = [-10 0 20];          %目标1径向速度10m/s,正号表示远离雷达，负号表示靠近
37 —    RCS = [1 1 1];           %目标1RCS
```

图 2-3　FMCW 仿真中目标的预设参数(上)；FMICW 仿真中目标的预设参数(下)

其实 RD 二维图就是一种图片。所以用机器学习领域中的图片分类识别方法处理是可行的。但在实际中，雷达得到的 RD 图大都是高分辨率的图像。所以需要用卷积算法提取输入特征降低数据量后，才可以对二维图进行分类。

RD 图中含有目标的距离与速度信息，人眼可以很直观地观察出距离与速度信息。但是目标的类型判断就不是很容易了。因为目标类型通常需要综合考量距离、速度、回波功率(亮斑大小)等信息，有经验的雷达观察员也许可以很快地给出判断结果，但是这并不是一种适合大量数数据分析的手段。

用人工智能的方法来实现上述功能是很合适的。只要用大量的二维图像训练模型，使其达到预设的识别目标准确率。其实，这个过程可抽象为根据图片特征判断图片类型的操作。因此在实践环节，本文将利用手写数字识别数据集来简要地说明图像识别分类原理。

Mnist 手写数字识别数据集分为训练集和测试集。训练集中有 60000 个 28×28像素的手写数字图片和对应的数字标签；测试集有 10000 个28×28像素的手写数字图片和对应的数字标签。相比于高分辨率的雷达的 RD 二维图，手写数字图片的像素级别很低。

2.2 基本原理

本节将着重介绍全连接神经网络、卷积神经网络两种人工智能领域内用于分类问题的处理方法。

2.2.1 人工神经网络概述

人工神经网络(Artifical Neural Networks，ANNs)是一种模仿动物神经网络行为特征，进行分布式并行信息处理的算法数学模型。这种网络依靠系统的复杂程度，通过调整内部大量结点之间的相互连接的关系，从而达到处理信息的目的，并具有自学习和自适应的能力。

人工神经网络是 20 世纪 80 年代以来人工智能领域兴起的研究热点。它从信息处理角度对人脑神经元网络进行抽象，建立某种简单模型，按不同的连接方式组成不同的网络。

在工程与学术界也常直接简称为神经网络或类神经网络。神经网络是一种运算模型，由大量的节点（或称神经元）之间相互连接构成。每个节点代表一种特

定的输出函数，称为激励函数(Activation Function)。每两个节点间的连接都代表一个对于通过该连接信号的加权值，称之为权重，这相当于人工神经网络的记忆。网络的输出则依网络的连接方式、权重值和激励函数的不同而不同。而网络自身通常都是对自然界某种算法或者函数的逼近，也可能是对一种逻辑策略的表达。

2.2.2 BP 神经网络

BP 网络(Back Propagation)是 1986 年由 Rumelhart 和 McCelland 为首的科学家小组提出，是一种按误差逆传播算法训练的多层前馈网络，是目前应用最广泛的神经网络模型之一。BP 网络能学习和存储大量的输入-输出模式映射关系，而无须事前揭示描述这种映射关系的数学方程。

BP 神经网络是一种具有三层或三层以上的多层神经网络，每一层都由若干个神经元组成，单个神经元模型如图 2-3 所示。

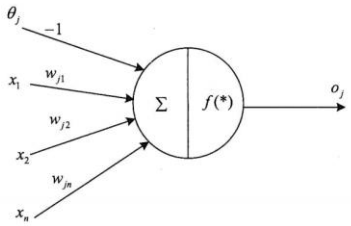


图 2-4　BP 神经元模型

其中，θ_i 表示偏置项；x_i 为输入特征；w_{ij} 为连接权重系数；$f(*)$ 表示某种激活函数；o_j 为神经元的输出。

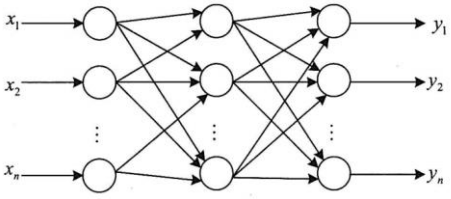


图 2-5 输入层、隐含层、输出层神经网络

简单地三层神经网络如图 2-4 所示。它的左右各层之间各个神经元实现**全连接**，即左层的每一个神经元与右层每个神经元都有连接，而上下各神经元之间无连接。BP 神经网络按“有导师”的学习方式进行训练，即在模型训练时利用已知正确的预测数据来修正模型权重参数。

当一些数据特征提供给网络后，各个状态的神经元的激活值将从输入层到隐

含层向输出层传播。然后，通过已知输出与实际输出的误差来逐层调节修正各连接权。由于这种修正过程是从输出到输入逐层进行的，所以称它为“误差逆传播算法"。随着这种误差逆传播训练的不断进行，网络对输入模式响应的正确率也不断提高，一般权重参数是根据**梯度下降法**更新的。

2.2.3 激活函数

激活函数是对神经元输出值的一种函数映射，除了 BP 神经网络外，卷积神经网络、循环卷积神经网络也要用到激活函数。常用的激活函数有以下几种。

1) sigmoid 函数

这种函数将正负实数域的输出值映射到 0-1 之间。其数学表达式及图形如下所示。

$$f(x)=\frac{1}{1+\exp(-x)}$$

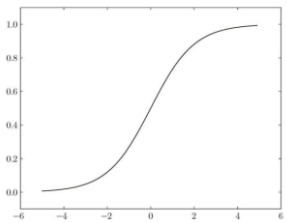


图 2-6 sigmoid 函数图

2) 阶跃函数

阶跃函数也将神经元输出值映射到 0-1 之间，但是其在 0 的左右变化剧烈，其实 sigmoid 是阶跃函数的一种优化，相比于阶跃函数，sigmoid 在 0 的附近变化比较缓和。这两种函数非常适合作二分类处理。阶跃函数的数学表达式和图形如下所示。

$$f(x)=\begin{cases}0, & x\le 0\\ 1, & x>0\end{cases}$$

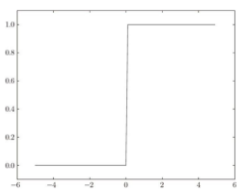


图 2-7 阶跃函数图

3) Relu 函数

Relu 函数是在 sigmoid 之后出现的很受欢迎的激活函数。此函数在输入大于 0 时，直接输出原值，当输入小于 0 时，输出恒为 0。其数学表达式及图形如下所示。

$$f(x)=\begin{cases}x, & x>0\\ 0, & x\le 0\end{cases}$$

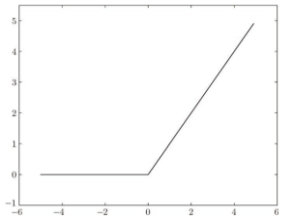


图 2-8　Relu 函数图

4) Softmax 函数

分类问题中常用 softmax 函数。这种函数可以将输出转为对应的概率值，其数学表达式如下。

$$y_k=\frac{\exp(a_k)}{\sum_{i=1}^{n}\exp(a_i)}$$

从上式可以看到，分子是某个神经元直接输出值的指数运算，分母是对输出层的所有直接输出值进行指数运算之后求和。这种运算会将输出层的某一个神经元输出转换为概率值，一般认为最大的那个概率值是模型所得到的结果。在与 one-hot 独热码后就可以利用程序代替人工进行分类判断。

独热码是有多少个状态就有多少比特的一种编码规则，而且只有一个比特为 1，其他全为 0 的一种码制。例如，有 6 个状态的独热码状态编码为：000001，000010，000100，001000，010000，100000。这个 1 正好是概率的最大值，因此只要将经过 softmax 的输出概率值与对应的独热码比较，概率相差最小的那一位就是分类结果。

比如经过 softmax 的输出概率为[0.1，0.2，0.7]，与之对应的三分类独热码是[0, 0, 1]，可以看到 0.7 与 1 最为接近，因此判断是第三类。数学中用来判断这种概率偏差大小的函数为交叉熵函数。

2.2.4 卷积神经网络

卷积神经网络(Convolutional Neural Network，CNN)被用于图像识别、语音识别等场合。深度学习领域的基础就是 CNN。

前述多层全连接神经网络已经可以进行分类识别等应用。采用全连接方式，如果数据量不很大，性能问题不大。如果遇到高维数据，性能可能会马上成为瓶颈。比如训练一张1000×1000像素的灰色图片，输入节点数就是1000×1000，如果隐含层节点是 100，那么输入层到隐含层间的权重矩阵就是1000×1000×100。

如还要增加隐含层，然后进行 BP，那结果可想而知。这还不是全部，采用全连接方式还容易导致过拟合。过拟合就是训练的时候效果很好，损失函数值可以降得很低，但是到测试数据集的时候表现就不那么好了，原因是过分依赖于现有训练数据集的特征造成的，解决方法是可以加大数据集或减少输入数据特征来进行训练。与过拟合相反的是欠拟合。

因此为了更有效地处理像图片、视频、音频、自然语言等大数据，必须另辟蹊径。经过多年不懈努力，人们终于找到了一些有效方法或工具。其中卷积神经网络、循环神经网络就是典型代表。接下来我们将介绍卷积神经网络，下一章将介绍基于循环神经网络的雷达目标航行点迹预测问题。

卷积神经网络的基本思路是在全连接神经网络前面加了一层数据处理层，被称为**卷积层**。其基本处理流程是利用小的数据处理框(卷积核)在原始输入数据上滑动，每走一步得到一个计算结果。例如下面的计算过程。

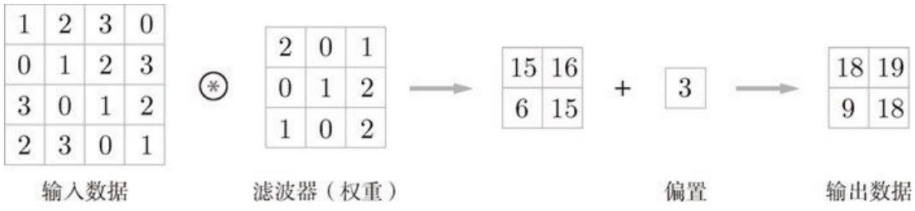


图 2-9 卷积核工作过程

卷积核在功能上等价于滤波器。由图可知，原始数据为结构为5×5，卷积核结构为3×3，让这个卷积核在原始数据上滑动计算，滑动步长为 1，滑动过程中，被卷积核覆盖到的元素都要与对应的卷积核元素相乘，所得结果再累加。如第一次卷积过程中得到的左上角元素 15 是按如下方式计算得来的。

$$15=1\times2+2\times0+3\times1+0\times0+1\times1+2\times2+3\times1+0\times0+1\times2$$

得到卷积结果后，再加上偏置项 3 就得到了最终值 18。实际的卷积步骤很复杂，涉及到输入数据量和结构，比如 RGB 彩色图像的三通道卷积、数据填充补 0 处理等，这里不详细说明。

卷积层得到的数据量依然很大，还要对结果进行一定的采样，比如最大值、最小值、均值采样，采样的过程被称为**池化**。池化后，还要对所得数据进行归一化等处理。经过多层卷积、池化处理后，数据量会显著减少，但是余下的数据中含有原始数据的大部分特征。将这样的数据再送入全连接网络处理就可以得到理想的结果。

2.3 基于 Tensorflow2.0 的目标分类实现

本节将基于鸢尾花数据集和手写数字识别数据集进行进行实践分类分析。代码中将不使用 Tensorflow 封装好的框架，而是采用最贴近底层的原理进行编程。

2.3.1 基本步骤

在分类问题中，文本分类与图片分类的步骤基本一致。基本步骤有数据预处理、模型训练和分类实现。

数据预处理的目的是使得数据集格式与使用的平台库相匹配。如果输入数据值过大会影响程序运行速度或达到内存上限造成数值溢出，因此将所有数据值与最大值进行比较，即归一化，是必要的。如果输入的是彩色图片，由于没用到彩色特征进行手写数字识别，所以需要对 RGB 三通道数据进行灰度化处理。处理之后的数据要进行拉直——形状调整，即每张图片排成一行送入模型进行训练。

图 2-10 分类问题基本步骤

在模型训练环节中，首先要输入预处理后的数据；接着送入全连接网络或进行卷积运算以压缩训练数据量；经过多批次循环训练后，根据梯度下降法得到训练后的最优参数组并保存下来以备断点续训或分类处理。最后调用保存的最优参数结合模型进行分类处理。

2.3.2 数据预处理

1) 鸢尾花数据集预处理

在进行编程之前，要导入相关的库和模块，具体的代码如下。

```
# 导入所需库和模块
import tensorflow as tf
from sklearn import datasets
from matplotlib import pyplot as plt
import numpy as np
```

利用 sklearn 库可直接导入鸢尾花数据集。

```
# 导入数据，分别为输入特征和标签
x_data = datasets.load_iris().data
y_data = datasets.load_iris().target
```

导入数据后需要对原始数据排列顺序进行打乱，目的是防止模型机械记忆。

```
# 随机打乱数据
# 使用相同的 seed，保证输入特征和标签一一对应
np.random.seed(116)
np.random.shuffle(x_data)
np.random.seed(116)
np.random.shuffle(y_data)
tf.random.set_seed(116)
```

将打乱后的数据前 120 组作为训练集，后 30 组作为测试集。

```
# 将打乱后的数据集分割为训练集和测试集，训练集为前 120 组，测试集为后 30 组
x_train = x_data[:-30]
y_train = y_data[:-30]
x_test = x_data[-30:]
y_test = y_data[-30:]
```

在送入模型之前要对数据特征进行类型转换以使得数据格式与模型匹配。最后，将输入数据与对应标签进行配对并分批次打包等待送入模型。

```
# 转换数据类型
x_train = tf.cast(x_train, tf.float32)
x_test = tf.cast(x_test, tf.float32)
# 输入特征和标签值一一对应，把数据集分批次，每个批次 32 组数据
train_db = tf.data.Dataset.from_tensor_slices((x_train, y_train)).batch(32)
test_db = tf.data.Dataset.from_tensor_slices((x_test, y_test)).batch(32)
```

2) 手写数字数据集预处理

所用模块库如下。

```
import tensorflow as tf
import os
import numpy as np
from matplotlib import pyplot as plt
```

导入数据并对其进行归一化处理。

```
# 导入手写数据集。训练集和测试集
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
# 对数据集的像素点进行归一化
x_train, x_test = x_train / 255.0, x_test / 255.0
```

2.3.3 模型建立与训练

1) 鸢尾花模型建立与训练

鸢尾花数据集的四个输入特征的含义分别是花萼长、花萼宽、花瓣长、花瓣宽的相应数值。而最终的目的是将鸢尾花分为三类，分别是狗尾草鸢尾花、杂色鸢尾花、弗吉尼亚鸢尾花，分别用编码 0、1、2、表示。因此输入特征数为 4，输出特征数为 3。

拟用一层全连接网络对鸢尾花进行分类。则输入层需含有 4 个神经元，输出层含有三个神经元，网络结构和权重示例计算过程如下所示。

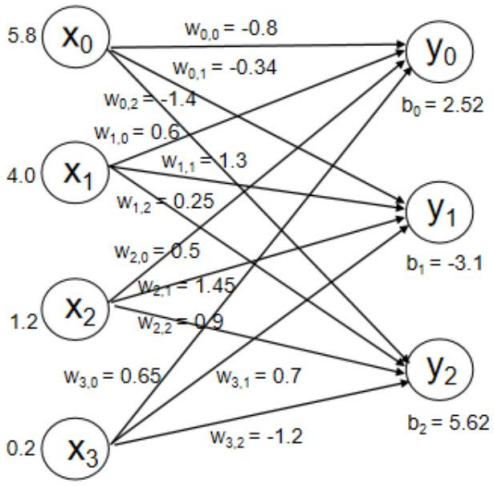


图 2-11 鸢尾花分类模型示例

训练之前要初始化权重参数。上述网络结果中前后神经元的连线数即权重数共为 12 个。利用函数随机生成 12 个初始权重参数如下所示。

```
# 生成神经网络的参数，4 个输入特征故，输出层为 3 个神经元
# 用 tf.Variable()标记参数可训练
w1 = tf.Variable(tf.random.truncated_normal([4, 3], stddev=0.1, seed=1))
b1 = tf.Variable(tf.random.truncated_normal([3], stddev=0.1, seed=1))
```

接下来要进行模型训练，模型训练的基本步骤式循环对每个输入数据特征进行处理，根据每次计算的结果对初始权重参数进行更新，直到所有数据循环遍历完毕。

```
for epoch in range(epoch):  #数据集级别的循环，每个 epoch 循环一次数据集
 for step, (x_train, y_train) in enumerate(train_db):  #batch 级别的循环
     with tf.GradientTape() as tape:  # with 结构记录梯度信息
         # [32 4] @ [4 3] + [1,3] => [32,3]
         y = tf.matmul(x_train, w1) + b1  # 神经网络乘加运算
         y = tf.nn.softmax(y)  # 使输出 y 符合概率分布
         y_ = tf.one_hot(y_train, depth=3)  # 将标签值转换为独热码格式
         loss = tf.reduce_mean(tf.square(y_ - y))  # 采用均方误差损失函数
         loss_all += loss.numpy()  # 将每个 step 计算出的 loss 累加
     # 计算 loss 对各个参数的梯度
     grads = tape.gradient(loss, [w1, b1])
     # 实现梯度更新 w1 = w1 - lr * w1_grad    b = b - lr * b_grad
     w1.assign_sub(lr * grads[0])  # 参数 w1 自更新
     b1.assign_sub(lr * grads[1])  # 参数 b 自更新
```

2) 手写数字识别模型建立与训练

在对模型送入图片数据之前，要将图片的行列数据进行拉直处理，即将所有像素点排成一行送入模型。本文的模型设置了一层全连接网络，共有 128 个神经元，激活函数为 Relu。由于是 10 分类问题，所以输出层共有 10 个神经元，激活函数为 softmax，这个激活函数可以将分类结果以概率的形式输出。

```
# 搭建训练模型
el = tf.keras.models.Sequential([
 tf.keras.layers.Flatten(),  # 拉直层
 # 全连接层：128 个神经元，激活函数 relu
 tf.keras.layers.Dense(128, activation='relu'),
 # 输出层：10 个神经元，激活函数 softmax
 tf.keras.layers.Dense(10, activation='softmax')
```

搭建好模型后，还要编译模型、设置断点续训参数保存路径并送入训练数据进行分批次训练。

```
# 编译模型
model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              metrics=['sparse_categorical_accuracy'])
# 设置断点续训保存参数的路径
checkpoint_save_path = "./checkpoint/mnist.ckpt"
if os.path.exists(checkpoint_save_path + '.index'):
    model.load_weights(checkpoint_save_path)
cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_save_path,
                                                 save_weights_only=True,
                                                 save_best_only=True)
# 给模型送入训练数据并分批次训练
history = model.fit(x_train, y_train, batch_size=32, epochs=5, validation_data=(x_test, y_test), validation_freq=1,
                    callbacks=[cp_callback])
```

训练过程中的损耗和分类准确性随着的变化曲线如下图所示。可以看到，准确性随着训练次数增加，损耗随着训练次数减少。

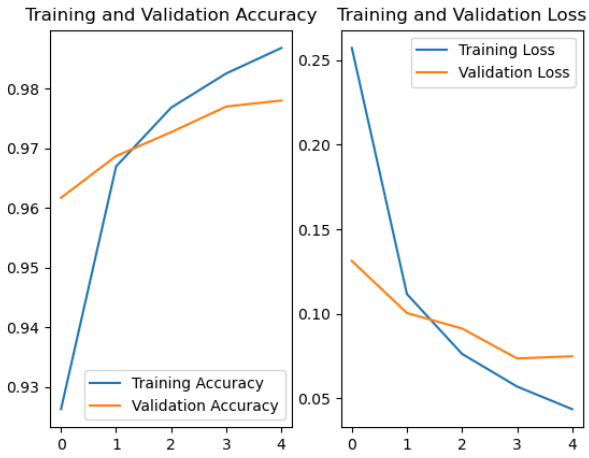


图 2-12 分类准确性(左)和损耗变化(右)图

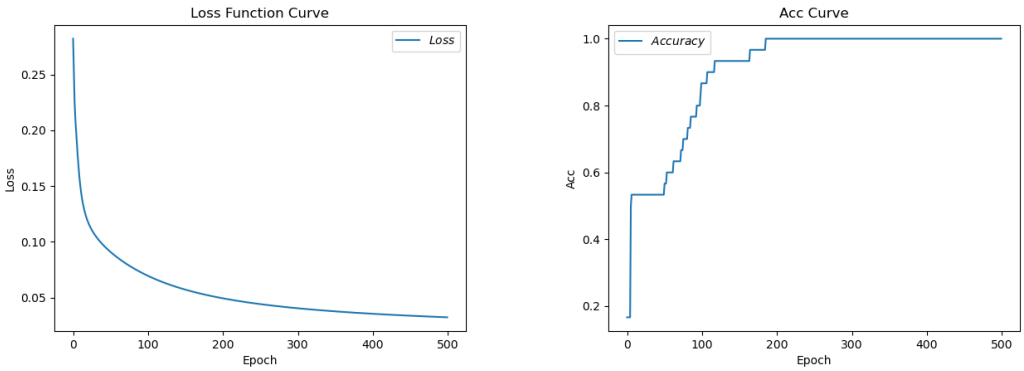
2.3.4 实现分类

1) 鸢尾花分类

训练好的模型的主要功能性参数就是权重参数。当用测试集实现分类时，只需将测试集输入特征与训练好的权重参数进行相应运算就可以得到分类结果。

```
for x_test, y_test in test_db:
    # 使用更新后的参数进行预测
    # 参数结构变化：[30 4] @ [4 3] + [ 1 3] = [30 3]
    y = tf.matmul(x_test, w1) + b1
    y = tf.nn.softmax(y)
    pred = tf.argmax(y, axis=1)  # 返回 y 中最大值的索引，即预测的分类
```

整个训练过程的损耗图和分类准确率曲线如下所示。



训练过程损耗图　　分类准确性曲线

图 2-13 鸢尾花模型损耗图和分类准确性曲线

从上图中可以看出，随着训练次数的递增，损耗不断减小；对测试集进行分类时，分类准确性随着训练轮数递增，最后达到绝对准确。说明模型训练效果已达到预期。

2) 手写数字识别

训练好模型后，就可以进行实际应用了，即输入一张或几张手写数字图片，让模型判断出该图片是数字几。完成这个过程的首要步骤是导入训练好的参数。

```
# 载入之前训练的参数
model.load_weights(model_save_path)
```

在输入待分类的手写图片时，要注意其格式要与训练用的图片格式相匹配。比如大小、灰度化、黑白底等。下边的代码说明了这个过程，其中 img 为用 PIL 图片处理库打开的图片对象。

```
  调整图片形状
img = img.resize((28, 28), Image.ANTIALIAS)
# 灰度化处理
img_arr = np.array(img.convert('L'))
# 图片黑白底变换
img_arr = 255 - img_arr
# 归一化
img_arr = img_arr / 255.0
```

本文用一幅图片来示例分类结果。选用的图片为

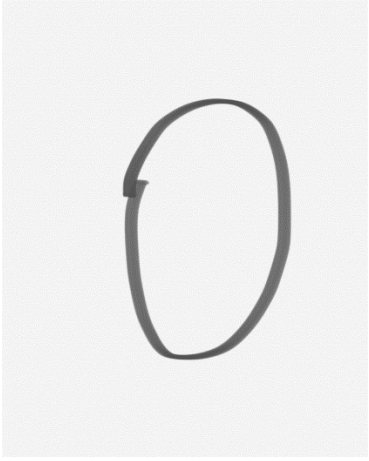


图 2-14 待分类手写数字图片

将上图的而图片送入模型进行分类后得到预测结果如下(命令行)。

```
the path of test picture:
D:\personal_multi_data\PostGraduateFiles\内大研究生课程\研一下\02 人工智能\课程报告及代码\0.png
img_arr: (28, 28)
x_predict: (1, 28, 28)
[0]  # 预测结果
```

可以看到，模型成功地预测到了图片中的数字 0。限于篇幅，本文不再赘述更多的手写数字图片验证。

第三章 预测问题

本章主要讨论的是如何用人工智能对雷达探测到的目标进行运动轨迹预测。对于高频地波雷达而言，部署的位置是海岸或舰载。因此其探测的目标通常是距离海岸几百公里范围内的舰船、海冰、低空飞行器等。合作船只会不断地向雷达发送 AIS 数据，AIS 数据中含有目标的经纬度、速度等信息。本章将利用函数生成一定数量的坐标点，故这些点的图像就是函数的图像子集。之后利用循环神经网络根据一批数据点预测下一点，如果预测合适，预测的点的轨迹应与函数图像一致。

3.1 点迹数据模拟

为了便于分析验证，特选择正弦函数作为模拟数据生成函数。每秒生成 1000 个点，即采样率为 1000Hz。正弦函数的频率 f 设为 10Hz，其数学表达式如下。

$$p(t) = a[\sin(2\pi ft)+1] = 0.5[\sin(200\pi t)+1] \tag{3.1}$$

式中，t 为时间向量，单位为秒；a 为幅值，此处设为 0.5，目的是使生成的坐标值落在 0-1 区间内，这样免去了数据归一化的麻烦。

利用 python 的第三方库 numpy 即可生成坐标点，用 matplotlib 库来查看函数图像。代码和图像如下所示。

```
1.  import numpy as np
2.  import matplotlib.pyplot as plt
3.  #  解决中文显示问题
4.  plt.rcParams['font.sans-serif'] = ['KaiTi'] # 指定默认字体
5.  # 解决保存图像是负号'-'显示为方块的问题
6.  plt.rcParams['axes.unicode_minus'] = False
7.  points = 3000  # 数据点数
8.  f = 10  # 函数频率
9.  t = np.linspace(0, 3, points)  # 时间向量
10. p = 0.5*( np.sin(2*np.pi*f*t)+1 )  # 生成的数据点
11. plt.plot(t, p)
12. plt.xlim([0, 1])
13. plt.title('目标运动轨迹模拟图')
14. plt.xlabel('时间/秒')
15. plt.ylabel('幅度/电压')
16. plt.show()
```

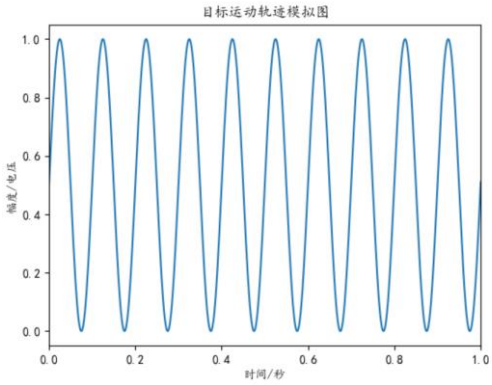


图 3-1 一秒内的模拟数据图

3.2 基本原理

上一章简要介绍了卷积神经网络，卷积神经网络利用卷积核的方式来共享参数，这使得参数量大大降低的同时还可利用位置信息，不过其输入大小是固定的。但是在语言处理、语音识别等方面，一段文档中每句话的长度不一样，且一句话的前后是有关系的，类似的数据还有很多，比如语音数据、翻译的语句等。像这样与先后顺序有关的数据我们称之为序列数据。处理这样的数据就不是卷积神经网络的特长了。

对于序列数据，我们可以使用循环神经网络（Recurrent Natural Network，RNN），它特别适合处理序列数据，RNN 是一种常用的神经网络结构，已经成功应用于自然语言处理（Neuro-Linguistic Programming，NLP）、语音识别、图片标注、机器翻译等众多时序问题中。

本节将介绍循环神经网络一般结构以及循环神经网络的衍生结构—长短期记忆网络（Long Short-Term Memory，LSTM）。

3.2.1 循环神经网络

图 3.2 左侧是循环神经网络的经典结构，从图中我们可以看到输入 x、隐含层、输出层等，这些与传统神经网络类似，不过自循环 W 却是它的一大特色。这个自循环直观理解就是神经元之间还有关联，这是传统神经网络、卷积神经网络所没有的。

其中 U 是输入到隐含层的权重矩阵，W 是状态到隐含层的权重矩阵，s 为状态，v 是隐含层到输出层的权重矩阵。图 3.2 左侧的图比较抽象，我们把它展开

成图 3.2 的右侧部分，这样就更好理解。

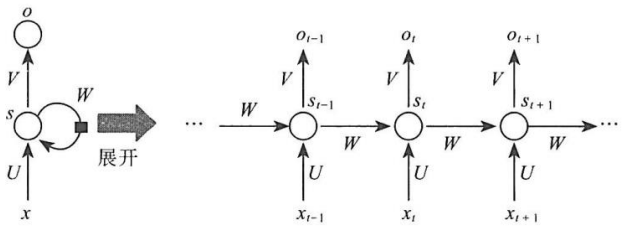


图 3-2　RNN 结构图

右侧部分是一个典型的 Elman 循环神经网络，从图中不难看出，它的共享参数方式是各个时间节点对应的 W、U、V 都是不变的，这个机制就像卷积神经网络的过滤器机制一样，通过这种方法实现参数共享，同时大大降低参数量。

3.2.2 长短时记忆网络(LSTM)

LSTM 最早由 Hochreiter & Schmidhuber（1997）提出，能够有效解决信息的长期依赖(遗忘)，避免梯度消失或爆炸。事实上，LSTM 的设计就是专门用于解决长期依赖问题的。与传统 RNN 相比，它在结构上的独特之处是它精巧地设计了循环体结构。

LSTM 用两个门来控制单元状态 c 的内容：一个是遗忘门（forget gate），它决定了上一时刻的单元状态 c_{t-1} 有多少保留到当前时刻 c_t ,；另一个是输入门（input gate），它决定了当前时刻网络的输入 x_t，有多少保存到单元状态 c_t。

LSTM 用输出门（output gate）来控制单元状态 c_t 有多少输出到 LSTM 的当前输出值 h_t。LSTM 的循环体结构如图 3.3 所示。

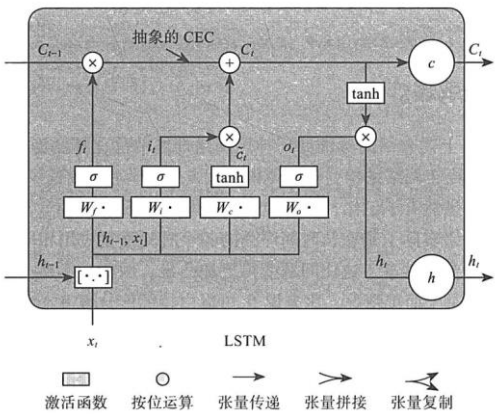


图 3-3　LSTM 循环神经网络的循环体

除了 LSTM 的四个门控制其状态外，LSTM 还有一个固定权值为 1 的自连接，以及一个线性激活函数，因此，其局部偏导数始终为 1。在反向传播阶段，这个所谓的常量误差传输子（Constant Error Carousel，CEC）如图 3.4 所示，它能够在许多时间步中携带误差而不会发生梯度消失或梯度爆炸。

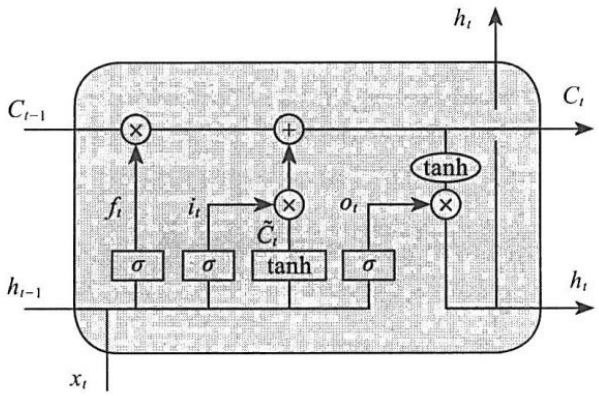


图 3-4　LSTM 状态的传递

以上所述呢欧容只是宏观介绍，具体内容还请查阅相关资料。

简而言之，LSTM 对神经元状态的修改是通过一种叫“门”的结构完成的，门使得信息可以有选择性地通过。LSTM 中门是由一个 sigmoid 函数和一个按位乘积运算元件构成的。sigmoid 函数使得其输出结果在 0 到 1 之间。sigmoid 的输出结果为 0 时，则不允许任何信息通过；sigmoid 为 1 时则允许全部信息通过；sigmoid 的输出位于（0，1）之间时，则允许信息部分通过。

LSTM 就是利用三个上述的门结构(输入门、遗忘门和输出门)来保护和控制神经元状态的改变。

3.3 基于 Tensorflow2.0 的点迹预测实现

在 Tensorflow2.0 的帮助下，可以很简单地搭建 RNN 模型。这一节将简明扼要地列出编程实现的框架图和部分代码及运行结果。

3.3.1 基本步骤

实际编程过程中的步骤与理论知识框架基本一致。

首先，需要对训练和测试的数据进行预处理，包括读取数据、归一化处理、形状调整。

接下来将调整好的数据送入模型输入端进行权重参数训练。训练过程中，还要对数据进行分批次分时段的训练以提高训练效率。

训练完成后，要对权重参数进行保存，目的是下次训练是在这次训练的基础上继续进行的，相当于断点续训。当然，保存的参数也可以直接用于预测，预测过程中，模型直接调用保存的参数即可根据输入数据计算出预测值。

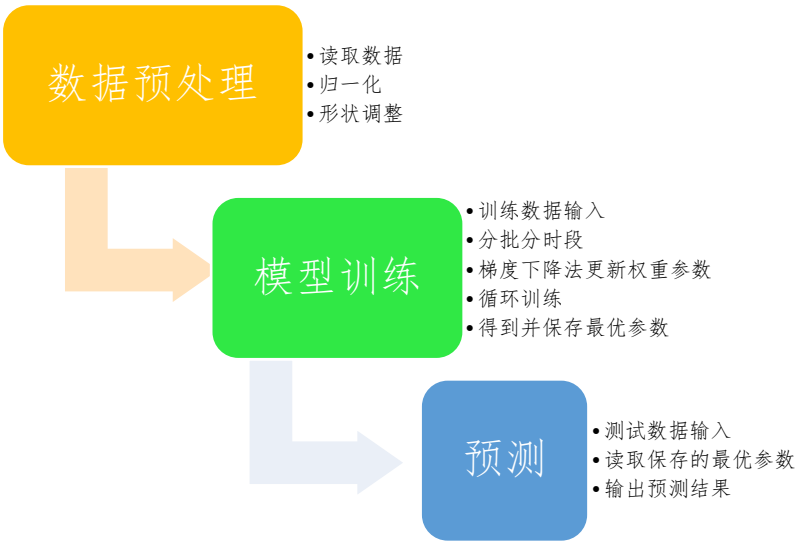


图 3-5 点迹预测基本步骤

3.3.2 数据预处理

本文一共模拟生成 3000 点数据，训练中取前 2700 点用于训练模型，后 300 点用于预测校验。在函数生成时，已将值域范围限制在 0-1 之间，即归一化。

```
# 模拟正弦函数生成数据点，值域映射到 0-1 之间
 0.5*( np.sin(2*np.pi*f*t)+1 )
```

这 2700 点数据要进一步整理成训练数据-标签对。本文取每 60 点数据组成一组，第 61 点数据作为标签，即根据前 60 点数据预测第 61 点的数据值。

```
# 前 2700 点数据作为训练数据
training_set = p[0:points-300]
# 后 300 点数据作为预测数据
test_set = p[points-300:]
x_train = []  # 训练集数据空间初始化
y_train = []  # 训练集标签空间初始化
x_test = []   # 测试集数据空间初始化
y_test = []   # 测试集标签空间初始化
```

```
#  前 60 点预测第 61 点，组成数据--标签对
for i in range(60, len(training_set)):
    x_train.append( training_set[i - 60 : i ])
    y_train.append( training_set[i] )
```

数据标签配对后要将顺序打乱，目的时防止模型的数据记忆性质破坏训练过程，即模型机械地记住数据标签对应关系而没有进行训练。打乱之后还要进行数据类型转换和形状调整。

```
# 对训练集进行打乱
np.random.seed(7)
np.random.shuffle(x_train)
np.random.seed(7)
np.random.shuffle(y_train)
tf.random.set_seed(7)
# 将训练集由 list 格式变为 array 格式
x_train, y_train = np.array(x_train), np.array(y_train)
# 使 x_train 符合 RNN 输入要求：[送入样本数，循环核时间步数，每个时间步输入点数]。
x_train = np.reshape(x_train, (x_train.shape[0], 60, 1))
```

对于测试集的 300 点数据预处理步骤于训练集方式相同，这里不再列出相关代码。

3.3.3 模型建立与训练

模型训练之前要搭建出模型，本文设置的模型有两层循环卷积层，一层全连接层。为了提高效率，RNN 中，随机舍弃 20%的记忆体。

```
# 搭建模型
model = tf.keras.Sequential([
    # 80 个记忆体的 RNN 第一层
    SimpleRNN(80, return_sequences=True),
    # 随机舍弃 0.2 的记忆体
    Dropout(0.2),
    # 100 个记忆体的 RNN 第二层
    SimpleRNN(100),
    Dropout(0.2),
    Dense(1)  # 一层全连接
])
```

模型建立后，要对整体模型进行编译。之后还要设置权重参数保存路径以便

于断点续训和参数提取预测。完成这些步骤后就可以为模型送入训练数据了。

```
#  模型编译，损失函数用均方误差
model.compile(optimizer=tf.keras.optimizers.Adam(0.001),
              loss='mean_squared_error')
#  权重保存路径
checkpoint_save_path = "./checkpoint/track_train.ckpt"
if os.path.exists(checkpoint_save_path + '.index'):
    print('-----load the model-----')
    model.load_weights(checkpoint_save_path)
# 保存训练参数
cp_callback = tf.keras.callbacks.ModelCheckpoint(
               filepath=checkpoint_save_path,
               save_weights_only=True,
               save_best_only=True,
               monitor='val_loss')
# 为模型输入数据并进行分批次的训练
history = model.fit(x_train, y_train, batch_size=64, epochs=50, validation_data=(x_test, y_test), validation_freq=1,  callbacks=[cp_callback])
# 输出模型基本信息
model.summary()
```

3.3.4 实现预测

预测环节比较简单。在已有模型的基础上，只需载入保存的最优权重参数并调用模型的预测函数即可完成预测任务。

```
# 测试集输入模型进行预测
predicted_track = model.predict(x_test)
real_track = y_test
# 画出真实数据和预测数据的对比曲线
plt.plot(real_track, color='blue', label='实际点迹')
plt.plot(predicted_track, color='red', linestyle=':', label='预测点迹', linewidth=3)
plt.title('实际点迹与预测点迹对比图')
plt.xlabel('时间点索引')
plt.ylabel('点迹值')
plt.legend()
plt.show()
```

最终得到的预测曲线与真实曲线的对比图如下所示。从中可以看出，预测的

曲线(红色)与真实曲线(蓝色)的变化趋势是一致的，这说明模型完成了预测功能。

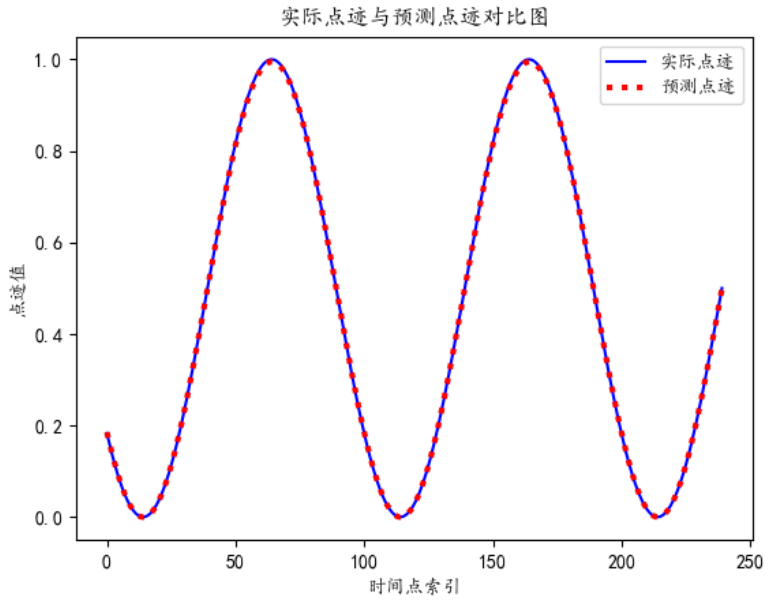


图 3-6　预测点迹与真实点迹对比图

还可以画出训练过程中损耗值变化的曲线，蓝色为训练数据的损耗，黄色线是测试数据的损耗，这个损耗图是经过几次训练后的数据，如下图所示。最初的数据损耗值会比图中所示大一些。

图 3-7　训练过程与测试过程的损耗变化图

第四章 分析与总结

人工智能可以处理的问题有很多，本文只是结合雷达领域问题进行了简要的理论分析和编程实践。通过分析可知，雷达数据处理中的目标分类识别与目标轨迹预测问题非常适合用人工智能方法来处理。

但是，雷达数据处理的关键所在是训练数据数量和质量。因为人工智能模型是用数据来分析数据的一种模式，如果用于训练的数据质量和数量不满足标准，那么模型的输出结果会与真实情况有很大偏差。因此可靠的雷达数据是人工智能模型搭建的前提。

真正的雷达数据不会像本文所述的这样简单，面对更复杂、更繁多的数据，相应的处理模型也会更复杂，这需要不断地根据实际需求进行模型调整。

评价模型处理性能优劣的指标是准确性和实效性。因为雷达大多是实时监测模式，如果模型处理速度太慢将不满足雷达需求。因此，人工智能在雷达数据处理领域的优化方向是处理速度、结果精准度、便捷性等。

以上所述内容都是在理论层面上的分析，实际应用时，还需考虑物理器件的精度、成本、性价比等问题。

由于个人不是人工智能方向，故本文分析的内容都是一些人工智能领域内典型的案例。改进之处是将人工智能于雷达数据处理结合起来，这样在后期如果能遇到类似问题便于用人工智能方法处理。

值得注意的是，本文虽说是人工智能在雷达数据处理中的应用分析，但是实例中使用的数据并不是真实的雷达数据，有的是结构类似的，有的是模拟的点迹。在日后的研究分析中，需结合实际的高频地波雷达数据进行分析。

致 谢

本次论文得以完善需要感谢以下单位或个人。

首先感谢内蒙古大学电子信息工程学院射流的人工智能课程，通过该课程使我了解了人工智能到底是什么。

其次要感谢人工智能课程主讲教师赵国亮老师，在课程讲授期间，我了解到了以前没接触过的 Latex 排版方式。个人安装实测后，该方法确实在排版方面有一定的应用价值，丰富了我的认知面。

再者，疫情期间，我的父母给与我很大的关心和支持，这有助于我完成这门课程的学习任务，在此表示感谢。

学习人工智能课程期间，我参考了很多互联网上的内容。这些资料纷繁杂乱，在此不一一列出出处。但特别感谢 CSDN 网站和大学生慕课网中北京大学推出的曹健老师的人工智能课程，通过学习该课程，我了解到了很多关于 TensorFlow 库的使用方法，本文的很多代码也是参考了该课程的一些实例。在此向该课程表示感谢。

参考文献

[1] Rejfek L , Nguyen T N , Chmelar P , et al. Neural Networks Application for Processing of the Data from the FMICW Radars[J]. Symmetry, 2019, 11(10):1308.

[2] Zhao J , Wu J , Guo X , et al. Prediction of radar sea clutter based on LSTM[J]. Journal of Ambient Intelligence and Humanized Computing, 2019(1).

[3] 杨博辰. 基于 AIS 的船舶轨迹分析的研究与应用[D].电子科技大学,2018.

[4] 罗永豪. 基于 AIS 数据的船舶航行轨迹预测[D].华南理工大学,2017.

[5] 陈志华. 基于海量 AIS 数据的内河船舶航迹预测[D].武汉理工大学,2018.

[6] 任宇翔,赵建森,刘卫,王胜正,韦雨含.基于 AIS 数据和 LSTM 网络的船舶航行动态预测[J].上海海事大学学报,2019,40(03):32-37.

[7] 甄荣,金永兴,胡勤友,施朝健,王胜正.基于 AIS 信息和 BP 神经网络的船舶航行行为预测[J].中国航海,2017,40(02):6-10.

[8] 徐国庆,马建文,吴晨辉,张安西.基于 Attenton-LSTM 神经网络的船舶航行预测[J].舰船科学技术,2019,41(23):177-180.

[9] 陈凯达,朱永生,闫柯,蔡依青,任智军,高大为.基于 LSTM 的船舶航迹预测[J].船海工程,2019,48(06):121-125.

[10] 权波,杨博辰,胡可奇,郭晨萱,李巧勤.基于 LSTM 的船舶航迹预测模型[J].计算机科学,2018,45(S2):126-131.

[11] 胡玉可,夏维,胡笑旋,孙海权,王云辉.基于循环神经网络的船舶航迹预测[J].系统工程与电子技术,2020,42(04):871-877.

[12] 杨金鸿,皇甫立,熊璋,许松,王新远.基于长短记忆网络的舰船航迹预测方法[J].舰船电子工程,2019,39(08):30-33+59.

[13] 谭伟,陆百川,黄美灵.神经网络结合遗传算法用于航迹预测[J].重庆交通大学学报(自然科学版),2010,29(01):147-150.

[14] 程子光,尹康银,叶全国,王前.基于神经网络的雷达实际探测距离研究[J].舰船电子工程,2020,40(02):57-61.

[15] Gao C , Liu H , Zhou S , et al. Maneuvering Target Tracking with Recurrent Neural Networks for Radar Application[C]. International Conference on Radar .2018

附 录

1) GitHub 源文件链接

https://github.com/gl15771340622/IMU_AI_CLASSFILE

2) Gitee 源文件链接

https://gitee.com/gaoliang0622/IMU_AI_CLASSFILE

3) 百度网盘源文件链接

链接：https://pan.baidu.com/s/1lu_ESsaqMxYs76cYK84RFw

提取码：srsh