

**毕业设计（论文）**

题 目 基于机器学习的异常检测算法的应用研究

电信 学院 计算机科学与技术　专业 计算机74 班

学生姓名 黄怀宇

学 号 2173310634

指导教师 李文

设计所在单位 计算机网络与工程研究所

2021 **年** 6 **月**

摘 要

在机器学习蓬勃发展的时代背景和社会背景下，异常检测作为云计算赋能大数据一个不可或缺的组成部分，其涵盖图景十分广阔。笔者围绕异常检测算法的应用场景，梳理细分异常检测算法的类别与特色，最后敲定主打无监督异常检测算法，就其涵盖领域进行深入研究，在分析评测主流算法的前提下，选取最符合应用场景的birch算法，基于用户感知度进行参数调整，从而实现算法的优化。

项目的目标，或者说异常检测算法的应用场景，是过滤微博垃圾评论。

首先，明确需求，即感知用户对垃圾评论的定义。其次，获取微博的垃圾评论，笔者采取Python的request, scrapy, selenium等框架，爬取了40余万条微博评论，当作后续训练的数据集。对于所有文本数据，先使用端到端的LTP分词，分别基于词性标注，命名实体识别，语义角色类型，语义依存关系生成相应的语法树或矩阵。

然后，基于用户感知对生成的数据进行降维，规整化，才可以进入下一层的birch（基于聚类的异常检测算法，使用sklearn框架），将训练好的模型用pickle存储，用Python和MySql可视化，再次进行基于用户感知度的参数调整。

最后，调通后端（node），前端（油猴脚本），算法（Python），数据库（Redis），实现基于机器学习的异常检测算法的微博垃圾评论过滤系统。

**关 键 词**：异常检测；无监督；LTP；Node；request；selenium；scrapy；redis；sklearn；birch；pickle

ABSTRACT

In the context of the booming era and society of machine learning, anomaly detection is an indispensable part of cloud computing to empower big data, and its coverage is very broad. The author focuses on the application scenarios of anomaly detection algorithms, sorts out the categories and characteristics of subdivided anomaly detection algorithms, and finally finalizes the main unsupervised anomaly detection algorithm, conducts in-depth research on its covered areas, and selects the most suitable application based on the analysis and evaluation of mainstream algorithms The birch algorithm of the scene adjusts the parameters based on the user's perception to realize the optimization of the algorithm.

The goal of the project, or the application scenario of the anomaly detection algorithm, is to filter Weibo spam comments.

First, clarify the needs, that is, to perceive the user's definition of spam. Secondly, to obtain spam comments on Weibo, the author uses Python's request, scrapy, selenium and other frameworks to crawl more than 400,000 Weibo comments as a data set for subsequent training. For all text data, first use end-to-end LTP segmentation, and generate corresponding syntax trees or matrices based on part-of-speech tagging, named entity recognition, semantic role types, and semantic dependencies.

Then, based on user perception, the generated data is reduced and regularized, and then it can enter the next layer of birch (clustering-based anomaly detection algorithm, using the sklearn framework), and store the trained model with pickle, using Python and MySql is visualized, and the parameters are adjusted based on user perception again.

Finally, the back-end (node), front-end (oil monkey script), algorithm (Python), and database (Redis) are mobilized to implement a Weibo spam comment filtering system based on machine learning anomaly detection algorithms.

**KEY WORDS**：Anomaly Detection; Non-supervision；LTP；Node；Request；Selenium；Scrapy；Redis；Sklearn；Birch；Pickle

目 录

[1 绪论 1](#_Toc10541211)

[1.1 课题背景及研究意义 1](#_Toc10541212)

[1.1.1 课题背景 1](#_Toc10541213)

[1.1.2 研究意义 1](#_Toc10541214)

[1.2 本文内容介绍及论文框架结构 2](#_Toc10541215)

[1.2.1 本文内容介绍 2](#_Toc10541216)

[1.2.2 论文框架结构 2](#_Toc10541217)

[2 相关算法 5](#_Toc10541218)

[2.1 主流的异常检测算法 5](#_Toc10541219)

[2.2 梳理算法方向 5](#_Toc10541220)

[2.2.1 K-Means算法 5](#_Toc10541221)

[2.2.2 LOF算法 5](#_Toc10541222)

[2.2.3 PCA算法 6](#_Toc10541223)

[2.2.4 Birch算法 6](#_Toc10541224)

[2.2.5 One-Class SVMs算法 7](#_Toc10541225)

[2.3 算法选择 8](#_Toc10541226)

[2.4 本章小结 9](#_Toc10541227)

[3 需求分析与系统结构设计 11](#_Toc10541228)

[3.1 需求分析 11](#_Toc10541229)

[3.2 系统总体结构设计 11](#_Toc10541230)

[3.2.1 数据集制作 11](#_Toc10541231)

[3.2.2 数据处理 11](#_Toc10541232)

[3.2.3 前后端交互 12](#_Toc10541233)

[3.3 系统模块设计 12](#_Toc10541234)

[3.3.1 爬取海量微博评论 13](#_Toc10541235)

[3.3.2 评论文本数据处理 13](#_Toc10541236)

[3.3.3 实时抓取微博评论 13](#_Toc10541237)

[3.3.4 实现后端和调通系统 13](#_Toc10541238)

[3.4 异常检测评价指标设计 14](#_Toc10541239)

[3.5 编码与调试设计 14](#_Toc10541240)

[4 系统开发及其实现 17](#_Toc10541241)

[4.1 引言 17](#_Toc10541242)

[4.2 开发与运行环境 17](#_Toc10541243)

[4.3 项目开源地址与本地部署说明 17](#_Toc10541244)

[4.4 数据集制作 18](#_Toc10541245)

[4.5 系统开发 19](#_Toc10541246)

[4.5.1 系统架构 19](#_Toc10541248)

[4.5.2 文本数据的向量化 20](#_Toc10541249)

[4.5.3 文本向量的聚类和持久化存储 21](#_Toc10541250)

[4.5.4 分类结果的可视化和参数调整 22](#_Toc10541251)

[4.5.5 前后端交互 23](#_Toc10541252)

[4.5.6 后端与算法端的交互 23](#_Toc10541253)

[4.6 项目结果说明和展示 24](#_Toc10541254)

[4.7 本章小结 27](#_Toc10541255)

[5 工作总结与展望 29](#_Toc10541256)

[5.1 工作总结 29](#_Toc10541257)

[5.2 结果展望 29](#_Toc10541258)

[参考文献 31](#_Toc10541259)

# 绪论

## 课题背景及研究意义

### 课题背景

异常检测，指的是识别罕见的数据项，事件或观察到的不同于总体分布的数据项[4]。异常的数据项通常被我们叫作离群点（outlier），具体表现为当前数据项的欧式距离相对于其它数据聚簇的欧氏距离明显偏大。1982年，Johnson[2] 定义了什么是异常：“异常事件作为数据集中的一个观测实例与其他实例相比看上去并不一致。” 2018年，Hawkins[3]定义了什么是离群点（outlier）：“离群点是一个观测值，它与其他观测值的差别如此之大，以至于怀疑它是由不同的机制产生的。”

互联网产业蓬勃发展，随之而来的是数据的指数式膨胀，千人千面，不同人产生的数据具有不同的特征，不同人对于数据评判的标准不尽相同。长尾效应[1]在互联网领域尤为突出，不仅仅局限于商业领域的小众市场，更是渗透于互联网的各个领域。比较成功的案例是YouTube和淘宝中的推荐算法，它们融合了多种聚类技术，比如基于用户的协同过滤算法、召回算法，本质是以平台数据优势赋能推荐算法，基于用户画像精准投放广告或推荐商业产品，底层逻辑是将用户精细化划分成一个个聚簇，对不同聚簇进行差异化处理，这就需要用到基于机器学习的聚类算法。

### 研究意义

异常检测算法的涵盖范围和应用场景十分广阔，从医学影像到文本分类，从推荐算法到流量特质识别。根据应用场景和训练模式的不同，可以根据是否对样本数据进行标注，或是标注的程度，将异常检测分成有监督的异常检测和无监督的异常检测。

互联网在全球的发展方兴未艾，在机器学习赋能大数据云计算的时代，样本的数量往往几十上百万。全监督，甚至是半监督异常检测，在人力成本日益上升的今天，对于中小型企业，应用上述方案，往往手足无措。在实际应用中，对于高维向量空间，人工标注往往会根据人的不同情感和价值取向而产生不同的分类结果。正如前面提到的，无监督异常检测算法由此成为这些应用场景的不二之选。

在现实情况中，异常检测问题往往是没有标签的，训练数据中并未标出哪些是异常点，因此必须使用无监督学习[5]。由此，本项目的研究聚焦于无监督学习。

传统的“分词-聚类”解决方案在数据体量小的时候速度尚可，但不适用于大数据业务场景，其消耗的算力非常恐怖（已经过笔者复现，在“3.2.2数据处理”提到）。笔者在传统的解决方案上进行改良，在“分词-聚类”之间加一层数据解构，在特定场景下，相较于了传统的PCA或LDA降维方法，大大缩短了数据训练的时间成本，节省了算力资源；同时还保留了评论文本特有的语义信息，做到了时间复杂度和异常检测效果双提升。

## 本文内容介绍及论文框架结构

### 本文内容介绍

基于聚类的异常检测算法是机器学习中的一个重要分支，是各种以长尾效应为商业模式的推荐算法的底层逻辑。微博垃圾评论的异常检测在应用场景上和上述商业案例具有很大的相似性，可以融合重组多种聚类算法，基于用户感知度评判评论文本的价值维度，在参数配置模块中调整评论文本评判的颗粒度，实现对微博评论是否是垃圾评论的判断。

由于应用场景的相似性，用户的评论在顶层设计上也可以复用上述商业模式的打法，采用基于机器学习的异常检测算法中的重要分支——聚类算法，进行文本聚类。

本项目在研究了主流异常检测算法的原理，比较了各类异常检测算法的差异后，将birch作为过滤微博垃圾评论系统的核心算法，通过简陋的数据可视化模块展示异常检测后的效果。同时，为了使项目落地，笔者在实现了基于异常检测算法的文本过滤算法的基础上，给算法添加了前后端。为了使用户具有良好的体验，笔者优化了数据的流动方式，采用Redis充当Cache，在评论数据无法命中Cache的情况下，才将评论导流到异常检测算法端，进行比较耗时的“预测”工作。笔者在本文依次介绍了对异常检测算法的研究，系统的设计和开发，各个模块的数据交互和调通，系统的优化和评价，以及对其未来的展望。

### 论文框架结构

本文内容结构安排如下：

第一章 绪论。在调研了工业界和学术圈主流异常检测算法的基础上，确定了研究的课题，以无监督异常检测算法为系统的底层逻辑，介绍了系统的总体架构。最后给出本文的内容，介绍框架结构。

第二章 相关算法。在研究了主流异常检测算法的几个主要方向，梳理各个异常检测算法的特点和应用场景后，说明了选择Birch算法的原因。

第三章 需求分析与系统结构设计。明确了系统的业务需求，设计了系统的总体架构和开发流程，划分了系统需要实现的模块。

第四章 系统开发及其实现。详述系统的模块划分、设计和实现。首先说明项目的环境部署与数据集的制作。然后阐述各个模块的结构和作用，细分数据在各个模块流动情况。最后展示异常检测的的效果和项目实现的结果。

第五章 工作总结与展望。归纳系统实现的工作流程和开发经验，总结项目进程推进过程中获得的成果，指明系统仍然存在的不足与缺憾，展望系统之后可以优化或重构的方向。

# 相关算法

## 主流的异常检测算法

这一数据挖掘的分支被定义后，经历了很长时间的发展，被研究出了多种类型的异常检测算法。

常用的经典的异常检测方法有：线性模型的，包括PCA（Principal Component Analysis，主成分分析）[10]、MCD（Minimum Covariance Determinant，最小协方差行列式）[11]、OCSVM（One-Class Support Vector Machines，一类支持向量机）[12]；基于邻近度的，包括LOF（Local Outlier Factor，局部离群因子检测）[13]、CBLOF（Clustering-Based Local Outlier Factor，基于聚类的局部离群因子）[14]、HBOS（Histogram-based Outlier Score，基于统计直方图的离群点得分）[15]、kNN（k Nearest Neighbors，k临近）[16]；基于概率统计的，如ABOD（Angle-Based Outlier Detection，基于角度的异常值检测）[17]、SOS（Stochastic Outlier Selection，随机离群值选择）[18]；以离群点集合为对象的，包括IForest（Isolation Forest，孤立森林）[19]、LSCP（Locally Selective Combination of Parallel Outlier Ensembles，并行离群集的局部选择性组合）[20]、XGBOD（Extreme Boosting Based Outlier Detection，基于极端增强的异常值检测）[21]；使用了神经网络的，包括使用AutoEncoder（自编码器）[22]的、使用对抗训练的MO\_GAAL（Multiple-Objective Generative Adversarial Active Learning，多目标生成对抗性主动学习）[23]。

判别结果

上述的这些异常检测算法的适用范围各不相同，但大多数都是在神经网络快速发展之前提出的，其中只有少数使用了深度学习的技术。深度学习模型在计算机视觉任务中被广泛使用，因其在特征的提取上有突出的优势，能够将图像中的信息更好地映射到潜在空间。

## 梳理算法方向

### K-Means算法

Goodfellow提出的生成对抗网络（GAN，Generative Adversarial Networks）[24]是近些年比较新的深度学习模型，首次出现在NIPS2014上。GAN最初设计的目的是生成现实世界所不存在的数据，使得人工智能可以模拟人类的创造力或者说想象力。这一模型被用于人工智能画家、数据增强等。

### LOF算法

GAN模型由两部分组成，即生成网络G和判别网络D。生成网络的功能是凭空捏造目标数据，判别网络的功能是判断一个数据是真实存在还是凭空捏造。生成网络G生成数据的来源是一个随机高斯噪声z。G的网络结构与自编码器的解码器网络相似，它接受z的输入，并生成数据，记作G(z)。判别网络D的输入为一个样本x，输出D(x)代表x为真实存在数据的概率，当输入的x为真实数据时，期望D(x)=1；当输入的x为G(z)时，期望D(x)=0。GAN的结构如图2-1所示。

生成网络G

随机分布的噪声向量z

生成图像

真实图像

判别网络D

判别结果

图 2-1 GAN生成对抗网络流程示意图

在训练过程中，生成网络G的目标是尽可能地使生成的G(z)更像真实的数据，以欺骗D网络，使D(G(z))=1。而对于判别网络D来说，它的目标则是尽量地把G(z)和x区分开，以使D(G(z))=0且D(x)=1。这样，G网络和D网络就组成了一组动态的博弈。这就是其名称中对抗的来源。最理想的状态下，G网络可以生成足以欺骗到D的数据G(z)，此时D难以判别G(z)是否为真实数据，即D(G(z))=0.5。此时训练的目的完成，即获得了足以用于凭空捏造数据的模型。

### PCA算法

卷积神经网络（Convolutional Neural Networks）是一类在层间使用了卷积计算且具有深度网络结构的前馈神经网络结构，是深度学习的经典算法之一。CNN可以接受原始图像的输入，避免了传统计算机视觉工作中的图像特征提取等预处理过程，因而被广泛的使用。

### Birch算法

CNN网络结构由输入和输出层及多个隐藏层组成。隐藏层包括卷积层、池化层、线性修正层、全连接层。

卷积层是CNN的核心，每个卷积层由一组卷积核组成，每个卷积核对整个输入进行卷积，计算卷积核与输入的点积，并输出计算后的二维激活图。卷积层对输入进行了更高层次的特征提取。每个卷积层对应单个图像的输入，产生等同于卷积核数的输出特征图，这些特征图将作为下一层的输入。

池化层的目的在于对图像进行下采样。在卷积层之后，特征图的维度很大，其中有大量的信息是没有太多用途或者重复的。此时需要对图像进行下采样，减小图像的尺寸，对特征实现降维。池化层通常的方法是将特征图划分为若干区域，然后对每一个区域取其最大值或平均值，作为这一区域的代表，这样就可以得到维度减小的特征图。通常在若干次卷积之后使用。

线性修正层是卷积层的激活函数，通常在卷积神经网络中使用ReLU激活函数，其对于输入的x的作用是取max(0,x)。

全连接层的作用是对经过了多层卷积和多层池化得到的若干高级特征进行全连接，使其以某种权重组合起来，以形成全局特征，用于最终的输出层。

### One-Class SVMs算法

使用深度学习来进行的计算机视觉任务广泛的使用CNN模型，当需要生成虚构的图像时，就可以将CNN和GAN结合起来。深度卷积生成对抗网络（Deep Convolutional Generative Adversarial Networks）[25]是这方面的一种尝试。DCGAN的原理与GAN是相同的，它只是把GAN中的生产网络G和辨别网络D换成了卷积神经网络。并对经典的卷积神经网络的结构做了一些调整，以提高生成样本的质量和训练的速度，其结构如图2-2。

输入矢量

转置卷积层

Batch-norm层

ReLU激活层

Tanh激活层

输出层

图 2-2 DCGAN深度卷积生成对抗网络的网络结构示意图

这些调整包括：

取消卷积神经网络的所有池化层，生成网络G中使用转置卷积（transposed convolutional layer）进行上采样；判别网络的卷积的滑动步长取卷积核宽度一半向上取整（每卷积计算后图像尺寸减半），在卷积计算的同时进行了特征图像的降维。

在生产网络G和判别网络D中使用batch normalization进行归一化处理。

去掉全连接层，使网络变为全卷积网络。这是因为DCGAN的输出结果不为分类，G的输出结果是图像，D的输出结果是图像的判别评分。

激活函数上，生产网络G使用ReLU作为激活函数，最后一层使用tanh；判别网络D使用LeakyReLU作为激活函数。

## 算法选择

自编码器是一种神经网络结构，其期望结果是能够使输出与输入相同，但是并不是机械地将输入直接作为输出。自编码器网络又两部分组成如图2-3所示，分别为编码器E网络和解码器D网络，这两部分通常是对称的。最初的自编码器只有三层，分别是输入层、隐藏层、输出层。由输入层到隐藏层的部分即为编码器网络；相对的，从隐藏层到输出层的部分即为解码器网络。

编码器E

解码器D

输入层

隐藏层

输出层

图 2-3 自编码器模型结构示意图

欠完备的自编码器指，隐藏层的维度小于输入输出的维度；过完备的自编码器则是相对的，指隐藏层维度大于输入输出层。异常检测中使用的自编码器通常是欠完备的自编码器。对于欠完备的自编码器，从输入层到隐藏层的这一过程实际上实现了对原始数据的降维，神经网络尽可能地在不损失原数据信息的基础上用更小的维度来描述原数据。在计算机视觉任务中，实际上就是一种特征提取的行为。

仅有单个隐藏层的自编码器通常在高维数据上表现不佳。栈式自编码器（Stacked Auto-Encoder，SAE），也称为深度自编码器，拥有更复杂的隐藏层，栈式即说明其是由多层堆叠而成。

使用自编码器的异常检测，其原理是理由了正常数据的“相似性”。在原始数据的高维层度，异常数据和正常数据间的差距难以被分辨。而在降维后的潜在矢量空间，因为维度的降低，其“相似性”降低，从而更容易被区分。若在训练自编码器时仅使用正常数据，那么次自编码器对异常数据的重现效果就会变差。对比原输入图像和输出图像间的差距，正常数据和异常数据就可以较清楚地区分开。

## 本章小结

本章首先介绍了一些现有的异常检测算法，其中使用大部分经典算法是在深度学习快速发展之前提出的。然后介绍了本文实现模型的核心技术，DCGAN深度卷积生成对抗网络。本文模型的各子网络结构参考DCGAN的模型。最后，介绍了自编码器，及其通常在异常检测中的使用形式。本模型的生成网络部分可以看做是使用了卷积和转置卷积的自编码器。

# 需求分析与系统结构设计

## 需求分析

本文目的是实现半监督的图像异常检测，输入为仅包含正常图像的数据集。测试时则同时包括正常图像和异常图像。问题数学描述如下：

有一个大的训练数据集，其中只包括M个正常数据，D = {X1,...,XM}

有一个较小的测试数据集，包括N个正常或异常的数据，D` = {(X`1,y`1),...,( X`N,y`N)}

其中M远大于N。

对于数据集，首先学习D中的数据分布流形，然后去检测D`中的离群点。

神经网络模型f学习正常数据的分布并最小化这些正常数据的异常得分。

对于给定的测试图像数据x`，给出的异常得分若高，则说明此图像可能存在异常。对此的评估标准是设置一个阈值φ，得分大于阈值φ看做异常。

## 系统总体结构设计

图3-1为此网络结构的简单描述。此模型由三部分子网络组成，分别是一个自编码器、一个额外的解码器、一个判别器。

### 数据集制作

第一个子网络在结构上是使用了深度卷积的对称的自编码器网络，起到GAN中的生成网络G的作用。此生成网络利用编码器GE学习输入数据并将特征提取到潜在矢量空间，编码器GE从潜在矢量空间重构输入图像。其内部具体形式为：生成器首先读取输入图像x，其中x∈Rw\*h\*c，然后传入编码器网络GE。编码器网络实质上是一个CNN网络。通过多次使用卷积层，及随后的Batch-norm层归一化和Leaky ReLU激活函数，GE将输入图像x压缩为潜在矢量z，其中z∈Rd。解码器GD部分采用DCGAN的生成器的结构[25]，使用转置卷积层、Batch-norm层归一化和ReLU激活函数，并在末尾加上一个Tanh层。此部分将潜在矢量z扩大铺平来还原x，将输出记作x`。基于以上这些，生产网络G通过编码器网络获得图像的潜在特征矢量，通过解码器获得重建图像。

### 3.2.2数据处理

第二个子网络是一个附加的编码器网络E，它将G网络重建的图像x`又压缩为潜在特征矢量。它在结构上和GE网络是相同的，但是网络的各部分参数和前者是相互独立的。E网络对x`压缩后获得的特征矢量z`与z有相同的维度，以便进行一致性比较。此部分是这个网络结构与其他基于自编码器的异常检测有区别的地方之一。其他的自编码器通常是最小化输入的原始图像与重建图像间的差距以提取正常数据的特征，此网络结构还同时最小化了原始图像与重建图像的潜在特征间的差距。

### 3.2.3前后端交互

第三个子网络是判别器网络D，其功能是对于输入的原始图像x和重构图像x`，能够分辨是真实的原始图像还是G网络的生成图像。此子网络结构使用DCGAN[25]中的判别器网络。

如图3-1所示，模型的整体神经网络结构确定为此。下一步的工作则是设计模型的学习目标。

卷积层

转置卷积层

Batch-norm层

ReLU激活层

Tanh激活层

输入输出层

LeakyReLU激活层

Softmax层

输入图像x

重构图像x`

潜在矢量z

重构图像潜在矢量z`

输入图像x及重构图像x`的特征

D(x)、D(G(x))

图 3-1 本文深度学习神经网络结构示意图

## 系统模块设计

我们期望模型能够做到，当一个异常图像数据x被输入到生成网络G时，即使编码器GE将这个输入图像映射到了潜在矢量空间，获得了其特征矢量z，解码器GD也无法重建这个异常图像。这是因为网络结构在训练时仅使用了正常样本进行建模，其各参数对于异常图像的重建工作是不适合的，G网络输出的重建图像失去了其异常性。并且，G网络的输出x`经过额外的编码器网络E，被映射到潜在矢量空间的特征矢量z`也会失去异常特征的表示，从而使得z和z`之间有较大的差异。当输入的图像在潜在矢量空间的这两个值有明显差异时，输入的x就可以被看作是异常图像。为了完成这一期望，本文将三个损失函数结合起来，建立目标函数，其中的每个损失函数都对单个子网络进行了优化。

### 爬取海量微博评论

与其他使用GAN网络进行异常检测的方法相似，本文也使用了特征匹配损失函数来作为GAN网络的对抗性损失函数。特征匹配损失函数可以降低GAN网络训练时的不稳定性[26]。与原始GAN网络中生成网络G的各参数更新取决于判别网络D对于其生成结果的判别不同，本文的生成网络G的更新基于判别网络D的内部表达。形式上，设函数f()输出为判别网络D的特征提取结束后的一层的输出，x为从输入数据分布pX中提取出的样本。特征匹配计算原始图像的特征表达与生成图像的特征表达之间的二范数。由此，本文的对抗性损失函数如(3-1)，这一部分体现了GAN网络结构中的对抗性训练的思想，之后生成网络D的损失将与此部分产生对抗。对抗性损失函数定义如下。

### 评论文本数据处理

本文的生成网络G，实质上是使用了深度卷积的自编码器。原始图像与重建的生成图像间的差异，期望尽可能的小。原始的自编码器是对这两部分进行二范数计算作为损失函数，但是本文采用的是一范数，即两图像在像素级的差的绝对值。这是因为[27]表明了直接使用一范数的效果好于二范数，重建图像中的不确定结果更少。虽然对抗性损失函数已经起到了对G网络优化的效果，但是重构损失函数(3-2)能够从自编码器的视角，使其重构图像的效果更好的实现。重构损失函数定义为如下。

### 实时抓取微博评论

上述两部分损失函数的优化目的是使生成网络G能够生成不仅仿真到可以通过判别网络D，还能够尽可能重构得与输入图像相同。由于本文还在G之后附加了一个编码器E，还需要对其进行优化，目的是最小化原图像x的潜在特征矢量z和重构的生成图像的潜在特征矢量z`的差距。本文使用二范数来衡量这一差距，附加编码器损失函数定义为如(3-3)。

在以上的学习目标下，生成网络G将学习如何生成正常图像的重构图像，及其在潜在矢量空间的特征的编码。但是这都是以正常数据为样本进行优化的，面对异常数据，G网络的重构效果和E网络的提取特征都会变得很差，而这正是模型期望的。总体的，生成器的损失函数如(3-4)。

### 实现后端和调通系统

因为判别网络D使用的是DCGAN[25]的结构，因此判别器损失函数也使用经典GAN的损失函数，即二进制交叉熵。二进制交叉熵的定义如(3-5)。判别网络D对于输入的真实图像和重构图像均进行评分计算，因为真实图像的期望D(x)=1，重构生成图像的期望D(G(x))=0，所以判别网络D的损失函数也有这两部分组成，定义为(3-6)。这与生成网络G的对抗性损失函数产生了对抗训练，因为此部分试图拉大原图像与重构图像的差距。

## 异常检测评价指标设计

在测试阶段，使用附加编码器的损失函数作为对于给定测试样例的异常评分。样例x`的异常评分sx`定义为(3-7)。

为了对测试集整体进行效果评估，模型对测试集D`中的每一个样本都进行异常得分的计算，得到一组异常得分。然后对这些异常得分做归一化处理(3-8)，使异常得分在表示概率范围的[0,1]之内。

## 计算流程设计

以上部分对神经网络的结构及各部分损失函数进行了设计，下一步是将这些组合起来，使整个神经网络能够进行运作。以一个训练图像和测试图像的计算过程来说明计算流程的设计。

首先进行工作的是生成网络G，对于输入的图像x，先通过编码器网络GE计算出其潜在特征矢量z。然后将潜在特征矢量z输入到结构及功能对称的解码器网络GD中，生成重构图像x`。下一步，将重构图像x`输入到附加的编码器网络E上，得到重构图像x`对应的潜在特征矢量z。第三步，将输入图像x和重构图像x`输入到判别网络D中，D网络输出这两图像的高度抽象的特征及对这两图像的判别结果。至此，计算(3-4)所需要的各损失函数的参数均已计算完成。分别计算三部分损失函数，然后按设置的权重组合起来，以得到生成网络G的优化目标。之后，由神经网络优化器自动进行优化。需要注意的是，虽然计算生成网络G时，判别网络D参与了计算，但优化生成网络G时，不对判别网络D的各参数进行优化。

生成网络G和判别网络D交替进行优化，当G网络的网络参数优化结束后，开始判别网络D的优化，D网络的损失函数(3-6)的计算，需要D网络对输入图像x和重构图像x`进行判别。因为不需要重构图像的潜在特征矢量，所以D网络的优化，附加编码器E不参与计算。与G网络的优化相似，训练D网络虽然需要G网络参与计算，但是优化时不对G网络的各参数进行优化。

为了加速模型的收敛，模型使用分批训练。每次计算一批图像的损失函数后，再进行优化。每一轮训练需将整个训练集分批输入计算，然后进行测试。

测试时，D网络不参与计算。对于测试样本,首先使用编码器GE计算其潜在特征矢量，然后利用此潜在特征矢量在解码器网络GD中生成重构图像,再将重构图像输入到附加编码器E中，计算得到。

# 系统开发及其实现

## 引言

本章参照上文提出的模型结构来实现基于深度学习的异常检测。此章首先介绍模型开发及运行环境，以及对模型工作效果的评测指标。接着将以一次运行的具体过程为例，来说明模型各处的具体计算流程。第一步，介绍数据集的来源和读取方法。第二步，对数据进行预处理，本文将分类数据集中的一类看做异常数据，其他类看做正常数据。此步将训练数据集处理为上文描述的仅含有正常数据的形式。第三步，开始模型训练，将处理好的正常数据作为输入，训练模型，并在每一轮训练结束后做一次测试，输出测试结果来评价当前模型性能。最后，在达到预设的训练轮数后停止，对整个训练过程模型性能的结果分析。

## 开发与运行环境

此模型的硬件开发环境为：

处理器：Intel Xeon E3-1231 v3

内存：8GB 1600MHz DDR3

软件开发环境为：

操作系统：Windows 10专业版 17763.503

编程语言：Python 3.6.5（Anaconda 5.2.0）

机器学习工具：TensorFlow1.11.0 ，Keras2.2.4

编辑工具：Microsoft Visual Studio 2017

运行环境为：

处理器：Intel Core i7-8700

内存：15.6GB

显示适配器：NVIDIA TITAN Xp

操作系统：ubuntu 16.04 LTS

## 项目开源地址与本地部署说明

异常检测常使用AUC值作为模型的评测指标，AUC全称为“The area under receiver operating characteristic curve”指ROC曲线下方的面积。对于二分类问题，预测模型会对每一个样本预测一个得分s，可以选取一个阈值t，让得分的样本预测为正，而其他样本预测为负。这样一来，根据预测结果和实际的标签，样本可以分为四类如表4-1。

表4‑1 单分类混淆矩阵

|  |  |  |  |
| --- | --- | --- | --- |
| 类别 | | 测试类别 | |
| 正常类 | 异常类 |
| 真实类别 | 正常类 | True Positive(TP) | False Negative(FN) |
| 异常类 | False Positive(FP) | True Negative(TN) |

真正例率被定义为(4-1)，假正例率则是(4-2)

由此，真正例率TPR和假正例率FPR都可以表示为阈值t的函数。如(4-3)(4-4)

其中N+(t),N−(t)分别为得分大于t的样本中正负样本数目，N+,N−为为总的正负样本数目

随着阈值t的变化，TPR和FPR在坐标图上形成一条曲线，这条曲线即为ROC曲线。对于一个随机的模型，其对正常数据和异常数据没有分辨能力，那么得分大于t的样本中，正负样本比例和数据集总体的正负样本比例应该是基本一致的，即(4-5)

可知此时有TPR=FPR，对应的ROC曲线是一条从(0, 0)到(1, 1)的直线。

反之，如果模型的性能达到了理想的最好情况，也就是说正负样本的得分可以完全分开，所以的正样本得分都高于负样本，那么此时的ROC曲线就会表现为一条连接了(0, 0),(0, 1),(1, 1)三个点的折线。因为此时不论t值为何，都会出现在TPR=0，或FPR=0中的一种情况。

实际情况中，模型的ROC曲线是一条上凸的曲线，介于随机和理想的ROC曲线之间。而ROC曲线下方的面积，即为AUC值。

## 数据集制作

本文选择的数据集为MNIST手写数字分类数据集[28]。此数据集被广泛用于分类问题和机器学习的教程。我们指定其中某一类为异常，将此类从训练集中除去，而保留其在测试集的位置。对于十个数字，每个数字都可以被视作异常。这样，就获得了十组训练集数据。对测试集的标签进行处理，将指定的异常类标记为1，其他正常类均标记为0。这样，就获得了与之前对应的十组测试集数据。

TensorFlow中有直接下载MNIST数据集的函数，直接调用即可。

from tensorflow.examples.tutorials.mnist import input\_data

…

mnist = input\_data.read\_data\_sets("mnist\_data/", one\_hot=False)

这样得到的mnist对象由训练样本、训练标签、测试样本、测试标签四部分组成。

## 系统开发

### 系统架构

首先，读取MNIST数据集，然后对数据集做预处理，将训练集中的某一个类别去掉，并调整所有图像的尺寸，以适应卷积网络的输入。

下一步，将处理后的训练数据输入模型进行训练，每一代训练结束后，输出当前的各损失函数及测试的AUC值。

当代数达到预设值时结束，并绘制AUC值随代数变化的曲线。

整体流程图如图4-1

数据预处理

输入数据进行训练

开始

是

否

读取数据集，及各超参数

训练代数是否达到预设值

输出当前损失函数及AUC值

输出整个训练过程的AUC值

图 4-1 整体流程图

### 文本数据的向量化

每一代开始训练之前，首先对训练集进行随机排序，以防止某些图像对整体的训练有较大的影响。为了尽快的收敛，我们使用分批训练的方式，每一批学习300个图像的输入，按批更新网络参数。每一个批次训练结束后，将此时的损失函数记录下来，待所有批次训练结束后计算本代损失函数的平均值。然后进行测试，每次不必对测试集中所有数据进行测试，本文设置每次取300个测试数据进行测试。测试后利用输出的异常评分集做AUC值计算并输出。训练时交替训练生成网络G和判别网络D，原始GAN模型是在多次训练生成网络G后才会训练一次判别网络D。因此，为了防止判别网络D收敛太快导致无法对抗训练，模型会在判别器损失函数过低时随机重置D网络的参数，以使对抗能够进行。

模型训练过程的流程图如图4-2所示。

输出尺寸为32\*32的单通道图像集

对输入的数据集进行随机排序

将数据集按批进行训练，每一批训练300张图像

计算此批次的损失函数，并做统计

训练集中的所有图像是否已全部训练

测试此代训练后模型的AUC值，并输出AUC值及损失函数的平均值。

训练生成网络G

训练判别网络D

图 4-2 训练模型流程图

### 文本向量的聚类和持久化存储

由上一章的损失函数定义(3-4)的各部分来源可知，模型训练G网络时，三个子网络均需要进行计算。对于单个的输入图像x，先将其送入编码器网络GE，经过深度卷积后获得其潜在特征矢量z。然后将矢量z送入解码器网络GD中，重建为与x有相同尺寸的生成图像x`。再将生成图像x`输入到附加编码器网络E中，得到其潜在特征矢量z`。然后将重建的生成图像x`和输入图像x均输入生成网络D中，得到对抗性损失函数。最后将各部分损失函数组合起来，作为G网络的损失函数进行优化。

对于单个的输入图像x，进行损失函数的计算

通过编码器网络GE得到潜在矢量z

通过解码器网络GD得到生成的重构图像x`

通过附加编码器网络E得到重构图像的潜在矢量z`

将输入图像x和生成图像x`输入判别网络D，得到判别结果D(x)和D(G(x))

使用这个过程得到的各数值，计算生成网络G的损失函数，并对G网络进行优化

图 4-3 生成网络G的训练流程

### 分类结果的可视化和参数调整

这两部分网络的输入均为32\*32的单色图像。首先是对输入图像进行卷积，本模型默认使用64个4\*4的卷积核进行卷积计算，步长为卷积核宽度的一半，即strides=2。为了使图像尺寸在卷积后还为2的整数次方，padding选择使用“same”，这样在卷积后获得的初层特征图的尺寸为16\*16；若使用“valid”，则会使下一次的卷积的边界数据产生浪费。第一个卷积层的输出尺寸为16\*16\*64。然后是一个LeakyReLU激活层，修正线性单元是神经网络中最常用的激活函数，因为其函数本身和导数均很简单，不包括复杂的数学运算，可以使计算速度提高。

然后进行两次卷积计算，因为其实现的函数相同，本文写在一个循环中。对于每一次循环，先是进行卷积计算，卷积核数量是上一次数量的2倍，输出尺寸是输入的一半。然后接下来是一个batch-norm层，batch normalization通过将输入的内容进行批量归一化，减少输入内部协变量加速神经网络的训练，以使其尽快收敛。之后再接一个LeakyReLU激活层。

最后，当输入的结构为4\*4\*256时，进行最后一层卷积，此次卷积padding设为“valid”，因为已经是最后一层了，卷积核数本文设为100，即潜在矢量空间的维度数。

图像的尺寸和通道数变化如图4-4。

32

32

16

16

8

8

4

4

64

128

256

100

图 4-4 编码器网络各层输入输出数据形状变化过程

### 前后端交互

解码器的大体过程与编码器相反，最初的输入为编码器的输出，即100维的潜在矢量。通过转置卷积，一步步将潜在矢量重构为与原输入图像尺寸相同的图像。第一个转置卷积层的输出尺寸为4\*4\*64，64为转置卷积的卷积核数，卷积核尺寸选用4\*4。Padding参数选用“valid”，因为转置卷积的第一层使用“same”参数会导致计算没有意义。卷积之后是一个批量归一化层，其目的是加速训练的收敛速度。激活函数与编码器不同，选用ReLU激活函数。

之后是这一过程的循环，每一次转置卷积计算，卷积核数为上一次的一半，步长为2，padding参数选用“same”。这样，每次卷积的输出结果尺寸会加倍，通道数会减半。循环的停止条件为生成图像尺寸达到原始图像的一半。此后，进行最后一层转置卷积，使用的激活函数为tanh。输出的格式与编码器的输入的格式完全相同，即尺寸为32\*32的单通道图像。

图像的尺寸和通道数变化如图4-5。

32

32

16

16

16

8

8

32

4

4

64

100

图 4-5 解码器网络各层输入输出数据形状变化过程

### 后端与算法端的交互

判别器实现上与编码器基本一致，主要区别在输出上。编码器的最后一层是将4\*4的特征图压缩为100维的特征图像。判别器则是直接将4\*4\*256的特征图作为输出，用于计算对抗性损失函数。然后将特征图用softmax全连接到一个一维的输出单元，将这个值作为判别网络D的判别结果。用于计算判别器损失函数。

## 项目结果说明和展示

此模型训练时使用了Adam优化器，其参数情况如下。

tf.train.AdamOptimizer( learning\_rate = 2e-3, beta1 = 0.5, beta2 = 0.999)

为了验证模型对于将各个数字设为异常时的判别效果，即模型是否具有普适性，本文利用MNIST数据集，产生了十组训练集和测试集。各组分别将一个数字作为异常，从训练集中剔除此数字的图像，将测试集中此数字对应的图像的标签设为1，表示异常，其他数字的标签则谁0，表示正常。对于十个数字各进行了多次试验，每个数字的最好结果如图4-6所示

图 4-6 MNIST数据集中的十个数字分别作为异常时最好的AUC值

当训练的AUC值不再有太大变化时，停止训练，测试目前的重构效果。可见，重构图像4-7（b）与原图4-7（a）极其相似。



(a)部分输入图像



(b)对应的重构图像

图 4-7 停止训练后，部分输入图像及其重构图像

实验时还发现，即使损失函数在随训练代数逐渐下降，但表示模型性能的AUC值的变化并不是一直在向更好的方向前进。如图4-8中有些情况下，前几代的AUC值反而是最好的。

图 4-8 选取数字0作为异常，其AUC值及损失函数与训练代数的关系

出现这个倾向，说明模型可能出现了过拟合现象，训练代数的增加以及无法对模型性能产生较好的优化。

最差的情况下，原本能够正常生成重构图像的网络，在某一代后突然无法生成重构图像，并且损失函数变的很大，AUC值则掉到0.5左右。如图4-9，在以数字2为异常时，出现了这种情况，这引起了实验的注意。经过思考，其原因与生成网络的三部分损失函数的权重设置中的重构损失函数的权重有关。因此，考虑调整重构损失函数的权重，测试模型性能的变化。

图 4-9 选取数字2作为异常，其AUC值及损失函数与训练代数的关系

在其他两部分损失函数不变的前提下，对重构损失函数进行调整，记录此时模型最好的AUC值的变化情况。如图4-10，最初的重构损失函数权重为1时，其最好AUC值为0.7604，且如图4-9，在训练的第三代出现了模型性能突降。在调整了重构损失函数为50后，其最好AUC值为0.9217，图4-6的结果记录的是调整后的结果。

图 4-10 重构损失函数权重对整体AUC值的影响

综上结果，本文的模型是可行的。对于MNIST数据集，其平均AUC值达到了0.7798。和表其他异常检测算法相比，基本接近其平均水平，并比部分算法性能要好。虽然不是最好的效果，但是证明了使用深度学习的异常检测算法是可行的。

表 4-2 与其他异常检测算法在MNIST数据集上的性能的对比

|  |  |  |  |
| --- | --- | --- | --- |
| 算法模型 | AUC值 | 算法模型 | AUC值 |
| ABOD | 0.7815 | MCD | 0.8666 |
| CBLOF | 0.8009 | OCSVM | 0.8529 |
| HBOS | 0.5742 | PCA | 0.8527 |
| 0IForest | 0.8159 | 平均性能 | 0.7898 |
| KNN | 0.8481 | 本文模型 | 0.7798 |
| LOF | 0.7161 |  |  |

## 本章小结

本章重点描述了此基于深度学习的异常检测模型的实现过程。首先介绍模型的开发与运行使用的软硬件环境，模型性能的评测指标，验证实验使用的数据集，模型的整体计算流程。接下来，针对其中的神经网络细节实现过程进行描述，同时给出了部分子网络的训练流程。模型搭建描述结束后，介绍在MNIST数据集上的运行结果，并对实验过程中出现的情况进行分析。最后将模型的结果与经典异常检测算法作对比。

# 工作总结与展望

## 工作总结

生成对抗网络[24]自2014年提出以来，被广泛地使用在各种深度学习算法之中。本文针对图像的异常检测问题，将深度卷积生成对抗网络与自编码器结合，实现了一个基于深度学习的半监督异常检测模型。并用此模型在MNIST数据集上进行了实验，确定了在MNIST数据集上较适合的超参数。实验结果证明，基于深度学习的异常检测算法与传统异常检测算法相比，有同等的性能效果，并且比部分传统异常检测算法性能要好。

本文的设计中，我的工作包括以下几部分。

(1)学习并了解包括生成对抗网络、卷积神经网络、自编码器等深度学习相关模型，及其发展历史背景、内部的数学原理、网络具体结构和设计方法。

(2)阅读并理解多篇与异常检测相关的论文，特别是使用深度学习的，并完成了对应的外文翻译工作。

(3)完成了模型实现，在参考现有的基于深度学习的异常检测算法的基础上，使用Python编程实现了能够有效工作的半监督异常检测模型。模型由生成网络、附加编码器网络、判别网络三部分子网络组成，在各子网络中通过使用TensorFlow和Keras深度学习工具搭建深度卷积网络和深度反卷积网络。

(4)用实现的模型在MNIST数据集上进行实验。在将各数字分别定义为异常时，模型均能够进行正常工作，即证明了模型的可行性。在发现重构图像过程出现问题后，寻找故障原因，调整损失函数的权重，解决最初设置的权重会出现的无法学习问题。最后，将结果与经典的异常检测算法在MNIST数据集上的效果做对比，本文模型的平均AUC值达到0.7798，达到经典算法的平均水平。

## 结果展望

(1)本文的学习方式是半监督学习，在实际的一些异常检测需求下，需要无监督学习的算法。目前还没有很好的使用深度学习的无监督异常检测算法。是否能够在现有模型的基础上尝试无监督的算法。本文的模型的性能并不突出，没有体现出使用深度学习较经典算法的优越性。

(2)本文模型检测的异常数据是图像异常数据。而对于金融资金异常，网络入侵异常、文本敏感词异常等情况不能直接使用。并且，即使调整了输入输出结构以适应其他异常检测问题，其准确度也无法得到保障。需要进一步实验来测试在其他类型数据上的效果。

(3)MNIST数据集作为一个较简单的深度学习数据集，虽然被广泛地使用在各种模型性能的评测中，但是和真实的异常检测需求难度上有差距。在MNIST数据集上的表现并不能充分表明能够在实际的问题上能够起到很好的作用。

(4)超参数的调整是通过实验获得的结果，没有数学理论能够充分说明怎样的模型可以得到最优结果，如果能够将模型的优化目标进行数学证明，则可能能解决这个问题。

# 参考文献

1. https://zh.wikipedia.org/wiki/%E9%95%BF%E5%B0%BE
2. Johnson R A, Wichern D W. Applied Multivariate Statistical Analysis, 6/E[J]. Technometrics, 2005, 47(4):517-517.
3. Hawkins D M. Identification of Outliers[J]. Biometrics, 2018, 37(4):860.
4. Y. Zhao, Z. Nasrullah, and Z. Li, “Pyod: A python toolbox for scalable outlier detection,” Journal of Machine Learning Research, vol. 20, no. 96, pp. 1–7, 2019.
5. 微调, 2019. https://www.zhihu.com/question/280696035, Last ac-cessed on 2019-05-04.
6. Hayes M A, Capretz M A M. Contextual Anomaly Detection in Big Sensor Data[C]// IEEE International Congress on Big Data. 2014.
7. Li X , Xue Y , Chen Y , et al. Context-Aware Anomaly Detection for Electronic Medical Record Systems[J]. 2011.
8. Zhang X , Wen G , Dai W . A Tensor Decomposition-Based Anomaly Detection Algorithm for Hyperspectral Image[J]. IEEE Transactions on Geoscience and Remote Sensing, 2016, 54(10):1-20.
9. Shyu M L, Chen S C, Sarinnapakorn K, et al. Principal Component-based Anomaly Detection Scheme[M]// Foundations and Novel Approaches in Data Mining. 2005:311-329.
10. Rousseeuw P J , Driessen K V . A Fast Algorithm for the Minimum Covariance Determinant Estimator[J]. Technometrics, 1999, 41(3):212-223.
11. Sch?Lkopf B , Platt J C , Shawe-Taylor J , et al. Estimating the Support of a High-Dimensional Distribution[J]. Neural Computation, 2001, 13(7):1443-1471.
12. Guan H , Li Q , Yan Z , et al. SLOF: Identify Density-Based Local Outliers in Big Data[C]// Web Information System & Application Conference. IEEE Computer Society, 2015.
13. He Z, Xu X, Deng S. Discovering cluster-based local outliers[J]. Pattern Recognition Letters, 2003, 24(9):1641-1650.
14. Goldstein M, Dengel A. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm[J]. KI-2012: Poster and Demo Track, 2012: 59-63.
15. Angiulli F, Pizzuti C. Fast outlier detection in high dimensional spaces[C]//European Conference on Principles of Data Mining and Knowledge Discovery. Springer, Berlin, Heidelberg, 2002: 15-27.
16. Kriegel H P, Zimek A. Angle-based outlier detection in high-dimensional data[C]//Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008: 444-452.
17. Janssens J H M, Huszár F, Postma E O, et al. Stochastic outlier selection[J]. Stochastic Outlier Selection, 2012.
18. Liu F T, Ting K M, Zhou Z H. Isolation-based anomaly detection[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2012, 6(1): 3.
19. Ramakrishnan J, Shaabani E, Li C, et al. Anomaly Detection for an E-commerce Pricing System[J]. arXiv preprint arXiv:1902.09566, 2019.
20. Zhao Y, Hryniewicki M K. XGBOD: improving supervised outlier detection with unsupervised representation learning[C]//2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018: 1-8.
21. Aggarwal C C. Outlier analysis[C]//Data mining. Springer, Cham, 2015: 237-263.
22. Liu Y, Li Z, Zhou C, et al. Generative adversarial active learning for unsupervised outlier detection[J]. IEEE Transactions on Knowledge and Data Engineering, 2019.
23. Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets. In Advances in neural information processing systems[J]. 2014.
24. Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks[J]. arXiv preprint arXiv:1511.06434, 2015.
25. Salimans T, Goodfellow I, Zaremba W, et al. Improved techniques for training gans[C]//Advances in neural information processing systems. 2016: 2234-2242.
26. Isola P, Zhu J Y, Zhou T, et al. Image-to-image translation with conditional adversarial networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 1125-1134.
27. LeCun Y, Cortes C, Burges C J. MNIST handwritten digit database[J]. AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist, 2010, 2: 18.