# 学士学位论文

## 高维大规模不平衡数据集 SMOTE 重采样算法


姓　　名：姜 昊 茗

院　　系：少年班学院

学　　号：PB13000422

导　　师：唐珂 教授


完成时间：二〇一七年五月

University of Science and Technology of China

A dissertation for bachelor's degree

# Innovative SMOTE for High Dimensional and Large Scaled Imbalanced Data Set


| | |
|---|---|
| Author : | Haoming Jiang |
| Department : | The School of Gifted Young |
| Student ID : | PB13000422 |
| Supervisor : | Ke Tang |
| Finished Time : | May, 2017 |

# 致　谢

白驹过隙时光飞逝，在中国科技大学本科的四年中，我学习到了很多，收获了很多。而在我这几年的学习生活中得到了很多老师和同学们的帮助。在这个毕业的时刻，我想借此机会表达我对他们的真诚的谢意。

首先我要感谢指导我毕业论文的唐珂老师和王硕老师，以及要感谢高屋建瓴地指导过我的姚新老师。早在演化计算的一门选修课中就结识了唐珂老师，这门课不仅让我学习了很多机器学习的算法，更让我感受到了这个领域的魅力。我于 2016 年暑假去了姚新老师在伯明翰大学的研究组里面，本论文就是基于当时的一些想法而成的。非常感谢伯明翰的王硕老师和姚新老师在期间给予我的关心和指导。

我还要感谢刘利刚老师，从他生动的计算机图形学的课堂上，我发现了数学有趣的一面。他给我学业和科研上的指导都让我受益终身，包括刘老师为人为师的态度都十分令我敬佩。他一直是我学习的榜样。

在学业上，我还要特别感谢刘恒昌老师，杨周旺老师的帮助。还要感谢陈洪嘉老师，陈老师的线性代数课程给我之后的学习科研都打下了坚实的基础。管理学院的韦来生老师，张曙光老师，毕绣春老师都给予了我很多学习和申请上的帮助，特此感谢。

另外感谢班主任郭民生老师多年对我学习生活上的关心。

感谢孙小山师兄，王士玮师兄，陈哲晖师兄，罗圣明师兄的指点和帮助。感谢宣婷婷，徐远哲，林展，王毅，苗震，郭昊翔，陈楚白，赵宇曦以及 223 宿舍全体同学，与你们在一起的这四年非常的开心，与你们的讨论也使我收益良多。

感谢科大给了我一个完满的本科生活。科大自由，学术，奋斗，富有激情的环境使我受益终身。

最后，感谢我家人二十多年来的鼓励和支持，你们永远使我坚强的后盾。

姜昊茗

2017 年 5 月 22 日

# 目 录

# 摘 要

随着数据科学和人工智能等领域的发展，社会科技正在进行一场变革。与此同时数据科学面临越来越多的挑战。本文主要研究其中两个很重要的挑战，一是如何对应处理越来越大的数据量，二是特征维数的增长以及越来越复杂的数据内在结构。本文着重于研究数据科学中一个非常重要的领域————不平衡学习算法。为了处理应对高维大规模不平衡数据集，新的高维大规模不平衡学习必然会是一个很重要的研究方向。本文以 R 语言为平台做了相关的工程性实践和算法方面的理论研究。具体来说工作如下：1. 对相关的算法 ( 不平衡学习算法，流行学习和降维，最近邻算法) 和工作平台 ( R 语言) 做了调研；2. 以 R 语言为平台建立了 R 语言软件包；3. 利用随机投影森林实现了快速的 SMOTE 算法；4. 利用自编码器的结构改进算法处理高维数据；5. 对大规模高维不平衡数据集做了相关的算法实验分析。

**关键词：** 不平衡学习, 流行学习, 降维, SMOTE, R 语言, 高维, 大规模, 机器学习, 数据科学

# ABSTRACT

With the development of data science and artificial intelligence, the technology is undergoing a revolution. At the same time, Data Science is now facing more and more challenges. This paper is trying to solve two of the most important challenges. One is the increasing scale of the data set, the other one is the increasing amount of the features of the data set. Since imbalanced learning is one of the most fundamental problem in data science, it also undergoing the same challenges. This paper aims at solving the challenges in the area of imbalance learning, which will definitely attract increasing attention. Related algorithms analysis and implementation were done on the platform of R, which is one the most widely used platform. Specifically, 1. A comprehensive survey on R platform and related algorithms, including imbalanced learning, manifold learning and dimension reduction and K-Nearest-Neighbors; 2. R packages construction on R platform; 3. Fast SMOTE algorithms via random projection forest; 4. Improvement on dealing with high dimensional data via the structure of auto-encoder; 5. Complementary experiment results.

**关键词：** Imbalanced Laerning, Manifold Learning, Dimension Reduction, SMOTE, R Language, High Dimensionality, Large Scale, Manchine Learning, Data Science

# 第一章　绪论

分类问题是机器学习和统计学里面最基本的问题，然而大多数的分类办法都是没有考虑到数据集的不平衡问题。而实际问题中不平衡数据集又是非常常见的。在不平衡类别的分类问题中，会出现在训练数据集中主要类的数量超过甚至远远超过其它类别的数据。数量多的类被称为多数类 (Majority Class)，其他类被称为少数类 (Minority Class)。如若不对非平衡类别做特殊处理的话，有时会导致严重的错误。一个教科书般的例子是在做癌症诊断的时候，假设群体中仅有百分之一的人患有癌症，倘若一个分类器将所有人都判别为健康这个多数类，判别器的准确率也可以达到百分之九十九。然而这个判别标准并不好，因为患有癌症的那个群体才是我们所更加关心的，这样的模型对判断病人是否患有癌症没有任何帮助。在实际的生产生活中很多时候就如上述例子一样，我们往往关心的是少数类。相比于直接使用准确率估计判别器效率，在不平衡类别数据的情况下，大家更多的是使用诸如 F1 Score，Recall，G-Means 等指标来分析分类器的效果。本文主要分析和改进的 SMOTE(Synthetic Minority Oversampling Technique, 合成少数类过采样技术) 算法 [1]，就是一种从数据层面解决不平衡学习问题的办法。

在处理高位大规模数据方面，近年来也有长足的发展。为了处理大规模的数据，有很多不同的优化计算方法被提了出来。本文中研究的是 SMOTE 算法，他的原理是基于现有数据中少数类数据的分布生成新的合成样本使训练集达到平衡。与之最为相关的一项技术是 KNN (K-Nearest-Neighbors, K 个最近邻算法)。而新提出许多包括 LargeVis[2]，SDNE[3] 在内处理大规模数据的算法中，KNN 也被广泛地应用于减少计算，加速优化等。所以在本工作中也调研参考比较了许多 KNN 算法。

高维的数据虽然含有的信息量很大，但因为维数高，信息有效的比例一般来说比较少，还会存在许多的噪音。在高维空间中数据的分布的极其非线性性也给机器学习算法带来很大的困难。很多时候就需要对数据进行降维。数据的降维就是为了对高纬度的数据找到一种良好的低维表示以便后续计算，这种表示又应当保留大部分原始数据的特征。从由经典统计理论角度出发的 PCA[4], 到后期提出的 t-SNE[5] 等非线性降维的手段都对这个问题给出了很多不同的解决手段。在面对高维非平衡数据集时，一个很自然的想法就是在对高维数据做了降维再进行非平衡学习算法。本文即使用了自编码器的结构 [6]，并针对任务目标提出了一种半监督的自编码器，以此对算法进行了改进。

R 语言作为一个传统的数据分析的平台，以其免费开源，便于使用的特点广泛地被数据科学领域的专业人士使用。为了应对大数据时代带来的新的变化，需要有更多的人投入对其的开发。根据调研发现现有的 SMOTE 算法效率低下，

非常耗时。在当前的大数据环境下，一个更加快速高效的 SMOTE 算法也是 R 语言开源社区的诉求。

概括来说本文针对两个数据集地两个特点：1. 高维，2. 大规模，设计了新的不平衡学习算法。并且基于 R 语言软件包的平台构建了 1. 快速 SMOTE 算法，2. 高维自编码 SMOTE 算法。

## 1.1 不平衡学习

近年来不平衡学习算法得到了越来越多的关注 [7]。实际上这个领域也已经发展过了很长一段时间，早在 2003ICML 会议上就组织过相关的研讨组 [8]。近年来在生产生活中，相关的不平衡学习算法也被广泛应用在故障诊断 [9][10]，医学分析 [11][12]，异常检测 [13] 等领域。

### 1.1.1 算法评估度量

为了应对不平衡学习的情况，首先要给出度量一个算法好坏的依据。从前述的例子中可以看出，通常使用的准确率 (Accuracy) 的度量并不理想。为此我们首先引入混淆矩阵的概念，如图1.1所示。虽然这样包含了所有的信息，但这不是一个单一的度量。不能明确的用来比较分类器的好坏。

| | | 预测类别 | | |
|---|---|---|---|---|
| | | 猫 | 狗 | 兔子 |
| 实际类别 | 猫 | 5 | 3 | 0 |
| | 狗 | 2 | 3 | 1 |
| | 兔子 | 0 | 2 | 11 |

**图 1.1** 混淆矩阵

对于不平衡学习算法的单一指标度量，主要使用的又有两种。一是受试者工作特征曲线 (ROC, Receiver Operating Characteristic) 和对应的曲线下面积 AUC (Area Under Curve)。曲线是根据 TPR(True Positive Rate), FPN(False Positive Rate) 在不同的分类器阈值下绘制出的。但是在多类问题和不同的分类器上，AUC 这样的指标有时后是不容易被扩展的。而且相对来说计算量也比较大。

另外一种单一度量指标是 F-Score 型指标，F-Score 是精确率 (Precision), 召回率 (Recall) 的调和平均数。

$$F_\beta = (1+\beta^2)\frac{precision \times recall}{\beta^2 \times precision + recall}$$

最为常用的就是 F1-Score (即 $\beta = 1$)。在多分类问题中也可以很容易地被推广为 G-Mean 指标，即不同类的召回率的几何平均值。其调和平均值也可以被拿来使用。本文主要以召回率和准确率为主要的评判指标。

### 1.1.2 不平衡学习算法

通常来说有两类策略来解决不平衡数据集：1. 数据层面的解决方案；2. 算法层面的解决方案。具体来说数据层面的解决方案就是通过对训练数据进行采样的操作，使训练集达到类别数量上的平衡。而算法层面的解决方案是通过对已有算法的修改达到可以更好地判别小类地效果。

在数据层面的解决方案中最为平常的是对少数类的随机过采样 (random over sampling)，即随机从少数类样本中重新抽取一部分复制成为新的样本添加进入原有的样本中。与之对应的是对多数类的随机降采样 (random under sampling)，即从多数类样本中，从中抽取一个部分作为多数类的代表。如上两种方案，第一个对分类器并没有贡献任何新的信息，第二个反而失去了很多珍贵的带标签数据。SMOTE 算法 [1] 针对少数类生成一些近似的样本，从而达到在平衡数据集的同时不损失信息，并且使得分类器接收到的少数类不仅是同样样本的复制。具体来说，SMOTE 算法找出少数类中每个点的几个最近邻，通过每个点和周围点的一个带随机权重的平均作为新生成的样本。这是一种非常传统且鲁棒的算法，故本文选取其为主要研究对象。这里在处理高维大规模数据集的时候，有两个问题：1. 构建准确的 KNN 图是非常耗时的，2. 处理高维数据的时候由于噪声和高维空间中距离度量不佳导致得到新数据点质量不佳。这也是本文主要想解决的两个问题。

在算法层面的解决方案中，很多研究者旨在对原有算法进行修改，如 z-SVM[14]，GSVM-RU[15] 等。很多时候这些修改算法仍需复杂的预处理步骤 [7]。另外一种是单类学习算法，如 CCNND[16]。单类学习算法的缺陷在于当算法需要向多数类学习信息时效率不高 [7]。另外一个重要的方法是代价敏感学习 (Cost-sensitive Learning)，如 Near Bayesian SVM[4]，Cost sensitive NN[17] 等。近年来发展得很快的一个方向是集成方法 (Ensemble Method)，如 SMOTEBoost[18]，RUSBoost[19]，AdaBoost.NC[20] 等。但是这类方法的缺陷也比较明显，为了达到更好的判别效果，所需的分类器较多，训练速度十分缓慢。

本文以 SMOTE 算法为核心，研究了如何使得 SMOTE 算法更加有效地应对高位大规模数据集。

## 1.2 KNN 算法

KNN 算法是 SMOTE 算法中最为耗时的一个步骤。而 KNN 算法也被广泛地应用在现在的机器学习算法中 [2]。最为直接的算法是暴力搜索，时间复杂度为 $O(N^2D)$，N 为数据个数，D 为数据维数。在数据量很多时候难以计算。另外一个非常经典的算法是树形算法，如 kd-tree，kd-tree 作为一个十分经典的算法被广为使用。ANN(A Library for Approximate Nearest Neighbor Searching)[21]

就是一个经典的寻找 k 近邻的 C++ 实现的库，其中就包含了 kd-tree 算法。除此之外还有许多被设计出适应各种问题的树形结构的算法，诸如 M-tree, vp-trees, cover trees, MVP Trees, 和 BK-trees 等等。然而这些算法在高维情况下效率会变慢，例如据研究表明 kd-tree 算法在维数高于 20 的时候就会效率变得和直接暴力搜索没有差异 [21]。

为了加速 KNN 图的计算，有许多近似算法被提出。LSH(Locality sensitive hashing) 是利用数据的对位置敏感的哈希值来快速寻找最近邻，一个很有效率的实现是 FALCONN 库 [22]。另外一个是随机投影森林的改进版本。[2] 关于各种 KNN 算法的比较,Erik Bernhardsson 提供了一份标准的基准参照 [23]，如图1.2所示。

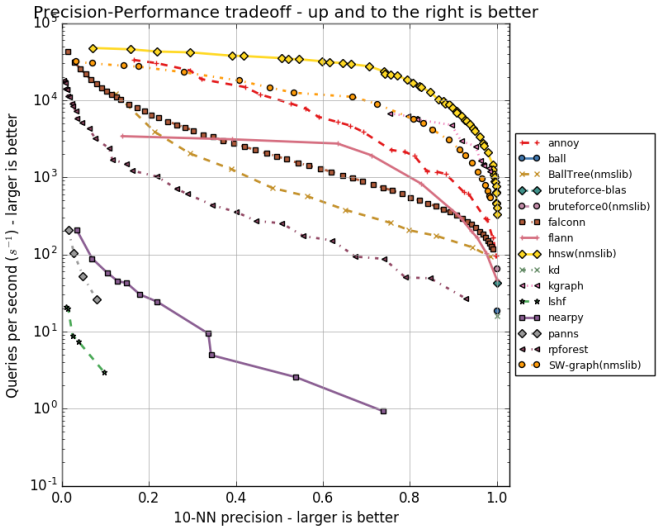


**图 1.2** Benchmarking nearest neighbors of Erik Bernhardsson

根据图1.2，可以得知目前为止 NMSLIB[24] 的表现最为优异。但诸如 NMSLIB 等先进的库本身比较庞杂，难以移植进入 R 语言平台。因此在本工作中主要引入整合了 ANN 库和改进的随机投影森林算法。

## 1.3 降维，可视化以及流形学习

数据的降维是一个被长期研究的领域。数据的可视化也就是将数据降维成二维或者三维得到数据的直观表达。流形学习也与数据的降维密不可分，流形学习的一个主要依据假设即为，高维的数据实际上分布在一个低维的流形上。数据的降维最早考追溯到 PCA 算法。

最早在 2000 年，有学者就提出了效果良好的非线性降维算法，Isomap[25]，与其同时提出的是局部线性嵌入算法，LLE[26]。相关的工作还有 LE(Laplacian eigenmaps)[27]。这是一批最早提出的十分经典的非线性降维的方法。

相比与上述方法一个里程碑式的算法被提出，即 t-SNE[5]，这是 SNE 算法 [28] 的一个变种。其降维效果非常好。近年来随着深度学习的发展诸如

SDNE[3] 的方法被提出用来寻找网络的降维，word2vec 和 doc2vec 被用来寻找自然语言的低维表示。另外一项工作是 LargeVis[2]，相比于 SNE，这个工作运用了很多加速的手段使得模型的训练效率提高很多。

由于许多降维的方法都是对一个已有的数据集做了整体的优化后进行降维操作如 t-SNE,PCA，而非给出一个显式的函数表达 $f : \mathbb{R}^n \to \mathbb{R}^m (n > m)$。然而对于一般的机器学习分类任务，目的是开发一种模型可以对未知的新数据做出处理。显然不可能将新来的数据加入原有的数据重新优化得到低维表示，重新训练分类器去判别新来的数据。其中一种选择方案是不改变训练数据集的低维表示，在优化时仅优化新数据点得到低维表示，但优化的过程由通常十分耗时。

所以综上考虑，在我的模型中采用了自编码器的结构 [6]。而在本工作中针对任务目标对自编码器进行了修改以期望达到更好的结果。

## 1.4 R 语言

R 语言是一种专门为统计和数据分析开发的跨平台的语言。在数据分析领域 R 语言以其开源，简单易用等特点而流行。R 语言中主要功能集中在工具包 (R Package) 内，而因为工具包的存在使用者使用起来也十分简便。而且因为他的开源特性，使得对功能的改良和调整变得十分容易。因此吸引了大量数据分析人才从事相关的开发工作。大量应对不同功能的工具包被开源社区开发出来，并且上传维护。

在 R 语言社区中，也有数个被开发出来带有 SMOTE 算法的工具包。如：DMwR[29]，smotefamily，unbalanced 等。而根据研究如上三个包中的 SMOTE 相关的源代码发现其实现大同小异，如 unbalanced 中的 SMOTE 算法几乎是从 DMwR 中移植而来。这几个 R 工具包中，DMwR 包是最为流行而且最为鲁棒的工具包。本文在比较时也主要以比较 DMwR 的实现为主。

# 第二章　R 语言包

本章主要介绍了如何在 R 语言平台上构建软件包，并且叙述了本论文的工作是如何构建在 R 语言平台上的。

本文针对实际应用中的高维大规模数据做了算法上的改进和分析。而且希望能够提供给 R 语言开源社区社区使用。为了复用代码，本文编写了对应的 R 语言软件包 (Packages)。在 R 语言中，最基础的可以共享复用的结构就是 R 语言包。[30] 一个 R 语言包包含了源代码，数据和对应的文档。截至 2015 年一月，已有超过 6000 个软件出现在 CRAN(Comprehensive R Archive Network) 上，供大家使用。本文选择使用的是 R-Studio 开发环境。

## 2.1　软件包结构

每个 R 语言用户最直接接触的是 R 语言的函数接口和对应的文档。最简单的软件包的开发是由开发者首先编写相关的 R 语言源代码，实现相关的函数功能，并且提供接口使用。期间可以通过编写 Description 文件链接其它的软件包。链接完成后，在 R 语言源代码中可以调用这些被链接的软件包的相关函数功能。在 R 语言源代码中对每个函数编写的注释可以通过 roxygen2 很轻松地导出成用户文档，方便用户查阅。比如软件包 smotefamily，DMwR，unbalanced 就是按照如上流程编写而成。如图2.1所示的是一个简单的 R 软件包结构。

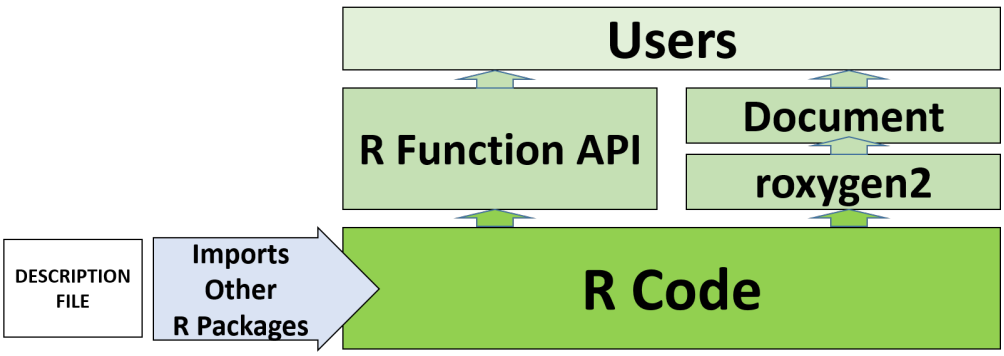


**图 2.1**　简单的 R 语言包结构

然而由于 R 语言本身为了保证易用性的特点，它在其他的方面有所牺牲。缺点最为明显的是两个方面，一是内存管理，二是代码执行的速度和效率。R 语言是起源于 AT&T 贝尔实验室所研发的 S 语言，这是一种解释型语言。解释型语言本身易用性很强，但是执行效率相比于编译型语言要相差一大截，如 C/C++ 等。所以在应对高维大规模数据集的时候，R 语言本身的缺点就成为了性能上的一个短板。通过使用一些 R 语言本身内置的向量计算可以提高代码的

运行效率，如 DMwR 中的 SMOTE 算法的实现。但是执行效率必然相比于编译型语言还是相去较远。

近年来这个问题也越来越受到人们的关注，Rcpp 工具 [31] 的出现无疑是一个对代码改进的突破口。本文通过编写底层 C++ 代码，R 语言仅作为上层语言，实现提供高效的函数接口和文档的功能。通过 Rcpp 可以很方便地实现 C 和 C++ 语言和 R 语言的对接。首先在 C 和 C++ 语言中编写相关的函数，编译出的可执行文件中的函数可以通过 Rcpp 轻松链接进入 R 语言环境，供 R 语言代码调用。Rcpp 同时也可以为 C++ 提供对 R 语言数据结构的操作，和提供 R 语言中的一些函数用法。

在另一方面，面向对象是 C++ 一个非常重要的特性，如何在 R 语言中使用和调用 C++ 中的类型对象 (class) 也是一个问题。近年来也有很多人开发相关的工具，本工作中对 C++ 中的类型的支持就是由 Rcpp Modules 实现的 [32]。

一个底层由 C/C++ 语言实现的 R 语言包结构如图2.2所示。

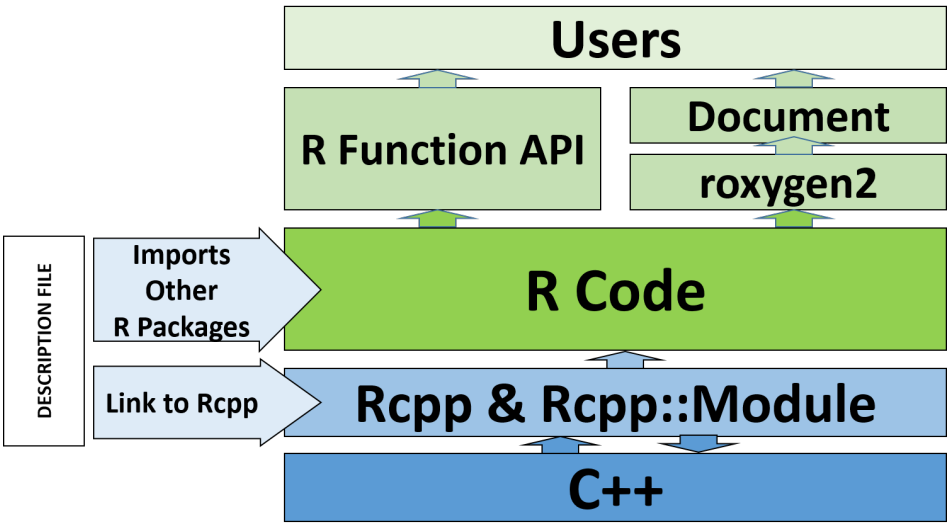


**图 2.2** 底层由 C/C++ 语言实现的 R 语言包结构

## 2.2 代码结构

回到本工作中，编写的软件包中主要实现了两个功能，一个是快速 SMOTE 算法，一个是自编码器 (autoencoder) 实现对高维数据的处理。

为了提供快速 SMOTE 算法相应的 K 近邻计算，我们首先通过整合 ANN 库进入 R 语言环境，以提供 kd-tree 等 K 近邻算法。然而由于高维原因，诸如 kd-tree 的传统 K 近邻计算算法效率非常低下。所以我们通过支持改进的随机投影森林的算法，来支持快速 SMOTE 算法。算法的具体内容参考章节三.

在自编码器的相关工作中涉及到神经网络的计算。所以首先考虑的是相关的深度学习平台，比如 MXNet。MXNet[33] 作为一个优秀的深度学习平台已经

提供了相对较好的 R 语言深度学习的支持。但是在实际工作过程中发现 MXNet 的 R 接口并不是十分灵活。为了尝试新的目标函数和优化办法，本工作仍然是直接在 C++ 中实现自编码器。由于在神经网络计算中涉及到大量的矩阵运算操作，工作中使用了 RcppArmadillo 库 [34] 作为矩阵运算的基础。Armadillo 是一个优秀的线性代数的 C++ 库，它旨在达到速度和易用性的一个平衡。在易用性方面它提供了类似 Matlab 中的 high-level 的语法和 API。而 RcppArmadillo 则为 R 语言提供了额外的支持。具体算法内容可以参考章节四。

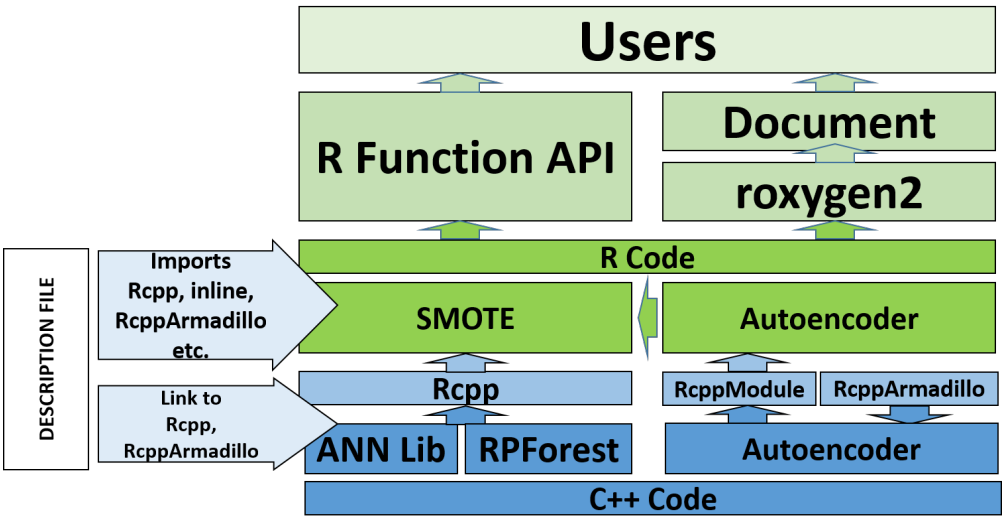


**图 2.3** 工程代码的大致基本结构

图2.3中展现的是工程中大致的代码结构。ANN 库 (ANN Lib)，改进版的随机森林 (RPForest) 都是在底层由 C++ 写成的高效代码提供最近邻的搜索功能。通过 Rcpp 提供向上的接口。自编码器 (autoencoder) 的代码也由底层的 C++ 代码写成，为了高效地运行矩阵计算，使用了 Armadillo 提供的线性代数操作。通过链接 RcppArmadillo 给底层地 C++ 提供对 Armadillo 和 R 数据结构的支持。在通过 RcppModule 为 C++ 的类对象提供向上对 R 语言的接口。最后由 R 语言编写 SMOTE 算法，和提供自编码器的封装。由此提供对用户的高效 R 语言接口和产生相应的文档。

# 第三章　快速 SMOTE 算法

SMOTE 算法是非平衡学习中的一项极其成功和重要的方法。SMOTE 算法旨在对数据集进行重采样来达到少数类和多数类的平衡。他的提出是为了克服单纯的随机过采样和随机降采样带来的一系列问题。单纯的随机降采样会损失大量的珍贵的带标签的数据，单纯的随机过采样又对分类器实质上贡献并不大。以 Naive-Bayesian 分类器为例子，单纯的随机过采样并不会改变条件概率，其效果只是单纯地去调节了先验概率，这和单纯地调节分类器地阈值相差并不大。

具体来说，SMOTE 算法通过对少数类分析生成新的合成样本来扩充新的数据集。SMOTE 算法首先根据重采样的比例 r% 从少数类数据集首先抽取一部分的点。当重采样比例超过百分之一百时，即 $r > 100$ 时，一般为了生成数据的多样性，先将少数类每个点抽取 $n$ 次，$n$ 为满足 $100n \leqslant r < 100(n+1)$ 的整数。余下的重采样部分不重复地从少数类样本中抽取一部分。对于每一个需要被重采样的点生成一个不同于他本身的新的合成样本。具体来说，首先对这样的样本 $x$，找到他的 k 个邻居。从中随机抽取一个邻居 $x_n$。然后新的合成样本由如下公示给出：

$$x_{new} = x + rand(d, [0,1]) \odot (x_n - x), \quad x_n \in kNeighbors(x)$$

其中 $rand(d, [0,1])$ 是 d 维的在 $[0,1]$ 区间上均匀分布的随机向量，d 是特征维数。$\odot$ 表示哈达玛乘积 (Hadamard product)，即对应元素相乘。

当应对大规模高维数据时，算法在性能上出现瓶颈。瓶颈主要由两个原因组成，1. 大规模高维数据会给算法增加很多计算量上的负担。2. 当数据分布在高维空间的时候，由于噪声和极其非线性的分布导致直接应用 smote 算法产生的 k 邻居质量不佳。

本章节针对第一个问题做了算法上的改进。即应用改进后的随机投影森林寻找近似的 K 邻居 [2]。

## 3.1　随机投影森林

首先我们使用欧几里得距离去度量空间中两个点的距离,$\|\mathbf{x}_i - \mathbf{x}_j\|$。给出数据的点集 $\{\mathbf{x}_i\}_{i=1,\dots,N}, \mathbf{x}_i \in \mathbb{R}^d$。构建一个准确的 KNN 图需要时间复杂度为 $O(N^2 d)$ 的计算量。如章节一中所介绍的有很多不同的近似计算方法被提了出来。也有很多很好的开源库实现了相应的功能，但大多数库结构比较庞杂，难以移植进 R 语言环境。为了简洁本文仅引入了改进后的随机投影森林算法。

根据 Erik Bernhardsson 提供的一份标准基准参照 [23]，随机投影森林 [35] 由非常良好的表现。这份基准参照中也研究了不同算法在处理高维数据时的表现。高维的数据往往会给计算带来影响，像是诸如 kd-tree，cover-tree 等树状结

构都会受到高维的影响。相关研究表明 kd-tree 在维数大于 20 的时候效率和直接暴力搜索相差无几。但是随机投影森林却能够和能顺利解决高维数据中寻找 k 最近邻的问题。

具体来说，随机投影森林由多个随机投影树构成。每一棵随机投影树的生成过程与 kd-tree 的生成过程十分相似，都是通过对空间的划分来将相近的点分在同一棵子树或者叶节点上面。开始时所有数据集中在唯一的根节点上，每一次对当前结点做一次划分，并且分别递归生成左子树和右子树。划分的依据是通过判断数据集是否在一个随机选取的超平面的上方。而这个超平面的选取规则如下，首先从当前集合内任取两个不重复的点，找到距离这两个平面相等的超平面即可。选取了超平面以后，就可以利用超平面把当前数据集一分为二，分别递归生成子树。如上过程不断重复直到当前结点所包含的样本个数足够少。构造完成后每一个样本都可以唯一地被归为某一棵子树。随机投影树的构造过程如算法3.1所示。

**input** : $\{\mathbf{x}_i\}_{i=1,\ldots,N}$
**output**: RPTree

MAKETREE(S):
**if** $|S| < MinSize$ **then**
  **return** $Leaf$
**end**
$Rule \leftarrow \text{CHOOSERULE}(S)$
$LeftTree \leftarrow \text{MAKETREE}(\{x \in S : Rule(x) = true\})$
$RightTree \leftarrow \text{MAKETREE}(\{x \in S : Rule(x) = false\})$
**return** $[Rule, LeftTree, RightTree]$

CHOOSERULE(S):
$x, y \leftarrow Sample(S, 2, withoutReplacement)$
$h \leftarrow MiddlePlane(x, y)$
$Rule(x) \leftarrow isAbovePlane(h, x)$
**return** $Rule$

**算法 3.1:** Construct RPTree

当一棵随机投影树被构造出来以后，对于需要寻找 K 最近邻的点可以通过在同属于同一个叶子结点的其他样本中寻找。因为根据如上的划分过程，留在同一个叶子结点的样本自然是相近的。如此可以大大减少计算量。然而如此构造随机投影树的过程和寻找 K 最近邻的过程并不能够保证找到的最近邻是真实的最近邻。所以在实际运用中通常构造多棵随机投影树来提高找到的 K 最近邻的准确度。具体来说在构造的多棵随机投影森林中的所有于目标样本处在同一个叶子节点的样本都可以被视为 K 最近邻的候选样本。从中再选取 K 个与目标节点最为接近的样本为 K 最近邻。

然而为了达到的准确率又要构建大量的随机投影树，这又是对效率上的一个削弱。为了克服上述的困难改进算法，即通过构造少量的随机投影树并

且达到足够好，足够准确的 K 最近邻。在此处引入邻居探索的技术 (Neighbor Exploring Techniques)[36]。邻居探索技术的主要目的是将已有的一个不是很准确的 K 最近邻图，即用少量的随机投影树生成的最近邻，得到一个更加准确的 K 最近邻图。它基于一个很简单的想法就是“自己邻居的邻居很有可能就是我的邻居”，即通过将邻居的邻居作为新的最近邻候选来更新自身的最近邻。通过邻居探索技术更新的 K 最近邻图会比原有的 K 最近邻图更加准确。在实际应用中可以更新多几轮来提升 K 最近邻图的准确率，相关实验研究表明通常只需几个循环即可达到很高的准确度 [2]。在实际实现中可以将每个结点的最近邻维护在一个固定大小的小根堆内进行存储和操作。

相关算法描述在算法3.2中。

**input** : $\{\mathbf{x}_i\}_{i=1,...,N}$, NT (Amount of Tree), K (Amount of Neighbor), $\mathrm{Iter}$ (Iterations in Neighbor Exploring).
**output**: Approximate K-nearest neighbor graph G

1 Build NT RPTrees $\{\mathrm{rptree}_j\}_{j=1,...,NT}$.
2 For each sample $\mathbf{x}_i$ construct a min heap $H_i$ to store K nearest neighbor.
3 Search nearest neighbor for each $\mathbf{x}_i$ in each $\mathrm{rptree}_j$ by updating $H_i$.
4 **for** $i = 1$ **to** $\mathrm{Iter}$ **do**
5 | Update $H_i$ for each sample $\mathbf{x}_i$ by exploring neighbors' neighbors $H_{H_i}$.
6 **end**

**算法 3.2:** KNN Search with RPForest and Neighbor Exploring Techniques

## 3.2 实验结果

快速 SMOTE 算法主要首先利用前述的随机投影森林和邻居探索技术来寻找每个样本 $\mathbf{x}_i$ 的高准确率的近似最近邻。并以此代替原有算法中的准确的最近邻，仍然应用章节开始时给出的公式生成新的少数类合成样本。

本文选取了 MNIST 数据做了相关的实验。MNIST 是一个手写数字的数据库，由纽约大学的 Yann LeCun 教授等人开发维护 [37]。数据库中包含了 60,000 个训练样本，以及 10,000 个测试用样本。每一个样本是一个 28*28 的灰度图，即样本的特征维数是 784 远远超过 kd-tree 建议处理的数据维数 (20 维)。每个像素的特征是一个灰度值从 0 到 255。而实际上这个数据集中每个类的数量大致相同。为了将这个数据集应用于测试不平衡算法，将“0”这一类视为少数类，其他类统一视为多数类进行计算。在训练样本中一共包含约 5,900 个少数类，不平衡比例大约为 1 比 9。为了至少每个少数类样本在从采样的时候能够被重采样一次，所以统一设置重采样率为 200%。K 统一设定为 5。随机森林的大小设为 20，循环迭代两次。表中除了 DMwR 和 unbalanced 包中的算法，其它算法结果均为十次平均所得。

**表 3.1** 运用不同 KNN 算法的 SMOTE 算法运行速度的比较结果

| 算法 | SMOTE(Brute Force) | SMOTE+kdTree | SMOTE+coverTree |
|---|---|---|---|
| 算法耗时 | 41.64 secs | 29.03 secs | 41.29 secs |
| 算法 | SMOTE+RPForest | SMOTE(DMwR) | SMOTE(unbalanced) |
| 算法耗时 | **19.23 secs** | 1.79 hours | 1.83 hours |

可以从表3.1中看出。原有的 R 语言包 (DMwR、unbalanced) 中实现的 SMOTE 算法耗时最长。实验证明底层使用 C++ 代码实现的 SMOTE 算法比 DMwR 中向量化的实现效率提高很多。利用移植进 R 语言环境 ANN 库实现的 kd-tree 算法确实能够起到一定的加速效果。另外一种 coverTree 算法，在应对高维数据是并没有起到明显的加速效果。而且可以看出利用改进的随机投影森林算法得到的快速 SMOTE 算法在速度上相比其它算法有显著的提高。

# 第四章　高维自编码 SMOTE

在之前的工作中，在底层如章节二所示使用 C++ 代码实现相关功能可以克服 R 语言本身处理不了大数据的问题。这是一个工程实现上的改进。另外在章节三中引入了随机投影森林算法提高了算法效率。虽然此处使用了估计的 k 最近邻，并没有使用准确的最近邻，但随机投影森林算法对最近邻估计的准确率很高 [2]，而且在生成合成样本时本身就带有一定的随机性，所以实际生成的新合成样本与原本 SMOTE 算法相差不大。这些工作并没有针对另外一个问题———数据的高纬度给 SMOTE 算法带来的本质上的影响做处理。本章主要针对处理高维数据，提出了高维自编码 SMOTE 的解决方案。

## 4.1　研究动机

在面对高维数据的时候这样重采样生成的样本有时候并不理想。比如在自然语言处理的情感分析中，Kaggle 平台上有一个非平衡的数据集，推特航空航空服务评价情感数据 (https://www.kaggle.com/crowdflower/twitter-airline-sentiment)。这是一个从推特平台上收集得到的大家对所乘坐的航班评价的微博数据，每一条数据被标定为一种情感，积极，中性或者消极。然而由于大家通常更加倾向于在推特平台上吐槽，导致数据集时不平衡的，即消极情感占多数。具体来说 14,485 条数据中，消极情感占 9082 条，中性情感占 3069 条，积极情感占 2334 条。为了方便研究，将积极情感这一类视为少数类，将中性和消极情感在实际试验中发现 SMOTE 算法对这个数据处理的效果并不理想。可以由表4.3中看出。为了研究算法的具体行为，我们研究了运用普通 SMOTE 算法前后训练的 Naive Bayesian 分类器的测试结果的混淆矩阵 (Confusion Matrix)。见表4.1和表4.2，这是多次试验结果中抽取的一次实验结果，与其它实验结果相差不大。可以看出在提升中性类判别结果的同时，降低了积极类的判别效果。

**表 4.1**　Confusion Matrix Before Applying SMOTE

| | | Predicted Labels | | |
|---|---|---|---|---|
| | | negative | neutral | positive |
| Actual Labels | negative | 518 | 40 | 22 |
| | neutral | 77 | 126 | 27 |
| | positive | 38 | 33 | 119 |

因为在试验中使用的词包模型 (Bag of Word Model)，在去除稀疏项以后每一条推特的维数仍然高达 1618 维。这是一个高维的数据。而且每一个样本之间的远近程度使用单纯的欧式度量时不太科学的。因为本身两句话可能仅仅因为一两个词语的不同，语句情感就可能发生完全的转变。而表示同一个语义的两

**表 4.2** Confusion Matrix After Applying SMOTE

| | | Predicted Labels | | |
|---|---|---|---|---|
| | | negative | neutral | positive |
| Actual Labels | negative | 510 | 55 | 23 |
| | neutral | 85 | 150 | 38 |
| | positive | 25 | 20 | 94 |

句话因为用词和表达方式不同反而可能会相去甚远。不仅仅是在自然语言处理中会出现这样的问题，当数据分布在高维空间的时候，由于噪声和极其非线性的分布都会导致直接应用 smote 算法产生的 k 邻居质量不佳。

本文通过构造自编码器模型来解决上述问题。

## 4.2 自编码器

为了处理这类高纬度数据，一个非常简单的想法就是通过降维，找到一个合理的低维表示，再利用低维表示去完成后续如判别分类等工作。在降维，可视化，流行学习领域里有很多相关的工作如 PCA，Isomap，SNE，t-SNE 等。其中很多工作都有非常优秀的降维效果，如 t-SNE，LargeVis 等。但是大部分工作都是将每个点的低维表示作为优化参数，根据整个数据集统一进行优化一个综合的目标函数，同时得到一个低维表示。然而这样的降维表示在我们的工作中并不适用。因为本质上我们还是希望解决诸如分类问题等机器学习问题，希望能够在获得新的样本的时候迅速做出预测等任务。显然是不可能每次将新的样本加入训练集中重新训练。况且对应的分类器也需要重新训练。一种方案是将新的样本加入训练集重新训练，但固定原本训练集的参数，仅仅优化新样本的参数。如此可以免去重新训练分类器的问题，但为了获得新样本的低维表示参数仍然要通过优化，需要大量的计算。

所以我们希望得到一个显式的降维函数表达 $f:\mathbb{R}^n \to \mathbb{R}^m (n > m)$。因此本文引入了自编码器来完成降维的部分。自编码器的编码器部分即为所要求得的显式的降维函数表达，$encoder:\mathbb{R}^n \to \mathbb{R}^m (n > m)$。通过编码器可以快速求得测试集样本的低维表示。

自编码器 (Auto-Encoder) 是一个非监督学习的神经网络，其结构如图4.1所示。神经网络的输入是一个高纬度的样本 $\mathbf{x}_i$。输出的是对输入的估计 $\hat{\mathbf{x}}_i$。中间层是样本抽取出的低维表示 $y_i$ 其结构分为两个部分，一个是编码器 (Encoder)，一个是解码器 (Decoder)。输入样本首先通过编码器编码成低维表示 $\mathbf{y}_i = encoder(\mathbf{x}_i)$, 再通过解码器解码根据低维表示尽量还原成原始输入 $\hat{x}_i = decoder(y_i)$

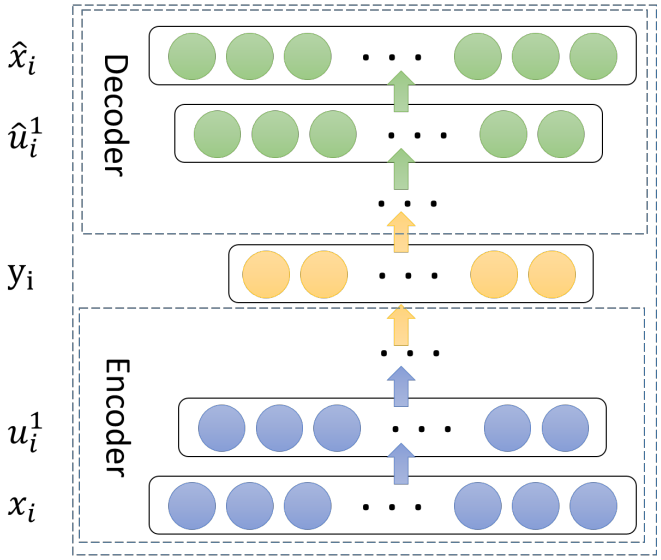


**图 4.1** 自编码器结构

对于编码器部分，它是由数个非线性函数叠加而成，具体来说：

$$\begin{aligned}\mathbf{u}_i^1 &= \sigma(W^1\mathbf{x}_i + \mathbf{b}^1)\\ \mathbf{u}_i^k &= \sigma(W^k\mathbf{u}_i^{k-1} + \mathbf{b}^k), k = 2, ..., K\\ \mathbf{y}_i &= \mathbf{u}_i^K\end{aligned}$$

其中 $\sigma$ 是激活函数，$W^k$ 是第 k 层的权重矩阵，$\mathbf{b}$ 是第 k 层的偏差值。对于激活函数一般来说可以使用，sigmoid 函数 $\sigma(x) = \frac{1}{1+\exp(-x)}$，tanh 函数 $\sigma(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$，relu 函数 $\sigma(x) = max(0, x)$。本文中实验主要使用 sigmoid 函数，但在实际 R 语言包编写中提供了不同的选择。

类似的对于解码器部分，相关计算如下：

$$\begin{aligned}\hat{\mathbf{u}}_i^{K-1} &= \sigma(\hat{W}^{K-1}\mathbf{y}_i + \hat{\mathbf{b}}^{K-1})\\ \hat{\mathbf{u}}_i^k &= \sigma(\hat{W}^k\hat{\mathbf{u}}_i^{k+1} + \hat{\mathbf{b}}^k), k = K-2, ..., 1, 0\\ \hat{\mathbf{x}}_i &= \hat{\mathbf{u}}_i^0\end{aligned}$$

自编码器的优化目标就是最小化解码后的估计值 $\hat{\mathbf{x}}_i$ 和 $\mathbf{x}_i$ 之间的误差。具体来说可以写成如下优化目标函数：

$$\mathcal{L}_{ae} = \sum_{i=1}^{i=N} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2$$

如前人工作所证明的 [38]，虽然最小化上述的重构误差并没有显式地将原有数据间的相似性保留在低维表示上，但是通过自编码器的自重构，它可以自

然地光滑地抓住数据流型，因此它能够自然地将数据间的相似性保留在低维表示上。

通常为了防止过拟合，需要在目标函数中加上一项正则化惩罚项，如下式所述。

$$\mathcal{L}_{reg} = \frac{1}{2}\sum_{k=1}^{i=K}(\|W^k\|_F^2 + \|\hat{W}^{k-1}\|_F^2)$$

## 4.3 半监督自编码器

然而由于此处是为了更好的进行 SMOTE 算法合成新的样本而训练自编码器的，所以数据中是带有标签的。利用这样的标签我们可以针对 SMOTE 算法做进一步的优化。在将数据进行低维编码以后 SMOTE 算法就会在低维流型上进行重采样，而此时的重采样过程仍然需要利用低维空间中的最近邻信息。一个直观的想法就是同样属于一个类的样本能够尽量在低维空间中距离较近，对于不同类别的样本希望低维表示距离较远。

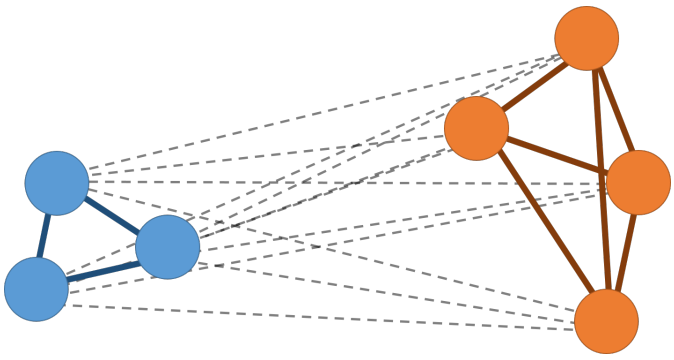

**图 4.2** 全链接图

考虑一个数据之间的全链接图，如图4.2中所示。其中灰色虚线的部分表示的是两个不同类之间的边，深色实线部分表示的是每个类自身连接的边。为了达到同一个类之间的数据之间更加接近，不同类之间数据疏远，我们考虑最大化如下目标函数：

$$\mathcal{O}_{sup} = \prod_{i\neq j, C(i)=C(j)} p(e_{ij} = true) \prod_{i\neq j, C(i)\neq C(j)} p(e_{ij} = false)$$

上述优化目标表示的就是观察到，一个在同一个类内部观测到所有边出现，不同类之间连边都不出现的图的概率密度。其中 $C(i)$ 表示第 i 个样本的类别。给定一对点 $(i,j)$，$p(e_{ij} = true) = f(\|\mathbf{y}_i - \mathbf{y}_j\|)$ 表示第 i 个样本和第 j 个样本之间观察到存在一条边相连的概率密度。$\mathbf{y}_i$ 即为前述的样本低维表示。通常来说可取 $f(x) = \frac{1}{1+x^2}$，或者 $f(x) = \frac{1}{1+\exp(x^2)}$[2]。本工作中此处选取 $f(x) = \frac{1}{1+x^2}$，

这个正好是 t 分布的概率密度函数。t 分布是一个重尾分布，在解决拥挤问题中由突出的表现 [5]。为了计算方便取其负对数为损失函数。

$$\mathcal{L}_{sup} = \sum_{i\neq j, C(i)=C(j)} -\log p(e_{ij} = true) + \sum_{i\neq j, C(i)\neq C(j)} -\log p(e_{ij} = false)$$

通过引入上述监督学习的部分，构成了半监督学习的自编码器。

总体的损失函数定义如下:

$$\mathcal{L}_{mix} = \mathcal{L}_{ae} + \alpha\mathcal{L}_{sup} + \lambda\mathcal{L}_{reg}$$

## 4.4 实验设定和结果

在训练半监督自编码器的过程中，使用了神经网络中常用的随机梯度下降的办法。所以我们首先计算相关的梯度：

$$\begin{aligned}
&\frac{\partial \mathcal{L}_{mix}}{\partial \hat{W}^k} = \frac{\partial \mathcal{L}_{ae}}{\partial \hat{W}^k} + \lambda\frac{\partial \mathcal{L}_{reg}}{\partial \hat{W}^k}, \quad \frac{\partial \mathcal{L}_{mix}}{\partial \hat{\mathbf{b}}^k} = \frac{\partial \mathcal{L}_{ae}}{\partial \hat{\mathbf{b}}^k} + \lambda\frac{\partial \mathcal{L}_{reg}}{\partial \hat{\mathbf{b}}^k}, \quad k = 0, ..., K-1\\
&\frac{\partial \mathcal{L}_{mix}}{\partial \hat{W}^k} = \frac{\partial \mathcal{L}_{ae}}{\partial \hat{W}^k} + \alpha\frac{\partial \mathcal{L}_{sup}}{\partial \hat{W}^k} + \lambda\frac{\partial \mathcal{L}_{reg}}{\partial \hat{W}^k},\\
&\frac{\partial \mathcal{L}_{mix}}{\partial \hat{\mathbf{b}}^k} = \frac{\partial \mathcal{L}_{ae}}{\partial \hat{\mathbf{b}}^k} + \alpha\frac{\partial \mathcal{L}_{sup}}{\partial \hat{\mathbf{b}}^k} + \lambda\frac{\partial \mathcal{L}_{reg}}{\partial \hat{\mathbf{b}}^k}, \quad k = 1, ..., K
\end{aligned}$$

首先看与 $\mathcal{L}_{ae}$ 相关的项目，由于设定数据矩阵 X，为每一行为一个样本，此处所有向量变更为行向量，

$$\frac{\partial \mathcal{L}_{ae}}{\partial \hat{W}^0} = (\hat{U}^1)^T \hat{G}_1, \quad \hat{G}_1 = 2(\hat{X} - X) \odot \sigma'(\hat{X})$$

其中 $\odot$ 为 Hadamard 乘积，并且在实际中为了计算方便 $\sigma'(x)$ 的值可以直接由 $\sigma(x)$ 得出，故可在此简单记述，不详细展开。对于偏差项：

$$\frac{\partial \mathcal{L}_{ae}}{\partial \hat{\mathbf{b}}^0} = \mathbf{1}\hat{G}_1$$

根据倒向传播技巧 (back-propagate) 可得计算下一层参数梯度的倒向传播更新公式：

$$\begin{aligned}
&\frac{\partial \mathcal{L}_{ae}}{\partial \hat{W}^k} = (\hat{U}^{k+1})^T \hat{G}_{k+1},\\
&\frac{\partial \mathcal{L}_{ae}}{\partial \hat{\mathbf{b}}^k} = \mathbf{1}\hat{G}_{k+1}, \quad \hat{G}_{k+1} = \hat{G}_k (\hat{W}^{k-1})^T \odot \sigma'(\hat{U}^k), \quad k = 1, ..., K-1
\end{aligned}$$

关于 $\mathcal{L}_{sup}$ 的项和 $\mathcal{L}_{ae}$ 类似，只需要将 $G_K$ 更换为 $G_K^* = \frac{\partial \mathcal{L}_{sup}}{\partial Y} \odot \sigma'(Y)$ 即可。关于 $\mathcal{L}_{reg}$ 项比较简单，也不在此赘述。

通过如上基于倒向传播技巧得到的梯度，我们可以使用随机梯度下降来最优化模型。算法流程如算法4.1所示。首先需要对参数进行初始化，对于使用

sigmoid 和 tanh 激活函数的参数的初始化可以参考文章 [39]。对于 relu 激活函数的参数的初始化可以参照文章 [40]。本文主要实验结果都以 sigmoid 为激活函数，根据 [39]，初始化采用在区间 $[-b, b], b = \sqrt{6/(\mathrm{Nnode}_k + \mathrm{Nnode}_{k+1})}$ 内的均匀分布初始化。对于偏差项都初始化为 0。由于目标函数比较复杂，非凸，函数很有可能会收敛到较差的局部最优点。所以在初始化之后需要进行一项非常重要的步骤，预训练。本文采用了逐层预训练的方式 [41]。相关研究表明 [42] 通过预训练神经网络，可以让参数达到一个较好的初始值。之后利用随机梯度下降进行微调可以快速的收敛到一个比较优的局部最优点。在网络结构的选取上，MNIST 采用了 $784 \to 100$ 的编码器结构，航空情感数据集采用了 $1618 \to 200 \to 100$。

**input** : $\{\mathbf{x}_i\}_{i=1,...,N}$ and $\{\mathrm{Nnode}_j\}_{i=1,...,K}$ (number of nodes in hidden layers) and $\mathrm{batchSize}$.
**output**: Low Dimensional Representations $\{\mathbf{y}_i\}_{i=1,...,N}$

Initialize the parameters
Pretrain the parameters
**repeat**
    Creat a mini batch data with the size of $\mathrm{batchSize}$
    Feedforward the data though the AutoEncoder
    Use the gradient to back-propagate through the entire network to get updated parameters
**until** *Reach the stopping criteria*;
Obtain the low dimensional representation $\{\mathbf{y}_i\}_{i=1,...,N}$.

**算法 4.1:** Semi-supervised Auto-Encoder

### 4.4.1 自编码器的重构

为了观察自编码器重构的结果，我们通过利用 MNIST 手写数字数据集训练自编码器可以观察到如下重构。如图4.3所示，上排是输入的图片数据，下排是输出的图片数据。说明自编码器的重构效果良好。

### 4.4.2 有监督损失函数

为了查看有监督损失函数是否能够使低维表示达到我们的期望，即使得数据的低维表示同类相近，异类相斥。此处进行了相关的对比实验。如图4.4所示。此处生成了两组在一个两百维的单位方形内均匀分布的点集，一组为 800 个点，一组为 200 个点。利用自编码器降至两维, 即采用了 (200,2,200) 的网络结构。图中为降维的结果。图中红色的是少数类，黑色的是多数类。而左图是加入了有监督损失函数的半监督模型，右图是原始的无监督模型。可以发现加入有监督的损失函数确实可以使得数据在低维空间中同类聚集，异类分散。模型收敛后，半监督模型和无监督模型的对数据重构的均方误差 (MSE, Mean Square

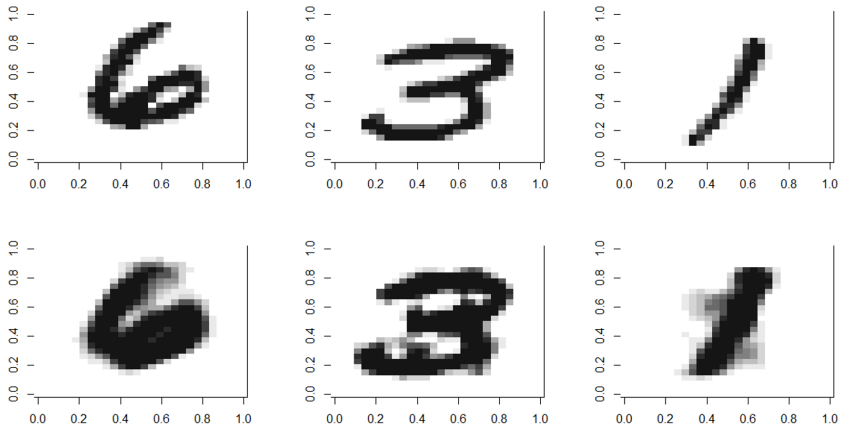

**图 4.3** 对手写数字集的重构

Error)，分别为 14.1469 和 14.1477，相差不大。说明有监督损失函数对重构质量的影响不大。

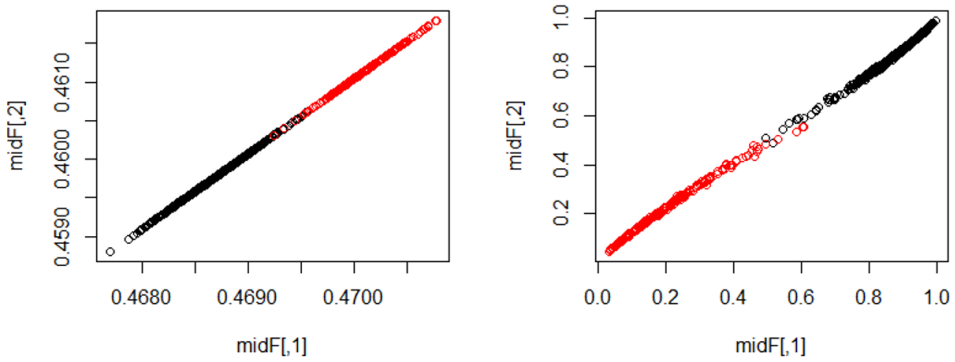


**图 4.4** 监督损失函数对降维的影响，半监督 (左)，无监督 (右)

### 4.4.3 可视化结果

为了验证半监督自编码器产生的低维表示是否可靠，此处我们使用了 t-sne 对两个数据集的低维表示进行了可视化。可视化结果如图4.5所示。

由于 MNIST 本身的样本之间差异较大，特征明显，可以看出不论是在原始数据集，无监督自编码器和半监督自编码器中的可视化效果差异不大。而对于推特情感数据，在原始的和普通的无监督自编码器中产生的特征的可视化效果都比较杂乱。半监督自编码器则能很好地将同类数据聚集在一起。同时对于信息的还原程度相差不大，对于情感数据无监督和半监督自编码器的均方误差 MSE 分别为 0.267 和 0.296。

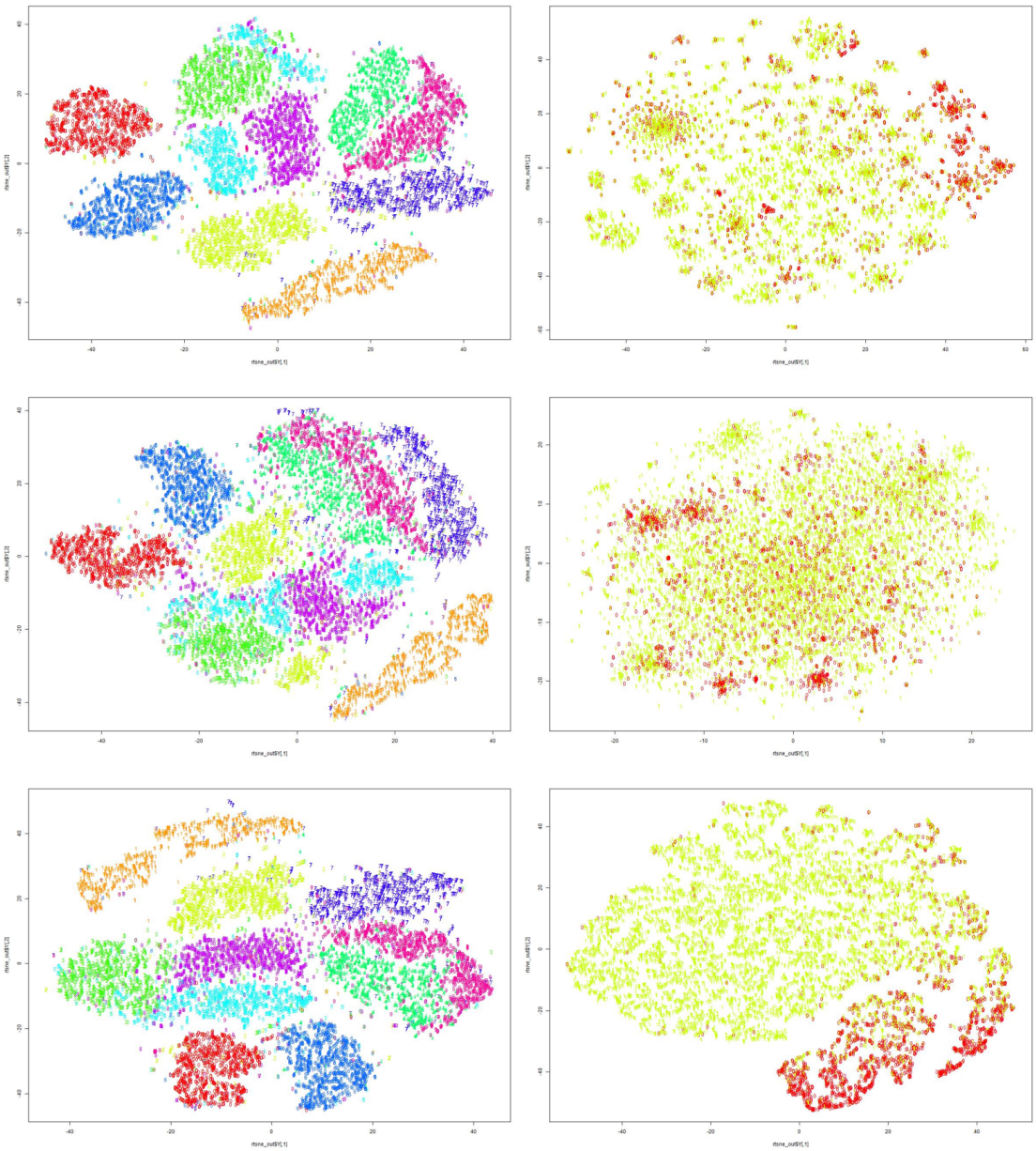

**图 4.5** 数据集 tsne 可视化结果　其中左列是 MNIST 数据集，右列是推特情感数据。第一排是原始数据集，第二排是无监督自编码器，第三排是半监督自编码器

### 4.4.4 分类判别实验

本文利用了 MNIST 数据库和航班评价微博数据做了相关的判别实验分析。MNIST 中包含了 60000 个 784 维的训练数据，10000 个测试数据。将 0 视为少数类以后不平衡比例约为一比九。微博数据中包含了 14485 条 1618 维的训练数据，每次从中划取 1000 个为测试数据。将积极类视为少数类以后不平衡比例约为一比五。在分类器上选择使用了神经网络 (MLP, Muti-Layer Perceptron) 和朴素贝叶斯方法 (Naive Bayes)。

表4.3列举了相关的分类判别实验，以此来说明不同 SMOTE 算法最后对分类器的影响。SMOTE 表示原始的 SMOTE 算法，ESMOTE 表示使用随机投影树

加速的快速 SMOTE 算法，AESMOTE 表示使用半监督自编码器降维的高维自编码 SMOTE 算法。可以看出使用随机投影树加速对于 SMOTE 算法的质量并没有显著的影响。

在分类器方面，由于神经网络在处理高维数据时本身就比其它分类器优秀，故它在直接对高维数据做分类效果比朴素贝叶斯分类器要好很多。朴素贝叶斯分类器虽然在除了使用高位自编码 SMOTE 之外的试验中达到了很高的召回率，但是其代价是严重削弱的总体的准确率。而在使用了高位自编码 SMOTE 算法以后，朴素贝叶斯的准确率可以达到非常高，召回率也相对优秀。可以看出高位自编码 SMOTE 算法能够提高 SMOTE 算法对高维数据的处理能力。

**表 4.3** Classification Experiment Results

| | Accuracy | Recall | Accuracy | Recall |
|---|---|---|---|---|
| | MLP | | NaiveBayesian | |
| Tweeter Airline | 0.911 | 0.637 | 0.180 | 1 |
| MNIST | 0.970 | 0.750 | 0.171 | 0.993 |
| | SMOTE+MLP | | SMOTE+Naive Bayes | |
| Tweeter Airline | 0.846 | 0.755 | 0.189 | 0.996 |
| MNIST | 0.968 | 0.978 | 0.204 | 0.994 |
| | ESMOTE+MLP | | ESMOTE+Naive Bayes | |
| Tweeter Airline | 0.852 | 0.767 | 0.188 | 0.997 |
| MNIST | 0.970 | 0.969 | 0.203 | 0.994 |
| | AESMOTE+MLP | | AESMOTE+Naive Bayes | |
| Tweeter Airline | 0.854 | 0.732 | **0.865** | 0.709 |
| MNIST | **0.992** | **0.980** | **0.970** | 0.888 |

# 第五章　总结

本文主要对高维大规模不平衡数据的 SMOTE 算法做了相关的研究。首先本工作详尽地调研了相关的算法和知识，如不平衡学习算法，流行学习和降维，最近邻算法。由于 R 语言在统计和数据分析领域的流型性，和他本身开源易用等特点，本工作选择 R 语言为工作平台。因此也调研了现有的 R 语言上的 SMOTE 算法，发现已有的实现效率十分低下。完全无法处理高维大规模数据。因此本人首先在 R 语言平台上建立 R 语言包，并且设计了 R 语言包的结构。针对 R 语言本身难以处理大规模数据的特点，底层使用了 C++ 代码对其进行了效率上的优化。在算法上首先利用随机投影森林实现了快速 SMOTE 算法，其算法速度在试验中被证明是相对最快的。而且根据分类判别方面的实验得出，基于近似 K 最近邻的快速 SMOTE 算法和准确 K 最近邻的原始 SMOTE 算法并无太大差异。再其次本文还探索了如何利用自编码器改进算法处理高维数据。具体来说，本文首先根据问题特点提出了半监督的自编码器模型。并计算了参数梯度，给出了可靠的优化办法。并做了相关的实验分析。

**未来工作**　由于本工作仍在开发阶段，代码和文档还有不完善之处。相关工作暂未公开放到 R 语言的开源社区中。今后希望相关工作能够进一步完善并且能够发布至 Github，CRAN 等平台，开源供广大的数据分析专业人士使用。

# 参考文献


[1] Chawla N V, Bowyer K W, Hall L O, et al. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 2002, 16:321–357.

[2] Tang J, Liu J, Zhang M, et al. Visualization Large-scale and High-dimensional Data. arXiv preprint arXiv:1602.00370, 2016..

[3] Wang D, Cui P, Zhu W. Structural deep network embedding. Proceedings of Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016. 1225–1234.

[4] Jolliffe I. Principal component analysis. Wiley Online Library, 2002.

[5] Maaten L v d, Hinton G. Visualizing data using t-SNE. Journal of Machine Learning Research, 2008, 9(Nov):2579–2605.

[6] Bengio Y, et al. Learning deep architectures for AI. Foundations and trends® in Machine Learning, 2009, 2(1):1–127.

[7] Ali A, Shamsuddin S M, Ralescu A L. Classification with class imbalance problem: A Review. Int. J. Advance Soft Compu. Appl, 2015, 7(3).

[8] Chawla N, Japkowicz N, Kolcz A. ICML’2003 Workshop on Learning from Imbalanced Data Sets (II). Proceedings of Proceedings available at:< http://www. site. uottawa. ca/~ nat/Workshop2003/workshop2003. html, 2003.

[9] Yang Z, Tang W, Shintemirov A, et al. Association rule mining-based dissolved gas analysis for fault diagnosis of power transformers. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2009, 39(6):597–610.

[10] Wang S, Minku L L, Yao X. Online class imbalance learning and its applications in fault detection. International Journal of Computational Intelligence and Applications, 2013, 12(04):1340001.

[11] Mazurowski M A, Habas P A, Zurada J M, et al. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. Neural networks, 2008, 21(2):427–436.

[12] Soler V, Cerquides J, Sabria J, et al. Imbalanced datasets classification by fuzzy rule extraction and genetic algorithms. Proceedings of Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on. IEEE, 2006. 330–336.

[13] Tavallaee M, Stakhanova N, Ghorbani A A. Toward credible evaluation of anomaly-based intrusion-detection methods. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2010, 40(5):516–524.

[14] Imam T, Ting K, Kamruzzaman J. z-SVM: an SVM for improved classification of imbalanced data. AI 2006: Advances in Artificial Intelligence, 2006. 264–273.

[15] Tang Y, Zhang Y Q, Chawla N V, et al. SVMs modeling for highly imbalanced classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2009, 39(1):281–288.

[16] Kriminger E, Príncipe J C, Lakshminarayan C. Nearest neighbor distributions for imbalanced classification. Proceedings of Neural Networks (IJCNN), The 2012 International Joint Conference on. IEEE, 2012. 1–5.

[17] Cao P, Zhao D, Zaïane O R. A PSO-based cost-sensitive neural network for imbalanced data classification. Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, 2013. 452–463.

[18] Chawla N V, Lazarevic A, Hall L O, et al. SMOTEBoost: Improving prediction of the minority class in boosting. Proceedings of European Conference on Principles of Data Mining and Knowledge Discovery. Springer, 2003. 107–119.

[19] Seiffert C, Khoshgoftaar T M, Van Hulse J, et al. RUSBoost: A hybrid approach to alleviating class imbalance. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2010, 40(1):185–197.

[20] Wang S, Chen H, Yao X. Negative correlation learning for classification ensembles. Proceedings of Neural Networks (IJCNN), The 2010 International Joint Conference on. IEEE, 2010. 1–8.

[21] Mount D M, Arya S. ANN: A library for approximate nearest neighbor searching (2006). URL http://www. cs. umd. edu/mount/ANN, 2012..

[22] Andoni A, Indyk P, Laarhoven T, et al. Practical and optimal LSH for angular distance. Proceedings of Advances in Neural Information Processing Systems, 2015. 1225–1233.
[23] Bernhardsson E. Benchmarking nearest neighbors. URL https://github.com/erikbern/ann-benchmarks, 2017..
[24] Boytsov L, Novak D, Malkov Y, et al. Off the Beaten Path: Let's Replace Term-Based Retrieval with k-NN Search. Proceedings of Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016, 2016. 1099–1108.
[25] Tenenbaum J B, De Silva V, Langford J C. A global geometric framework for nonlinear dimensionality reduction. science, 2000, 290(5500):2319–2323.
[26] Roweis S T, Saul L K. Nonlinear dimensionality reduction by locally linear embedding. science, 2000, 290(5500):2323–2326.
[27] Belkin M, Niyogi P. Laplacian eigenmaps and spectral techniques for embedding and clustering. Proceedings of NIPS, volume 14, 2001. 585–591.
[28] Hinton G, Roweis S. Stochastic neighbor embedding. Proceedings of NIPS, volume 15, 2002. 833–840.
[29] Torgo L, Torgo M L. Package 'DMwR'. Comprehensive R Archive Network, 2013..
[30] Wickham H. R packages. " O'Reilly Media, Inc.", 2015.
[31] Eddelbuettel D, Francois R, Leeuw J D, et al. Rcpp: Seamless R and C++ Integration. Journal of Statistical Software, 2011, 40(i08).
[32] Eddelbuettel D, François R. Exposing C++ functions and classes with Rcpp modules. 2010..
[33] Chen T, Li M, Li Y, et al. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. Statistics, 2015..
[34] Eddelbuettel D, Sanderson C. RcppArmadillo: Accelerating R with high-performance C++linear algebra. Computational Statistics & Data Analysis, 2014, 71(March):1054–1063.
[35] Dasgupta S, Freund Y. Random projection trees and low dimensional manifolds. Proceedings of Proceedings of the fortieth annual ACM symposium on Theory of computing. ACM, 2008. 537–546.
[36] Dong W, Moses C, Li K. Efficient k-nearest neighbor graph construction for generic similarity measures. Proceedings of International Conference on World Wide Web, 2011. 577–586.
[37] Lecun Y, Cortes C. The mnist database of handwritten digits. 2010..
[38] Shaw B, Jebara T. Structure preserving embedding. Proceedings of International Conference on Machine Learning, 2009. 937–944.
[39] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. Journal of Machine Learning Research, 2010, 9:249–256.
[40] He K, Zhang X, Ren S, et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. 2015. 1026–1034.
[41] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets. Neural Computation, 2006, 18(7):1527.
[42] Erhan D, Bengio Y, Courville A, et al. Why Does Unsupervised Pre-training Help Deep Learning? Journal of Machine Learning Research, 2010, 11(3):625–660.
[43] Chawla N, Japkowicz N, Kolcz A. Special issue on learning from imbalanced datasets, sigkdd explorations. Proceedings of ACM SIGKDD, 2004.
[44] Ng A. Sparse autoencoder. CS294A Lecture notes, 2011, 72(2011):1–19.