# 基于vae的不平衡分类

# Abstract

分类问题是机器学习中非常重要的一部分，而在不平衡分类问题中，传统的机器学习方法容易忽视少数类样本的识别，造成少数类识别率很低，在本文中，利用变分自编码器对少数类样本进行扩充，并提出了采用自编码器的重建模型进行分类，结果表明，本文算法在特定的数据集上可以取得目前比较好的效果。

关键字：不平衡分类，生成模型，变分自编码器，重建模型

# Introduction

分类问题是机器学习中非常重要的一部分，目前，大多数的分类问题都是假设不同类别的样本数目是相等的，但是现实问题中，人们比较关心的数据往往是比较稀少的，例如，信用卡欺诈行为检测，医学疾病诊断等，由于数据集中需要关注的类别中样本数量很稀少，导致分类器对少数类关注度不够，因此识别效果不佳。

传统的不平衡问题通常的解决方案有两种思路：数据层面和算法层面，数据层面的主要是采样，从采样的对象上，可以分为过采样、欠采样和混合采样等等。过采样是指在模型训练过程中针对比较容易被忽视的少数类样本，增加其训练次数的过程，常见的过采样方法有简单复制样本的重采样、针对样本进行线性插值的SMOTE[1]等，过采样可以有效提高分类器对少数类的关注度，但是简单复制样本并不会加入额外信息，而SMOTE这种随机插值的方法，则缺乏典型性且拥有较强的随机性。欠采样[2]是指针对大量的多数类样本，对其进行删除的方法，欠采样可以很快达到平衡状态，但可能有丢失有价值样本的风险。混合采样[3]则是对以上两种采样方式的统一，也存在着丢失珍贵样本的风险。

从采样方式上，可以分为有信息的采样和随机采样，随机采样是指利用样本直接进行采样，包括简单重复、线性插值、非线性插值等；有信息的采样指的是利用样本中的分布信息，对其分布进行拟合，并根据建模内容进行采样，例如论文中涉及到的高斯分布、威布尔分布等，这些算法较随机插值方法有了比较大的进步，因为利用了样本分布信息，但是在这些方案中存在的问题同样也很明显：算法假设样本服从某种先验分布，例如正态分布等，然后利用EM算法对分布中的参数进行计算，当样本分布确实为该假设分布时，这种算法通常能获得比较好的效果，但是如果样本先验分布不服从假设分布时，这种过采样算法能够获得的提升效果往往有限，而在数据类型的数据集中，我们难以获得样本的真实分布情况，因而这种基于分布的算法效果不是很稳定。

算法层面的是大多是基于代价敏感的思路，即针对少数类和多数类的分类结果进行不同的分类惩罚，比较典型的有直接利用F1值作为反向传播标准的代价函数，还有各种集成算法、启发式算法等，多是针对少数类的分类错误进行更大的代价惩罚，例如结构化SVM，就是可以使用少数类的F1值对分类面进行改进，因而在少数类的分类效果上较好的性能。

由于目前的过采样方法在生成数据时多是采用直接复制或者简单的线性插值的方法，没有充分利用样本之间的分布信息，也无法保证插值样本的合理性，本文中针对不平衡分类中的少数类样本难以识别的问题，根据样本之间的分布关系，利用神经网络中非常具有生成能力的vae模型对少数类样本进行扩充，并针对插值样本的合理性问题，提出了利用独立的第三方分类器，对生成样本进行评价，实验结果证明该模型可以有效提高分类器的F1值。

本文的贡献如下：

1. 利用vae对少数类样本分布进行建模，并利用该模型对少数类样本过采样，解决分类问题中的不平衡问题。
2. 对vae中采样生成的少数类，本文中提出了有效的机制进行评价，以避免加入不好的样本进入到原始样本集中，提高了分类性能，并减少过拟合的风险。

本文的结构如下：2中介绍了本文的相关工作，主要是vae等；3中介绍了本文方法及对该方法的分析；4中介绍了基本数据集和实验结果等；5中为论文结论和将来的扩展。

# Related work

## Vae

自编码器是Lecun在1987年提出的，其表型形式如下图所示，ad中的隐层变量可以认为是原始空间中变量的压缩编码，由于实数的无限性，理论上该类形式的编码可以存储无限信息。但这些信息在特定结构的ad中，都是由原始的X决定的，无法随意生成，因此2013年KM对其进行改进，将变分推断加入到ad中，并提出了重参数化的机制，使得变分推断可以和随机梯度下降相结合，因而产生了vae，在vae中，我们假设变量是由隐层的压缩编码决定的，而该压缩编码服从某种特定的分布，因此在已知该特定分布的情况下，可以根据其CDF进行采样，理论上可以产生无限样本。

Vae的结构如图所示：

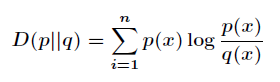
利用神经网络生成样本，假设样本的最后表现形式由某些潜在的变量决定，设其为z，并对其先验分布和后验分布用神经网络进行模拟。神经网络的生成模型优势在于其输出维度任意，因此几乎可以生成任意维度的数据。

生成模型是根据观测数据得出其分布，并生成观测数据中未曾出现的有效数据，一个自然的想法是在原始样本空间中计算P（X）以估计其出现的概率大小，但是因为不知道数据分布，所以P（x）无法计算，根据贝叶斯公式，假设z为潜在变量，

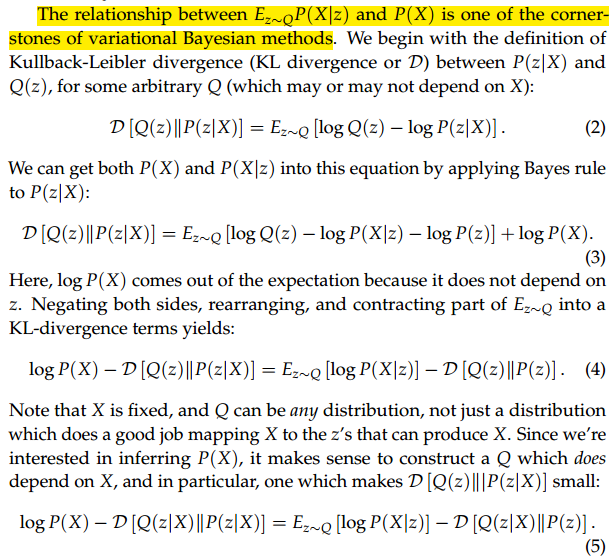
但对于大多数的z来说，都无法生成可靠的样本，即趋于0，则也趋于0，为了简化计算，假设是相同分布的，则只需要计算，我们采用直接对较大值的z进行计算和采样。

由自编码器的编码部分可得，设其产生的分布为，采用KL散度来计算分布拟合误差

KL散度定义：

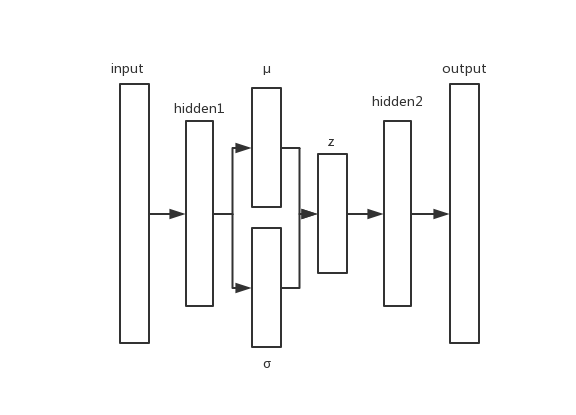
从等式中可以看出，如果pq比较接近的话，KL散度会趋于0.

计算



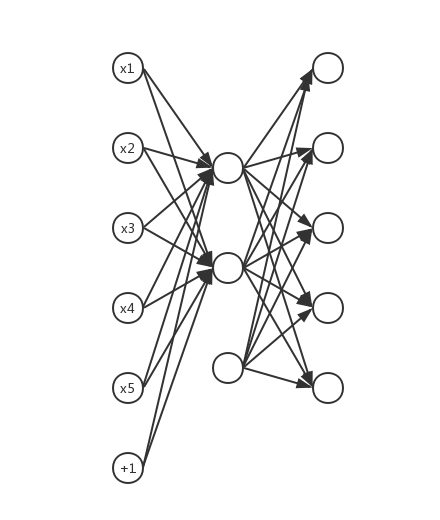
从等式（5）中可以看到，只要趋于0，则达到了我们最大化P（x）的目的。

但是我们对潜在变量的形式和分布都不清楚，需要利用神经网络的拟合能力对其真正的分布进行拟合，因此vae的网络结构需要多隐藏层，以对z的真实分布进行拟合。



## 自编码器

自编码器是由LeCun在1987年提出的，最开始用于对数据的降维或特征学习，网络架构如图:



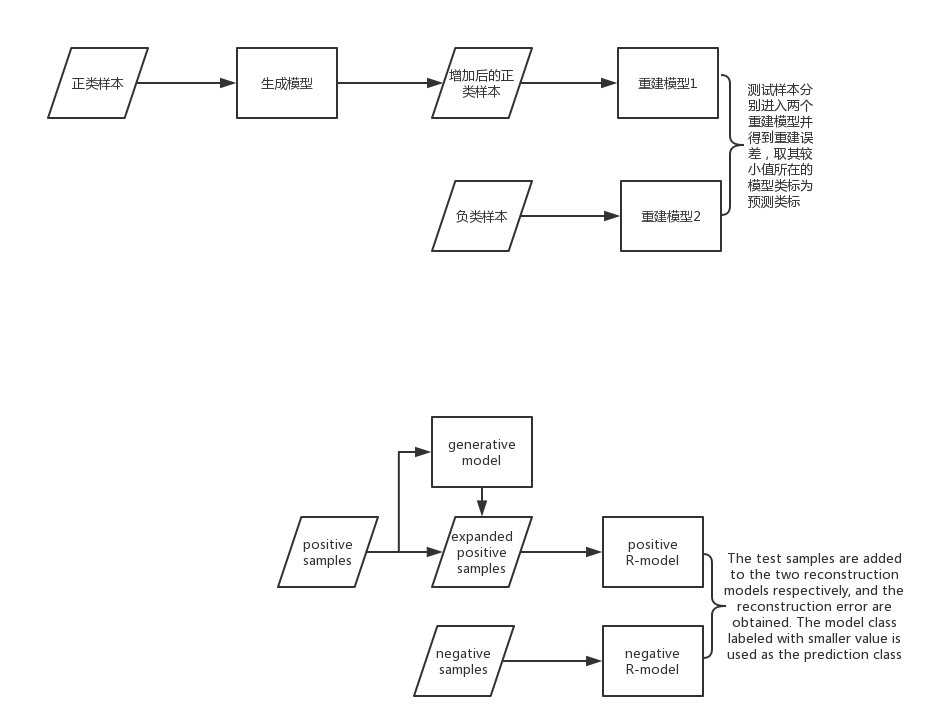
自编码器的损失函数为网络输出与输入的误差，该网络可以看成是由两部分组成，由输入到隐藏层的编码器和由隐藏层映射到输出的解码器，此外还发展出许多变种，例如对网络权重进行惩罚的稀疏自编码器[5]，要求网络权重趋于0，这样的网络结构对输入的小范围变化更具有鲁棒性。还有在输入中增加噪声的去噪自编码器[6]，在输入中加入噪声，要求自编码器对被噪声污染过的输入进行重建，误差仍为输出与原始输入的误差，该网络结构能够对噪声污染的输入进行重建，也因此网络的鲁棒性较好。

自编码器的使用分为有监督类和无监督类型，自编码器本身是无监督网络，但是由于目前的无监督学习发展结果并不很好，所以大部分的自编码器在使用时仍然加入了有监督的机制，有很多的不平衡分类采用了自编器进行分类，多数是采用先利用逐层训练的方法对自编码器进行预训练，之后采用有监督微调的机制，即在自编码器的隐层输出上加上一层softmax层，根据其输出的预测类标进行微调[7]。

本文[7]中采用了SMOTE+有监督自编码器的架构，首先对数据采取SMOTE方式进行过采样至数据集平衡，后采用AE对数据进行重建，最后加入softmax层进行分类。结果显示该模型对提高不平衡分类效果有一定的好处，并在UCI数据集上的auc值取得了比较高的测试结果。

针对模型的任务相关性不强的问题，本文[8]提出在AE的训练loss中加入有监督分量，即在模型中添加一个softmax层，并在训练中根据其分类结果，调整自编码器的参数。传统的训练过程是逐层训练，最后添加一个softmax层，而本文[8]则在训练过程中直接添加了softmax层，并且将其分类结果直接添加入loss函数。

在这篇文章中，主要是分析二分类中的不平衡问题，因此本文训练了两个模型分别为正类样本的重建模型和负类样本的重建模型，分类流程如下：



算法描述：

对原始数据进行划分，分割出训练集和测试集

分开训练集中的正类样本和负类样本，将正类样本输入到生成模型中，产生一定数量的正类生成样本，并根据选择机制将生成的样本筛选，得出信息量较大的样本。

在生成模型中，本文采用了传统的vae作为生成网络，网络结构如图1所示

在传统的分类算法中由于目标函数倾向于最小化整体的错误率，但在不平衡样本分类中，少数类样本的错误代价更大，在文献[10]中说明在不平衡分类中，阈值设为0.5是不合理的，并提出了一个寻找最佳分类阈值的novel架构，同时，该文献还提出了在极端情况下，AUC值的不可靠性，因此在实验结果分析中，只采用了F-value和gmean等能够分析整体分类情况的预测指标，同时，本文结果采取了加权投票机制，并对比在不同的阈值下，本文算法的性能。

# Method

由于原始的vae是应用于图像生成中，因此可以天然可视化；但本文中直接使用了序列数据，因而需要对其进行比较多的改进：

首先，由于序列数据中可能存在着离散特征，而vae中使用的随机梯度下降使得所生成的特征必定是连续可微分的，因此在vae开始训练前，需要将这部分特征进行筛选，并在生成后对生成的样本进行1NN分类，将其最靠近的原始样本的离散特征部分和生成的连续特征组合成新的合成样本；在机器学习中，因为我们是假设所有的样本都是独立同分布的，因此在样本生成后，由于无法保证这些随机采样的样本均是正类样本，所以我们需要对这些生成样本再进行筛选，筛选机制是利用无类标的迁移学习，将那些能够提高分类性能的合成样本筛选出来，加入到最后的生成样本中。该算法的流程图如下：

由于样本数量稀少，无法可靠判断某个特征是否为离散特征，因此本文中认定已经出现的特征值少于2个的，均认定其为离散变量，实际上，如果某个特征在各类样本中只有1个值的话，该类特征对于分类是没有作用的。

# Experiment

数据集描述

文中实验数据来自于UCI机器学习数据库，都是比较常用的二分类不平衡数据集，具体描述如表中所示

表1 数据集描述

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 数据集 | 样本总数 | 属性数 | 少数类 | 多数类 | 不平衡率 |
| Ionosphere | 351 | 34 | 126 | 225 | 1:1.8 |
| german | 1000 | 24 | 300 | 700 | 1:2.3 |
| wpbc | 198 | 33 | 47 | 151 | 1:3.2 |

基于混淆矩阵的评估指标

在二分类中，常常使用混淆矩阵来评估分类器的性能，其定义如下

表2 二分类问题的混淆矩阵

|  |  |  |
| --- | --- | --- |
|  | Positive prediction | Negative prediction |
| Positive class | True positive(TP) | False negative(FN) |
| Negative class | False positive(FP) | True negative(TN) |

其中，TP代表正类被正确预测，判定为正类，TN则是负类被正确预测为负类的样本数，FN则表示正类被错误判定为负类，而FP则为负类被错误判定为负类的情况，在传统的分类方法中，通常采用以整体准确率作为评价指标，而在不平衡问题中，由于正类的数量较少，则采用整体准确率为评价指标会导致分类器对少数类不敏感，极端情况下，如果数据集中仅仅包含1%的少数类，如果分类器将所有样本全部判定为多数类时，整体准确率仍然可以达到99%，但是这对我们关心的少数类是非常不利的，因此传统的分类算法会导致少数类容易被分成多数类，导致少数类的识别率较低，目前出现了一些新的不平衡数据的分类评价指标，根据混淆矩阵中可以计算准确率和召回率等，例如 AUC、 F-value 和 G-mean 等方法[11]。

F-value 是衡量准确率和召回率的分类评价指标， 比较偏向对少数类的分类性能评价，定义如下：

其中，准确率，召回率,β取值为[0,+∞]。本实验中取β=1，此时的F-value表示召回率和准确率之间权重的平均。

G-mean表示少数类分类精度和多数类分类精度的几何平均值，用来评价分类器的整体的分类性能，其定义如下：

G-mean表示只有在少数类和多数类分类精度同时都高的情况下，此时，G-mean的值最大。

实验结果及分析

本文的对比算法为论文中实现的SMOTE-SDAE算法[7]，以及传统的朴素贝叶斯算法，算法设置如下：

本文中采用10-折交叉验证，对少数类和多数类同时进行分割，以保证数据分布同原始分布较一致，本文的算法vae中采用的是三层隐藏层的设置，其中包括产生潜在变量z的隐藏层，而重建模型则是含有一层隐藏层的DAE，利用重建误差来进行分类，同时还对少数类和多数类的重建误差有加权的投票机制。

不同算法的F1值比较 %

|  |  |  |  |
| --- | --- | --- | --- |
|  | ionosphere | german | wpbc |
| SMOTE-SDAE | 78.42 | **81.92** | 83.71 |
| VAE-R | **92.29** | 47.59 | **86.98** |
| Naïve Bayes | 90.99 | 58.07 | 76.38 |

不同算法的gmean值比较 %

|  |  |  |  |
| --- | --- | --- | --- |
|  | ionosphere | german | wpbc |
| VAE-R | **86.08** | 35.34 | 9.1 |
| SMOTE-SDAE | 76.15 | **84.50** | **65.24** |

从实验数据中可以看到，在加入比较有效的生成样本的前提下，本文提出的分类器可以获得比较好的结果，但是该分类器倾向于将样本划分成少数类，因此在样本数量较少的情况下，多数类容易全部被误分，gmean值很容易为0。

本文提出的算法在样本总体数量较少的情况下，能够得到比较好的效果，即少数类是绝对稀缺的情况，因此在ionosphere和wpbc数据集上的F1值比较高，能达到目前state-of-art水平，但是在german这种样本数量较多的数据集中，算法的重建模型容量不足以拟合整个数据集，会造成随机分类的情况。

生成样本质量对分类结果的影响

该实验基于ionosphere数据集，由于vae中假设是高斯分布，因此在符合高斯分布的情况下，能够获得比较好的结果，在F1值和G-mean都能取得比较高的效果。

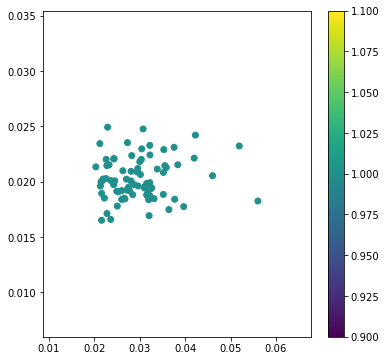
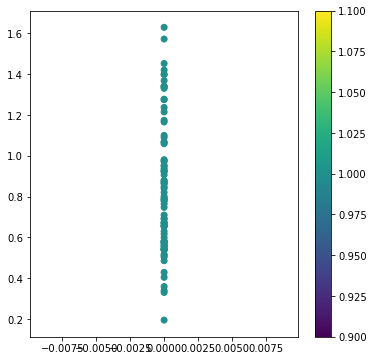
 

图1中的测试结果为F1=0.84，图2结果为F1=0.79，从中可以看出，虽然在假设中认为无论的分布形式如何，都能通过神经网络将其映射成真实的潜变量分布，但是实际上由于神经网络的容量问题，且神经网络容易陷入局部极小值，不一定能将完全无偏差的映射到真实的潜变量分布，也因此会造成生成样本的质量不佳问题。

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Method | Ionosphere | | | German | | | wpbc | | | satimage | | |
|  | F1 | AUC | gmean | F1 | AUC | gmean |  |  |  | F1 | AUC | gmean | |
| 本文方法 | 0.876285294416 | 0.946385679538 | 0.893525858509 |  |  |  |  |  |  | 0.507584388669 | 0.908044007126 | 0.8245453368 | |
| 高斯贝叶斯 | 0.811949278791 | 0.935222205331 | 0.839234927297 |  |  |  |  |  |  |  |  |  | |

**Segment** 单倍采样

mean F1: 0.500044392626 mean AUC: 0.908667927026 mean gmean: 0.815389629799

不采用mean F1: 0.472241062803 mean AUC: 0.908282882197 mean gmean: 0.824115335389

采样至与多数类一致

mean F1: 0.534049274178 mean AUC: 0.908140575286 mean gmean: 0.809373602351

mean F1: 0.472241062803 mean AUC: 0.908282882197 mean gmean: 0.824115335389

**wpbc**（vae,smote,无采样）

##########################zhouying###################################

mean F1: 0.418597089185 mean AUC: 0.679333333333 mean gmean: 0.562038054871

##########################zhouying###################################

mean F1: 0.413424548448 mean AUC: 0.656666666667 mean gmean: 0.536602522928

##########################zhouying###################################

mean F1: 0.40663003663 mean AUC: 0.66325 mean gmean: 0.552930042654

**satimage**

##naive-bayes+vae#########

mean F1: 0.527982278034 mean AUC: 0.908573108731 mean gmean: 0.817048170228

###naive-bayes########

mean F1: 0.472241062803 mean AUC: 0.908282882197 mean gmean: 0.824115335389

#####desion-tree+vae###########

mean F1: 0.472902263483 mean AUC: 0.715364883444 mean gmean: 0.671208774182

#####desion-tree###########

mean F1: 0.482577139108 mean AUC: 0.719497467065 mean gmean: 0.673346470786

**sonar**

##naive-bayes+vae#########

mean F1: 0.633796157059 mean AUC: 0.685252525253 mean gmean: 0.621010622268

###naive-bayes########

mean F1: 0.630129252898 mean AUC: 0.735101010101 mean gmean: 0.568492193988

#####desion-tree+vae###########

mean F1: 0.681675434988 mean AUC: 0.718888888889 mean gmean: 0.707685346894

#####desion-tree###########

mean F1: 0.557012087328 mean AUC: 0.593484848485 mean gmean: 0.567804318474

#####3-NN+vae###########

mean F1: 0.734302698479 mean AUC: 0.755101010101 mean gmean: 0.628800442083

#####3-NN###########

mean F1: 0.609974267219 mean AUC: 0.717664141414 mean gmean: 0.625461262596

**vehicle**

##naive-bayes+vae#########

mean F1: 0.578904640209 mean AUC: 0.83030041751 mean gmean: 0.747592690545

###naive-bayes########

mean F1: 0.541271883001 mean AUC: 0.815019420547 mean gmean: 0.711768469124

#####desion-tree+vae###########

mean F1: 0.858201199662 mean AUC: 0.906482793522 mean gmean: 0.90313293843

#####desion-tree###########

mean F1: 0.860951797688 mean AUC: 0.914570470648 mean gmean: 0.912546990481

#####3-NN+vae###########

mean F1: 0.88415220962 mean AUC: 0.979714290233 mean gmean: 0.938869294995

#####3-NN###########

mean F1: 0.872825407597 mean AUC: 0.97412496837 mean gmean: 0.915172594287

**ionosphere**

##naive-bayes+vae#########

mean F1: 0.872511908381 mean AUC: 0.945863737712 mean gmean: 0.886474293625

###naive-bayes########

mean F1: 0.794044685292 mean AUC: 0.935222205331 mean gmean: 0.824486059181

#####desion-tree+vae###########

mean F1: 0.819656414439 mean AUC: 0.859244197831 mean gmean: 0.856885988726

#####desion-tree###########

mean F1: 0.825208458208 mean AUC: 0.864414208979 mean gmean: 0.859892323013

#####3-NN+vae###########

mean F1: 0.790534823681 mean AUC: 0.900785446438 mean gmean: 0.825497041452

#####3-NN###########

mean F1: 0.730153793575 mean AUC: 0.902009856086 mean gmean: 0.765699158425

**diabetes**

##naive-bayes+vae#########

mean F1: 0.656929208727 mean AUC: 0.820245014245 mean gmean: 0.733016401461

###naive-bayes########

mean F1: 0.628791254285 mean AUC: 0.816065527066 mean gmean: 0.706039250773

#####desion-tree+vae###########

mean F1: 0.606401190992 mean AUC: 0.697615384615 mean gmean: 0.689427126237

#####desion-tree###########

mean F1: 0.563720708989 mean AUC: 0.666398860399 mean gmean: 0.655568089674

#####3-NN+vae###########

mean F1: 0.622565185065 mean AUC: 0.766303418803 mean gmean: 0.703473474807

#####3-NN###########

mean F1: 0.608424459364 mean AUC: 0.763501424501 mean gmean: 0.690124556817

##naive-bayes+vae#########

mean F1: 0.661078619953 mean AUC: 0.818660968661 mean gmean: 0.736432055798

###naive-bayes########

mean F1: 0.628791254285 mean AUC: 0.816065527066 mean gmean: 0.706039250773

#####desion-tree+vae###########

mean F1: 0.575213487082 mean AUC: 0.673800569801 mean gmean: 0.664750982799

#####desion-tree###########

mean F1: 0.552462745814 mean AUC: 0.657772079772 mean gmean: 0.647504741637

#####3-NN+vae###########

mean F1: 0.632346313415 mean AUC: 0.77947008547 mean gmean: 0.710919054641

#####3-NN###########

mean F1: 0.608424459364 mean AUC: 0.763501424501 mean gmean: 0.690124556817

segment-challenge

#####desion-tree+vae###########

mean F1: **0.975433664663** mean AUC: 0.985567978533 mean gmean: 0.985361429209

#####desion-tree###########

mean F1: 0.969654129654 mean AUC: 0.980567978533 mean gmean: 0.980101991425

##naive-bayes+vae#########

mean F1: **0.67224340894** mean AUC: 0.976652185024 mean gmean: 0.908233345033

###naive-bayes########

mean F1: 0.609158615611 mean AUC: 0.976434960388 mean gmean: 0.887662310237

#####3-NN+vae###########

mean F1: 0.974944472639 mean AUC: 0.991996124031 mean gmean: 0.985278667603

#####3-NN###########

mean F1: 0.969944472639 mean AUC: 0.991920734304 mean gmean: 0.980346451613

# Conclusion

在这篇论文中，我们利用生成模型来代替传统的重采样机制，这样可以充分利用数据集中的信息，同时，利用自编码器对不同类别的样本的特征获取，根据其重建误差来对样本进行分类，这样，分类器是有两个相对独立的自编码器组成，能够有效防止分类器对少数类的忽视，但是本文算法中，投票机制中的权重会在最终结果上有非常大的影响，同时，在不加权时，本文中的算法会倾向于将多数类误分成少数类，也会造成分类器的性能下降，在生成样本的质量衡量标准上仍然有很多不足，目前只能依靠分类器的结果来衡量生成样本质量，在以后的研究中应当会改进。

Reference

[1] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321–357, 2011.

[2] D. L. Donoho and J. Tanner, “Precise Undersampling Theorems,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 913–924, 2010.

[3] I. Bilinskis, *Hybrid Sampling*. John Wiley & Sons, Ltd, 2007.

[4] C. Doersch, “Tutorial on Variational Autoencoders,” 2016.

[5] A. Ng, “Sparse autoencoder,” 2011.

[6] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *International Conference on Machine Learning*, 2008, pp. 1096–1103.

[7] 张成刚, 宋佳智, 姜静清, and 裴志利, “一种改进的降噪自编码神经网络不平衡数据分类算法,” 计算机应用研究, no. 5, pp. 1329–1332, 2017.

[8] Y. Sun, H. Mao, Y. Sang, and Z. Yi, “Explicit guiding auto-encoders for learning meaningful representation,” *Neural Computing & Applications*, vol. 28, no. 3, pp. 429–436, 2017.

[9] M. Hayat, M. Bennamoun, and S. An, “Learning Non-linear Reconstruction Models for Image Set Classification,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 37, no. 4, pp. 713–727, 2015.

[10] Q. Zou, S. Xie, Z. Lin, M. Wu, and Y. Ju, “Finding the Best Classification Threshold in Imbalanced Classification ☆,” *Big Data Research*, vol. 5, pp. 2–8, 2016.

[11] G. Menardi and N. Torelli, “Training and assessing classification rules with imbalanced data,” *Data Mining & Knowledge Discovery*, vol. 28, no. 1, pp. 92–122, 2014.