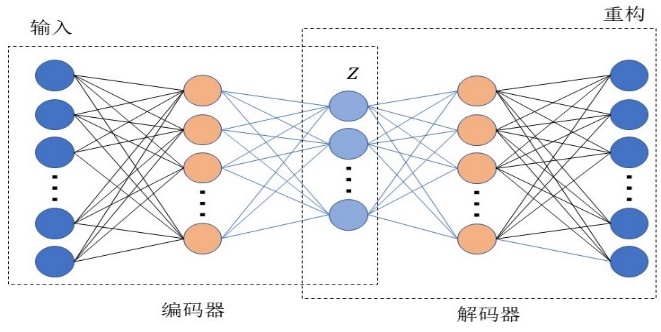
**基于自编码器的异常检测报告**

**绪论--自编码器简介**

自编码器是一类特殊的端到端的神经网络结构。它的特殊之处在于输出端的维度和输入端的维度是相同的，而且输出一般是输入的重构，损失函数一般是重构误差。模型结构如下：



自编码器分为两部分：编码器和解码器。编码器将输入向量编码为较低维度的编码向量Z,

解码器将Z解码为重构向量。

从结构可以看出，自编码器实质上是一个无监督模型，它不需要数据的标签指导模型的训练，而是通过重构自身的方式训练使得模型收敛。而自编码器在重构自身的同时得到了一个有用向量——编码向量Z。

编码向量Z也称为隐向量(latent variable)，它所在的空间称为隐空间（latent space）。隐向量Z的维度一般低于输入向量的维度。这样的设置的原因是：输入向量的特征是有冗余的，因此我们要将输入用作后续任务时，通过选择（仅保留一些现有特征）或通过提取（基于就特征组合生成数量更少的新特征）进行降维，来保留关键的主要特征。而如果将Z的维度设置为比输入的更高，那么则会增加无效特征而无益于后续任务。因此编码器就是在降维或者数据压缩，解码器则是解压缩。

当然，根据初始数据分布，隐空间大小和编码器的选择，压缩是有损的，即一部分信息在编码过程中丢失，并且在解码时无法恢复。我们考虑一种情形是不是可以让编码器做到无损压缩？即： 或,所以.换句话说，是的逆映射或反函数。在一般的自编码器中，编码和解码都是非线性映射，非线性体现在激活函数，但是要知道函数与反函数的图像是关于y=x对称的。而目前使用的激活函数如Relu(), tanh(), sigmoid()的反函数不存在或者会使得梯度爆炸。因此，只有一种情形使得是的逆映射或反函数，那就是f为线性映射。

这种情形则会出现两个问题：1.在求逆映射时，要求矩阵的逆，而且矩阵的维度和输入向量维度相同，可能会维度过大而计算困难，或者不可逆。2. 也是最重要的问题，在数据压缩中，线性压缩导致学习的编码向量太简单，无法满足后续任务。神经网络中，我们一般为了提取深层特征，增加网络的深度和非线性性。而这样我们为了无损恢复数据，而无法提取好的编码表示，得不偿失。

因此，我们通过建立非线性的网络，损失一部分信息，提取更有用的信息。在现有的模型和参数设置下，使得解压缩函数尽可能近似于压缩函数的逆函数。

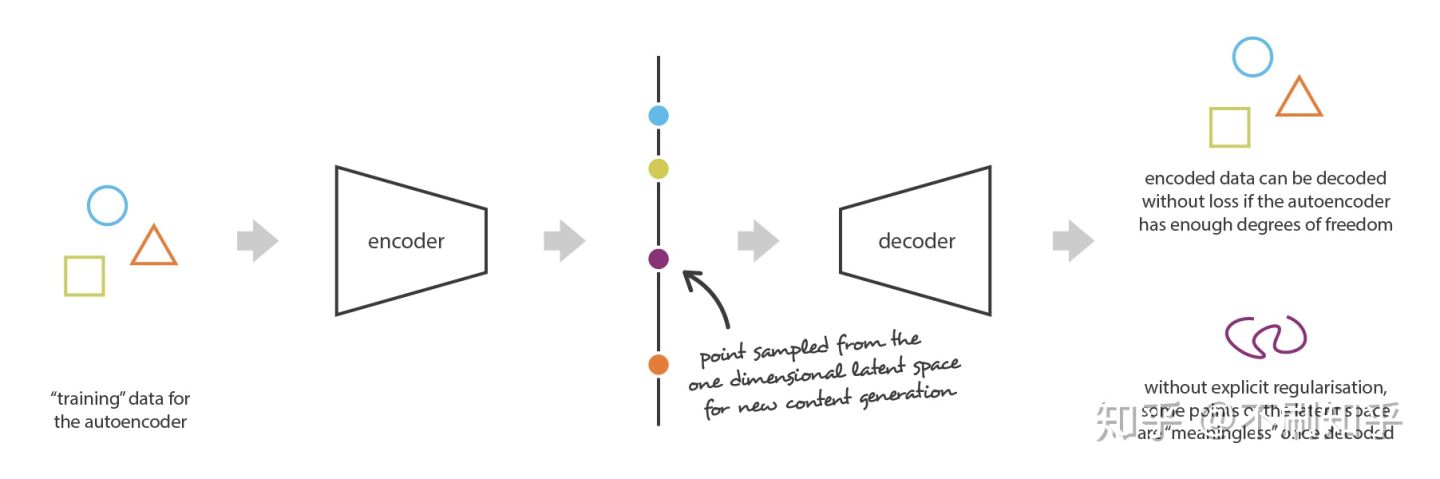
线性降维的典型代表是PCA，它是通过正交投影获得降维向量，但是值得指出的是，线性自编码器同样可以选择不同的基向量来描述最佳隐空间，我们最终获得的新特征不必是相互独立的（神经网络没有正交约束）。

**自编码器用于内容生成的局限性**

通过对自编码器的熟悉，自然会产生一个想法：”自编码器和内容生成之间有什么联系？“ 但事实是，我们虽然训练好了一个自编码器以及一个隐空间，仍然没有办法产生任何新内容。如果隐空间足够规则，我们可以从隐空间中随机取一个点并将其解码来获得新的内容，就像生成对抗网络的生成器一样。

但是，**自编码器的隐空间的规则性是一个难点**，它取决于初始空间中数据的分布，隐空间大学和编码器的结构。因此，很难先验地确保编码器智能地组织隐空间。

为了说明这一点，让我们看一个示例。该示例中，我们描述了一种强大的编码器和解码器，可以将任何N个初始训练数据放到实轴上（每个数据点都被编码为实值）并可以没有任何损失地解码。在这种情况下，自编码器的高自由度使得可以在没有信息损失的情况下进行编码和解码（尽管隐空间的维数较低）但会导致**严重的过拟合**，这意味着隐空间的某些点将在解码时给出无意义的内容。尽管这种精挑细选的一维例子太过极端，但可以注意到自编码器的隐空间规则性问题是普遍的，值得特别注意



隐空间的隐变量之间缺乏结构是很正常的。的确，在自编码器的训练任务中，没有什么东西保证能够得到这种结构：**自编码器仅以尽可能少的损失为目标进行训练，而不管隐空间如何组织**。因此，如果我们对架构的定义不小心，那么在训练过程中，网络很自然地会利用任何过拟合的可能性来尽可能地完成其任务……除非我们明确对其进行规范化。

因此，为了能够将我们的自编码器的解码器用于生成目的，我们必须确保隐空间足够规则。**获得这种规律性的一种可能方案是在训练过程中引入显式的正规化**。因此，**变分自编码器**（VAE）可以定义为一种自编码器，其训练经过正规化以避免过度拟合，并确保隐空间具有能够进行数据生成过程的良好属性

为了使生成过程成为可能，我们期望隐空间具有规则性，这可以通过两个主要属性表示：**连续性**（continuity，隐空间中的两个相邻点解码后不应呈现两个完全不同的内容）和**完整性**（completeness，针对给定的分布，从隐空间采样的点在解码后应提供“有意义”的内容）。这就是VAE模型的独特之处，本文不做详述。

**综上，自编码器的隐空间由于缺乏规则性，无法用于内容生产。所以隐空间中的隐变量仅能作为提取了输入主要特征的编码向量。**

**1.问题描述**

在光伏发电等环境下，一般每隔一段时间会检测设备，然后输出一组检测数据。技术人员会用检测数据分析设备是否运转正常，来决定是否需要维护设备。因此，异常检测在工程中显得异常重要。在实际生产中，异常数据可能是无规律、随时出现，因此，我们需要设计一个模型及时，准确的实时检测出异常数据。

异常检测的方法已有很多，传统的方法有单类别的SVM，K-means等。随着深度学习的发展，也有很多学者尝试采用自编码器（AE）进行异常检测。自编码器的检测方法如下：

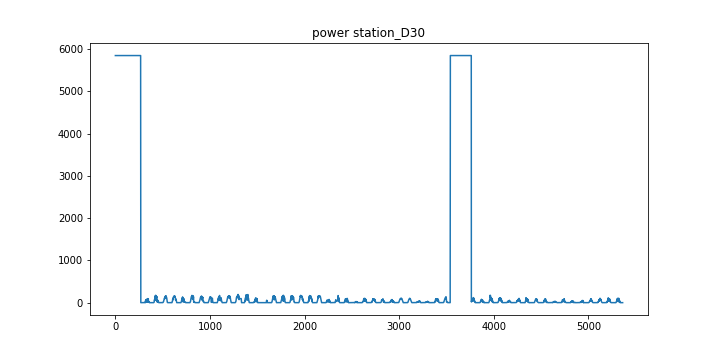
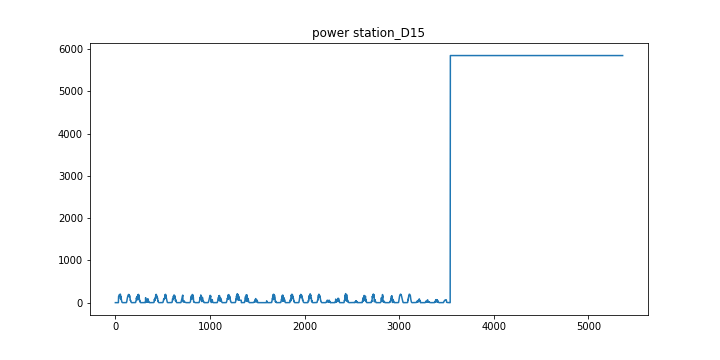
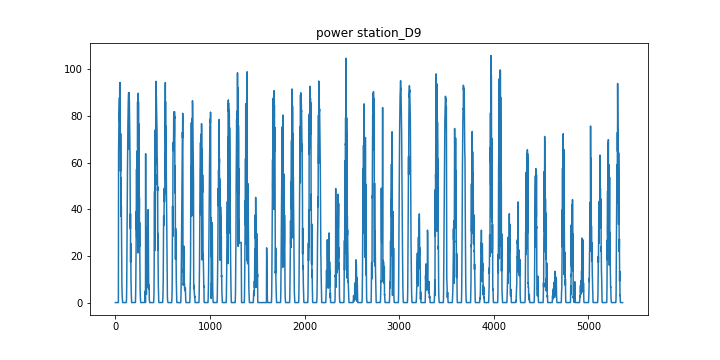
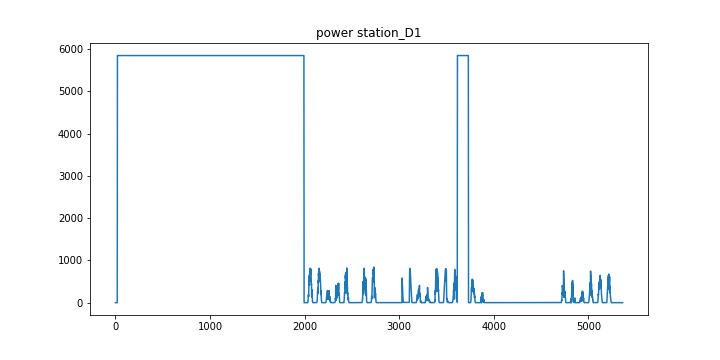
首先，从原始数据中人为地找出正常数据和异常数据；然后，用正常数据训练自编码器，使得重构数据与原始数据尽可能近似，这样做旨在使模型学习到正常数据的隐空间分布，而异常数据则是这个隐空间的离群点，重构得到的数据则会偏离输入的异常数据。此时，重构数据与异常数据的重构误差会相比较变得较大。这样，通过设立阈值，就可以将异常数据分离出来。

上述其实是理想的状态，即人为地可以将异常数据和正常数据找出，并用正常数据训练网络。但是，如果我们考虑一个更复杂的情形：我们无法从庞杂的数据中分辨出异常数据，或者这个过程是极其耗时耗力的（通常，这个情形更符合实际的生产生活），那么上述方法显得过于简单。因此，如何让模型自行地、不借助人力地分离正常数据和异常数据显得非常重要。

以下模型设计均建立在自编码器框架下。

**2.数据描述**

实验数据采用园区光伏发电数据。该数据集每隔15分钟记录一条，共5363条（删除完缺失数据），数据的属性包括：年，月，日，具体时间，和31个发电站数据。每个发电站的数据如下图（选取部分）：



我们可以发现一些的数据会出现5850（这个数字远大于其他数据，视作异常），因此定义：一条记录中，如果至少一个发电站的数据出现5850，则视作这条记录为异常数据。这样我们人为对数据基于上述定义赋予标签，并在模型训练和预测后，计算检测的准确率。

**3.模型搭建**

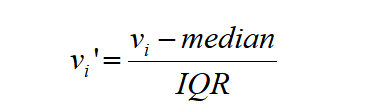
（1）基于MLP的自编码器：

自编码器的结构如图，输入31维，隐变量3维。

**4.实验结果**

（1）预处理

由于每个发电站的数据范围不一致，因此采用归一化。因为数据中存在大量的异常数据，因此采用Robust scaler将数据归一化。robust scaler的数学公式为：



median:表示中位数；IQR:表示四分位距。

（2）数据分割

首先将归一化后的数据集前3500条作为训练数据集，剩下的作为预测数据集，然后进行训练预测。而且数据在批次化时，没有将数据打乱，目的是保留数据本身的顺序，有利于训练。

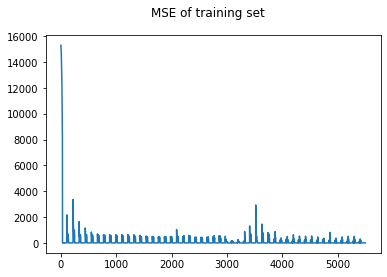
（3）网络训练：

网络损失函数采用MSE，优化器为Adam，训练的网络参数如下：

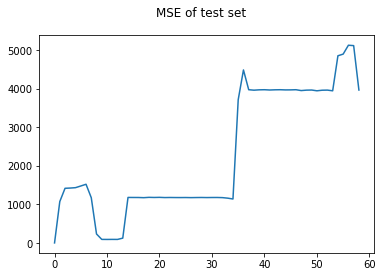
epoch = 50

batch size = 32

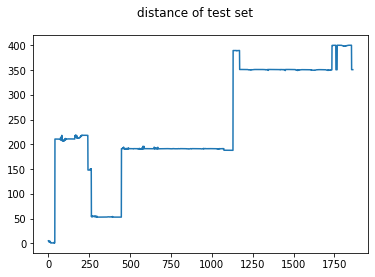
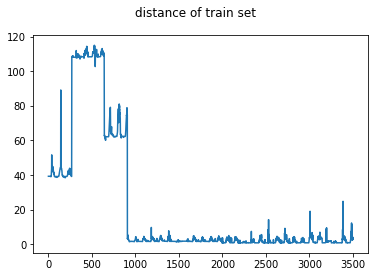
训练误差如图所示：



测试误差如图所示：



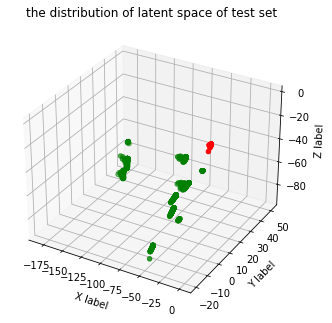
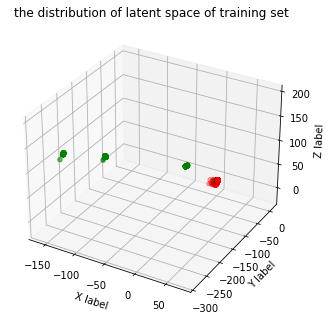
然后计算每条数据与重构的近似程度，采用欧氏距离衡量，测量结果如下：



如图，我们将距离大于50的设为异常数据，由此我们计算检测的准确率 ，结果如下:

训练集的准确率为69.0%，测试集的准确率为100%。

训练数据的隐空间分布图:



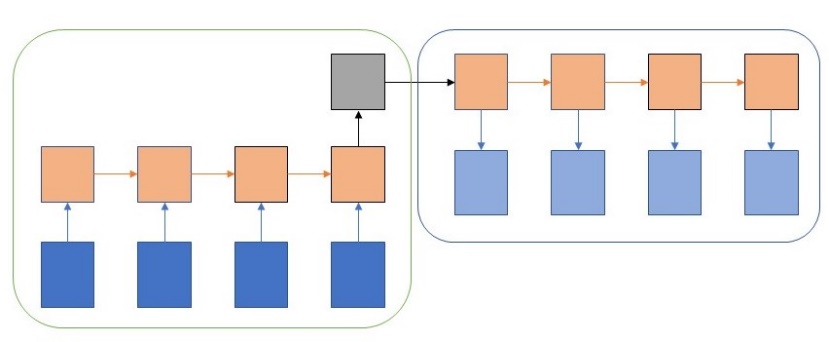
从图中的看出异常数据和正常数据的隐空间分布是分离的，但是正常数据的分布过于分散。

【解释】测试集准确率100%的原因是：测试集选取的是数据集的最后，正常数据和异常数据排列整齐，而且多为异常数据所以检测效果很好。

问题：如何提高训练集的准确率？（提升方法）

【模型提升】：在上述模型设计中，没有将数据的序列性信息在网络中体现，所以对于时间序列等异常检测使用LSTM Autoencoder进行异常检测。

模型框架为：



**问题**

考虑一下我们需要探究的问题，我想到的有：

1）通过自编码训练，模型学到的是什么性质的映射关系？目的是不是构成某种意义下的分类？

非线性映射。目的不是构成分类，**自编码器仅以尽可能少的损失为目标进行训练，而不管隐空间如何组织。**

2）模型的区分效果怎么样？区分能力怎么样？从正常到异常的边界是这样的？

正常到异常的边界是难以计算的。原因如下：

设输入：, 输出为. 在上述计算中通过阈值k来确定异常数据，公式为：

现在有两组数据,分别是正常和异常：

那么在的连线上有一个点会使得

这个点可以表达为:

所以我们只需求出即可，但是由于是非线性映射，即：

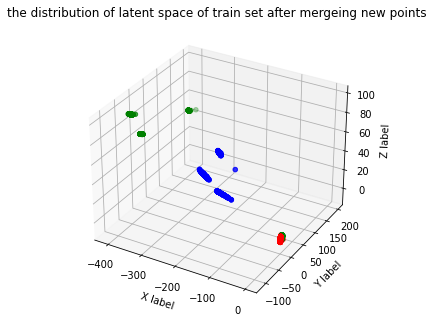
所以是难以求解的。

解析法失效，可以使用遍历的方式，取一个近似的值。

另外，上述的结论中说到，自编码器的隐空间是缺乏规则的，所以正常与异常的边界也是难以用规则的，统一的几何图形划分。

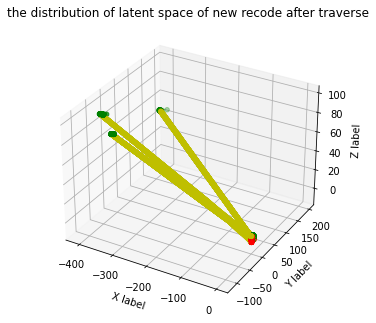
**边界可视化**

为了观察正常数据与异常数据在编码空间的阈值边界，进行了以下可视化：



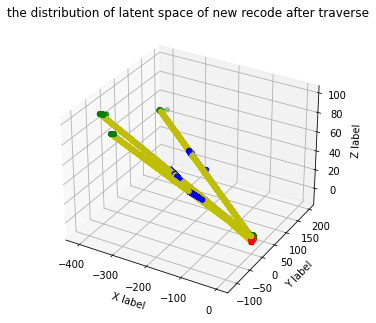
图a

如图a所示，绿色点代表正常数据的编码向量，红色点代表异常数据的编码向量，蓝色点表示阈值边界点（此模型设置的阈值为50）。



图b

如图b所示，绿色点和红色点分别代表正常和异常。我们通过上述线性插值的方式，从正常数据出发到异常数据结束，采样1000个点，然后进行编码，得到的编码向量在图b中显示为黄色点。我们发现，采样点从正常编码向量到异常向量是线性连续变化的。



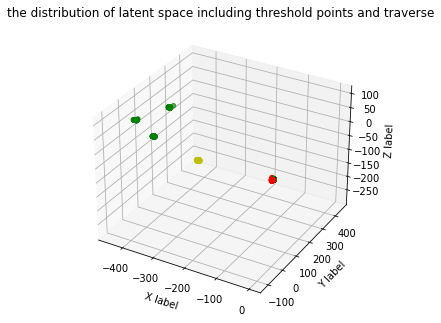
图c

如图c所示，将图a,图b中的阈值边界点和所有的插值点表示在一张图中。（阈值边界点是从插值点中选取的距离等于阈值的点，用蓝色表示以示区分）。

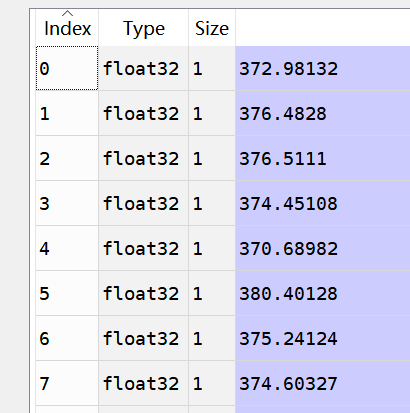
从上图可以看出：1）原始空间的数据映射的编码空间是非连续的，相似的数据聚成簇。2）原始空间线性变化的点在编码空间也是线性变化的。3）线性插值得到的阈值点在正常编码与异常编码连线的某处也聚成簇。

如果是不在数据集中数据，输入编码器会产生什么结果呢？

我们用多元高斯分布，随机生成1000组31维的数据，输入编码器，得到编码向量：



如图所示，黄色簇为高斯分布（loc = 100 ,std = 1）数据的编码向量。我们由此可以得出结论：在原数据距离相近的点，在编码之后也会聚成簇。



图上是计算的部分距离，发现均大于50, 所以说按照实验设定，这些点均为异常点。但是如果不提前设定，从图中是无法得知是不是异常值。所以可以得出结论：编码器仅是将原始数据中近似的数据在编码空间中聚成簇，在低维空间显示了不同类数据的差异性，在3维以下可以可视化展示。

3）编码输出可以怎样利用？

降维后的编码向量可以用于需要低维数据（数据可视化，数据存储，降低计算复杂度…）的场景中。

4）网络超参数、网络节点数、网络结构、网络类型，会有怎样的影响，有怎样的差别？

如果不是对隐空间进行规范化的设计，其他设计的影响，仅是针对损失函数是不是更小了，学到的隐变量是不是代表数据更多的信息了。

5）训练损失函数，有怎样的影响与差别？

自编码器的损失设计是重构误差，不同的设计仅是不同的距离表示方式，没有很大的差别。

另，在这个具体问题中，31个站点而不是每一站点独立处理，是要利用个别站点异常其他大多站点正常，有可能明显处分出故障站点。最终应该是还要确定出具体故障站点的。

31个站点，每时刻判断每个站点是否异常？

1. 可以建立31个自编码模型，
2. 建立一个自编码模型，对31个输入，重构，每个输入检验重构误差，阈值判别

？自编码应用中，是否输入与输出（重构）必须是同维度的？

必须

在上面模型2）中，每一时刻有31个输入，如果将时间序列作为输入，31个时间序列，自编码模型如何构成？

？对时间序列，一维或高维，自编码模型如何构成？重构什么？只是重构最后一个时刻的输出吗？

思路不变，重构的是每一个时刻的输入。输入多少时刻，重构多少时刻。

分阶段建模处理：

1. 先分离5850  
   自编码模型1：可以识别定位5850异常
2. 去除5850之后，在分析异常  
   自编码模型2