

**汇总报告**

|  |  |
| --- | --- |
| **课程名称：** | **智能软件项目实训** |
| **学生姓名：** | **孙武杰** |
| **学生学号：** | **201630665632** |
| **学生班级：** | **软件工程3班** |
| **开课学期：** | **2018-2019学年第二学期** |

**软件学院**

**2019年6月**

目录

[一、实训任务 3](#_Toc11939435)

[二、相关概念 4](#_Toc11939436)

[三、实验内容 22](#_Toc11939437)

# 一、实训任务

选定一个振动或者加速度类型数据集，使用深度置信网模型进行特征级融合。

**过程**

调研深度置信网络模型在机械故障诊断中应用的国内外发展现状，至少复现出近期发表的一篇论文。

**要求**

使用深度置信网，对选定的数据集进行数据融合处理，得到最终融合诊断结果。其中，可能会有数据特征提取处理，该步骤可以按照已有文献中给定的常用方法实现。

**附加**

如提前完成，可以深入调研深度置信网在机械故障诊断中的应用，并对比相应的优化方法，提出自己的优化思路，或者在诊断模型上的优化创新。

**参考数据集**

轴承故障数据集（美国凯斯西储大学电气工程实验室提供的滚动轴承振动信号数据） [http://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website](http://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website%20)。

**完成情况**

使用MATLAB（用于数据预处理）和Python（用于特征提取、训练DBN），复现《The optimized deep belief networks with improved logistic Sigmoid units and their application in fault diagnosis for planetary gearboxes of wind turbines》及《Improved classification with semi-supervised deep belief network》两篇论文，并撰写一篇小论文《Bearing Fault Diagnosis Based on Deep Belief Network》，所有的文档（包括报告、代码、论文等）可见于<https://github.com/wjsunscut/Intelligent-Software-Project-Training>。

# 二、相关概念

在本次实训中，主要内容分为三个部分。第一部分，数据预处理；第二部分，特征提取；第三部分，训练DBN。对于我个人而言，本次实训的主要任务为特征提取和训练DBN，因此数据预处理部分使用了MATLAB自带的函数，但依然需要对其原理进行了解。因此，在下面的内容中，我将分别对数据预处理、特征提取、训练DBN的相关概念进行记录和学习。

1. **数据预处理**

在数据预处理阶段，主要使用了小波包分解与广义回归神经网络。

**小波包分解**

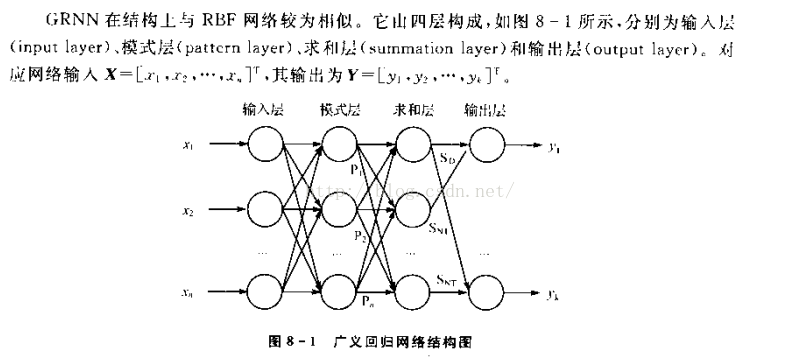
*小波包分解（wavelet packet decomposition）也可称为小波包（wavelet packet）或子带树（subband tree）及最佳子带树结构（optimal subband tree structuring）。其概念是用分析树来表示小波包，即利用多次叠代的小波转换分析输入讯号的细节部分。*

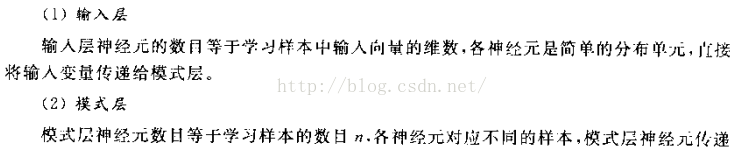
*从函数理论的角度来看，小波包分解是将信号投影到小波包基函数张成的空间中。从信号处理的角度来看，它是让信号通过一系列中心频率不同但带宽相同的滤波器。*

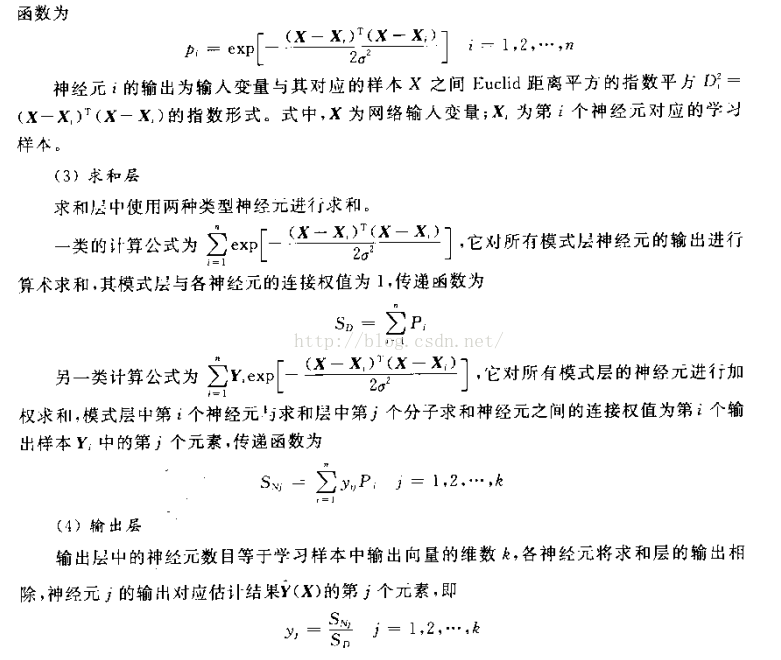
**广义回归神经网络**

*在数学建模领域，径向基函数网络（Radial basis function network，缩写 RBF network）是一种使用径向基函数作为激活函数的人工神经网络。径向基函数网络的输出是输入的径向基函数和神经元参数的线性组合，广义回归神经网是基于径向基函数网络一种改进。广义回归神经网络也可以通过径向基神经元和线性神经元来设计。*

*GRNN是RBF的一种改进，结构相似。区别就在于多了一层加和层，而去掉了隐含层与输出层的权值连接（对高斯权值的最小二乘叠加）。*







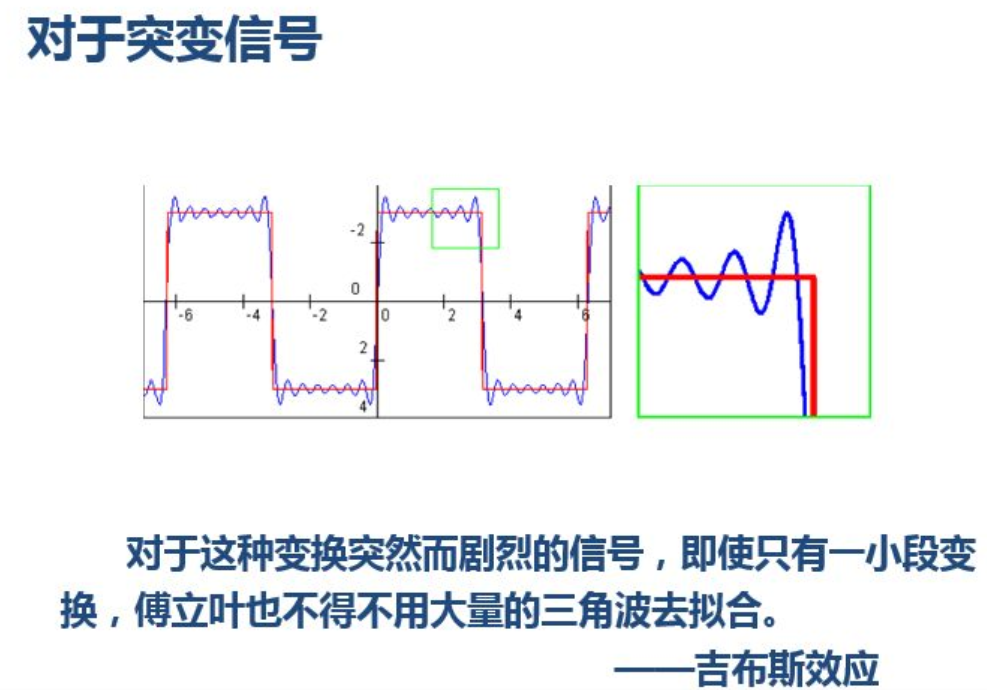
由于小波包应用对存储花费大，而神经网络在自主学习、非线性映射、容错能力等方面显得更胜一筹，因此将二者结合。

小波包分解由小波分解优化而来，且与傅里叶变换息息相关，因此需要对相关概念进行了解。

**傅里叶变换**

<https://www.cnblogs.com/h2zZhou/p/8405717.html>对傅里叶变化进行了说明。

但傅里叶变换处理非平稳信号有天生缺陷。它只能获取一段信号总体上包含哪些频率的成分，但是对各成分出现的时刻并无所知。在傅里叶变换中，积分是从负无穷到正无穷的，所以积分在所有时间类都是有效的。意思就是说，无论什么时候频率分量发生了改变都会全局性的影响积分的结果，同样的道理也适用于离散型的傅里叶变换。因此，傅里叶变换不适合分析非平稳信号（分布参数或者分布律随时间发生变化的信号）。因此，提出了短时傅里叶变换。



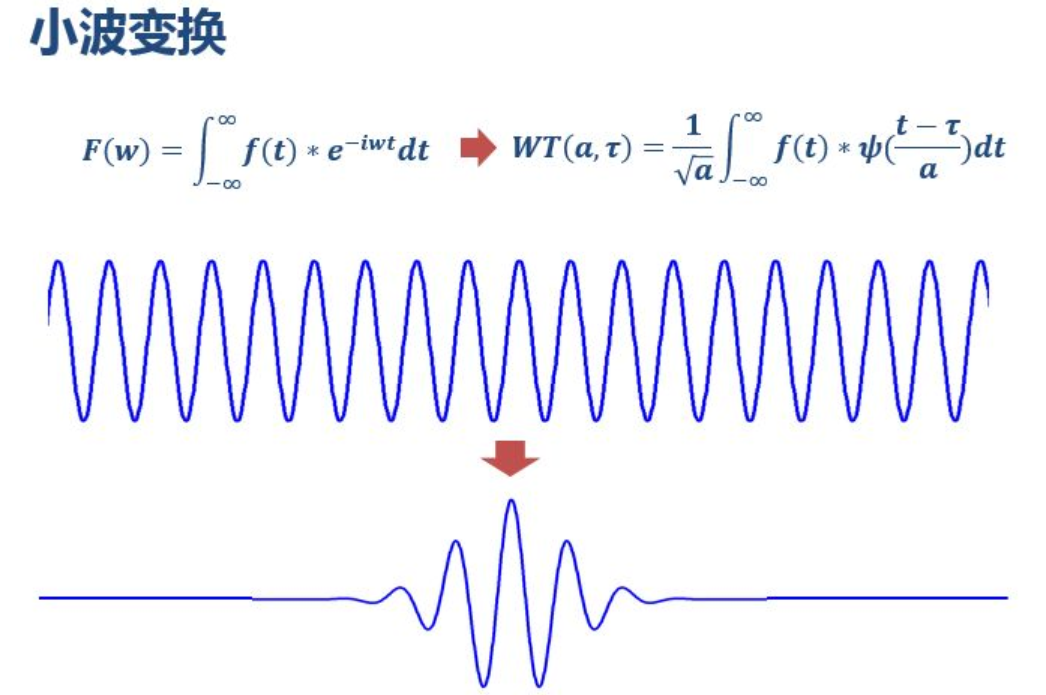
**短时傅里叶变换**

在短时傅立叶变换中，信号被分为足够小的片段，这些片段的信号都可以看成平稳信号，克服了傅里叶变换在处理非平稳信号时的缺陷。这里需要用到窗函数，但其宽度不容易确定。为了获得信号的平稳性，我们必须要有一个宽度足够短的窗函数，在这个很短的时间内，信号是平稳的。窗口越短，时间分辨率越高，信号的稳定性越高，但是频率分别率却越来越低。

**小波变换**

*小波就是小区域、长度有限、均值为0的波形。小波变换就是选择适当的基本小波或母小波ψ(t)，通过对基本小波的平移、伸缩而形成一系列的小波，这簇小波作为基可以构成一系列嵌套的（信号）子空间，然后将欲分析的信号（例如图像）投影到各个大小不同的（信号）子空间之中，以观察相应的特性。这样，就相当于我们用不同的焦距去观察一个物体，可从宏观到微观，从概貌到细节观察得十分详尽。*

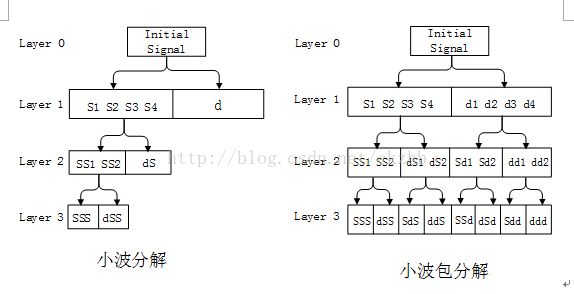
平移、伸缩是小波变换的一个特点，因而可以在不同的频率范围，不同的时间（空间）位置对信号进行各种分析，通过这种多分辨率分析，在高频信号中获得一个好的时间分辨率和较差的频率分辨率，低频信号中获得较好的频率分辨率和较高的时间分辨率 ，明显地解决了傅里叶变换应用与非平稳信号的弊端。



小波变换有两个变量：尺度a（scale）和平移量τ（translation）。尺度a控制小波函数的伸缩，平移量τ控制小波函数的平移。尺度a就对应于频率（反比），平移量τ就对应于时间。

**小波包分解**

小波包是为了克服小波分解在高频段的频率分辨率较差，而在低频段的时间分辨率较差的问题的基础上而提出的。它是一种更精细的信号分析的方法，提高了信号的时域分辨率。

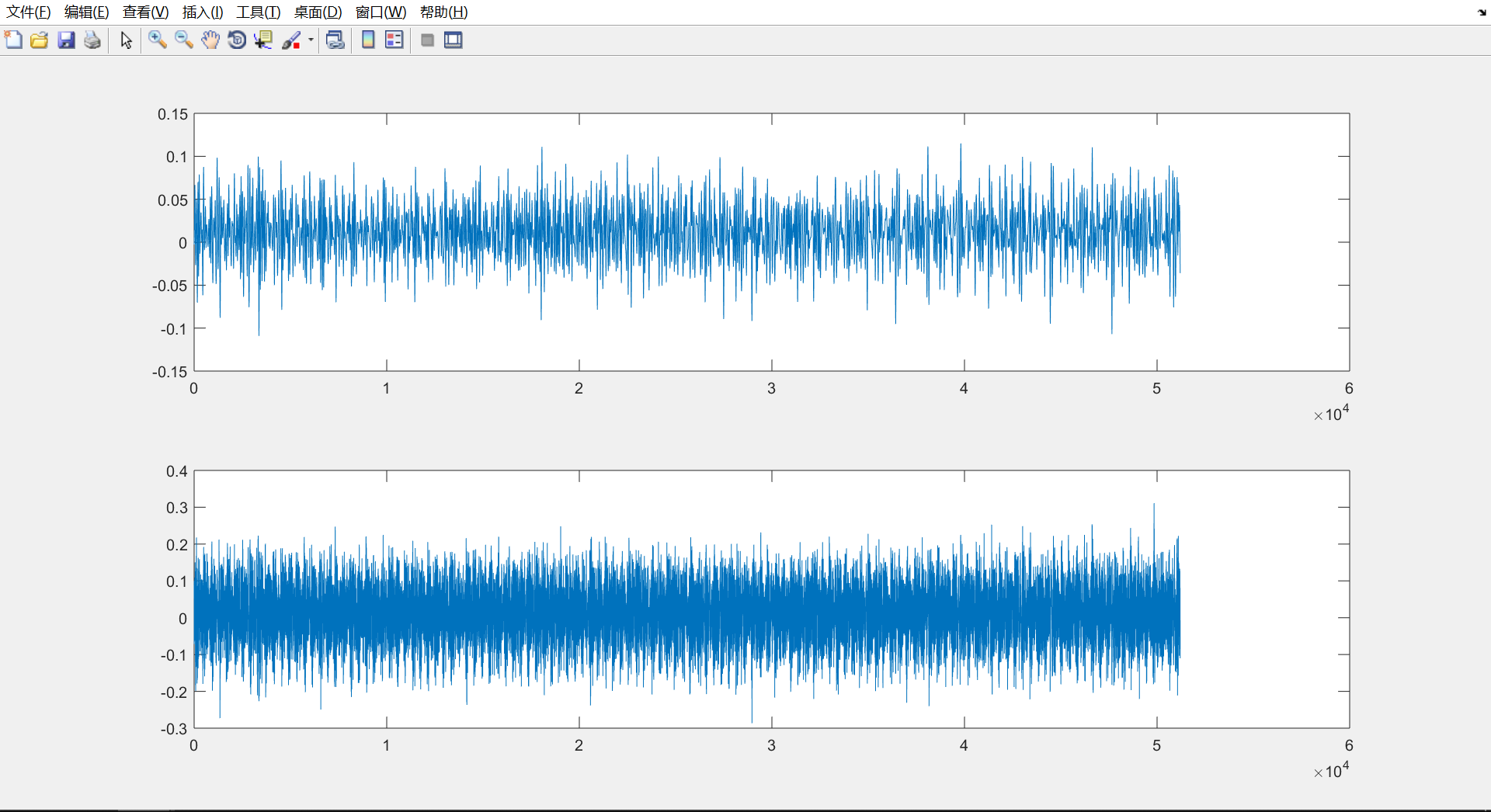


**小波包分解使用方法**

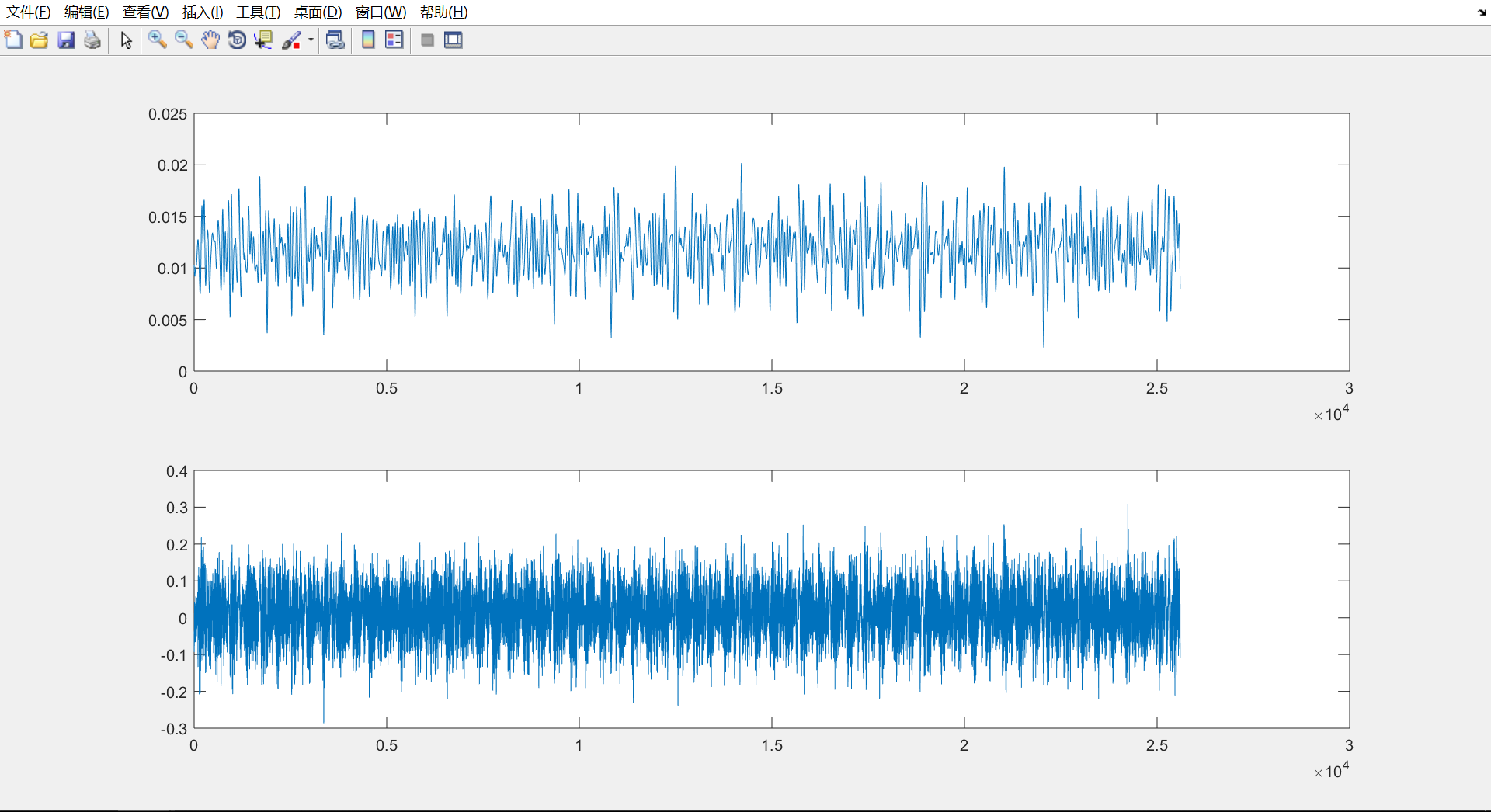
MATLAB中自带的与小波包分解相关的wavedec、appcoef、detcoef、wbmpen、wdencmp函数，可以快速方便地实现小波包分解，具体可见代码<https://github.com/wjsunscut/Intelligent-Software-Project-Training/blob/master/Code/preprocessing.m>。

**原本实现方法**

原本的降噪过程使用了GRNN，参考文献为《基于小波包-神经网络的MEMS加速度计零漂补偿》，但因为较难验证其正确性，所以最后还是选用了MATLAB自带的函数以实现降噪，附上原本的降噪结果。



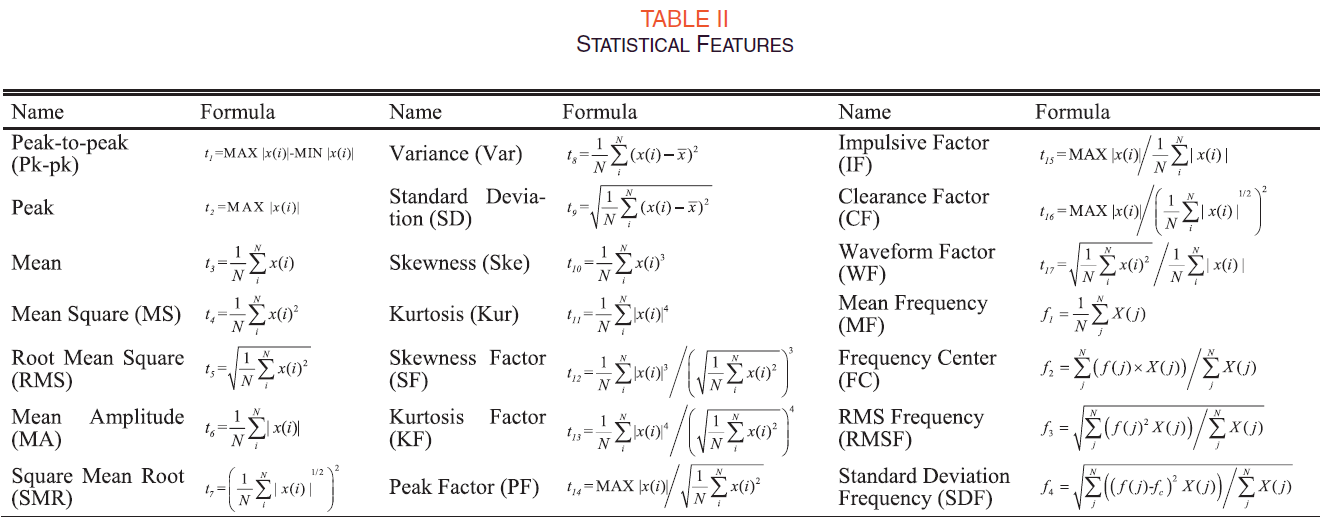
**降噪结果，50\*1024个样本**



**GRNN结果，前25\*1024用于学习，后25\*1024用于检验**

1. **特征提取**

在降噪完成后，可以将N个连续的加速度数据组合（在下面的实验内容中取N=2048），构成一条数据，并对每条数据提取特征。根据什么来提取特征可以网上查找，在本次实验中，主要是根据下表的前17项提取了特征（即时域信号的特征），并未对频域信号提取特征。提取的特征越多，越能区分相应的故障类型。还根据论文《Bearing performance degradation assessment using long short-term memory recurrent network》附加了波形熵特征。



1. **训练DBN**

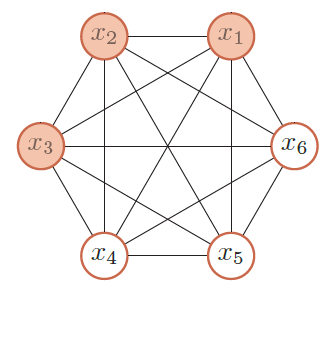
DBN由RBM和分类器构成。首先需要对RBM进行了解。

**玻尔兹曼分布**

能量收敛到最小后，热平衡趋于稳定，也就是说，在能量最少的时候，网络最稳定，此时网络最优。此时模型能量收敛到的分布，即为玻尔兹曼分布（吉布斯分布）。

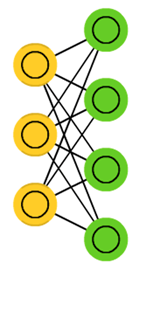
**玻尔兹曼机（Boltzmann Machines，简称BM）**

玻尔兹曼机（BM）是由随机神经元全连接组成的反馈神经网络，且对称连接，由可见层、隐层组成，BM可以看做是一个无向图，如下图所示：



其中，x1、x2、x3为可见层，x4、x5、x6为隐层。其缺点为训练时间长。

**受限玻尔兹曼机（Restricted Boltzmann Machines，简称RBM）**

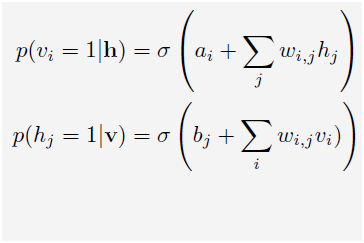


**RBM结构**

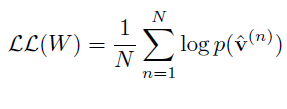
受限玻尔兹曼机（RBM）同样具有一个可见层，一个隐层，但层内无连接，层与层之间全连接，节点变量仍然取值为0或1，是一个二分图。也就是将玻尔兹曼机（BM）的层内连接去掉，对连接进行限制，就变成了受限玻尔兹曼机（RBM），这样就使得计算量大大减小，使用起来也就方便了很多。

受限玻尔兹曼机（RBM）的特点是：在给定可见层单元状态（输入数据）时，各隐层单元的激活条件是独立的（层内无连接），同样，在给定隐层单元状态时，可见层单元的激活条件也是独立的。

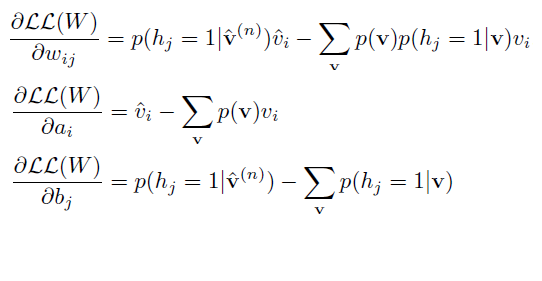
跟玻尔兹曼机（BM）类似，根据玻尔兹曼分布，可见层（变量为v，偏置量为a）、隐层（变量为h，偏置量为b）的概率为：



训练样本的对数似然函数为：

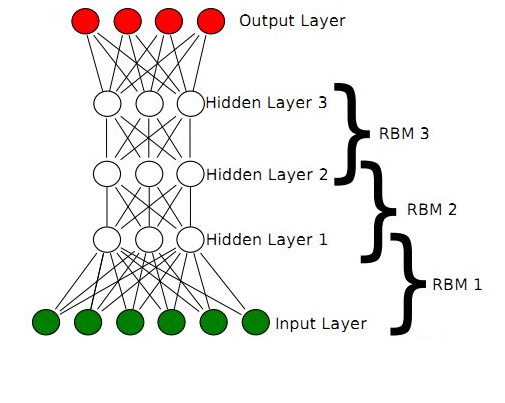


求导数：



RBM的本质是非监督学习的利器，可以用于降维（隐层设置少一点）、学习提取特征（隐层输出就是特征）、自编码器（AutoEncoder）以及深度信念网络（多个RBM堆叠而成）等等。

**深度信念网络（Deep Belief Network，简称DBN）**



**DBN结构**

此处参考了多篇文章

<https://blog.csdn.net/u013631121/article/details/76652647>

<https://blog.csdn.net/OPPOA113/article/details/42486833>

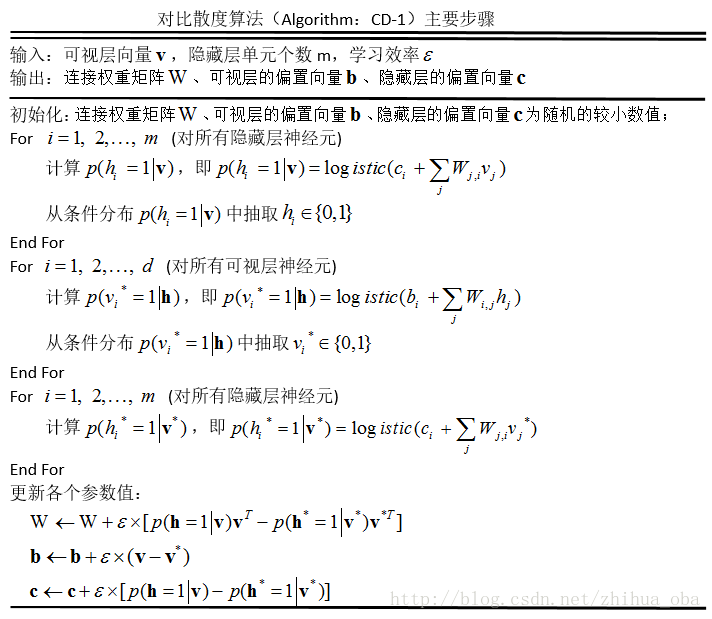
<https://blog.csdn.net/Rainbow0210/article/details/53010694>

<https://blog.csdn.net/zhihua_oba/article/details/69487730>

<https://www.cnblogs.com/pythonlearing/p/9979161.html>

按照我的理解，DBN其实就是在进行分类之前，首先使用RBM进行预训练，从而优化了初始的权重、偏置，使在进行训练时有一个较好的初始起点，避免因初始化过差而陷入局部最优解，相比传统方法更能找到全局最优解，从而提升性能。

在训练过程中，首先将可视层向量值映射给隐含层单元；然后可视层单元由隐含层单元重建；这些新可视层单元再次映射给隐含层单元，这样就获取新的隐含层单元。执行这种反复步骤叫做吉布斯采样。吉布斯采样的计算是耗时的，而对比散度算法的提出使得通过一次吉布斯采样，就能取得较好的效果。主要步骤如下。这样子，我们就能快速地初始化相应变量，使DBN具有较好的效果。具体的推导步骤在SSDBN内。



**DBN与传统BP算法对比**

**传统BP算法的缺陷：**

（1）需要为训练提供一个有标签的样本集；

（2）学习过程较慢；

（3）不适当的参数选择会导致学习收敛于局部最优解。

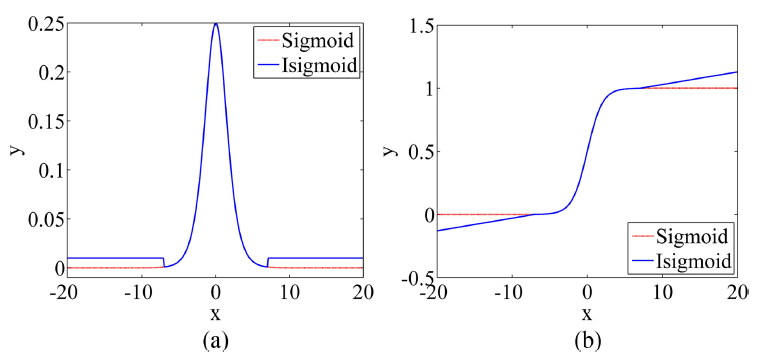
**DBN特点：**

（1）通过一个非监督贪婪逐层方法去预训练获得生成模型的权值；

（2）Gibbs采样使训练时间显著减少，因为只需要单个步骤就可以接近最大似然学习。在预训练后，DBN可以通过利用带标签数据用BP算法去对判别性能做调整。DBNs的BP算法只需要对权值参数空间进行一个局部的搜索，这相比前向神经网络来说，训练是要快的，而且收敛的时间也少；  
（3）RBM 网络训练模型的过程可以看作对一个深层 BP 网络权值参数的初始化，使DBN 克服了BP 网络因随机初始化权值参数而容易陷入局部最优和训练时间长的缺点。

**ISigmoid**

**ISigmoid就是将Sigmoid与LReLU结合，综合二者的优势。**



**a图为二者导数对比，b图为二者对比**

**选择Sigmoid函数作为激活函数的优势**

（1）可以引入非线性；

（2）容易求导；

（3）可以将实数压缩至(0,1)。

**梯度消失**

*神经网络主要的训练方法是BP算法，BP算法的基础是导数的链式法则，也就是多个导数的乘积。而Sigmoid的导数最大为0.25，且大部分数值都被推向两侧饱和区域，这就导致大部分数值经过Sigmoid激活函数之后，其导数都非常小，多个小于等于0.25的数值相乘，其运算结果很小。随着神经网络层数的加深，梯度后向传播到浅层网络时，基本无法引起参数的扰动，也就是没有将loss的信息传递到浅层网络，这样网络就无法训练学习了。这就是所谓的梯度消失。*

**Sigmoid与ReLU对比**

1. Sigmoid函数在压缩数据“幅度”方面有优势，对于深度网络，使用Sigmoid函数可以保证数据幅度不会有问题，幅度稳住后就不会有太大失误。
2. Sigmoid存在梯度消失的问题，在反向传播上有劣势。
3. ReLU不会对数据做幅度压缩，所以随着深度网络层数加深，数据的幅度会越来越大，最终影响模型的表现。
4. ReLU在反向传导时，能够将梯度信息“完完全全”地传递到浅层网络。

ReLU和LReLU不适用于DBN，因为它们无法充分发挥RBM预训练的作用。

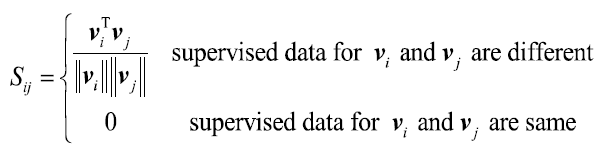
因此在文章中结合了LReL和Sigmoid，在|x|>α时，fI’>fS’，权重变化值又与f’有关，则权重变化更快，收敛更快。同时减少了梯度消失的情况。

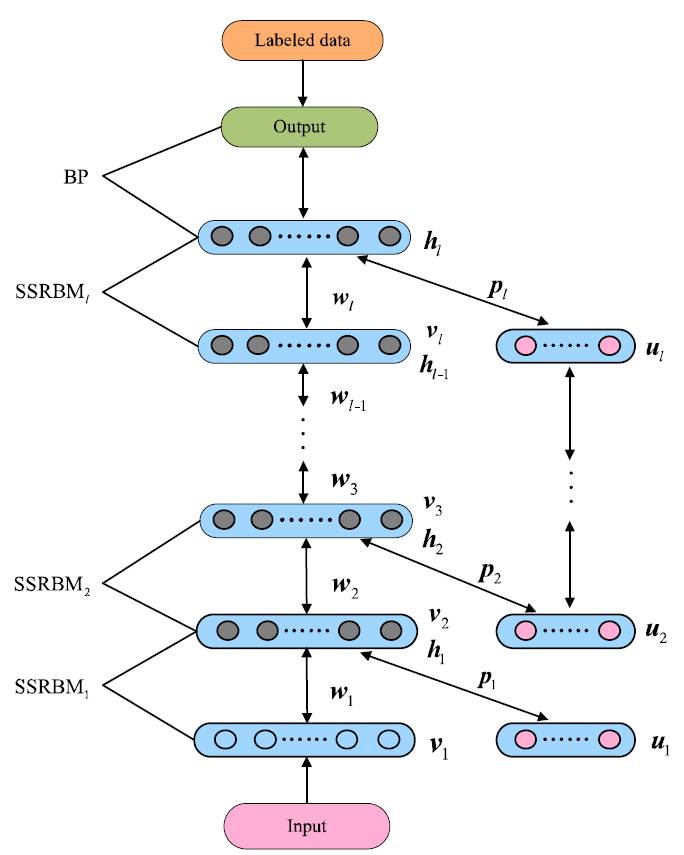
**纠错**

文中公式15的第三条公式中的Sigmoid(a)应为Sigmoid(-a)。

**SSDBN**

SSDBN就是在DBN的RBM中，加入了半监督学习的内容。其通过判断可见层单元数据是否标签相同，从而构造矩阵S，对其进行奇异值分解，提取出S中重要的特征，构成层u，与层v、层h一起构成了SSRBM。层u与层v的单元数应该一致。





**SSDBN结构**

**奇异值分解**

SVD可以理解为：将一个比较复杂的矩阵用更小更简单的3个子矩阵的相乘来表示，这3个小矩阵描述了大矩阵重要的特性，可使用Python中的np.linalg.svd(input)。

**公式推导**

根据文章

<https://blog.csdn.net/itplus/article/details/19168937>

<https://blog.csdn.net/itplus/article/details/19168989>

<https://blog.csdn.net/itplus/article/details/19169027>

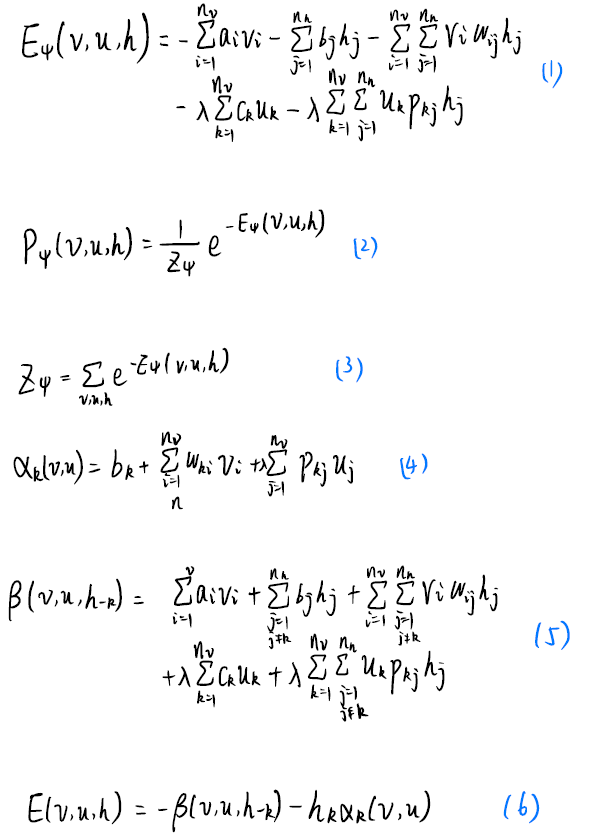
<https://blog.csdn.net/itplus/article/details/19207371>

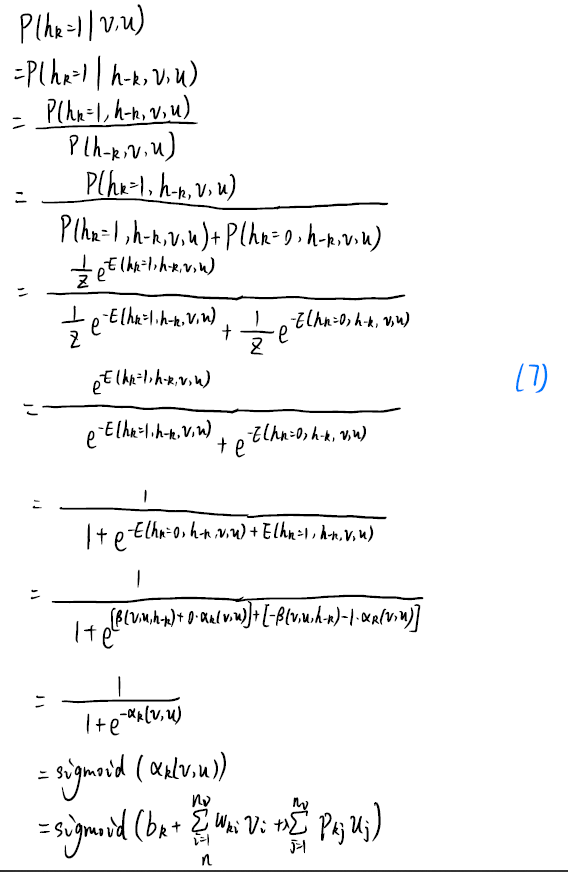
<https://blog.csdn.net/itplus/article/details/19408143>

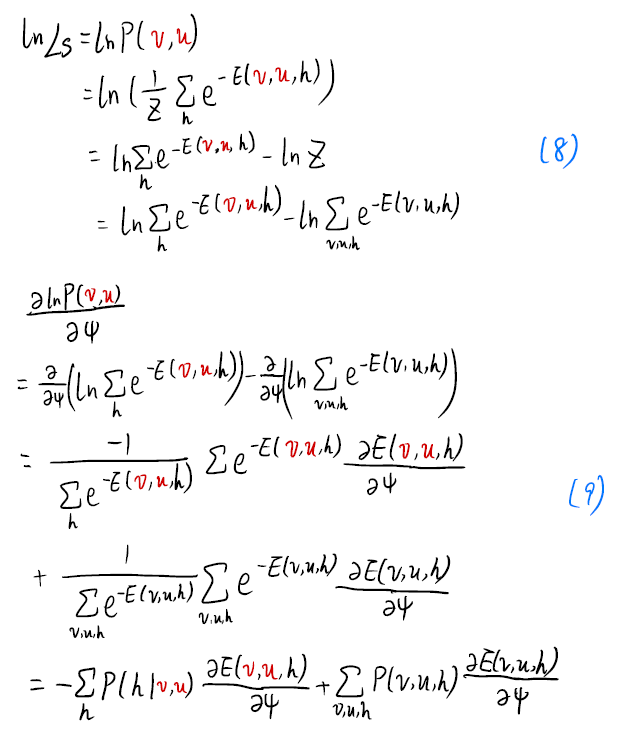
<https://blog.csdn.net/itplus/article/details/19408701>

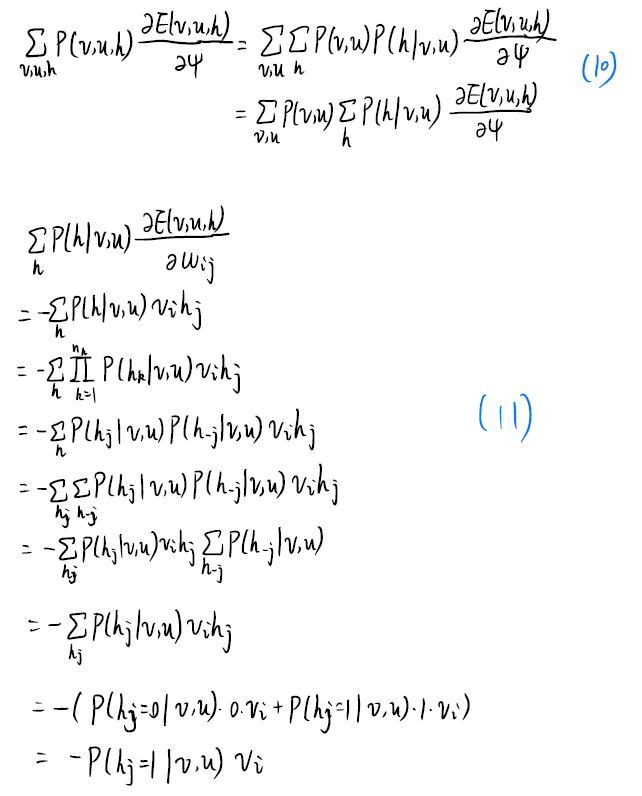
<https://blog.csdn.net/itplus/article/details/19408773>

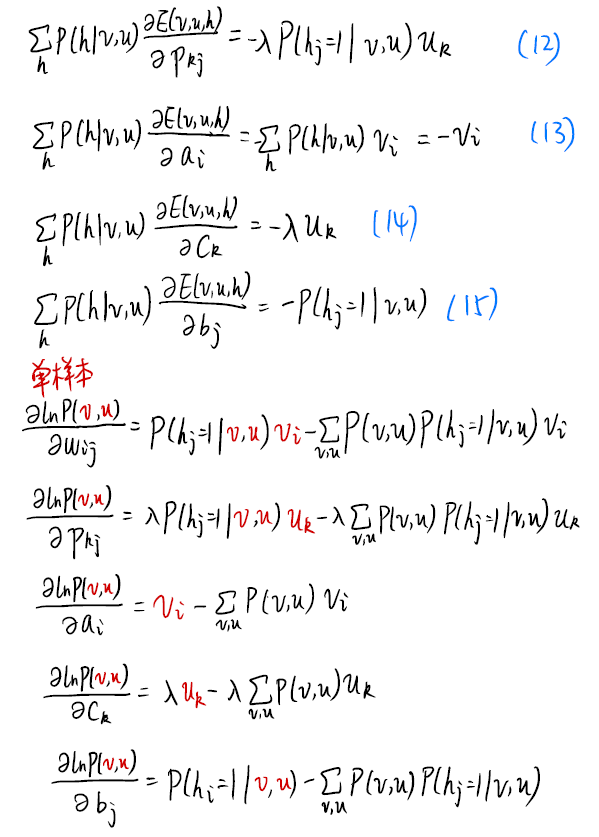
对文章内涉及的公式部分进行推导，文章部分公式错误，在实验中，参考的是该报告中推导得出的公式。

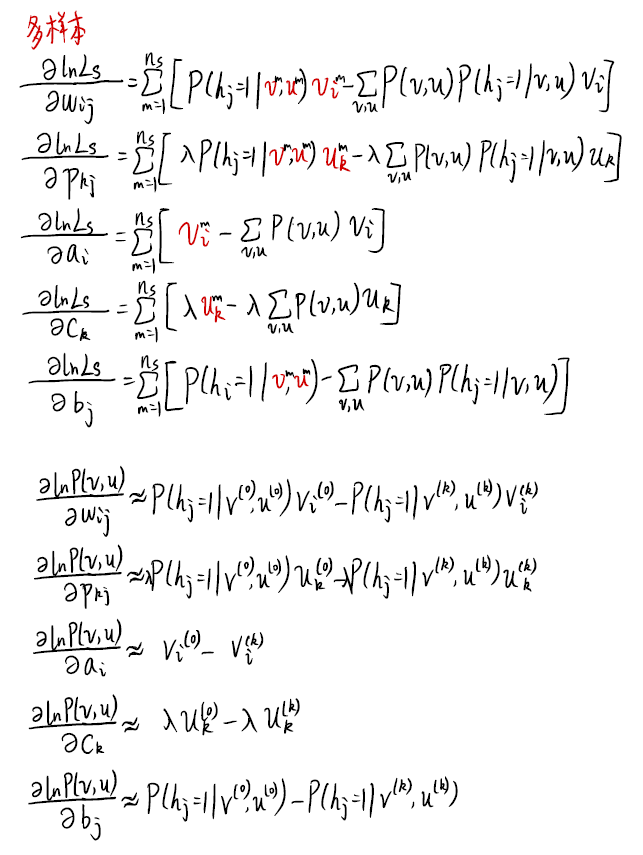










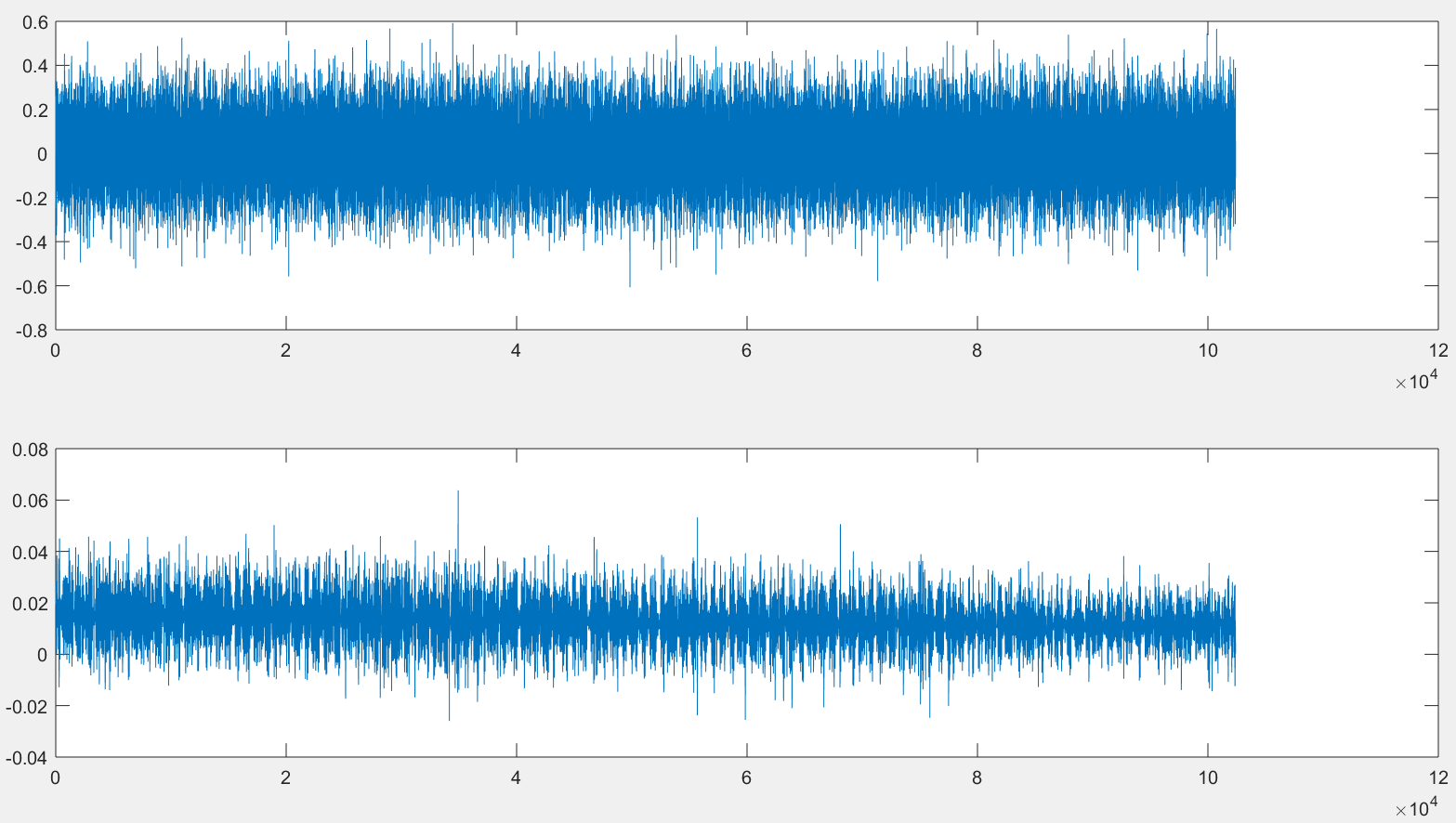


# 三、实验内容

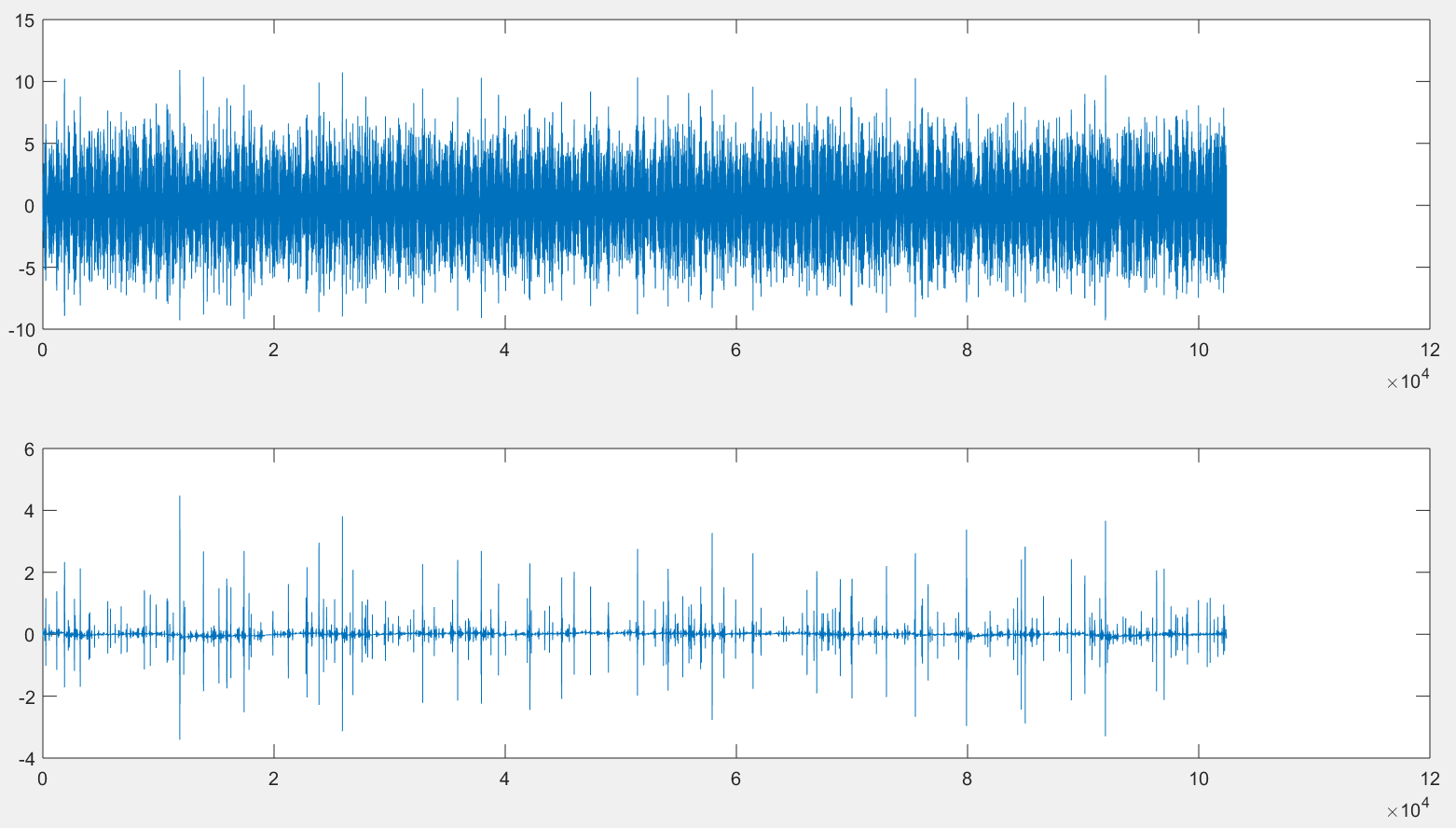
将数据分为ABCD四类，ABC对应不同负荷（0-2hp），D为三者总和。以2048个点的窗口处理数据，每种负荷的每种故障类型可得到50个，共10种故障类型，则共有1500条数据，此处参考了论文《基于DBN的故障特征提取及诊断方法研究》。在下列的结果中，如未特别说明，都是以60%的D为训练集，40%的D为测试集得到的结果。提取的特征为36种（因为有可能降噪去除了有用的信号，所以将原始数据也提取特征，即2\*18种特征），RBM的单元数为36->18,18->13,13->10,最后分类器为10->10。每次循环随机生成训练集与测试集，取50次的均值。

1. **数据预处理**

对不同的数据进行降噪，降噪效果可能存在差异，如下面2张图所示，其中每张图的上子图为原始数据，下子图为降噪数据。



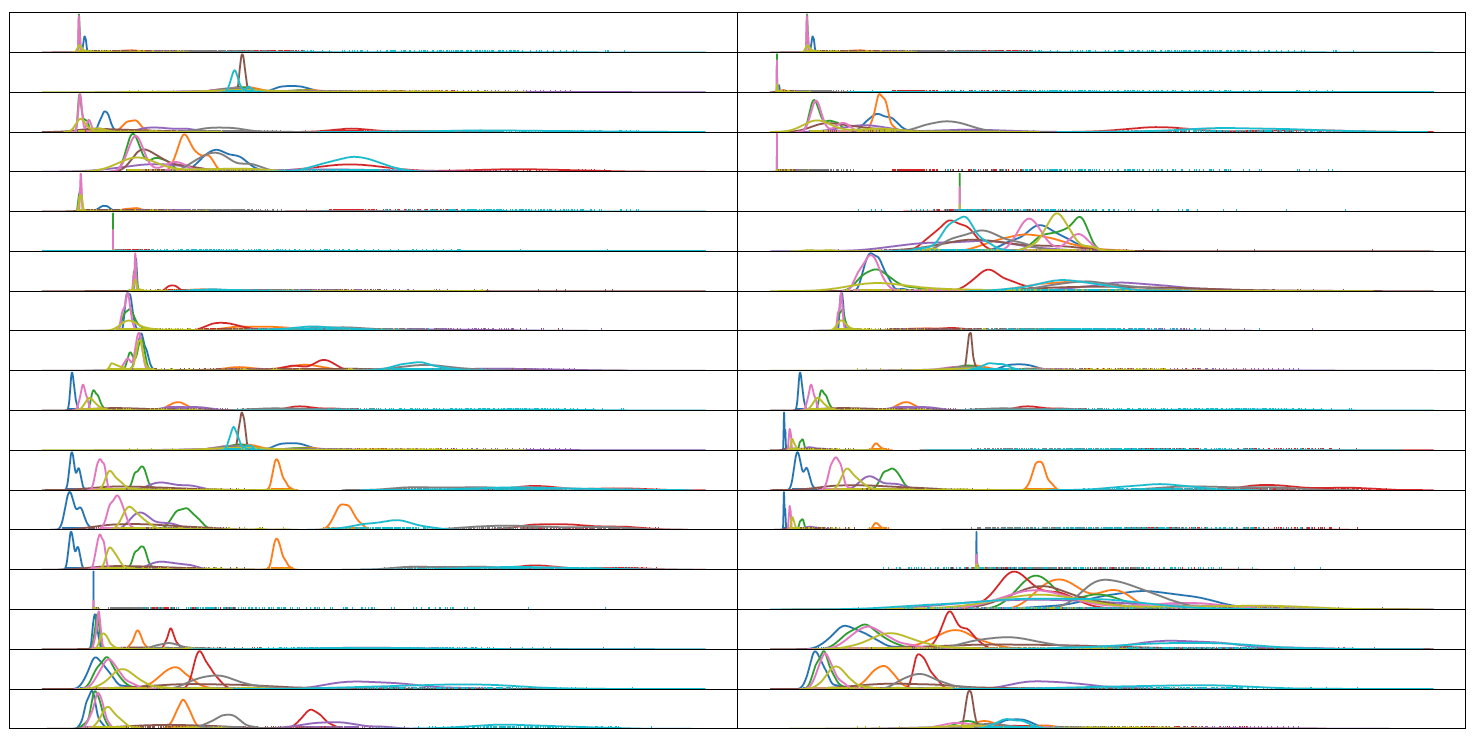
**对B007\_0的X118\_DE\_time降噪结果**



**对B028\_0的X048\_DE\_time降噪结果**

1. **特征提取**

在得到降噪数据后，我们对其提取特征。提取得到的特征如下图所示。每一个坐标系代表了一个单独特征的分布。在一个坐标系中，不同的颜色代表了不同的故障类型，我们可以看到，不同的颜色有着不同的分布，因此我们可以根据这些特征来对故障类型进行区分。每一行的两列表示相同特征降噪前后的特征分布差异。可见降噪可使不同的故障特征差异增大或减小。



**提取得到的特征分布**

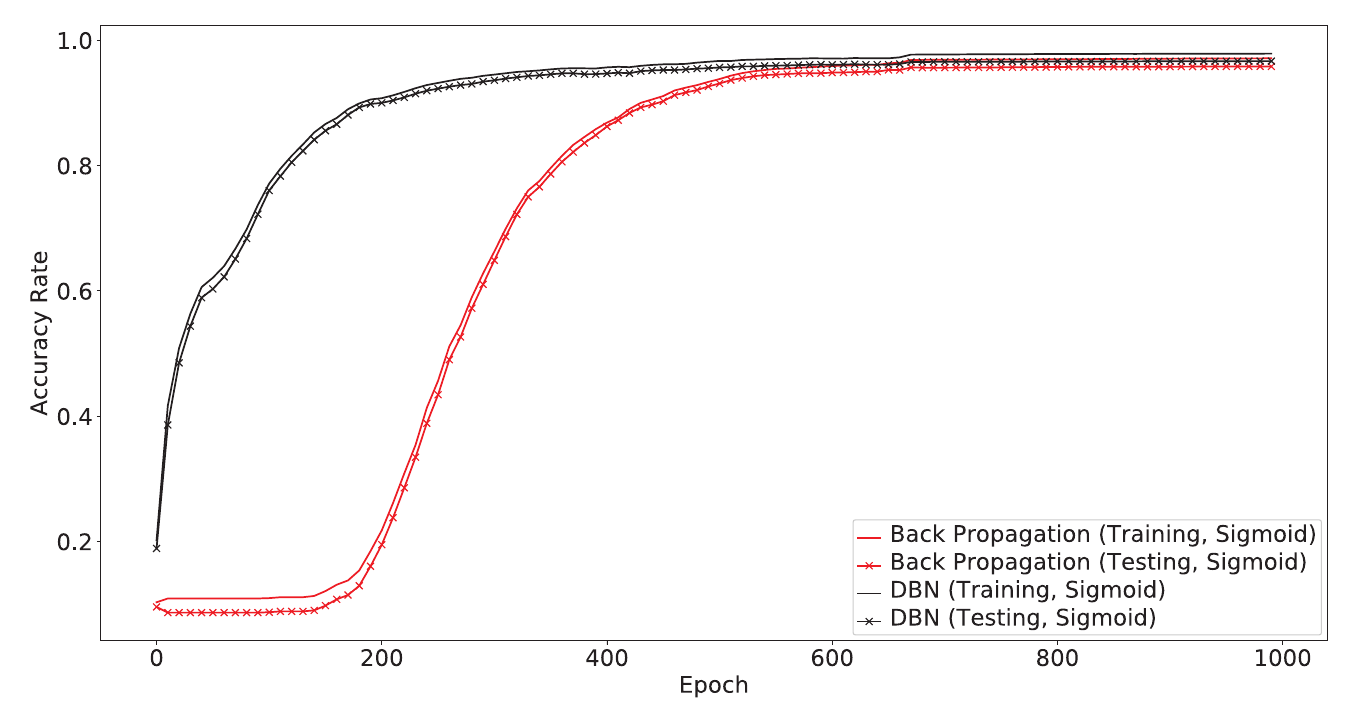
1. **训练DBN**

前6个报告或多或少地涉及了DBN的训练结果，但是由于在对代码的正确性、性能进行不断的优化，因此可能前6个报告中的结果是未多次测试的，抑或是不准确的，所以在汇总报告中，对一些重要的结果进行展示，建议以该报告为参考。

考虑到本次实训主要是进行复现，而非提出优化方法，因此主要目的不是找到最高正确率，而是比较提出的优化方法是否对原有的方法有改进。因此，在结果中，着重比较起始正确率、收敛速度、最终正确率等。实验结果为取50次的平均值。在所有的不同方法的对比图中，黑色曲线总是表示理论上应该表现更好的方法。

**BP（Sigmoid）与DBN（Sigmoid）**

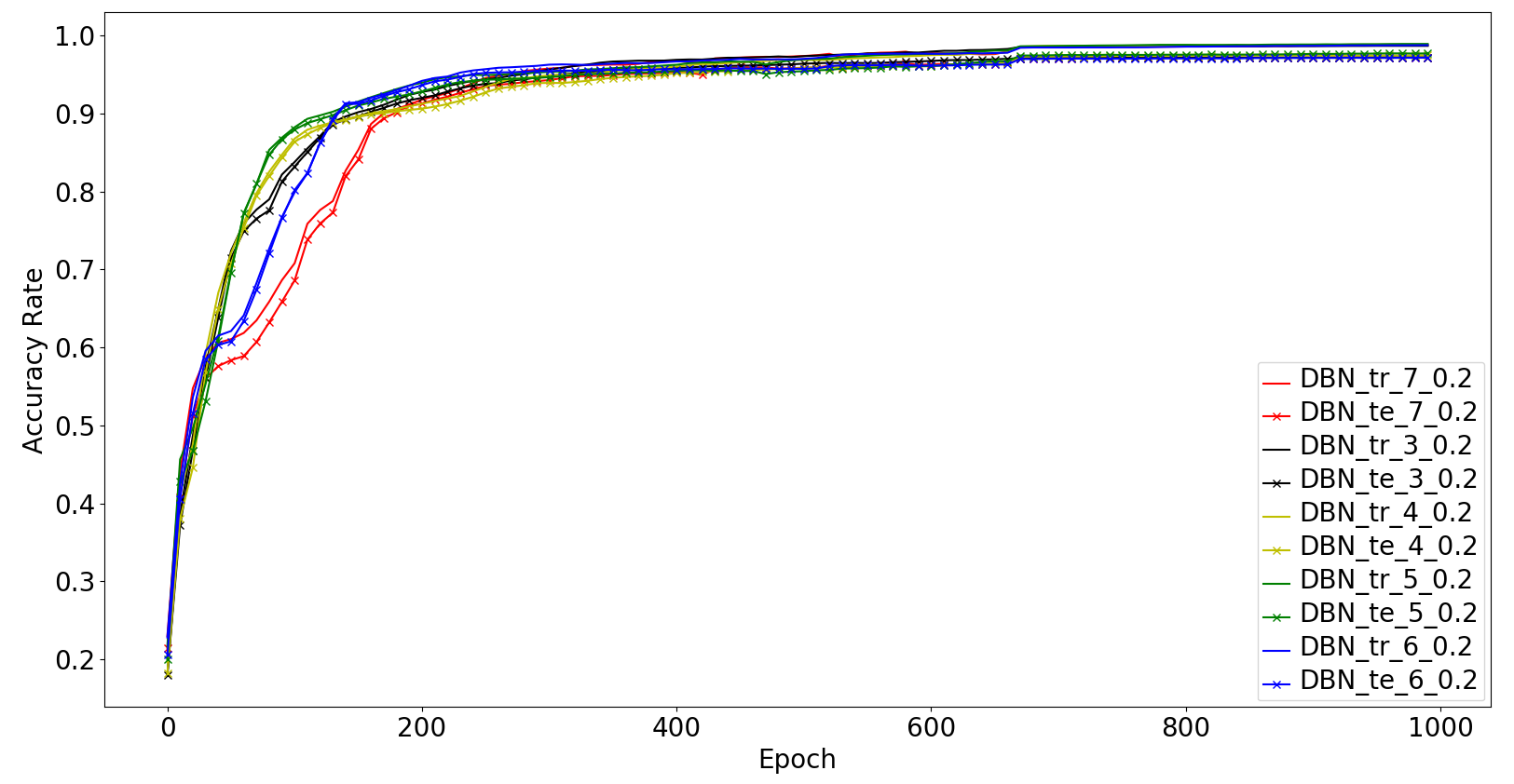
可见在前200个epoch，BP神经网络没有显著的正确率提升，概率为随机选择（1/10=10%），而DBN的初始正确率就是20%，可见预训练确实有效果，且最终正确率DBN高于BP。



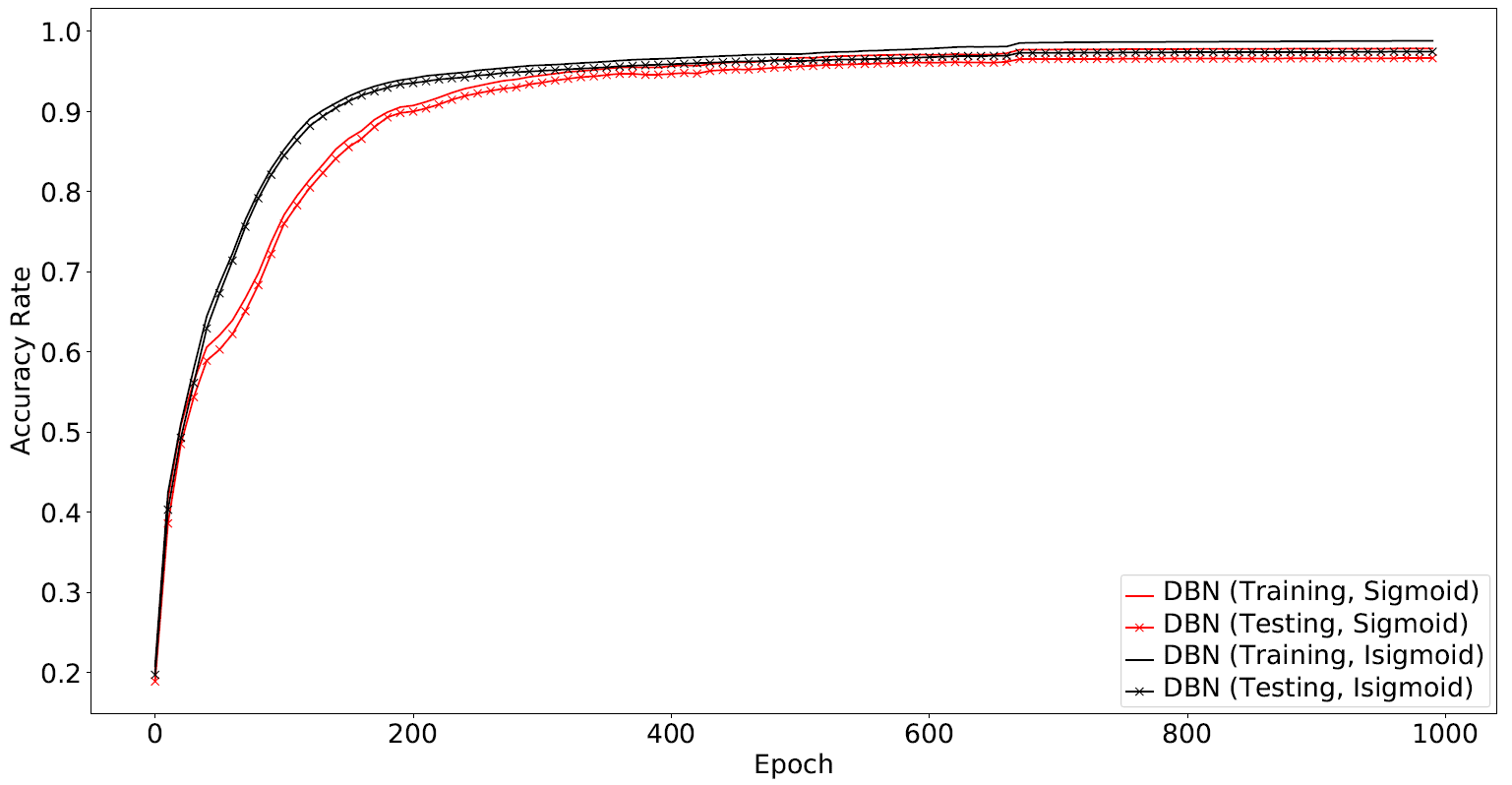
**BP与DBN正确率对比**

**DBN（Sigmoid）与DBN（Isigmoid）**

在使用Isigmoid时，DBN的收敛更快，且最终正确率更高，选取a=5，alpha=0.2。

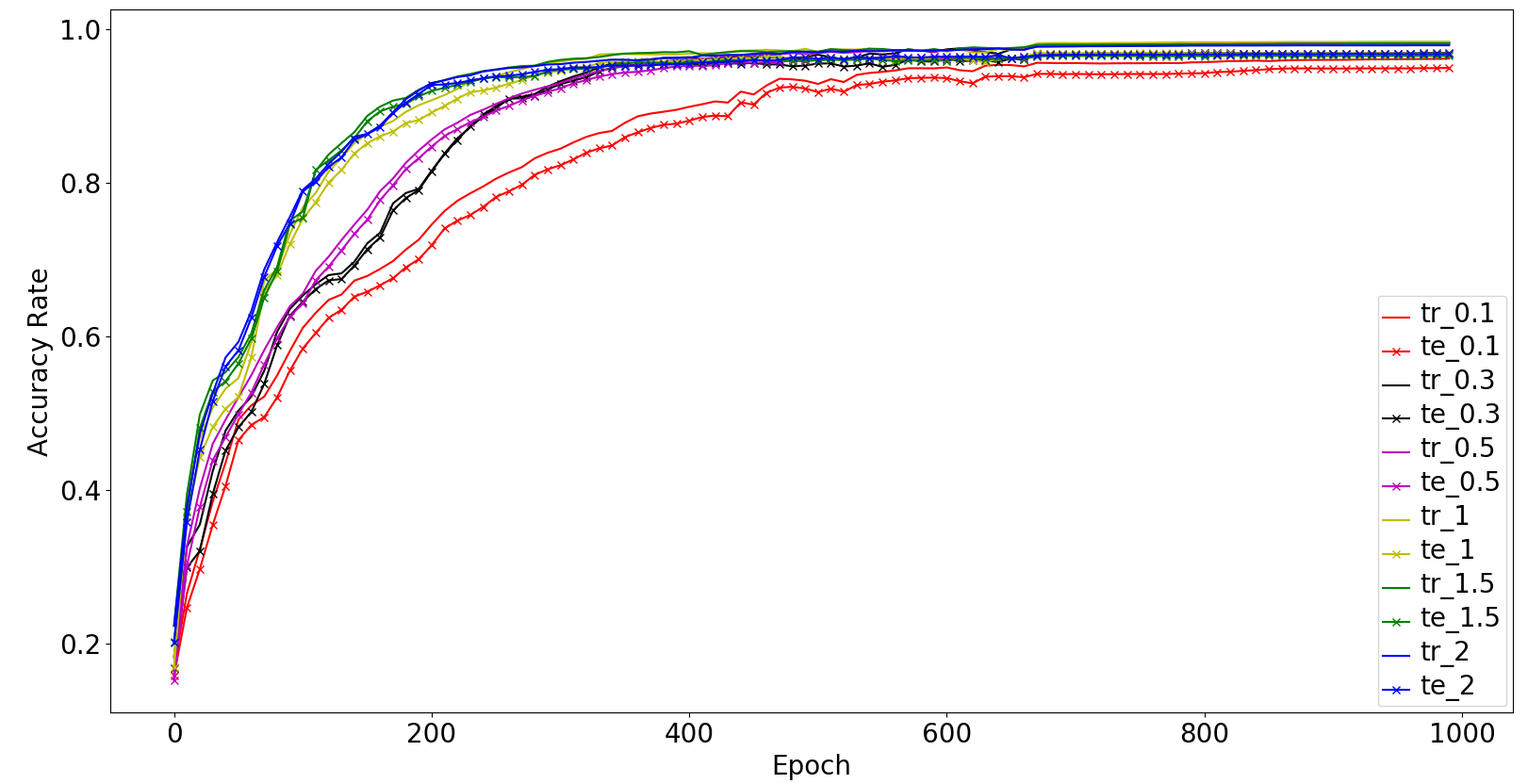


**寻找Isigmoid的最优参数**

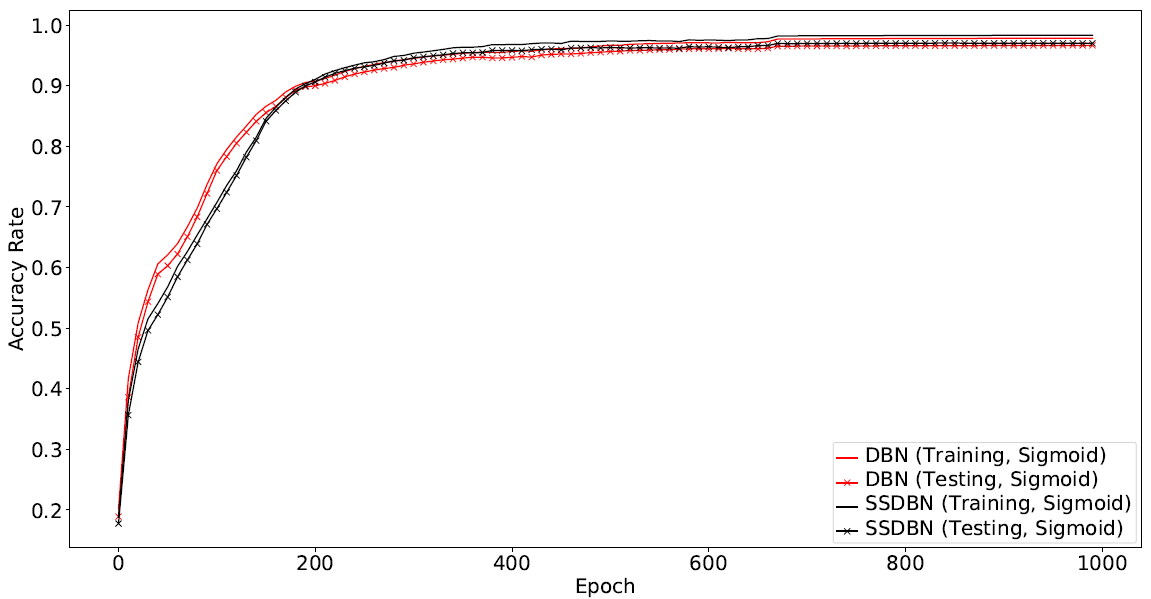
  
**DBN分别使用Sigmoid与Isigmoid时的正确率对比**

**DBN（Sigmoid）与SSDBN（Sigmoid）**

在使用SSDBN时，其正确率更高，但是其收敛速度不如DBN，考虑可能与数据集相关，选取propagation=1。

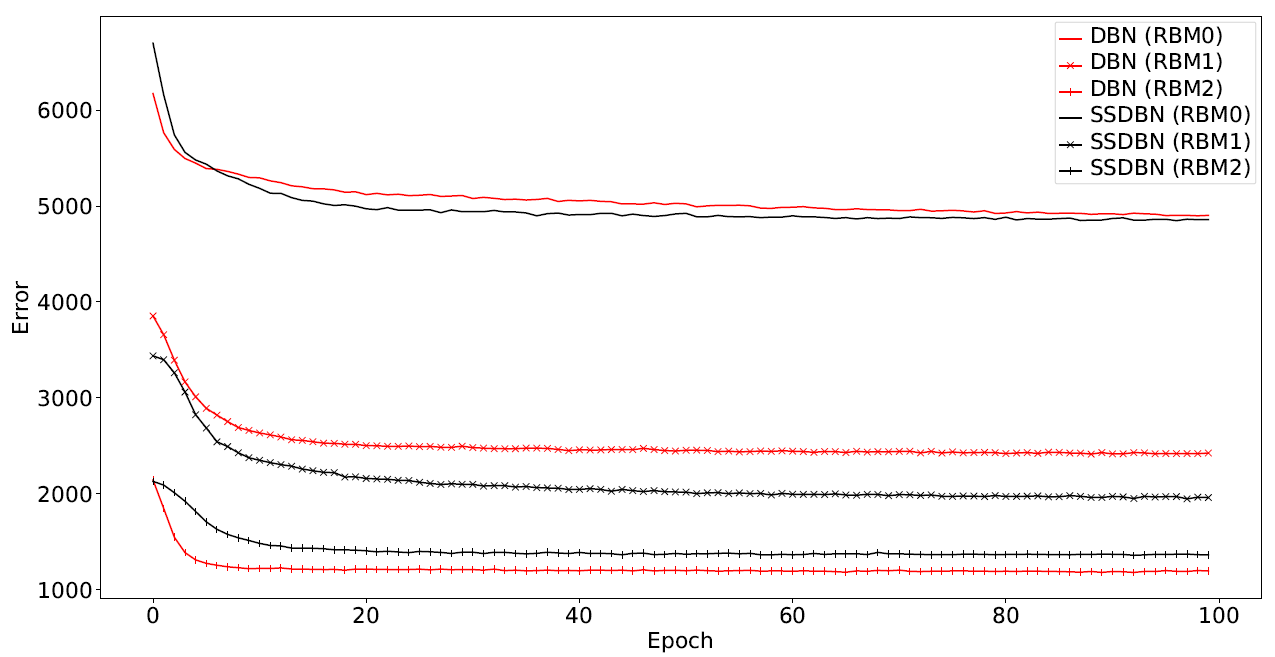


**寻找SSDBN最优参数**



**DBN与SSDBN正确率对比**

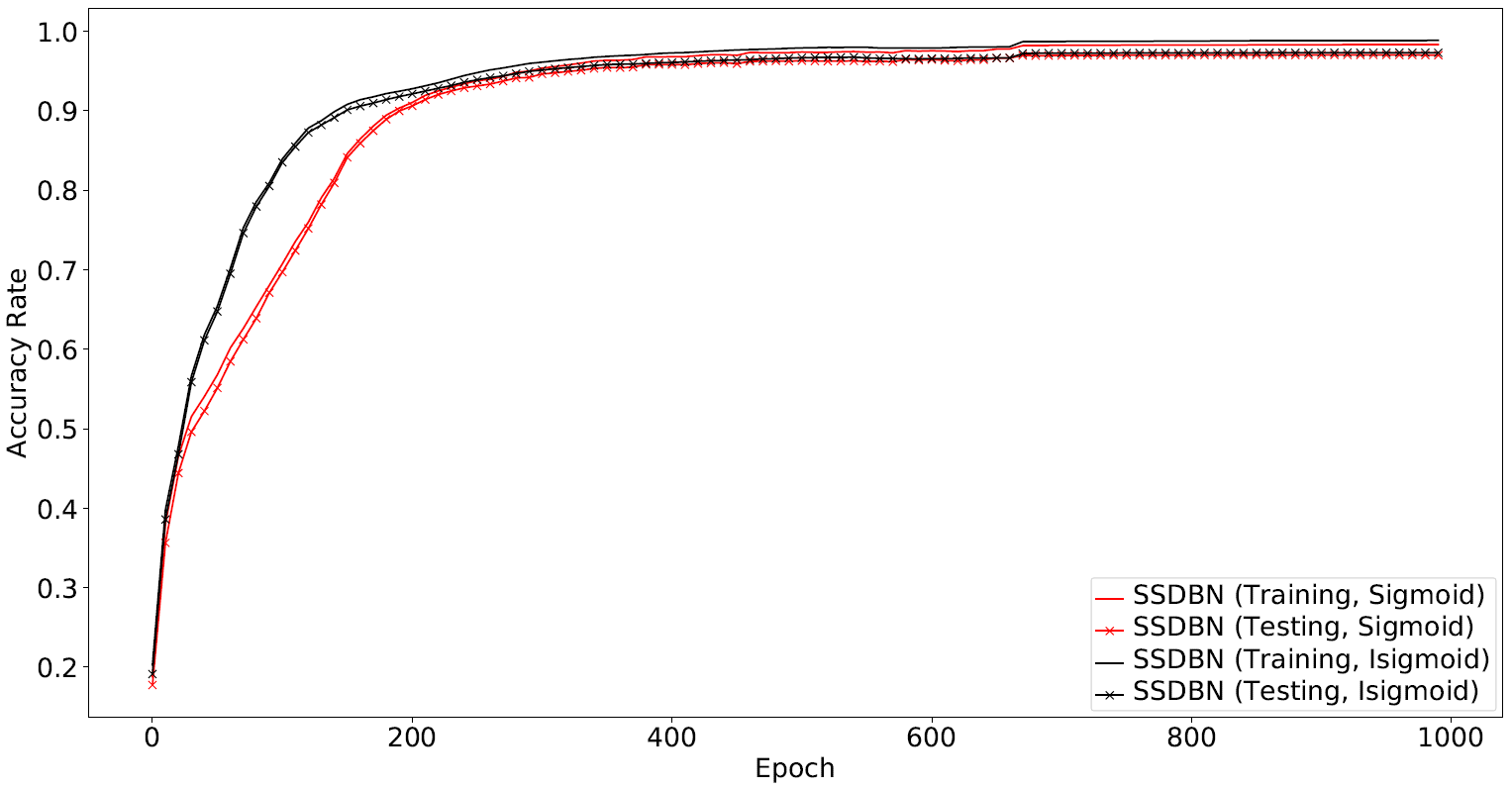
另外在论文中提到了重构误差的差异，因此计算得到了每个RBM的重构误差，可见在前两个RBM中，SSDBN的重构误差确实低于DBN。



**DBN与SSDBN的重构误差对比**

**SSDBN（Sigmoid）与SSDBN（Isigmoid）**

最后将两篇论文的方法结合，使用SSDBN（Isigmoid）进行训练、分类，可见其进一步提升了分类准确率。



**SSDBN（Sigmoid和Isigmoid）正确率对比**

最终分类正确率的对比结果为SSDBN（Isigmoid）> SSDBN（Sigmoid）> DBN（Sigmoid）> BP（Sigmoid）。

考虑到美观和一些不必要的对比，并未将所有的实验结果添加到报告中，但在<https://github.com/wjsunscut/Intelligent-Software-Project-Training/tree/master/Code/result>中有着大量实验结果。