

# 清华大学

# 模式识别

## 结课报告

姓 名： 李浩然、石婉欣、胡玉明

学 号： 2017210914、2017210917、2017210912

日 期： 2018 年 1 月 14 日

指导老师： 卢宗庆

任务分工： 方案讨论与确认：胡玉明、李浩然、石婉欣

数据处理与编程：李浩然

撰写报告与汇总：石婉欣

## 1、 实验方法原理

该部分主要阐明实验所采用的一些方法，主要针对数据的处理，最终训练模型从而实现预测功能。

### 1.1 连续与离散傅里叶变换

傅立叶变换是一种分析信号的方法，它可分析信号的成分，也可用这些成分合成信号。

$f(t)$ 是  $t$  的周期函数，如果  $t$  满足以下条件：在一个以  $2T$  为周期内  $f(x)$ 连续或只有有限个第一类间断点，且  $f(x)$  单调或可划分成有限个单调区间，则  $F(x)$  以  $2T$  为周期的傅里叶级数收敛，和函数  $S(x)$  也是以  $2T$  为周期的周期函数，且在那些间断点上函数是有限值；在一个周期内具有有限个极值点；绝对可积。下列公式即称为积分运算  $f(t)$ 的傅立叶变换，

$$F(\omega) = \mathcal{F}[f(t)] = \int_{-\infty}^{\infty} f(t)e^{-i\omega t} dt$$
$$f(t) = \mathcal{F}^{-1}[F(\omega)] = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)e^{i\omega t} d\omega$$

$f(t)$ 式的积分运算叫做  $F(\omega)$ 的傅立叶逆变换。 $F(\omega)$ 叫做  $f(t)$ 的像函数， $f(t)$ 叫做  $F(\omega)$ 的像原函数。 $F(\omega)$ 是  $f(t)$ 的像。 $f(t)$ 是  $F(\omega)$ 原像。

由于采样值为离散数据，故而该实验采用的是离散傅里叶变换。离散傅里叶变换(DFT)，是傅里叶变换在时域和频域上都呈现离散的形式，将时域信号的采样变换为在离散时间傅里叶变换(DTFT)频域的采样。在实际应用中通常采用快速傅里叶变换以高效计算 DFT。

$$X(e^{j\omega}) = DTFT[x(n)] = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}$$

简言之，傅里叶变化就是将基于时域的特征转换成基于频域的特征。而离散傅里叶变换就是在其基础上对离散数据基于时间进行傅里叶变换。

### 1.2 主成分分析 PCA

从下图不难看出，主成分分析是将原来的变量重新组合成一组新的互相无关的变量，同时根据实际需要从中可以取出几个较少的变量尽可能多地反映原来变量的信息的统计方法。

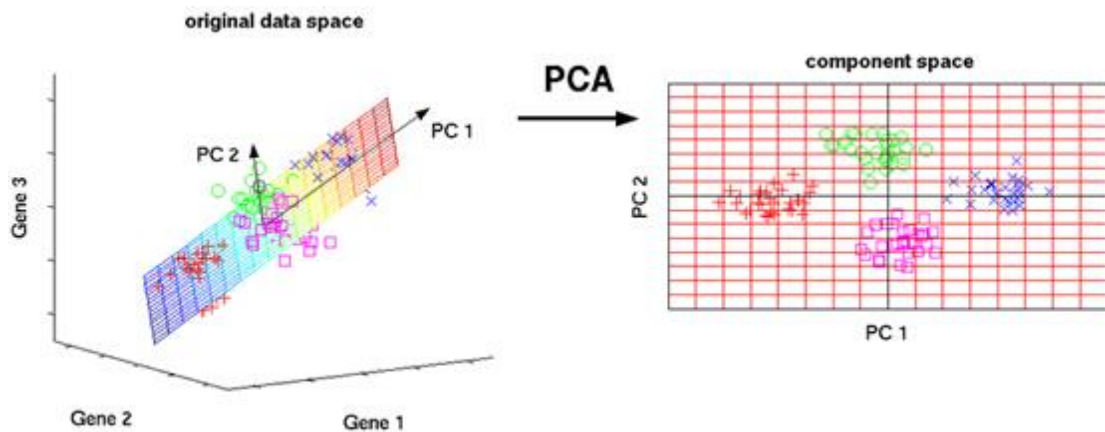


图 1

主成分分析能降低所研究的数据空间的维数。即用研究  $m$  维的  $Y$  空间代替  $p$  维的  $X$  空间( $m < p$ )，而低维的  $Y$  空间代替高维的  $x$  空间所损失的信息很少。

### 1.3 人工神经网络的基本思想

根据对自然神经系统构造和机理的认识, 神经系统是由大量的神经细胞构成的复杂的网络, 人们对这一网络建立一定的数学模型和算法, 设法使它能够实现诸如基于数据的模式识别、函数映射等带有“智能”的功能。

#### 1.3.1 全连接神经网络的含义

对  $n-1$  层和  $n$  层而言,  $n-1$  层的任意一个节点, 都和第  $n$  层所有节点有连接。即第  $n$  层的每个节点在进行计算的时候, 激活函数的输入是  $n-1$  层所有节点的加权。

全连接是较好的神经网络模式, 但是网络很大的时候, 训练速度会很慢。部分连接就是人为切断某两个节点直接的连接, 这样训练时计算量将大为减小。

#### 1.3.2 BP 神经网络

单个感知器能够解决线性可分的问题, 但是也只能解决所谓一阶谓词逻辑问题。所以提出了多层模型: 前一层神经元的输出是后一层神经元的输入, 最后一层只有一个神经元, 它接收来自前一层的  $n$  个输入, 给出作为决策的一个输出。

## 2、实验函数

该部分主要阐明实验过程中调用 Matlab 平台的相关函数及使用原理。

### 2.1 feedforwardnet(hiddenSizes,trainFcn)

网络包含一系列的层次，第一层与网络输入连接，接下来的层次与上一次连接。最后一层产生网络的输出。**feedforward** 网络可以用作输入和输出的映射，只含有一个隐含层的神经网络可以拟合任意有限的输入输出映射问题。输入的变量有两个可以选择，前者是对隐层的设置，可以有多层并且设置每层的神经元个数。后者可以设置训练函数，默认为 **trainlm**。

hiddenSizes	Row vector of one or more hidden layer sizes (default = 10)
trainFcn	Training function (default = 'trainlm')

### 2.2 Matlab 自带神经网络库中可用的训练函数

训练方法	训练函数
梯度下降法	traingd
有动量的梯度下降法	traingdm
自适应 lr 梯度下降法	traingda
自适应 lr 动量梯度下降法	traingdx
弹性梯度下降法	trainrp
Fletcher-Reeves 共轭梯度法	traincgf
Ploak-Ribiere 共轭梯度法	traincgp
Powell-Beale 共轭梯度法	traincgb
量化共轭梯度法	trainscg
拟牛顿算法	trainbfg
一步正割算法	trainoss
Levenberg-Marquardt	trainlm

## 2.3 性能函数 (performance function)

Mean-Square Error( MSE) 均方误差是反映估计量与被估计量之间差异程度的度量; Sum of Squares for Error(SSE)是误差项平方和; Mean Squared Error with Regularization performance function(MSEREG)是正则化性能函数均方误差。

## 2.4 BP 网络常用传递函数

Log-sigmoid 型函数的输入值可取任意值, 输出值在 0 和 1 之间;

tan-sigmoid 型传递函数 tansig 的输入值可取任意值, 输出值在-1 到+1 之间;

purelin 线性传递函数的输入与输出值可取任意值。

BP 网络通常有一个或多个隐层, 该层中的神经元均采用 sigmoid 型传递函数, 输出层的神经元则采用线性传递函数, 整个网络的输出可以取任意值。

# 3、实验背景

根据脑电波活动为特征可以将睡眠分为四个阶段。

第一阶段: 开始进入睡眠模式。此时脑波是  $\alpha$  波。进入第一阶段后期, 脑波变成了 3.5 到 7.5 赫兹之间的  $\theta$  波。

第二阶段: 这个阶段的脑波很不规则, 会出现叫 K-复合波的低幅 EEG 波。

第三阶段: 这一阶段我们发出高幅波 (3.5 赫兹)、 $\delta$  活动的 EEG 信号。

第四阶段: 这一阶段出现更多的  $\theta$  波干扰了正常的  $\delta$  波。人的眼睛开始快速地来回动, 称作快速眼动 (REM)。

# 4、实验数据

旧数据: 20 个人的脑电波波形数据, 19 个人的数据作为训练集, 最后一个人的数据作为测试集。数据为 mat 文件, 标签为 txt 文件。每 1000 个数据对应一个标签, 例如 5768000 维数据对应 5768 维标签。

新数据: 40 个人的脑电波波形数据, 39 个人的数据作为训练集, 最后一个人的数据作训练模型的预测。数据为 mat 文件, 每 3000 个数据对应一个标签, 所以每个人的数据都是  $3001 \times n$  的矩阵。

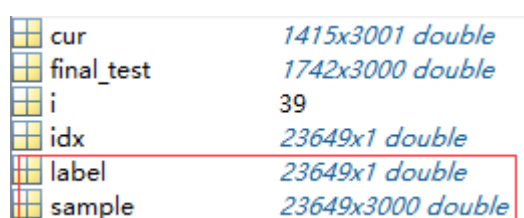
## 5、实验过程

该部分主要呈现实验的具体流程，介绍实验使用的基本方法与基本工具，并且阐明各个环节中细节，展示主要解决的几个问题。

### 5.1 训练数据提取与处理

#### 5.1.1 训练特征与标签分离

首先将 39 个人的所有数据合并，然后过滤掉第四和第五周期的数据。继而将标签与特征数据分离，易于后期进行训练。



cur	1415x3001 double
final_test	1742x3000 double
i	39
idx	23649x1 double
label	23649x1 double
sample	23649x3000 double

最终将数据分成两部分，一部分是训练特征，另一部分是标签。此外还有一个最终测试样本，但是不包含标签。

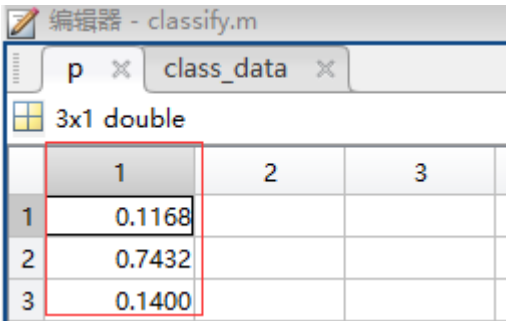
```
sample = [];  
label = [];  
load('data/train.mat');  
load('data/test.mat');  
vari = whos('-file', 'data/train.mat');  
for i = 1:length(vari)  
    cur = eval(vari(i).name);  
    sample = [sample; cur(:,1:3000)];  
    label = [label; cur(:,3001)];  
end  
idx = find(label > 0 & label < 4);  
sample = sample(idx,:);  
label = label(idx);  
test_sample = final_test;
```

#### 5.1.2 三类训练样本的比例对比

为了更直观的呈现新给的样本集合中每一类样本所占有的比例，以明确后续训练过程中的细节，我组尝试将三类样本占总样本的比例算出，可知，第一类样本占有 11.68%，第二类

占有 74.2%，第三类占有 14.00%，由于第二类样本较多，另外两类的样本显然较少，这也是我组在进行后续优化所要考虑的问题。

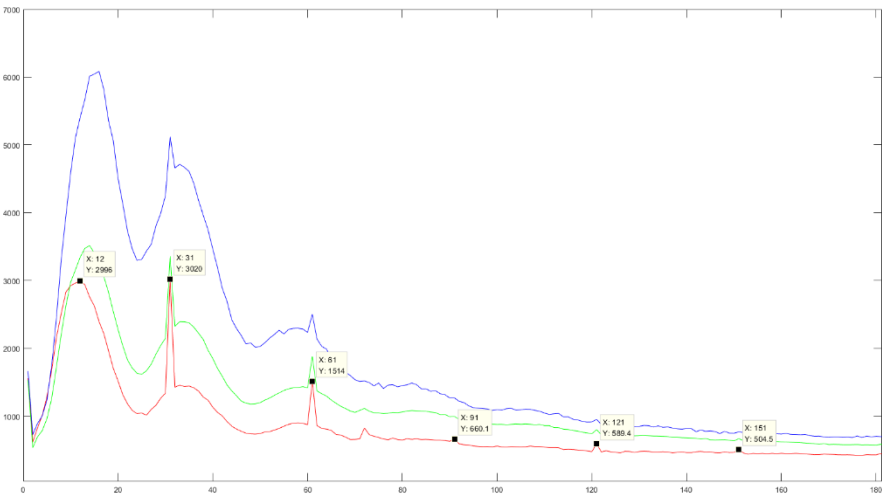
```
k = 3;
m = size(label);
p = zeros(k, 1);
tot = 0;
class_data = cell(k, 1);
for i = 1:k
    indices = find(label == i);
    t_sample = sample(indices(:,:),:);
    p(i) = size(t_sample, 1);
    tot = tot + p(i);
    class_data{i} = t_sample;
end
p = p ./ tot;
```



	1	2	3
1	0.1168		
2	0.7432		
3	0.1400		

### 5.2 进行 DTFT 变换

正如前文所述，利用计算机的 FFT 变换，将原本样本的时域特征变换成基于频域的特征。在完成 FFT 变换后，我组绘制出三类样本所分别对应的平均 FFT 示意图。进行对图片的观察后，我们得出了一系列可供参考的信息。



从上图我们可以清晰地看出三类样本在进行快速傅里叶变换后，取出每一类样本的 fft 平均值，绘制出的示意图有三个尖峰。这就意味着这三个频域[1, 20]、[25, 40]、[50, 70]所对应的基于频域的特征函数能较好地还原出源像。

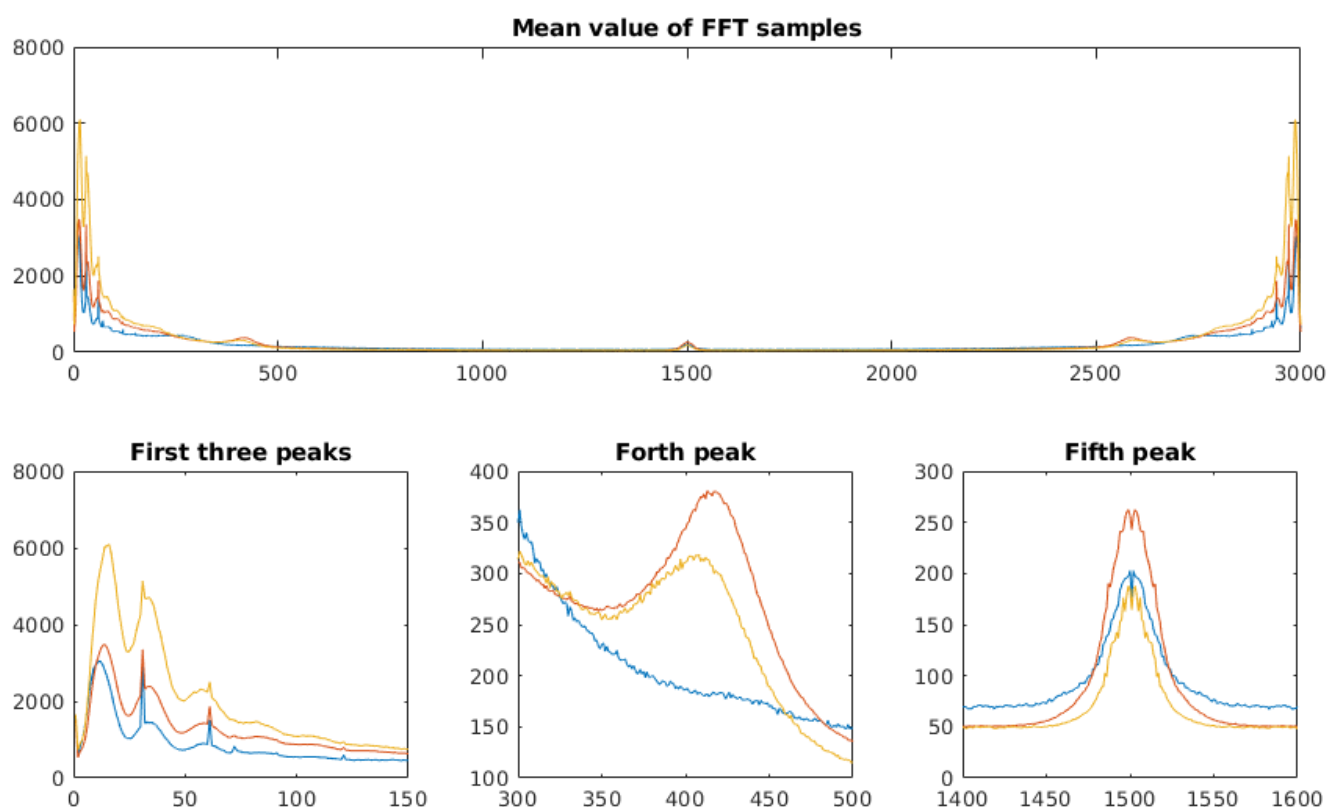
### 5.2.1 单样本特征曲线抖动问题

由于单个样本的 FFT 曲线抖动较为剧烈,所以本文使用两次中值滤波(`medfilt2(X, [1 3])`)使得特征曲线变得较为平滑。事实上,中值滤波是一种基于排序统计理论的,能有效抑制噪声的非线性信号处理技术。其原理在于把数字序列中一点的值用该点的一个邻域中各点值的中值代替,让周围的值接近的真实值,从而消除孤立的噪声点。

```
编辑器 - F:\matlab\work\2017210917_石婉欣_第四次实验报告\PR-Final-master\data_process\p
fil_sca.m × linear_scale.m × pca_trans.m × pca_vec.m × process.m ×
- p1 = 1:20;
- p2 = 25:40;
- p3 = 50:70;
- p4 = 350:480;
- p5 = 1450:1550;
- sam = medfilt2(fft_sample, [1 3])
- sam = medfilt2(sam, [1 3]);
- mx1 = max(sam(:, p1), [], 2);
- mx2 = max(sam(:, p2), [], 2);
```

### 5.2.2 提取峰值对应的主频

此外通过对该示意图的缩放,我们还可以看到在另外两个频域内[350,480]、[1450,1550]也有两个尖峰,说明这两个频域所对应的特征函数也可以较好地匹配源像信息。





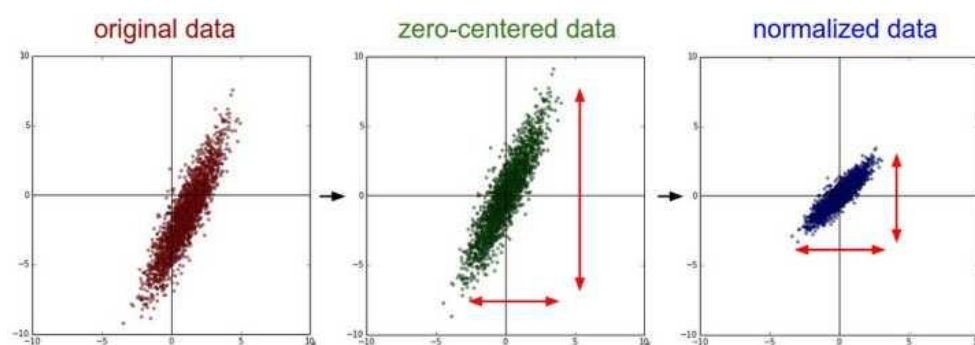
所以通过上述观察信息，我们可以得知以上五个峰值信息能够比较完整的还原出三种睡眠时期的频谱信息。所以我组决定取五个尖峰所对应的频域信息作为训练特征。所以取出这五个区间的 `fft` 样本值如图所示：

变量 - fft_sample							
res    fft_sample							
16554x3000 double							
	1	2	3	4	5	6	7
1	422.3687	145.1134	332.3692	552.4210	579.1014	1.5349e+03	1.6702e+03
2	7.2954e+03	344.5652	1.1621e+03	395.3624	1.0875e+03	958.2963	720.7581
3	572.8606	296.8642	441.2350	484.8582	568.3730	828.9204	1.2748e+03
4	342.2364	842.7495	200.2033	905.4041	589.3659	2.2325e+03	224.2300
5	2.8848e+03	90.2320	270.4100	120.0723	568.1672	462.3037	405.2444
6	995.8154	695.4350	740.6249	853.1600	938.4482	645.0761	2.1640e+03
7	240.5143	285.9578	939.7206	304.2622	946.6742	1.2724e+03	1.7757e+03
8	3.3388e+03	513.0401	758.5731	987.6735	1.0572e+03	2.0417e+03	1.8966e+03
9	70.4889	55.2139	85.0078	116.2190	408.3707	646.0191	477.1162

### 5.3 数据正则化处理

本实验在进行主成分分析前，对各个特征做标准正态化(normalize)。因为 PCA 算法选择每个分量的时候，都尽量使得投影其上的数据尽量分散，所以如果有个维度的数据过于分散，PCA 在选取投影向量的时候就会偏向于选取该特征所在的分量，从而丢失了较多其他维度的信息。事实上，我们得到的样本数据都是多个维度的，即一个样本是用多个特征来表征的。但是，这些特征的量纲和数值的量级不一定一样。如果直接使用原始的数据值，那么他们对预测目标的影响程度将是不一样的。而通过标准化处理，可以使得不同的特征具有相同的尺度。这样，在使用梯度下降法学习参数的时候，不同特征对参数的影响程度就一样了。

为了解决上述问题，我组将特征数据进行均值为 0、方差为 1 的正态分布转换。

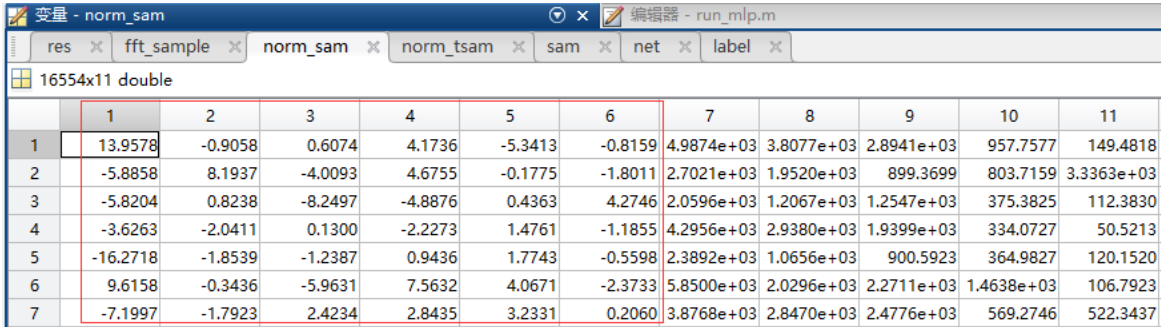


## 5.4 PCA 处理

在进行 PCA 处理之前，先采取正则化，避免过拟合问题。此外，如同前文所述，PCA 有助于提取主成分，将主特征提取出来用于训练从而避免维度爆炸问题。当然，针对测试数据，也需要进行相同的处理，为后续测试验证做准备。

从下图中可看到经过处理的数据，前六维是 PCA 所提取出来的六个主成分，而后五个特征则是基于傅里叶变换得到的五个峰值所对应的频域信息，经过这样的数据处理，不经有效地降低了数据维度，解决了计算量过于庞大的问题。而且这些特征信息能较为有效地代表原有数据的特征。

```
norm_sample = pca_trans(process(sample), vec, pca_param);  
norm_tsampl = pca_trans(process(test_sample), vec, pca_param);
```



	1	2	3	4	5	6	7	8	9	10	11
1	13.9578	-0.9058	0.6074	4.1736	-5.3413	-0.8159	4.9874e+03	3.8077e+03	2.8941e+03	957.7577	149.4818
2	-5.8858	8.1937	-4.0093	4.6755	-0.1775	-1.8011	2.7021e+03	1.9520e+03	899.3699	803.7159	3.3363e+03
3	-5.8204	0.8238	-8.2497	-4.8876	0.4363	4.2746	2.0596e+03	1.2067e+03	1.2547e+03	375.3825	112.3830
4	-3.6263	-2.0411	0.1300	-2.2273	1.4761	-1.1855	4.2956e+03	2.9380e+03	1.9399e+03	334.0727	50.5213
5	-16.2718	-1.8539	-1.2387	0.9436	1.7743	-0.5598	2.3892e+03	1.0656e+03	900.5923	364.9827	120.1520
6	9.6158	-0.3436	-5.9631	7.5632	4.0671	-2.3733	5.8500e+03	2.0296e+03	2.2711e+03	1.4638e+03	106.7923
7	-7.1997	-1.7923	2.4234	2.8435	3.2331	0.2060	3.8768e+03	2.8470e+03	2.4776e+03	569.2746	522.3437

## 5.5 移除异常值

另外本实验做了剔除异常值的工作，可知在实际采样过程中不可避免地可能产生人为或者机械干扰，导致会有部分数据丧失真实特征。当然我们通过一定的过滤算法，能够比较有效地筛选一些异常点。

本实验编写相应的过滤函数，将数据进行线性伸缩变换，能够将有效数据圈定在一定范围内，由此异常值就自然突出。

```
function [sca_sam, param] = linear_scale(sample, range, param)  
    scale = range.u - range.l;  
    bias = range.l;  
    if size(param, 1) == 0  
        mi = min(sample);  
        ma = max(sample);  
    else  
        mi = param.mi;  
        ma = param.ma;  
    end  
    sca_sam = scale * (sample - mi) ./ (ma - mi) + bias;  
    param.mi = mi;  
    param.ma = ma;  
end
```

然后建立过滤函数，将数据特征样本、标签、测试数据、测试数据样本带入，最终通过主函数调用完成异常点筛选。

```
function [sam, lab, tsam, tlab] = fil_sca(sample, label, test_data, test_label)

% remove outlier
% linear scale to unit range
if ~exist('sam', 'var')
    [sam, lab, tsam, tlab] = fil_sca(norm_sam, label, norm_tsam, tlabel);
end
```

## 5.6 附：数据归一化处理

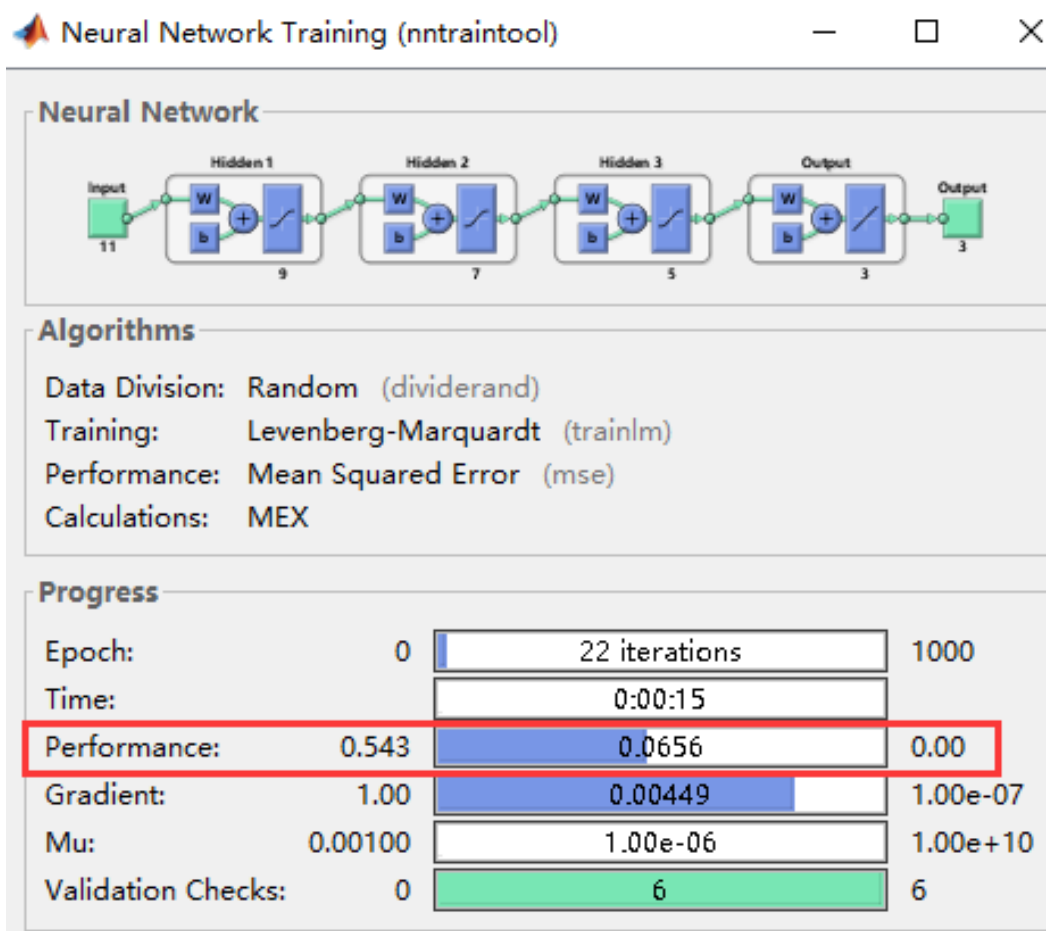
本实验为采用数据归一化，我组认为在神经网络训练中也有必要进行数据归一。

数据归一化，就是将数据映射到[0,1]或[-1,1]区间或更小的区间，比如(0.1,0.9)，需要将训练和测试数据进行归一化的原因如下：

- 1.输入数据的单位不一样，有些数据的范围可能特别大，导致的结果是神经网络收敛慢、训练时间长；
- 2.数据范围大的输入在模式分类中的作用可能会偏大，而数据范围小的输入作用就可能会偏小；
- 3.由于神经网络输出层的激活函数的值域是有限制的，因此需要将网络训练的目标数据映射到激活函数的值域。例如神经网络的输出层若采用 S 形激活函数，由于 S 形函数的值域限制在(0,1)，也就是说神经网络的输出只能限制在(0,1)，所以训练数据的输出就要归一化到[0,1]区间；
- 4.S 形激活函数在(0,1)区间以外区域很平缓，区分度太小。例如 S 形函数  $f(x)$  在参数  $a=1$  时， $f(100)$  与  $f(5)$  只相差 0.0067。

## 5.7 创建神经网络进行训练

本实验采用 `feedforwardnet` 创建训练网络，该网络结构为五层网络，出去输入层和输出层，还有中间三层隐层。从下图中可以看出输入层为 11 维，即所选出的 6 维主成分及 5 维傅里叶变换主频。中间层分别为 9 维、7 维、5 维，通过原本 11 维的特征，依次递减，既能有效地保持数据特征，又能逐渐收敛，得到结果输出。



此外，同样可以通过 nntool 图形界面创建相应的网络结构，并调节相关参数。

Create Network or Data

Network Data

Name: network1

### Network Properties

Network Type: Feed-forward backprop

Input data: input

Target data: target

Training function: TRAINLM

Adaption learning function: LEARNGDM

Performance function: MSE

Number of layers: 3

Properties for: Layer 2

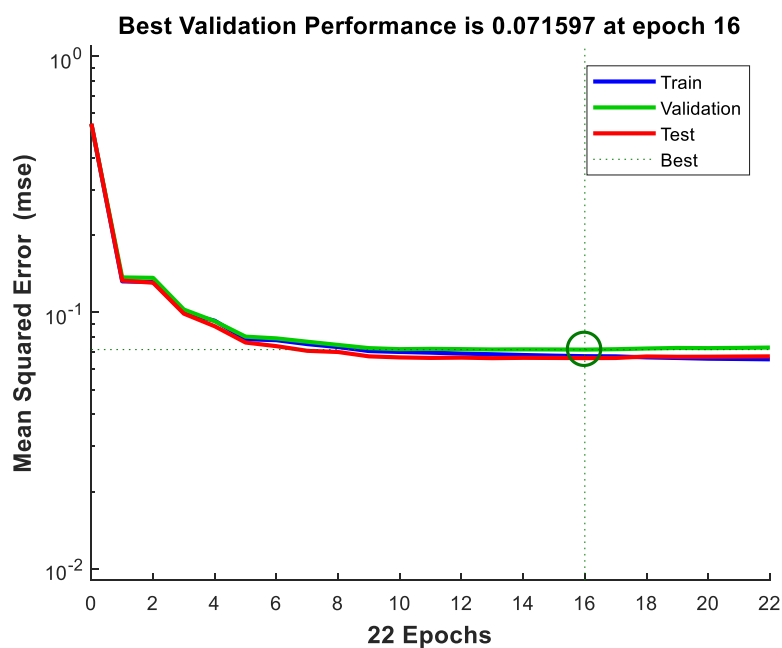
Number of neurons: 6

Transfer Function: TANSIG

## 6、实验结果

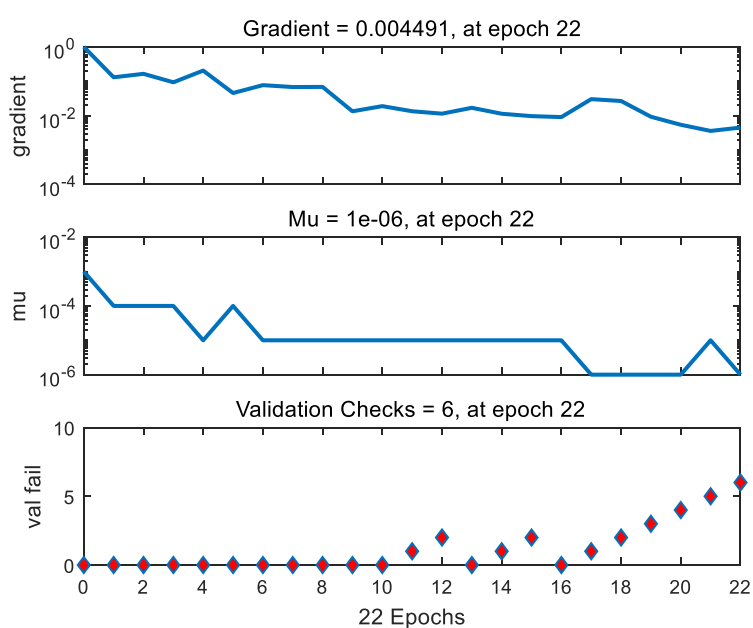
### 6.1 性能收敛

下图所示为训练过程示意图，从下图可以看出在进行 16 次迭代时，通过 MEX 性能函数已能看出在不断的训练中，性能达到 0.071.不难看出该网络模型的收敛速度较快。



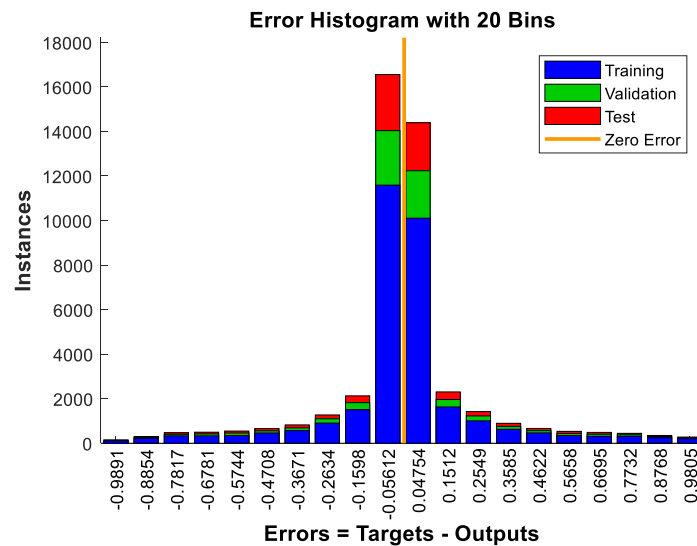
### 6.2 训练状态变化示意

此外，由于 matlab 自带的图形界面绘制工具比较丰富，我们可以比较容易地捕获到相关参数变化的时序图：



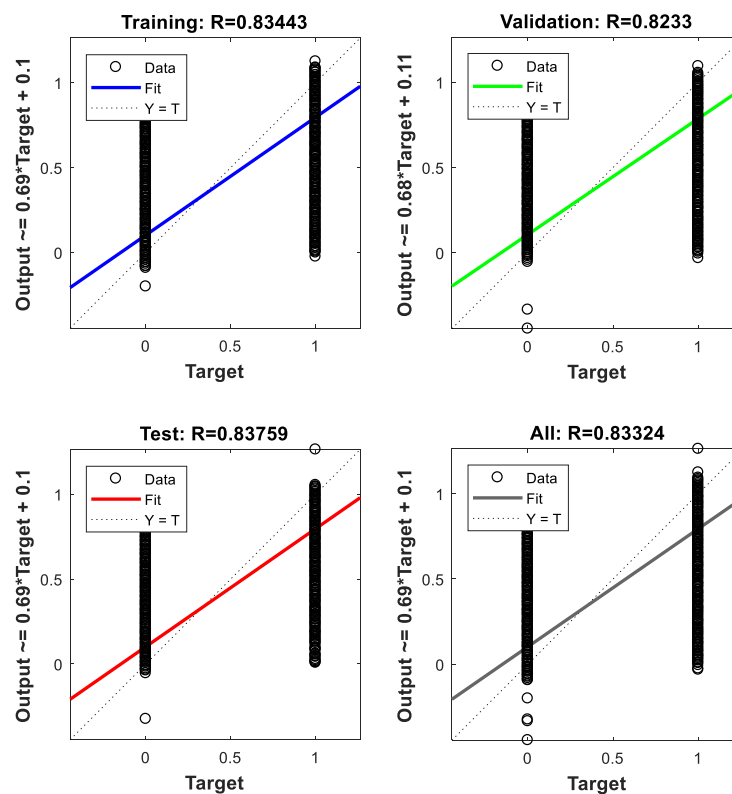
### 6.3 验证结果实例

通过将预期结果与实际结果进行对比也可以用于评价训练学习过程的变化以及优劣更替，如下图所示即为误差柱状图：

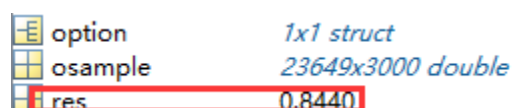


### 6.4 模型精确率 84%+

最终测试精确度可以从下方的回归图看出此次效率已在 83%以上，当然本次实验最佳模型可以高达 85%.



我组通过多次训练，对网络参数进行修改，并且多次重新调整训练数据的维度和数量之后，可以将训练模型的准确率基本稳定在 84%，并且最佳模型高达 85%。为了尽可能使得分类效果较好，所以将各个参数的阈值调得较为细致，通过当迭代到一定程度，模型自动停止训练。故在 20 次训练中，每次基本能在迭代到第 20 轮左右（最佳验证性能采用 MSE 标准）达到最佳性能。



option	1x1 struct
osample	23649x3000 double
res	0.8440

最终我组进行正确率统计，该模型经过测试样本验证，最佳状态时能够达到 85% 左右的正确率。

## 7、总结

本次实验，我组采用傅里叶变换提取主频，借助 Matlab 平台的 fft 工具有效计算。另外在进行主成分分析前，对数据进行正则化避免过拟合现象。为了防止异常值的影响，我组实验对异常数据也进行过滤处理。最终通过五层神经网络——输入、输出及三层隐藏层，完成对预测模型的训练，达到 85% 的最佳预测精确度。

该实验中，我组经过多次讨论，协同合作展开各项工作，包括确定实验方案、探索实验方法、编写程序、验证测试、撰写报告、汇总修缮等。

我组成员均在此过程中获益良多，真正在实验中各司其职并相互帮助。不仅仅是得出实验结果，而是在整个实验过程中经历困惑然后豁然开朗。

最后对老师和助教致以诚挚感谢，与我们以耐心指导，授我们以专业知识！