Test\_example\_CNN:

* 1设置CNN的基本参数规格，如卷积、降采样层的数量，卷积核的大小、降采样的降幅
* 2 cnnsetup函数 初始化卷积核、偏置等
* 3 cnntrain函数 训练cnn，把训练数据分成batch，然后调用   
  + 3.1cnnff 完成训练的前向过程
  + 3.2 cnnbp计算并传递神经网络的error，并计算梯度（权重的修改量）
  + 3.3 cnnapplygrads 把计算出来的梯度加到原始模型上去
* 4 cnntest 函数，测试当前模型的准确率

主要工作：

设置网络结构及训练参数

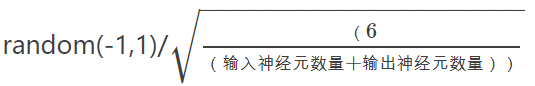
初始化网络，对数据进行批训练，验证模型准确率。

绘制均方误差曲线

Cnnsetup.m

该函数用于初始化CNN的参数。   
**设置各层的mapsize大小，   
初始化卷积层的卷积核、bias**   
**尾部单层感知机的参数设置\*（输出层）**

bias统一初始化为0

权重设置为：

对于卷积核权重，输入输出为fan\_in, fan\_out   
fan\_out = net.layers{l}.outputmaps \* net.layers{l}.kernelsize ^ 2;

卷积核初始化，1层卷积为1\*6个卷积核，2层卷积一共6\*12=72个卷积核。对于每个卷积输出featuremap

fan\_in = 表示该层的一个输出map，所对应的所有卷积核，包含的神经元的总数。1\*25,6\*25   
fan\_in = numInputmaps \* net.layers{l}.kernelsize ^ 2;   
fin =1\*25 or 6\*25   
fout=1\*6\*25 or 6\*12\*25

net.layers{l}.k{i}{j} = (rand(net.layers{l}.kernelsize) - 0.5) \* 2 \* sqrt(6 / (fan\_in + fan\_out));

**卷积降采样的参数初始化**

**尾部单层感知机的参数（权重和偏量）设置**

cnntrain.m

该函数用于训练CNN。   
**生成随机序列**，每次选取一个batch（50）个样本进行训练。   
**批训练**：计算50个随机样本的梯度，求和之后一次性更新到模型权重中。   
在批训练过程中调用：   
Cnnff.m 完成前向过程   
Cnnbp.m 完成误差传导和梯度计算过程   
Cnnapplygrads.m 把计算出来的梯度加到原始模型上去

## cnnff.m

## 取得CNN的输入 输入map，是一个的矩阵（矩阵维度随情况变化）

## 两次卷积核降采样层处理

## 尾部单层感知机的数据处理需要把subFeatureMap2连接成为一个(4\*4)\*12=192的向量，但是由于采用了50样本批训练的方法，subFeatureMap2被拼合成为一个192\*50的特征向量fv； fv作为单层感知机的输入，全连接的方式得到输出层

## cnnbp.m

该函数实现2部分功能，计算并传递误差，计算梯度

**计算误差和LossFunction**

**计算尾部单层感知机的误差**

**改变误差矩阵形状**,把单层感知机的输入层featureVector的误差矩阵，恢复为subFeatureMap2的4\*4二维矩阵形式

**误差在特征提取网络【卷积降采样层】的传播？？？**

如果本层是卷积层，它的误差是从后一层（降采样层）传过来，*误差传播实际上是用降采样的反向过程，也就是降采样层的误差复制为2\*2=4份。卷积层的输入是经过sigmoid处理的，所以，从降采样层扩充来的误差要经过sigmoid求导处理。*

如果本层是降采样层，他的误差是从后一层（卷积层）传过来，*误差传播实际是用卷积的反向过程，也就是卷积层的误差，反卷积（卷积核转180度）卷积层的误差，*

**计算特征抽取层（**卷积+降采样**）和尾部单层感知机的梯度**

## cnnapplygrads.m

该函数完成权重修改，更新模型的功能   
1更新特征抽取层的权重 weight+bias   
2 更新末尾单层感知机的权重 weight+bias

## cnntest.m

验证测试样本的准确率

计算预测错误的样本数量

计算错误率