# 利用LSTM网络和风险估计损失函数做股票交易

## 脚本框架（按照执行顺序）

### 1、rawdata.py

功能：将原始数据转换为按照日期排序的列表。

输入：[stock\_code].csv

输出：sorted\_data （列表类型）

1. 构建RawData类，包含6个变量：date、open、high、close、low、volume
2. 每条记录转换为一个RawData实例。
3. 列表sorted\_data按日期顺序存储实例，并返回。

### 2、chart.py

功能：根据需要提取sorted\_data每一个实例的变量，产生新变量并返回。

输入：sorted\_data

输出：moving\_features，moving\_labels

1. 提取sorted\_data列表的各个实例变量，得到按日期排序的date、open、high、close、low、volume共6个列表（list.shape = [1, data\_len]），并分别转换为numpy.asarray矩阵类型（asarray.shape = [1, data\_len]，其中data\_len等于日期数）。
2. 提供11种类型共61个的特征值，使用6个实例变量矩阵（numpy.asarray）产生新的特征值（利用ta-lib库），并合并为feature（numpy.asarray矩阵类型，shape = [feature\_num, data\_len]）。
3. 设置window值（本例设置为30天），将feature矩阵划分成n个（**x**）[feature\_num, window]矩阵，组成moving\_features（numpy.asarray矩阵，shape = [n, feature\_num, window]）。
4. 如果flatten = True, 利用numpy.ndarray.flatten(“F”)转化为按列展平的一位数组[1, feature\_num \* window]；组成moving\_features（numpy.asarray矩阵，shape = [n, feature\_num \* window]）
5. 同样，设置window值，利用close\_price产生window天内的价格变化值（**y**），得到moving\_labels（numpy.asarray矩阵，shape = [1, n]）。

### 3、smart\_trade.py

功能：构建LSTM(含DropoutWrapper)模型及损失函数。（LSTM结构详细见代码）。

训练模型，最小化LOSS函数（Loss = -100. \* mean(P \* (R - cost))）与优化权值和偏置值，并保存；

利用训练获取的优化参数，预测股票交易。

输入：

num\_step：时间步长数

input\_size：每个时间步长每个对象输入的特征值数量

train\_steps：训练迭代次数

batch\_size：每个时间步长将提供相应的batch\_size个对象

learning\_rate：初始学习速率

hidden\_size：LSTM单元数量

nclasses：输出值的数量

validation\_size：测试集规模

keep\_prob：保留系数（DropoutWrapper）

cost：成本

decay\_step：衰变速度

decay\_rate：衰变系数

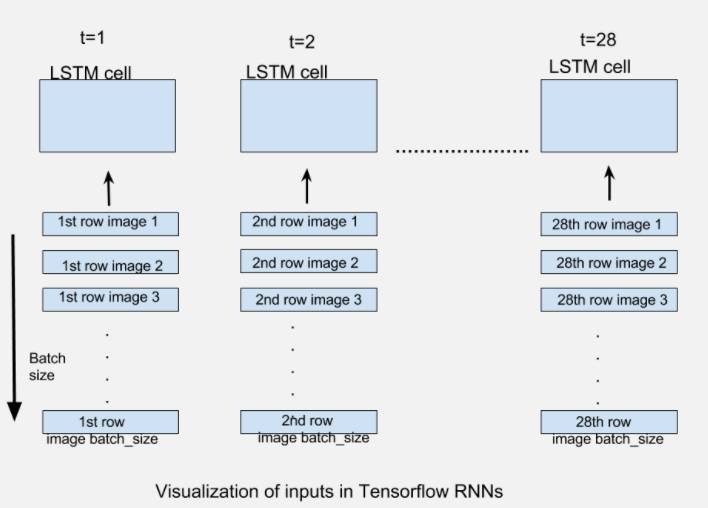
1. 训练：
2. 将特征集合划分成训练集train和测试集val（train + val = n）
3. 将moving\_features分成train\_features（shape = [train, feature\_num, window]）和val\_features（shape = [val, feature\_num, window]）。
4. 将moving\_labels分成train\_labels（shape = [1, train]）和val\_labels（shape = [1, val]）。
5. numpy.transpose()，转置训练集和测试集，得到train\_features（shape = [train, window, feature\_num]）和val\_features（shape = [val, window, feature\_num]）；train\_labels（shape = [train, 1]）和val\_features（shape = [val, 1]）
6. 将train\_features和train\_labels合并在DataSet类实例train\_set，在每次迭代训练中输入batch\_size个x’（转置）和y’（转置）。
7. 设置参数，将train\_set输入LSTM模型进行迭代训练，train\_labels和val\_labels进行优化。
8. 预测：
9. 将moving\_features和moving\_labels转置为shape = [n, window, feature\_num]和shape = [n, 1]。
10. 截取moving\_features和moving\_labels后第一维的val个数据集，合并在DataSet类实例val\_set，在每次迭代训练中输入batch\_size个x’（转置）和y’（转置）。
11. 设置参数，将val\_set输入LSTM模型进行预测。

### 4、dataset.py

功能：对**smart\_trade.py**处理的train\_features和train\_labels整合在DataSet类中，利用next\_batch在每次训练迭代输入batch\_size个x’和y’。batch\_features（shape = [batch\_size, window, feature\_num]）和batch\_lables（shape = [batch\_size, 1]）。

## LSTM模型的输入格式：

对于我们的MNIST图像的情况，我们有大小为28X28的图像。它们可以被推断为具有28行28像素的图像。我们将通过28个时间步骤展开我们的网络，使得在每个时间步长，我们可以输入一行28像素（input\_size），从而通过28个时间步长输入完整的图像。如果我们提供batch\_size图像的数量，每个时间步长将提供相应的batch\_size图像行。下图应该可以解释上述描述：



### 输入形式：

每次迭代输入batch\_features

(shape:[batch\_size, num\_step, input\_size]) 形式，其中

batch\_size: 每个时间步长将提供相应的batch\_size个对象

num\_step: 时间步长数

input\_size: 输入的特征数量

转换成inputs 参数格式: **inputs = tf.unstack(raw\_data ,num\_step,1)**

inputs参数是为了接受形状张量列表[batch\_size,input\_size]。该列表的长度是网络展开的时间步长数num\_step，即该列表的每个元素对应于我们展开网络的相应时间步长的输入。

### 特征值类型：