西门子CPU模块6ES7416-2XK02-0AB0

产品名称	西门子CPU模块6ES7416-2XK02-0AB0
公司名称	湖南西控自动化设备有限公司
价格	.00/件
规格参数	西门子:授权代理商 S7-400:一级代理商 德国:售后保障服务
公司地址	中国（湖南）自由贸易试验区长沙片区开元东路1306号开阳智能制造产业园（一期）4#栋301
联系电话	17838383235 17838383235

产品详情

机器学习与工业(一)-预测性维护

前 言：

随着自动化软件、大数据等技术的成熟，越来越多的工业场合开始将人工智能和机器学习引入到工业控制中去，比如我们常见的软测量、MPC控制、APC控制系统、寿命预测、状态检测、设备运行状态检测等系统。本次探讨内容主要以笼统的理论和一个简单的案例为探讨方向。 这一篇我们主要就针对如何进行相关知识的了解（仅属于个人观点和积累）首先我们需要的基础知识：

1、Python、Matlab、Mathematica、go、scala等至少掌握一门基础编程语言（只要会用就行）

2、了解机器学习的原理

3、了解深度学习的理论知识。推荐美国吴恩达的机器学习、深度学习的系列课程视频的观看笔记。

4、高数、概率论和线性代数的基础知识。

5、一颗好学的心

本文需要掌握的基础准备：

由于Matlab有点贵，就大学用了一段时间；本文以Python为例，需要掌握的基础准备：1、python的基础语法2、常用的数据结构3、常用的数据处理相关库3.1、Numpy:数据类型的转换；3.2、Pandas :读取原始的数据文件，数据集的处理，数据集后处理后的特征数据（如果是时间序列）处理，读取起来也是非常方便的。3.3、matplotlib:

Python绘图库的扛把子;后续特征的简单可视化，神经网络的训练过程的Loss，轴承或设备Remianing Useful Life （RUL）的可视化也是用得到的。3.4、Tensorflow：常用的谷歌开源深度学习框架、CNN、RNN等基础包。3.5、pyTorch：4、常用训练数据集网站：国内有阿里的天池，Jdata、国外的kaggle，nasa，DataFountain等，这些网站可以找到很多的数据集，用于训练或测试自己的模型。5、了解常见的机器学习算法监督学习：朴素贝叶斯、SVM、决策树、k-近邻算法等；非监督学习：PCA、SVD、K-Means；近比较火的：LSTM、GNU等；

有了这些基础知识之后我们要怎么用呢？

来，一起学吧！

以一个常见的深度学习进行预测性维修为例（核心代码来自微软demo）

LSTM模型属于RNN，而RNN适用于处理序列型数据，所以在常见的预测性维护和寿命预测中比较适用，这里有个题外话，股票预测也属于时间序列的一种，所以都会忍不住用LSTM去做股票预测，经过小伙伴长时间的测试发现股票预测目前还是不太现实，或者说我们技术还没学到家，目前做的比较多的还是策略。

步：针对数据集进行导入，和标准化（就是取列名）

这一步需要根据数据集的不同，输入方式的不同进行自定义进行。如果是CSV的话，只要简单的取列名就行了（数据处理代码部分自己写的）self.tapeNum = self.fileData[0] + self.fileData[1]*256 self.TapeNumCurrent = 0 self.arrayIndex = self.fileData[2:2997] self.fileJudge = self.fileData[2999] + self.fileData[3000]*256 self.reel = self.fileData[3200:3599] self.dataList = [] for i in range(0, self.tapeNum):   dataTtrack = DataTtrack() if i == 0:   dataTtrack.headBegin = 3600   else: dataTtrack.headBegin = self.dataList[i - 1].dataBegin + self.dataList[i-1].dataNum*2 dataTtrack.dataBegin = dataTtrack.headBegin + 240   dataTtrack.dataNum = self.fileData[dataTtrack.headBegin + 114] + self.fileData[dataTtrack.headBegin+115] * 256 dataTtrack.sampleInterval = (self.fileData[dataTtrack.headBegin+116] + self.fileData[dataTtrack.headBegin+117] * 256)/5   dataTtrack.waveSpeed = self.fileData[dataTtrack.headBegin + 238] + self.fileData[dataTtrack.headBegin+239] * 256   dataTtrack.data = [] for j in range(dataTtrack.dataBegin, dataTtrack.dataBegin+dataTtrack.dataNum,2): dataCache = self.fileData[j] + self.fileData[j+1]*256   if self.fileData[j+1] >= 128 : dataCache -=65536   dataTtrack.data.append(dataCache) self.dataList.append(dataTtrack)

第二步：定义任务，一般情况下就以下这三种任务模式，选二分类吧。

1、利用回归类预测剩余使用寿命（RUL）或故障时间；2、二分类预测设别是否会在特定时间范围内失效；3、多类分类：预测设备是否会在不同的时间窗口失效；

计算RUL

```
# Data Labeling - generate column RUL
rul = pd.DataFrame(train_df.groupby('id')['cycle'].max()).reset_index() rul.columns = ['id', 'max']
train_df = train_df.merge(rul, on=['id'], how='left')
train_df['RUL'] = train_df['max'] - train_df['cycle'] train_df.drop('max', axis=1, inplace=True)
```

train_df.head()

创建标签

generate label columns for training data w1 = 30 w0 = 15
train_df['label1'] = np.where(train_df['RUL'] <= w1, 1, 0) train_df['label2'] = train_df['label1']
train_df.loc[train_df['RUL'] <= w0, 'label2'] = 2 train_df.head()

第三步：创建模型

说到寿命预测、我们常用的就是LSTM模型。

标准化

# MinMax normalization train_df['cycle_norm'] = train_df['cycle']
cols_normalize = train_df.columns.difference(['id','cycle','RUL','label1','label2'])
min_max_scaler = preprocessing.MinMaxScaler()
norm_train_df = pd.DataFrame(min_max_scaler.fit_transform(train_df[cols_normalize]),
columns=cols_normalize,
index=train_df.index) join_df = train_df[train_df.columns.difference(cols_normalize)].join(norm_train_df)
train_df = join_df.reindex(columns = train_df.columns) train_df.head()

创建3D数据（样本、时间步、特征）

def gen_sequence(id_df, seq_length, seq_cols): data_array = id_df[seq_cols].values
num_elements = data_array.shape[0] for start, stop in zip(range(0, num_elements-
seq_length), range(seq_length, num_elements)): yield data_array[start:stop, :]

创建LSTM模型，我们这里选用TensorFlow里的Keras

# build the network nb_features = seq_array.shape[2] nb_out = label_array.shape[1]
model = Sequential() model.add(LSTM(input_shape=(sequence_length, nb_features),
units=100, return_sequences=True)) model.add(Dropout(0.2)) model.add(LSTM(units=50,
return_sequences=False)) model.add(Dropout(0.2)) model.add(Dense(units=nb_out, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

第四步：模型训练和预测

fit the network 这里有因为Tensorflow版本问题，有过修正
model.fit(seq_array, label_array, epochs=10, batch_size=200, validation_split=0.05, verbose=1,
  callbacks = [keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience=0, ve
rbose=0, mode='auto')]) # training metrics predict_x=model.predict(seq_array,verbose=1, batch_size=200)
# make predictions and compute confusion matrix y_pred=(predict_x>= 0.5).astype("int32")
y_true = label_array print('Confusion matrix\n- x-axis is true labels.\n- y-
axis is predicted labels') cm = confusion_matrix(y_true, y_pred)