Arkenstone

slip the surly bond of earth to touch the face of god

昵称：Arkenstone
园龄：1年5个月
粉丝：23
关注：0
+加关注

< 2017年10月 >

日	一	二	三	四	五	六
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

Python中利用LSTM模型进行时间序列预测分析

时间序列模型

时间序列预测分析就是利用过去一段时间内某事件时间的特征来预测未来一段时间内该事件的特征。这是一类相对比较复杂的预测建模问题，和回归分析模型的预测不同，时间序列模型是依赖于事件发生的先后顺序的，同样大小的值改变顺序后输入模型产生的结果是不同的。

举个栗子：根据过去两年某股票的每天的股价数据推测之后一周的股价变化；根据过去2年某店铺每周想消费人数预测下周来店消费的人数等等

RNN 和 LSTM 模型

时间序列模型最常用最强大的的工具就是递归神经网络（recurrent neural network, RNN）。相比与普通神经网络的各计算结果之间相互独立的特点，RNN的每一次隐含层的计算结果都与当前输入以及上一次的隐含层结果相关。通过这种方法，RNN的计算结果便具备了记忆之前几次结果的特点。

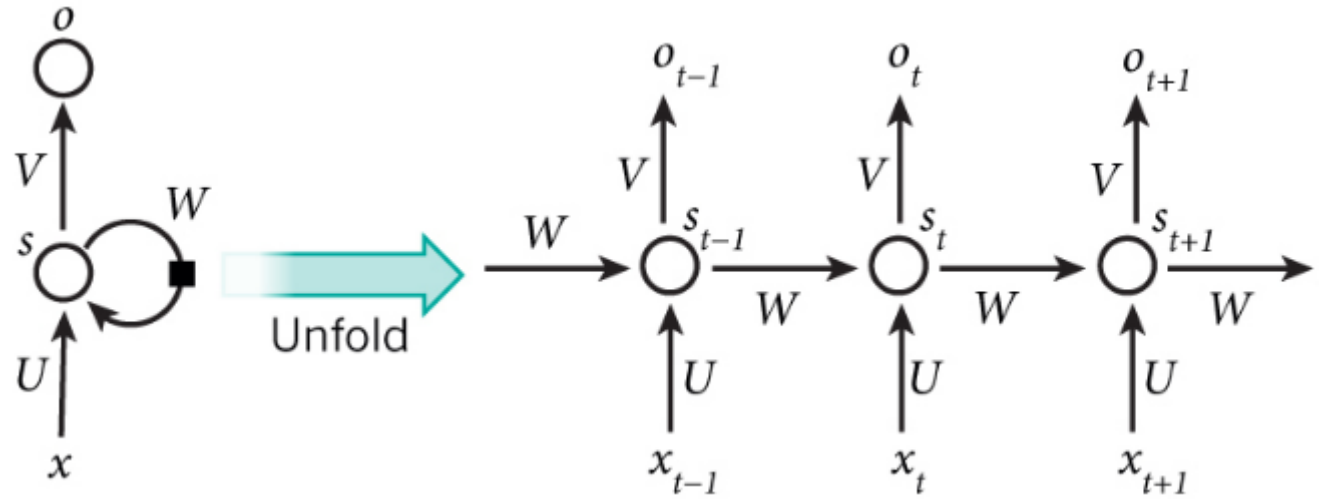
python(31)
opencv(5)
R(5)
Tensorflow(5)
Stanford NLP(4)
git(3)
机器学习(3)
多线程(2)
Ubuntu(2)
统计检验(2)
更多

随笔档案

2017年9月 (2)
2017年8月 (4)
2017年7月 (3)
2017年6月 (4)
2017年5月 (5)
2017年4月 (2)
2017年3月 (2)
2017年2月 (2)
2017年1月 (5)
2016年12月 (4)
2016年11月 (7)
2016年10月 (4)
2016年9月 (2)
2016年8月 (6)
2016年7月 (1)
2016年6月 (3)
2016年5月 (4)
2016年4月 (4)

最新评论

典型的RNN网路结构如下：



右侧为计算时便于理解记忆而产开的结构。简单说，x为输入层，o为输出层，s为隐含层，而t指第几次的计算；V,W,U为权重，其中计算第t次的隐含层状态时为St = f(U*Xt + W*St-1)，实现当前输入结果与之前的计算挂钩的目的。对RNN想要更深入的了解可以戳**这里**。

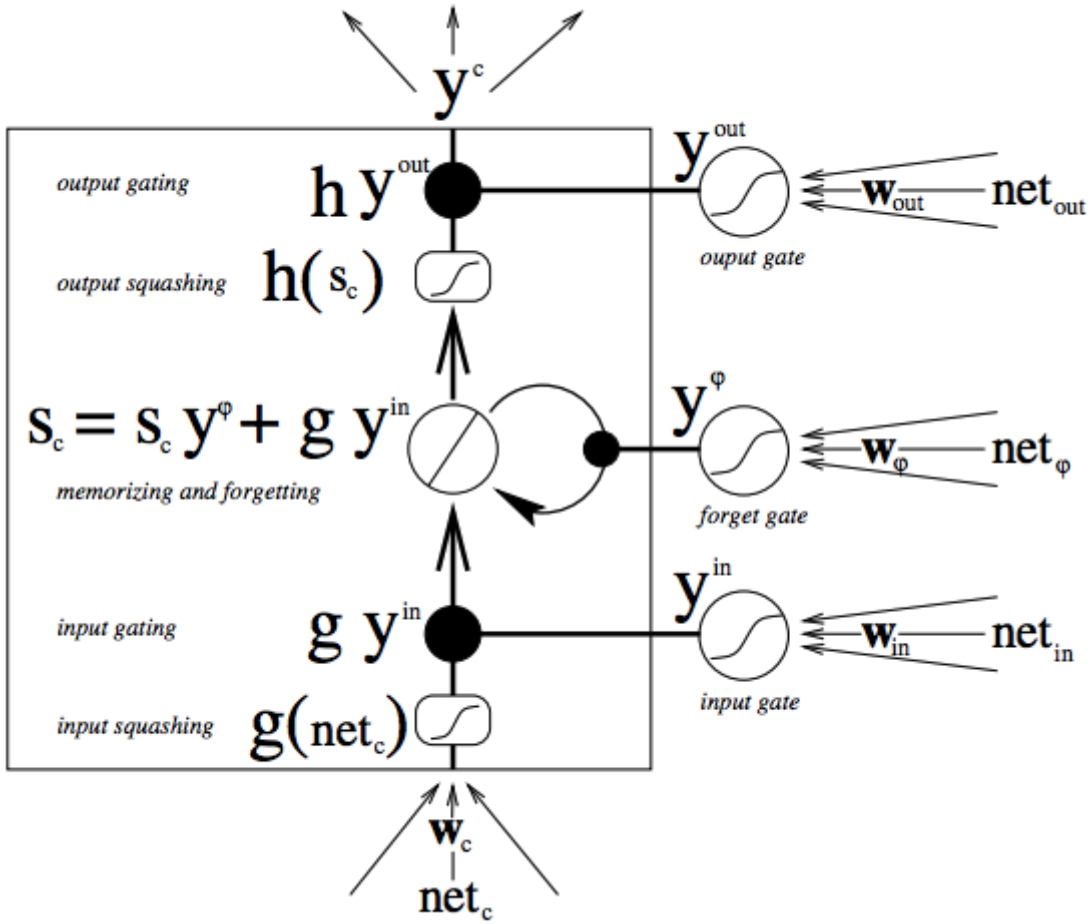
RNN的局限：

由于RNN模型如果需要实现长期记忆的话需要将当前的隐含态的计算与前n次的计算挂钩，即St = f(U*Xt + W1*St-1 + W2*St-2 + ... + Wn*St-n)，那样的话计算量会呈指数式增长，导致模型训练的时间大幅增加，因此RNN模型一般直接用来进行长期记忆计算。

LSTM模型

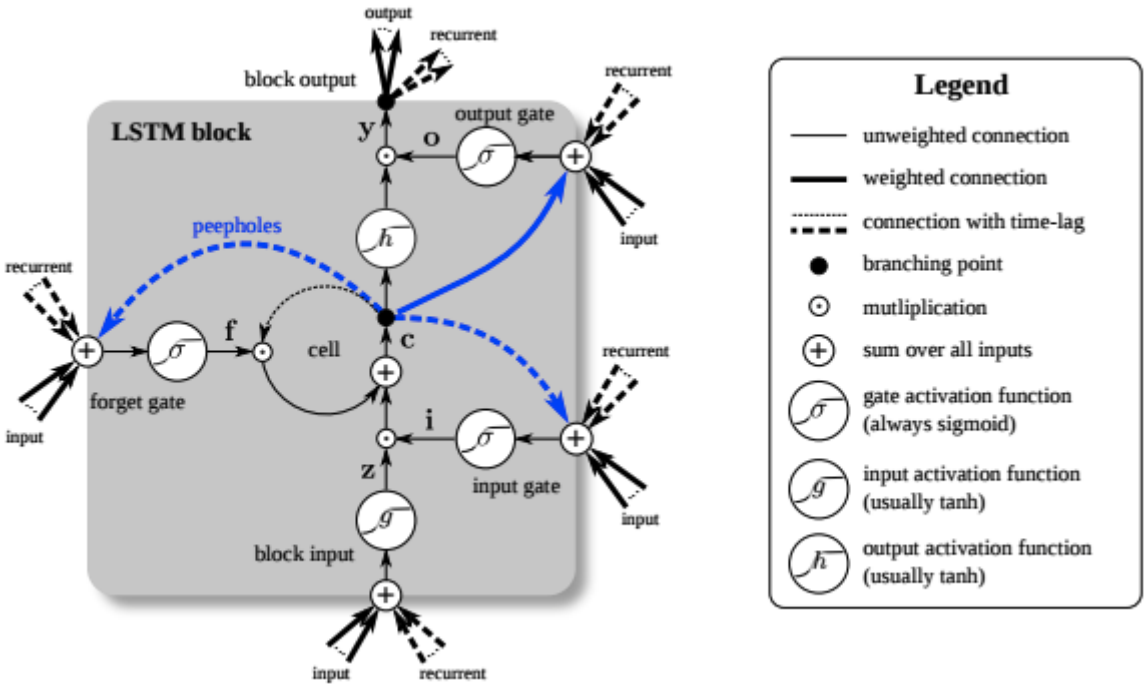
LSTM（Long Short-Term Memory）模型是一种RNN的变型，最早由Juergen Schmidhuber提出的。经典的LSTM模型结构如下：

1. Re:Python中利用LSTM模型进行时间序列预测分析
@cherry_whu这个一般是凭经验值来设定的，也可以用网格搜索或者随机收缩的进行训练然后根据训练结果来决定...
--Arkenstone
2. Re:Python中利用LSTM模型进行时间序列预测分析
您好，请问一下，LSTM层的Input_dim和output_dim是怎么确定呢？
--cherry_whu
3. Re:Ubuntu16.04下安装OpenCV3.2.0
@管真多这句命令是在动态库添加opencv编译产生的模块的路径。原文有点错误：CMAKE_INSTALL_PREFIX=/usr/local的时候是这种写法（默认情况），但是如果使用anaconda中.....
--Arkenstone
4. Re:Ubuntu16.04下安装OpenCV3.2.0
问一下：sudo /bin/bash -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'这句是什么意思。 路径怎么改。。。我按照这个.....
--管真多
5. Re:用python将MSCOCO和Caltech行人检测数据集转化成VOC格式
@破晓。前面150-153行的输入输出路径确认下，我这边的话seq路径下包括像这样：annotations set00 set01 set02 set03 set04 set05 set06 set0.....



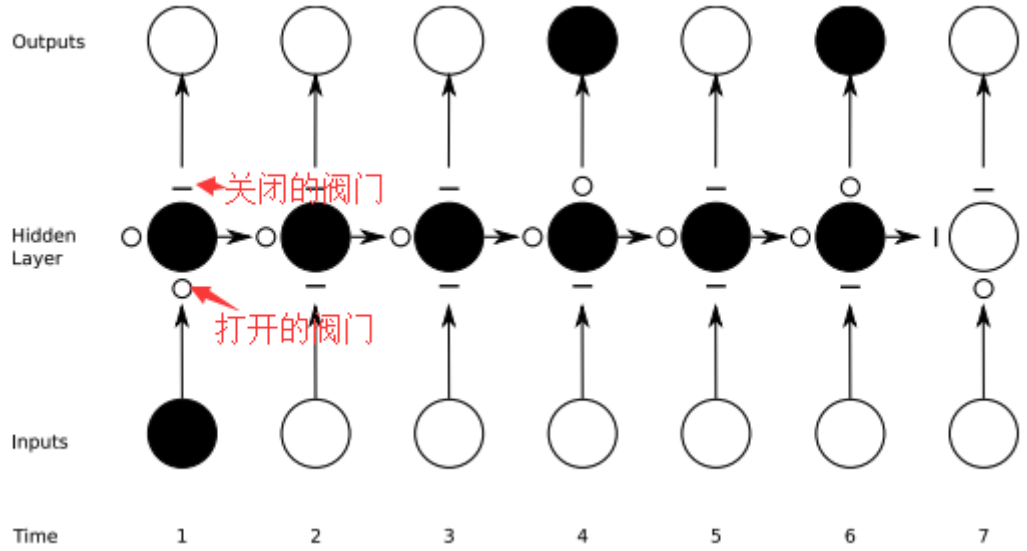
LSTM的特点就是在RNN结构以外添加了各层的阀门节点。阀门有3类：遗忘阀门（forget gate），输入阀门（input gate）和输出阀门（output gate）。这些阀门可以打开或关闭，用于将判断模型网络的记忆态（之前网络的状态）在该层输出的结果是否达到阈值从而加入到当前该层的计算中。如图中所示，阀门节点利用sigmoid函数将网络的记忆态作为输入计算；如果输出结果达到阈值则将该阀门输出与当前层的的计算结果相乘作为下一层的输入（**PS：这里的相乘是在指矩阵中的逐元素相乘**）；如果没有达到阈值则将该输出结果遗忘掉。每一层包括阀门节点的权重都会在每一次模型反向传播训练过程中更新。更具体的LSTM的判断计算过程如下图所示：

--Arkenstone



LSTM模型的记忆功能就是由这些阀门节点实现的。当阀门打开的时候，前面模型的训练结果就会关联到当前的模型计算，而当阀门关闭的时候之前的计算结果就不再影响当前的计算。因此，通过调节阀门的开关我们就可以实现早期序列对最终结果的影响。而当你不不希望之前结果对之后产生影响，比如自然语言处理中的开始分析新段落或新章节，那么把阀门关掉即可。（对LSTM想要更具体的了解可以戳这里）

下图具体演示了阀门是如何工作的：通过阀门控制使序列第1的输入的变量影响到了序列第4,6的的变量计算结果。



黑色实心圆代表对该节点的计算结果输出到下一层或下一次计算；空心圆则表示该节点的计算结果没有输入到网络或者没有从上一次收到信号。

Python中实现LSTM模型搭建

Python中有不少包可以直接调用来构建LSTM模型，比如pybrain, kears, tensorflow, cikit-neuralnetwork等（更多戳**这里**）。这里我们选用**keras**。（PS：如果操作系统用的linux或者mac，强推Tensorflow！！！）

因为LSTM神经网络模型的训练可以通过调整很多参数来优化，例如activation函数，LSTM层数，输入输出的变量维度等，调节过程相当复杂。这里只举一个最简单的应用例子来描述LSTM的搭建过程。

应用实例

基于某家店的某顾客的历史消费的时间推测该顾客前下次来店的时间。具体数据如下所示：

```
消费时间
2015-05-15 14:03:51
2015-05-15 15:32:46
2015-06-28 18:00:17
2015-07-16 21:27:18
2015-07-16 22:04:51
2015-09-08 14:59:56
```

```
..
..
```

具体操作：

1. 原始数据转化

首先需要将时间点数据进行数值化。将具体时间转化为时间段用于表示该用户相邻两次消费的时间间隔，然后再导入模型进行训练是比较常用的手段。转化后的数据如下：

```
消费间隔
0
44
18
0
54
..
..
```

2.生成模型训练数据集（确定训练集的窗口长度）

这里的窗口指需要几次消费间隔用来预测下一次的消费间隔。这里我们先采用窗口长度为3， 即用t-2, t-1,t次的消费间隔进行模型训练，然后用t+1次间隔对结果进行验证。数据集格式如下：X为训练数据，Y为验证数据。

PS： 这里说确定也不太合适，因为窗口长度需要根据模型验证结果进行调整的。

```
X1    X2    X3    Y
0    44    18    0
44    18    0    54
..
..
```

注：直接这样预测一般精度会比较差，可以把预测值Y根据数值bin到几类，然后用转换成one-hot标签再来训练会比较好。比如如果把Y按数值范围分到五类（1：0-20，2：20-40，3：40-60，4：60-80，5：80-100）上式可化为：

```
X1    X2    X3    Y
0    44    18    0
44    18    0    4
...
```

Y转化成one-hot以后则是(关于one-hot编码可以参考这里)

```
1    0    0    0    0
0    0    0    0    1
```

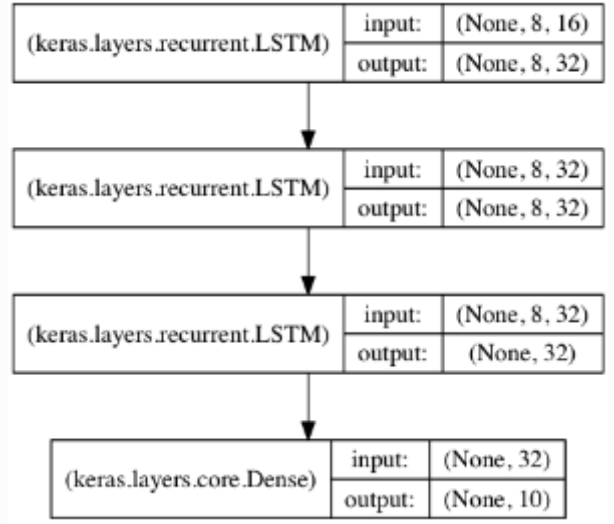
```
...
```

3. 网络模型结构的确定和调整

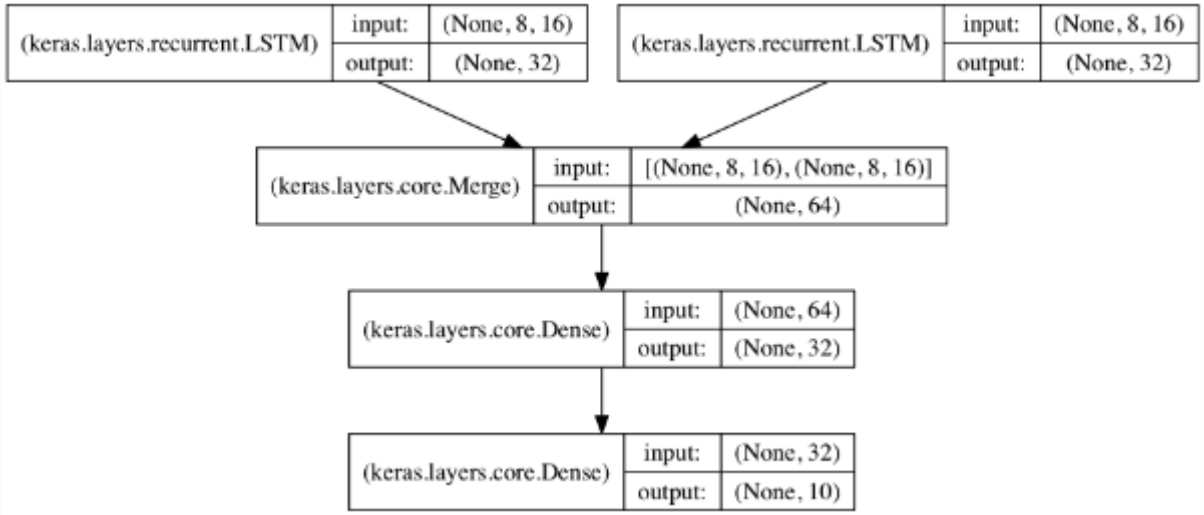
这里我们使用python的keras库。（用java的同学可以参考下deeplearning4j这个库）。网络的训练过程设计到许多参数的调整：比如

- 需要确定LSTM模块的激活函数（activation fucntion）（keras中默认的是tanh）；
- 确定接收LSTM输出的完全连接人工神经网络（fully-connected artificial neural network）的激活函数（keras中默认为linear）；
- 确定每一层网络节点的舍弃率（为了防止过度拟合（overfit）），这里我们默认值设定为0.2；
- 确定误差的计算方式，这里我们使用均方误差（mean squared error）；
- 确定权重参数的迭代更新方式，这里我们采用RMSprop算法，通常用于RNN网络。
- 确定模型训练的epoch和batch size（关于模型的这两个参数具体解释戳**这里**）

  一般来说LSTM模块的层数越多（**一般不超过3层**，再多训练的时候就比较难收敛），对高级别的时间表示的学习能力越强；同时，最后会加一层普通的神经网路层用于输出结果的降维。典型结构如下：



如果需要将多个序列进行同一个模型的训练，可以将序列分别输入到独立的LSTM模块然后输出结果合并后输入到普通层。结构如下：



4. 模型训练和结果预测

将上述数据集按4:1的比例随机拆分为训练集和验证集，这是为了防止过度拟合。训练模型。然后将数据的X列作为参数导入模型便可得到预测值，与实际的Y值相比便可得到该模型的优劣。

实现代码

1. 时间间隔序列格式化成所需的训练集格式

```
import pandas as pd
import numpy as np

def create_interval_dataset(dataset, look_back):
    """
    :param dataset: input array of time intervals
    :param look_back: each training set feature length
    :return: convert an array of values into a dataset matrix.
    """
    dataX, dataY = [], []
    for i in range(len(dataset) - look_back):
        dataX.append(dataset[i:i+look_back])
        dataY.append(dataset[i+look_back])
    return np.asarray(dataX), np.asarray(dataY)

df = pd.read_csv("path-to-your-time-interval-file")
```

```
dataset_init = np.asarray(df)    # if only 1 column
dataX, dataY = create_interval_dataset(dataset, lookback=3)    # look back if the training set sequence length
```

这里的输入数据来源是csv文件，如果输入数据是来自数据库的话可以参考**这里**

1. LSTM网络结构搭建

```
import pandas as pd
import numpy as np
import random
from keras.models import Sequential, model_from_json
from keras.layers import Dense, LSTM, Dropout

class NeuralNetwork():
    def __init__(self, **kwargs):
        """
        :param **kwargs: output_dim=4: output dimension of LSTM layer; activation_lstm='tanh': activation function for LSTM layers; activation_dense='relu': activation function for Dense layer; activation_last='sigmoid': activation function for last layer; drop_out=0.2: fraction of input units to drop; np_epoch=10, the number of epoches to train the model. epoch is one forward pass and one backward pass of all the training examples; batch_size=32: number of samples per gradient update. The higher the batch size, the more memory space you'll need; loss='mean_square_error': loss function; optimizer='rmsprop'
        """
        self.output_dim = kwargs.get('output_dim', 8)
        self.activation_lstm = kwargs.get('activation_lstm', 'relu')
        self.activation_dense = kwargs.get('activation_dense', 'relu')
        self.activation_last = kwargs.get('activation_last', 'softmax')    # softmax for multiple output
        self.dense_layer = kwargs.get('dense_layer', 2)     # at least 2 layers
        self.lstm_layer = kwargs.get('lstm_layer', 2)
        self.drop_out = kwargs.get('drop_out', 0.2)
        self.nb_epoch = kwargs.get('nb_epoch', 10)
        self.batch_size = kwargs.get('batch_size', 100)
        self.loss = kwargs.get('loss', 'categorical_crossentropy')
        self.optimizer = kwargs.get('optimizer', 'rmsprop')

        def NN_model(self, trainX, trainY, testX, testY):
        """
```

```
        :param trainX: training data set
        :param trainY: expect value of training data
        :param testX: test data set
        :param testY: epect value of test data
        :return: model after training
        """
        print "Training model is LSTM network!"
        input_dim = trainX[1].shape[1]
        output_dim = trainY.shape[1] # one-hot label
        # print predefined parameters of current model:
        model = Sequential()
        # applying a LSTM layer with x dim output and y dim input. Use dropout parameter to avoid
overfitting
        model.add(LSTM(output_dim=self.output_dim,
                       input_dim=input_dim,
                       activation=self.activation_lstm,
                       dropout_U=self.drop_out,
                       return_sequences=True))
        for i in range(self.lstm_layer-2):
            model.add(LSTM(output_dim=self.output_dim,
                       input_dim=self.output_dim,
                       activation=self.activation_lstm,
                       dropout_U=self.drop_out,
                       return_sequences=True))
        # argument return_sequences should be false in last lstm layer to avoid input dimension
incompatibility with dense layer
        model.add(LSTM(output_dim=self.output_dim,
                       input_dim=self.output_dim,
                       activation=self.activation_lstm,
                       dropout_U=self.drop_out))
        for i in range(self.dense_layer-1):
            model.add(Dense(output_dim=self.output_dim,
                        activation=self.activation_last))
        model.add(Dense(output_dim=output_dim,
                        input_dim=self.output_dim,
                        activation=self.activation_last))
        # configure the learning process
        model.compile(loss=self.loss, optimizer=self.optimizer, metrics=['accuracy'])
        # train the model with fixed number of epoches
```

```
        model.fit(x=trainX, y=trainY, nb_epoch=self.nb_epoch, batch_size=self.batch_size,
validation_data=(testX, testY))
        # store model to json file
        model_json = model.to_json()
        with open(model_path, "w") as json_file:
            json_file.write(model_json)
        # store model weights to hdf5 file
        if model_weight_path:
            if os.path.exists(model_weight_path):
                os.remove(model_weight_path)
            model.save_weights(model_weight_path) # eg: model_weight.h5
        return model
```

这里写的只涉及LSTM网络的结构搭建，至于如何把数据处理规范化成网络所需的结构以及把模型预测结果与实际值比较统计的可视化，就需要根据实际情况做调整了。具体脚本可以参考下**这个**

参考文档：

[力推]： Understanding LSTMs

1. Keras Documnet
2. What is batch size in neural network?
3. Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras
4. Save Your Neural Network Model to JSON
5. RECURRENT NEURAL NETWORKS TUTORIAL, PART 1 – INTRODUCTION TO RNNS
6. A Beginner's Guide to Recurrent Networks and LSTMs
7. Pybrain time series prediction using LSTM recurrent nets
8. PyBrain Document
9. Recurrent neural network for predicting next value in a sequence
10. What are some good Python libraries that implement LSTM networks?

标签: python, Keras, 机器学习, LSTM, 时间序列分析

Arkenstone
关注 - 0
粉丝 - 23

0 0

+加关注

« 上一篇：Windows下运行python脚本报错“ImportError： No Module named ...”的解决方法

» 下一篇：python中利用logging包进行日志记录时的logging.level设置选择

posted on 2016-08-23 13:06 Arkenstone 阅读(32063) 评论(29) 编辑 收藏

评论:

#1楼 2016-09-20 08:51 | zhangge1980

请教一下楼主：例子给出根据历史t-m,...,t-2, t-1,t的数据预测下一个时刻t+1的数据，可否用多个历史数据一次预测出未来t+1，t+2， t+3,...t+n个的数据而不是只预测下一个的数据？比如用过去2年每天的股票数据来预测未来一周的，而不是未来一天的？非常感谢。

支持(0) 反对(0)

#2楼[楼主] 2016-09-26 20:13 | Arkenstone

@ zhangge1980

这个当然可以啊。因为输入数据和验证数据都是矩阵形式，文中预测下一次无非是输出的m x 1维的数据，你想要预测多天的无非就是输出结果是m x n维的数据，原理是一样的。

不过要注意的是应为lstm是依赖序列关系的，所以你的下一次预测结果也会影响之后的预测，如果当中有偏差，这个偏差会累积到之后的预测结果，因此长时间的预测的准确性都是不高，但如果只是看趋势的话倒影响不大。

支持(0) 反对(0)

#3楼 2016-11-21 22:11 | maocy

请教楼主，如果数据是多维的 请问如何调整？ 如用前5天的开盘价、收盘价、交易量去预测第6天的收盘价

支持(0) 反对(0)

#4楼[楼主] 2016-11-23 01:21 | Arkenstone

@ maocy

如果有多个维度可以用这几个维度的序列分别训练对应的lstm模块，然后把它们的输出用一层/多层dense层合并到一起即可。可以参考keras文档guide to sequential model的这个部分：Two merged LSTM encoders for classification over two parallel sequences

支持(0) 反对(0)

#5楼 2016-11-24 22:33 | maocy
那请教一下， 首层input_shape=(8,32) ，这个8不是时间长度 32不是代表数据维度吗？ 能否给个QQ之类的 我的654946450 Q/微信 谢谢
支持(0) 反对(0)

#6楼[楼主] 2016-11-25 00:45 | Arkenstone
@ maocy
是的，你的说法没错。单独训练再融合的效果和你事先把数据合并一起训练的效果是一样的。
支持(0) 反对(0)

#7楼 2016-12-14 16:19 | rain1
请问楼主，你的原始数据处理完是什么样子的呢？
支持(0) 反对(0)

#8楼[楼主] 2016-12-16 00:39 | Arkenstone
@ rain1
就是具体步骤第二点里面给的那样呀：
X1 X2 X3 Y
0 44 18 0
44 18 0 54
...
支持(0) 反对(0)

#9楼 2016-12-16 18:21 | melisa
请问一下楼主，给lstm的数据就只是一维的吗？就是一列数据窗口数据咯？
支持(0) 反对(0)

#10楼[楼主] 2016-12-17 16:54 | Arkenstone
@ melisa
如果你的输入数据有多维，那么只需将模型网络的输入节点维度调整即可；或者将多维数据拆分成一维的单独训练之后合并

支持(0) 反对(0)

#11楼 2017-04-26 20:35 | 心冷2080
厉害了我的哥

支持(0) 反对(0)

#12楼 2017-05-06 10:32 | DDD_D
数据链接失效了 打不开啊 求源数据啊！

支持(0) 反对(0)

#13楼[楼主] 2017-05-07 01:05 | Arkenstone
@ DDD_D
之前把代码调整了挺多的，提供的只是从数据库提取数据的参考代码病不是数
据：https://github.com/CasiaFan/time_seires_prediction_using_lstm/blob/master/preprocessing.py

支持(0) 反对(0)

#14楼 2017-05-11 15:10 | onionwyl
楼主，麻烦问一下，我现在想做的是通过一定的时刻通过同一个路段的行车时间来预测未来同一时刻通过该路段的行车时间，我的数据集该如何处理成训练集呢？
是用t(n-2) t(n-1) t(n) 作为X，t(n+1)作为Y么？

支持(0) 反对(0)

#15楼[楼主] 2017-05-12 16:49 | Arkenstone
@ onionwyl
这个看你具体需求进行调整，如果是预测未来某一个时间点可以用t时刻前n个节点作为输入，t时刻作为y；如果是预测某段时间，那么就用

sequence-to-sequence作为输入输出，t时刻前n个作为输入，t时刻后m个作为输出。就是别忘了把网络的结构进行相应的调整。

支持(0) 反对(0)

#16楼 2017-05-16 00:11 | onionwyl

麻烦再问个问题，t时刻前n个节点是作为一个n维输入输入进去的么？还是每一个节点对应的output是0？

支持(0) 反对(0)

#17楼[楼主] 2017-05-18 10:12 | Arkenstone

@ onionwyl

对于每个测试样本，输入为有n个元素的一维向量，输出可以为数值所对应one-hot编码

支持(0) 反对(0)

#18楼 2017-06-15 17:33 | msw1992

楼主，麻烦问一下，如果我现在有一个时间序列A需要进行预测，同时它受到两个时间序列B和C的影响，请问我要怎么构建LSTM的输入？

支持(0) 反对(0)

#19楼[楼主] 2017-06-26 22:41 | Arkenstone

@ msw1992

如果A, B, C是同类型的数据（比如都是整数型），那么你可以把这三组数按顺序合并然后作为一个整体输入（从3个n x 1维向量变成一个n x 3维向量）；如果不是同类型的数据，可以构建3个lstm分支分别以A，B，C作为输入，然后将输入到全连接层之前的向量合并成一个（3个m x 1变成一个3m x 1的向量）。

支持(0) 反对(0)

#20楼 2017-07-03 18:38 | msw1992

楼主，请问做预测的时候数据要归一化么？我看有别的作者写一定要将数据归一化，代码如下：

```
def normalise_windows(window_data):
normalised_data = []
for window in window_data:
normalised_window = [((float(p) / float(window[0])) - 1) for p in window]
```

normalised_data.append(normalised_window)
return normalised_data
我测试过，如果不做归一化效果特别差。我不是很能理解。

支持(0) 反对(0)

#21楼[楼主] 2017-07-06 15:01 | Arkenstone
@ msw1992
是的，实际情况一般都是要normalization，normalization的代码在[这里](https://github.com/CasiaFan/time_seires_prediction_using_lstm/blob/master/preprocessing.py#L109-L124),一般是对全局数据做统一的normalization，这样原始数据和归一化后的数据转换就很方便，而且原始数据特征保存比较好

支持(0) 反对(0)

#22楼 2017-07-06 16:38 | msw1992
谢谢楼主，我还有一个疑问。请问，如果有三个时间序列A,B,C，用B,C来拟合A，即A=f(B,C)，通过训练数据得到训练后的网络。训练的时候数据是经过了normalization的，此时如果拿已经训练好的模型来计算序列A的时候，此时并不知道时间序列A的均值和标准差，但通过网络得到的结果是normalization之后的结果，请问这时候怎么将其反归一化得到最后的结果呢？

支持(0) 反对(0)

#23楼[楼主] 2017-07-11 13:13 | Arkenstone
@ msw1992
一般是将历史数据计算出的均值和方差作为当前测试数据默认值来进行处理的

支持(0) 反对(0)

#24楼 2017-07-12 12:29 | msw1992
谢谢楼主，但是这样得到的结果不会误差很大么？

支持(0) 反对(0)

#25楼[楼主] 2017-07-13 09:55 | Arkenstone

@ msw1992
这种处理就是假设测试和训练的数据的分布是近似的，那么对于一般情况下同一类型数据这种假设是可行的

支持(0) 反对(0)

#26楼 2017-08-04 16:19 | ypaafw
楼主你好，我想请问一下，关于LSTM，有没有什么经典的网络结构设置，类似于CNN中的ResNet， VGG等,另外楼主在文中说LSTM结构越多，能够解释更加复杂的时序模型，但我在其他地方看到说，LSTM一般就是2到3层，更多的话，难以训练，我想请问你能否推荐一下经典的关于LSTM应用的文章，多谢

支持(0) 反对(0)

#27楼[楼主] 2017-08-08 22:07 | Arkenstone
@ ypaafw
根据这篇文章(http://colah.github.io/posts/2015-08-Understanding-LSTMs)的说法，几乎每种LSTM的应用都有微小的区别，比较著名的变种结构有peehole connection lstm，以及GRU, Depth gated RNN, clockwork RNN之类的。

第二点你说的很对的，一般lstm层不会超过3层，再多就比较难收敛了。关于LSMT应用的话其实有很多，最有名的应该是14年的seq2seq机器翻译(https://arxiv.org/pdf/1409.3215.pdf)跟语音识别（http://ieeexplore.ieee.org/abstract/document/6638947/），还有image caption（https://arxiv.org/pdf/1411.4555v2.pdf）跟现在很火的图像合成（https://arxiv.org/pdf/1502.04623v2.pdf）跟GAN。。基本凡是可以生成序列的模型都会用到。

支持(0) 反对(0)

#28楼 2017-09-25 17:33 | cherry_whu
您好，请问一下，LSTM层的Input_dim和output_dim是怎么确定呢？

支持(0) 反对(0)

#29楼[楼主] 2017-10-16 00:02 | Arkenstone
@ cherry_whu
这个一般是凭经验值来设定的，也可以用网格搜索或者随机收缩的进行训练然后根据训练结果来决定

支持(0) 反对(0)

· 做到这一点，你也可以成为优秀的程序员
· 写给立志做码农的大学生
» 更多知识库文章...