

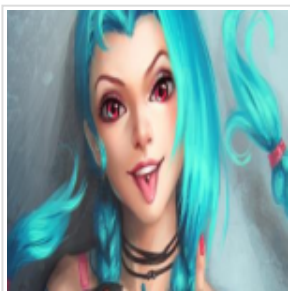
## You will~My hand

目录视图

摘要视图

RSS 订阅

## 个人资料



ma416539432

访问：11364次

积分：175

等级：BLOG &gt; 2

排名：千里之外

原创：4篇 转载：1篇

译文：2篇 评论：2条

## 文章搜索

异步赠书：9月重磅新书升级，本本经典 【免费直播】Python最佳学习路线！ 程序员9月书讯 节后荐书：Python、PyQt5、Kotlin（评论送书）

## 在keras 上实践,通过keras例子来理解lstm循环神经网络

标签：神经网络

2016-12-07 19:05

5719人阅读

评论

分类：机器学习工具（3） 机器学习原理（5） 机器学习实践（5）

目录(?)

[+]

本文是对这篇博文的翻译和实践：

<http://machinelearningmastery.com/understanding-stateful-lstm-recurrent-neural-networks-python-keras/>

关闭

阅读本文以后，你将要知道：

- 一）怎么在keras上实习一个普通的lstm循环神经网络。
- 二）在lstm中怎样小心的利用好时间状态特征
- 三）怎样在lstm上实现状态的预测。

本文在一个很简单的例子上说明lstm的使用和lstm的特点，通过对这个简化序列预测问题和序列预测问题有更高的理解和使用。

先导入我们会用到的包。



## 文章分类

java (0)

推荐系统原理 (1)

机器学习原理 (6)

机器学习实践 (6)

机器学习工具 (4)

推荐系统实践 (0)

统计学的知识 (0)

深度学习 (0)

数据结构与算法 (0)

sql (0)

## 文章存档

2016年12月 (3)

2016年11月 (4)

## 阅读排行

在keras 上实践,通过kera (5682)

机器学习之降维方法总结 (2509)

从数据预处理到特征工程 (952)

用gensim的word2vector (864)

sklearn 的优雅数据挖掘 (691)

ML机器学习基于树的家族 (244)

随机森林—bagging算法 (208)

## 评论排行

```

1 import numpy
2 from keras.models import Sequential
3 from keras.layers import Dense
4 from keras.layers import LSTM
5 from keras.utils import np_utils

```

然后我们把我们的数据集—一个字母表转化为keras可以处理的形式。

```

1 # define the raw dataset
2 alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
3 # create mapping of characters to integers (0-25) and the reverse
4 char_to_int = dict((c, i) for i, c in enumerate(alphabet))
5 int_to_char = dict((i, c) for i, c in enumerate(alphabet))

```

ok, 现在我们在来制造我的input和output

数据集。

我们先创造这样一个数据集, 用一个字母, 来预测下一个字母是什么。

```

1 # prepare the dataset of input to output pairs encoded as integers
2 seq_length = 1
3 dataX = []
4 dataY = []
5 for i in range(0, len(alphabet) - seq_length, 1):
6     seq_in = alphabet[i:i + seq_length]
7     seq_out = alphabet[i + seq_length]
8     dataX.append([char_to_int[char] for char in seq_in])
9     dataY.append(char_to_int[seq_out])
10    print seq_in, '->', seq_out

```

我们运行上面的代码, 来观察现在我们的input和output数据集是这样一种情况

:

关闭



在keras 上实践,通过kera (2)  
ML机器学习基于树的家族 (0)  
随机森林—bagging算法 (0)  
机器学习之降维方法总结 (0)  
从数据预处理到特征工程 (0)  
用gensim的word2vector (0)  
sklearn 的优雅数据挖掘 (0)

### 推荐文章

\* 【观点】第二十三期：程序员应该如何积累财富？  
\* Android检查更新下载安装  
\* 动手打造史上最简单的Recycleview 侧滑菜单  
\* TCP网络通讯如何解决分包粘包问题  
\* SDCC 2017之大数据技术实战线上峰会  
\* 快速集成一个视频直播功能

### 最新评论

在keras 上实践,通过keras例子来 szq261299: 这么好的文章居然没人点赞嘛，楼主 我是按照你的代码一行行敲下来的，在前面一个个细节介绍时候，漏掉了关...

在keras 上实践,通过keras例子来 DragonFlyingST: 用训练样本作为测试样本，正确率当然是100%了，这显然是不合理啊。

```
1 A -> B
2 B -> C
3 C -> D
4 D -> E
5 E -> F
6 F -> G
7 G -> H
8 H -> I
9 I -> J
10 J -> K
11 K -> L
12 L -> M
13 M -> N
14 N -> O
15 O -> P
16 P -> Q
17 Q -> R
18 R -> S
19 S -> T
20 T -> U
21 U -> V
22 V -> W
23 W -> X
24 X -> Y
25 Y -> Z
```

input是一个一个字母的序列，output是一个一个的序列。

ok，就在这样的数据集上来应用我们的lstm。看看会有什么结果？

这时候dataX是一个一个用字母组成的序列，但是还要转换一下格式，才能用到keras上。

```
1 # reshape X to be [samples, time steps, features]
2 X = numpy.reshape(dataX, (len(dataX), seq_length, 1))
```

关闭



然后我们需要把我们的整数值归一化到0 ~ 1的区间上。

```
1 # normalize
2 X = X / float(len(alphabet))
```

最后，我们可以把这个问题当作是一个序列的分类问题，26个不同的字母代表了不同的类别，我们用keras的内置的 to\_categorical()函数把datay进行 one——hot编码，作为输出层的结果。

## 一个字母——一个字母映射！

我们通过lstm在这个问题上的预测，会发现这对lstm循环网络来说是很难解决的问题。

keras上lstm用于上述问题的代码如下：

```
1 # create and fit the model
2 model = Sequential()
3 model.add(LSTM(32, input_shape=(X.shape[1], X.shape[2])))
4 model.add(Dense(y.shape[1], activation='softmax'))
5 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
6 model.fit(X, y, nb_epoch=500, batch_size=1, verbose=2)
```

对与这个小问题，我用了掉炸天的Adam  
算法，经历了500次的迭代，最后我们得到了：

关闭



```
Epoch 492/500
0s - loss: 1.7121 - acc: 0.7600
Epoch 493/500
0s - loss: 1.7100 - acc: 0.7600
Epoch 494/500
0s - loss: 1.7130 - acc: 0.7600
Epoch 495/500
0s - loss: 1.7077 - acc: 0.8000
Epoch 496/500
0s - loss: 1.7083 - acc: 0.8000
Epoch 497/500
0s - loss: 1.7076 - acc: 0.7600
Epoch 498/500
0s - loss: 1.7084 - acc: 0.7600
Epoch 499/500
0s - loss: 1.7059 - acc: 0.7600
Epoch 500/500
0s - loss: 1.7060 - acc: 0.7600
```

这样的结果，真是日了狗了。

我们可以看到这样的问题对lstm循环神经网络来说真的难以处理。

原因是可怜的lstm单元根本没有可以利用的上下文信息。

这是对lstm的愚弄，把它当成了普通的多层感知机来对待。

## Three-Char Feature Window to One-C

[关闭](#)

这个标题不知道怎么翻译。。

我们把输入从一个字符升到三个字符。

```
1 # prepare the dataset of input to output pairs encoded as integers
2 seq_length = 3
```

就像这样：

```
1 ABC -> D
2 BCD -> E
3 CDE -> F
```

全部的代码如下：

```
1 # Naive LSTM to learn three-char window to one-char mapping
2 import numpy
3 from keras.models import Sequential
4 from keras.layers import Dense
5 from keras.layers import LSTM
6 from keras.utils import np_utils
7 # fix random seed for reproducibility
8 numpy.random.seed(7)
9 # define the raw dataset
10 alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
11 # create mapping of characters to integers (0-25) and the reverse
12 char_to_int = dict((c, i) for i, c in enumerate(alphabet))
13 int_to_char = dict((i, c) for i, c in enumerate(alphabet))
14 # prepare the dataset of input to output pairs encoded as integers
15 seq_length = 3
16 dataX = []
17 dataY = []
18 for i in range(0, len(alphabet) - seq_length, 1):
19     seq_in = alphabet[i:i + seq_length]
20     seq_out = alphabet[i + seq_length]
```

关闭





```

21 dataX.append([char_to_int[char] for char in seq_in])
22 dataY.append(char_to_int[seq_out])
23 print seq_in, '->', seq_out
24 # reshape X to be [samples, time steps, features]
25 X = numpy.reshape(dataX, (len(dataX), 1, seq_length))
26 # normalize
27 X = X / float(len(alphabet))
28 # one hot encode the output variable
29 y = np_utils.to_categorical(dataY)
30 # create and fit the model
31 model = Sequential()
32 model.add(LSTM(32, input_shape=(X.shape[1], X.shape[2])))
33 model.add(Dense(y.shape[1], activation='softmax'))
34 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
35 model.fit(X, y, nb_epoch=500, batch_size=1, verbose=2)
36 # summarize performance of the model
37 scores = model.evaluate(X, y, verbose=0)
38 print("Model Accuracy: %.2f%%" % (scores[1]*100))
39 # demonstrate some model predictions
40 for pattern in dataX:
41     x = numpy.reshape(pattern, (1, 1, len(pattern)))
42     x = x / float(len(alphabet))
43     prediction = model.predict(x, verbose=0)
44     index = numpy.argmax(prediction)
45     result = int_to_char[index]
46     seq_in = [int_to_char[value] for value in pattern]
47     print seq_in, "->", result

```

运行结果如下：

```

1 Model Accuracy: 86.96%
2 ['A', 'B', 'C'] -> D
3 ['B', 'C', 'D'] -> E
4 ['C', 'D', 'E'] -> F
5 ['D', 'E', 'F'] -> G

```

关闭



```
6 ['E', 'F', 'G'] -> H
7 ['F', 'G', 'H'] -> I
8 ['G', 'H', 'I'] -> J
9 ['H', 'I', 'J'] -> K
10 ['I', 'J', 'K'] -> L
11 ['J', 'K', 'L'] -> M
12 ['K', 'L', 'M'] -> N
13 ['L', 'M', 'N'] -> O
14 ['M', 'N', 'O'] -> P
15 ['N', 'O', 'P'] -> Q
16 ['O', 'P', 'Q'] -> R
17 ['P', 'Q', 'R'] -> S
18 ['Q', 'R', 'S'] -> T
19 ['R', 'S', 'T'] -> U
20 ['S', 'T', 'U'] -> V
21 ['T', 'U', 'V'] -> Y
22 ['U', 'V', 'W'] -> Z
23 ['V', 'W', 'X'] -> Z
24 ['W', 'X', 'Y'] -> Z
```

通过，结果我们发现有了一点点的提升，但是这一点点的提升未必是真实的，梯度下降算法本来就是具有随机性的。

也就是说我们再一次的错误使用了lstm循环神经网络。

我们确实给了上下文，但是并不是合适的方式，

我们把ABC当成了一个特征，而不是在一个时间端上的三个独立的特征。 In steps of one feature rather than one time step of separate features.（原文是这么说的）

## keras实践循环的正确打开方式！

在keras中，利用lstm的关键是以时间序列的方法来提供上下文，而不是像其他features的方式。

[关闭](#)



这次我们还是采用这样的训练方式，

```
1 ABC -> D
2 BCD -> E
3 CDE -> F
4 DEF -> G
```

我们这次唯一改变的地方是下面这里：

```
1 # reshape X to be [samples, time steps, features]
2 X = numpy.reshape(dataX, (len(dataX), seq_length, 1))
```

timesteps这个参数，我们设置了3，而不是前面的1。

也就是说我们把ABC 看成独立的三个特征 A B C组成的时间序列，而不是把ABC看成一个总的特征。

这就是keras中lastm循环神经网络的正确打开的方式。

我的理解是，

这样在训练 ABC——D的时候，BCD，CDE，都可以发挥作用。而最开始那只是利用了ABC——D这样一个训练样本。

完整代码如下：

```
1 # Naive LSTM to learn three-char time steps to one-char mapping
2 import numpy
3 from keras.models import Sequential
4 from keras.layers import Dense
5 from keras.layers import LSTM
6 from keras.utils import np_utils
7 # fix random seed for reproducibility
```

关闭



```
8  numpy.random.seed(7)
9  # define the raw dataset
10 alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
11 # create mapping of characters to integers (0-25) and the reverse
12 char_to_int = dict((c, i) for i, c in enumerate(alphabet))
13 int_to_char = dict((i, c) for i, c in enumerate(alphabet))
14 # prepare the dataset of input to output pairs encoded as integers
15 seq_length = 3
16 dataX = []
17 dataY = []
18 for i in range(0, len(alphabet) - seq_length, 1):
19     seq_in = alphabet[i:i + seq_length]
20     seq_out = alphabet[i + seq_length]
21     dataX.append([char_to_int[char] for char in seq_in])
22     dataY.append(char_to_int[seq_out])
23     print seq_in, '->', seq_out
24 # reshape X to be [samples, time steps, features]
25 X = numpy.reshape(dataX, (len(dataX), seq_length, 1))
26 # normalize
27 X = X / float(len(alphabet))
28 # one hot encode the output variable
29 y = np_utils.to_categorical(dataY)
30 # create and fit the model
31 model = Sequential()
32 model.add(LSTM(32, input_shape=(X.shape[1], X.shape[2])))
33 model.add(Dense(y.shape[1], activation='softmax'))
34 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
35 model.fit(X, y, nb_epoch=500, batch_size=1, verbose=2)
36 # summarize performance of the model
37 scores = model.evaluate(X, y, verbose=0)
38 print("Model Accuracy: %.2f%%" % (scores[1]*100))
39 # demonstrate some model predictions
40 for pattern in dataX:
41     x = numpy.reshape(pattern, (1, len(pattern), 1))
42     x = x / float(len(alphabet))
```

关闭



```
43 prediction = model.predict(x, verbose=0)
44 index = numpy.argmax(prediction)
45 result = int_to_char[index]
46 seq_in = [int_to_char[value] for value in pattern]
47 print seq_in, "->", result
```

最终的训练结果是

```
1 Model Accuracy: 100.00%
2 ['A', 'B', 'C'] -> D
3 ['B', 'C', 'D'] -> E
4 ['C', 'D', 'E'] -> F
5 ['D', 'E', 'F'] -> G
6 ['E', 'F', 'G'] -> H
7 ['F', 'G', 'H'] -> I
8 ['G', 'H', 'I'] -> J
9 ['H', 'I', 'J'] -> K
10 ['I', 'J', 'K'] -> L
11 ['J', 'K', 'L'] -> M
12 ['K', 'L', 'M'] -> N
13 ['L', 'M', 'N'] -> O
14 ['M', 'N', 'O'] -> P
15 ['N', 'O', 'P'] -> Q
16 ['O', 'P', 'Q'] -> R
17 ['P', 'Q', 'R'] -> S
18 ['Q', 'R', 'S'] -> T
19 ['R', 'S', 'T'] -> U
20 ['S', 'T', 'U'] -> V
21 ['T', 'U', 'V'] -> W
22 ['U', 'V', 'W'] -> X
23 ['V', 'W', 'X'] -> Y
24 ['W', 'X', 'Y'] -> Z
```

关闭



66666！吊的不行！但是我们还没有展示出循环神经网络的强大之处，因为上面这个问题我们用多层感知器，足够多的神经元，足够多的迭代次数也可以很好的解决。（三层神经网络拟合任意可以表示的函数）  
那么接下来就是展示循环神经网络的独到之处！！

## LSTM State Within A Batch

keras实现的lstm在每一个batch以后，都重置了lstm的状态。

顶 踩  
0 0

上一篇 用gensim的word2vector实现词嵌入

下一篇 sklearn 的优雅数据挖掘流程

关闭

### 相关文章推荐

- 基于Theano的深度学习(Deep Learning)框架Keras...
- keras系列 - 安装和...
- Presto服务治理与架构优化在京东的实践应用--王...
- 10小时深入掌握 K...
- 详细解读简单的lstm的实例
- keras + lstm 情感...
- 【免费直播】Python最佳学习路线--韦玮
- JDK9新特性
- 详细说明用keras建立训练自己数据的LSTM----语音..
- keras在构建LSTM



- JS-SDK开发与微信支付
- Keras框架下LSTM的一种实现
- keras中LSTM文本挖掘
- [NLP][Python]基于keras和LSTM的文本生成
- Spring Cloud微服务真实场景实战解析
- 用gensim的word2vector实现词嵌入

### 查看评论

2楼 [szq261299](#) 2017-04-13 11:02发表



这么好的文章居然没人点赞嘛，楼主 我是按照你的代码一行行敲下来的，在前面一个个细节介绍时候，漏掉了关于y的生成命令说明

1楼 [DragonFlyingST](#) 2017-04-11 15:42发表



用训练样本作为测试样本，正确率当然是100%了，这显然是不合理啊。

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

网站客服

杂志客服

微博客服

[webmaster@csdn.net](mailto:webmaster@csdn.net)

400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知

关闭

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved

