MBA em Ciência de Dados

Redes Neurais e Arquiteturas Profundas

Módulo VI - Redes neurais para dados sequenciais

Avaliação

Moacir Antonelli Ponti

CeMEAI - ICMC/USP São Carlos

As respostas devem ser dadas no Moodle, use esse notebook apenas para gerar o código necessário para obter as respostas

---

Questão 1)

Considere o uso de uma rede neural não recorrente (por exemplo uma rede densa) para o problema de, dado um caracter atual, predizer os k próximos caracteres de uma sentença, sendo formulado como um problema de regressão. Por outro lado, podemos também utilizar arquiteturas recorrentes. Qual alternativa está correta na comparação da adequação das duas abordagens ao problema?

(a) As unidades recorrentes possuem memória que considera a ordem que as instâncias aparecem nas iterações. Redes com unidades densas assumem independência entre as instâncias podendo considerar os dados de entrada em ordem arbitrária, e portanto as densas serão sempre melhores para problemas como o de predição de sequências.
(b) Unidades recorrentes cumprem melhor os requisitos de predizer sequências pois mantêm um sumário que está relacionado ao contexto de um caracter e sua ordem em uma sentença.
(c) Ambas são igualmente adequadas ao problema e devem produzir resultados similares, desde que os dados sejam preparados para cada caso
(d) As unidades densas capturam contexto e as relações de ordenação entre as instâncias, sendo mais adequadas para esse problema.

Questão 2)

Dentre as unidades recorrentes mais conhecidas, destacam-se a RNN tradicional, a GRU e a LSTM, em ordem da unidade com menor número de parâmetros, para aquela com maior número de parâmetros e portanto de maior capacidade. A GRU foi proposta mais recentemente do que as outras, ainda que tenha menos graus de liberdade (capacidade) do que a LSTM. Qual a principal motivação de usar GRU frente à LSTM?

(a) O mecanismo de esquecimento e atualização de novas informações nos sumários da

LSTM se tornou obsoleto.
(b) A GRU produz resultados significativamente melhores na maior parte dos casos reais do que a LSTM, ainda que seja mais demorada para treinar por iteração.
(c) A GRU é mais adequada apenas para problemas envolvendo sequência-para-sequência na estratégia encoder-decoder, nas demais tarefas a LSTM é sempre superior.
(d) A GRU possui menos parâmetros, mas mantendo um mecanismo de adição de informação e esquecimento ao sumário, e por isso é geralmente mais fácil de otimizar do que a LSTM.

Questão 3)

Escolha a alternativa que descreve a forma de implementação do mecanismo básico de atenção.

(a) Comparação da similaridade entre um vetor s e um conjunto de vetores h_i. Estas similaridades, depois de normalizadas, são utilizadas para realizar uma soma ponderada dos vetores h_i, resultando em um vetor c de contexto
(b) Por meio de uma unidade recorrente, computa-se uma série de sumários h ao longo de diversas iterações. Esses sumários são então processados por uma camada densa que computa a atenção com base na entrada
(c) Utilizando uma convolução unidimensional nos vetores a e b individualmente, e depois somando-os
(d) O mecanismo é implementado por meio do alinhamento entre dois vetores, a e b, comumente aplicado por meio de um produto interno. Após esse processo utiliza-se um autoencoder com erro médio quadrático para aprender a atenção do vetor a em relação ao vetor b

---

Questão 4)

Carregue a base de dados `starbucks.csv` , conforme abaixo, com uma divisão hold-out utilizando os 85% exemplos iniciais para treinamento e os restantes para teste e normalize no intervalo 0-1.

Grafe a série temporal, e note que há uma modificação brusca na série relativa ao teste, próxima ao ponto 950, que torna difícil sua predição.

Crie duas redes recorrentes, uma baseada em LSTM e outra em GRU, para predizer, ponto-a-ponto, a série temporal de testes. Elas devem conter ativações do tipo ReLU:

- Camada recorrente (LSTM/GRU) com 16 unidades
- Camada recorrente (LSTM/GRU) com 8 unidades
- Dropout de 0.2
- Camada densa de predição

Treine as duas redes com função de perda MSE, batch_size = 1, por 15 épocas, utilizando Adam com taxa de aprendizado 0.001. Antes de instanciar cada modelo, compilar e treinar,

inicialize as sementes com seed(2) e set_seed(2).

Calcule o MSE de teste (considere arredondamento para 4 casas decimais) e trace o gráfico comparando os dados reais com os preditos pelas duas redes. O resultado obtido foi:

(a) GRU com menor erro do que LSTM, mas apenas LSTM foi capaz de melhor predizer o pico na parte final da série
(b) GRU com menor erro do que LSTM, mas ambas falharam em predizer o pico na parte final da série
(c) LSTM com menor erro do que GRU, e apenas LSTM foi capaz de melhor predizer o pico na parte final da série
(d) LSTM com menor erro do que GRU, mas apenas GRU foi capaz de melhor predizer o pico na parte final da série

In [1]:
```
import numpy as np
from pandas import read_csv

import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras

from numpy.random import seed
from tensorflow.random import set_seed

from sklearn.metrics import mean_squared_error

df = read_csv('starbucks.csv')
var = df.columns.values[1]

porc_treinamento = 85
```

```
<ipython-input-1-db539b2e533e>:3: FutureWarning: The pandas.datetime class
is deprecated and will be removed from pandas in a future version. Import f
rom datetime module instead.
  from pandas import datetime
```

In [2]:
```
series = np.array(df[var])
plt.plot(series)
N = series.shape[0]

print("Série: ", var)
print("Tamanho da série: ", N)
```

```
Série:  Close
Tamanho da série:  1006
```

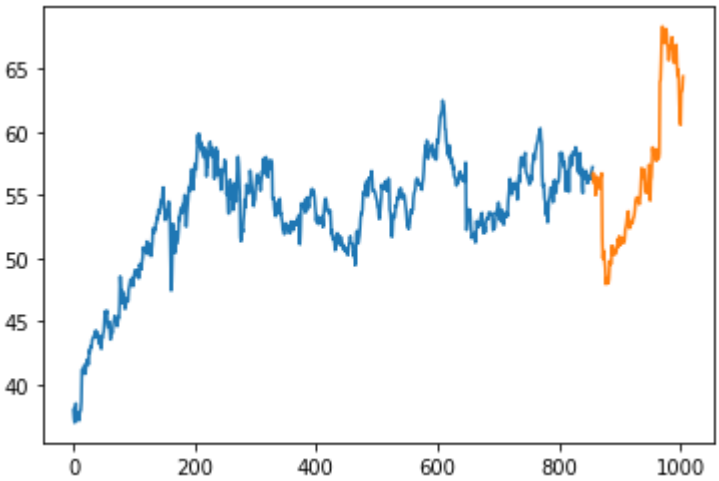
In [3]:
```
# calcula tamanhos dos dados de treinamento (n1) e teste (n2)
n1 = int(series.shape[0]*(porc_treinamento/100.0))
n2 = int(series.shape[0]*(1-(porc_treinamento/100.0)))

# divide dados de treinamento e teste
train, test = series[0:-n2], series[-n2:]

print("Exemplos de Treinamento: ", n1)
print("Exemplos de Teste: ", n2)
plt.plot(np.arange(0, n1+1), train)
plt.plot(np.arange(n1,n1+n2), test)
```

```
Exemplos de Treinamento:  855
Exemplos de Teste:  150
```

Out[3]:
```
[<matplotlib.lines.Line2D at 0x7fe3d69cd340>]
```



In [4]:
```
# modifica uma serie temporal tornando-a
# um problema de aprendizado supervisionado
def timeseries_to_supervised(series, look_back=1):
    x = series[:-look_back]
    y = np.array(series[look_back:], copy=True)
    return x,y

def normalize(train, test):
    d_max = np.max(train)
    d_min = np.min(train)
    return ((train - d_min) / (d_max-d_min)), ((test - d_min) / (d_max-d_m
```

Questão 5)

Carregue o word embedding `glove_s50` em Português do NILC, conforme visto em aula, e obtenha os word embeddings das palavras: 'celular', 'triste', 'arte', 'livro', 'feliz', 'cansado', 'teclado', 'música', 'estudo', 'escritório'

Produza uma projeção em 2 dimensões utilizando o PCA com random_state=1 e visualize os embeddings das palavras em um scatterplot.

Compute as distâncias (Euclidiana) entre "celular", "triste", "arte" e as outras palavras descritas acima, considerando a projeção obtida. Quais palavras (excluindo elas mesmas) estão mais proximas?

(a) celular-estudo, triste-cansado, arte-música

(b) celular-escritório, triste-cansado, arte-livro

(c) celular-teclado, triste-feliz, arte-celular

(d) celular-teclado, triste-feliz, arte-música

In [15]:
```
import os
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import pandas as pd

from tensorflow import keras
from tensorflow.keras import layers
from numpy.random import seed
from tensorflow.random import set_seed
```

In [16]:
```
# descomente para baixar na primeira vez
#!wget http://143.107.183.175:22980/download.php?file=embeddings/glove/glo
```

In [17]:
```
#!mv download.php?file=embeddings%2Fglove%2Fglove_s50.zip glove_s50.zip
#!unzip -q glove_s50.zip
```

In [18]:
```
path_to_glove_file = "./glove_s50.txt"

embeddings_index = {}
with open(path_to_glove_file) as f:
    for line in f:
        word, coefs = line.split(maxsplit=1)
        coefs = np.fromstring(coefs, "f", sep=" ")
        embeddings_index[word] = coefs

print("Encontrados %s word vectors." % len(embeddings_index))
```

```
<ipython-input-18-ecfee5d4ff7a>:7: DeprecationWarning: string or file could
not be read to its end due to unmatched data; this will raise a ValueError
in the future.
  coefs = np.fromstring(coefs, "f", sep=" ")
Encontrados 929594 word vectors.
```