Trường Đại học Công nghệ Thông tin
Khoa Khoa học và Kỹ thuật Thông tin

DS201 - Deep Learning trong Khoa học dữ liệu
Bài thực hành 5. MẠNG NEURAL HỒI QUY (RNN)

Mục tiêu

- Ôn tập kiến thức cơ bản về mô hình mạng neural hồi quy (RNN).
- Biết cách xây dựng mô hình mạng neural hồi quy cơ bản và sử dụng một số mô hình mạng neural hồi quy nổi tiếng.
- Áp dụng các mô hình trên vào bài toán phân tích cảm xúc.

1. BỘ DỮ LIỆU

Phân tích cảm xúc (Sentiment Analysis) là một kỹ thuật xác định và phân tích cảm xúc của người dùng dựa vào một đoạn văn hoặc một câu văn.

Bộ dữ liệu IMDB Movies Reviews được xây dựng nhằm phục vụ cho bài toán phân tích cảm xúc của người dùng đối với các bộ phim trên IMDB dựa vào các bình luận (review) của họ.

Có hai nhãn cảm xúc chính trong bộ dữ liệu:

- **Tích cực (positive)** - ký hiệu là 1.
- **Tiêu cực (negative)** - ký hiệu là 0.

Bộ dữ liệu này được hỗ trợ sẵn trong thư viện Keras.
Để load bộ dữ liệu này, chúng ta thực hiện như sau:

```
from keras.datasets import imdb
(training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=10000)
```

(?) Cho biết số lượng phần tử của tập train và tập test?

Bộ dữ liệu này gồm hai phần:

- **Tập train**: gồm 25,000 câu bình luận đã được mã hoá (encode) theo thứ tự từ điển.
- **Tập test**: gồm 25,000 câu bình luận đã được mã hoá (encode) theo thứ tự từ điển.

(?) Vẽ biểu đồ cột thể hiện số lượng nhãn của mỗi tập dữ liệu và biểu đồ tròn thể hiện tỉ lệ phân bố nhãn của mỗi tập dữ liệu?

Để in ra câu dữ liệu đầu tiên trong tập train, có thể sử dụng câu lệnh sau:

```
training_data[0]
```

(?) In ra 5 câuđầu tiên trong tập train và tập test kèm nhãn tương ứng?

Ví dụ

Giả sử tập từ vựng của một bộ dữ liệu có dạng sau:

```
{ "#": 0
  "this": 1,
   "is": 2,
```

```
  "A": 3,
  "B": 4,
  "C": 5 }
```

Một câu văn bản: "this is C" sẽ được mã hoá thành vector như sau: [1, 2, 5].

Để chuyển câu dữ liệu training_data[0] về dạng văn bản, cần phải có tập từ vựng với các chỉ số đi kèm, được gọi là word_index.

Để lấy word index của bộ dữ liệu, chúng ta thực hiện như sau:

```
index = imdb.get_word_index()
```

(?) Cho biết 5 từ đầu tiên của tập từ vựng?

(?) Cho biết tập từ vựng của bộ dữ liệu có bao nhiêu từ?

Dựa trên tập từ vựng, chúng ta có thể chuyển câu dữ liệu đầu tiên trong tập train về dạng văn bản (decode) như sau:

```
reverse_index = dict([(value, key) for (key, value) in index.items()])
decoded = " ".join([reverse_index.get(i, "#") for i in training_data[0]])
```

(?) Cho biết độ dài của câu văn vừa được decode?

(?) Decode 5 câu tiếp theo trong tập train?

Padding chiều dài của sequence

Để đảm bảo các sequence đều có độ dài như nhau.

Ví dụ

Chọn độ dài sequence là 200.

```
from keras.preprocessing.sequence import pad_sequences
maxlen = 200
X_train = pad_sequences(training_data, maxlen=maxlen)
X_test = pad_sequences(testing_data, maxlen=maxlen)
```

Xử lý cho nhãn y_train và y_test:

```
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(training_targets,num_classes=2)
y_test = testing_targets
```

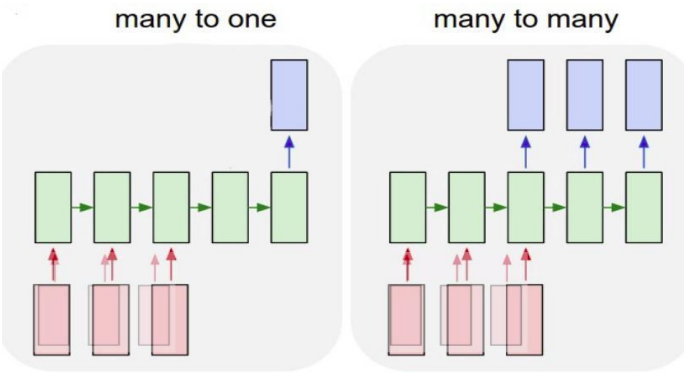
2. XÂY DỰNG MÔ HÌNH

SimpleRNN

Là layer gồm các units kết nối với nhau theo phương pháp Fully-connected.

Các tham số quan trọng bao gồm:

- **units:** số lượng units trong layer.
- **activation:** hàm kích hoạt.
- **dropout:** sử dụng dropount hay không.
- **return_sequences:** trả về output là một sequence (đối với Many-to-many) hoặc là một giá trị (Many-to-one).



- **recurrent_initializer**, **kernel_initializer** và **bias_initializer**: khởi tạo giá trị.
- **recurrent_regularizer**, **kernel_regularizer**: chuẩn hoá mạng neural.

Xem thêm: https://keras.io/api/layers/recurrent_layers/simple_rnn/.

Xây dựng mạng neural hồi quy đơn giản gồm lớp với 200 units (ứng với độ dài sequence)

SimpleRNN cần đầu vào có shape như sau: [batch, timesteps, feature].

Ý nghĩa như sau:

- **batch**: số lượng sample - điểm dữ liệu (đã chia theo kích thước mỗi batch, có thể xem lại batch and mini-batch training để hiểu rõ hơn).
- **timesteps**: Số lượng từ, ký tự trong câu, ở đây timesteps chính là độ dài chuỗi.
- **features**: số lượng features đưa vào mỗi steps trong sequence, features sẽ bằng với **chiều (dimension) của word embedding**.

Mô hình đơn giản ban đầu sẽ dùng features với kích thước là 1. Do đó, chúng ta sẽ expand kích thước của dữ liệu ban đầu bằng cách dùng np.expand_dims trong thư viện numpy (xem lại bài thực hành 3).

```
import numpy as np

X_train_new = np.expand_dims(X_train, axis=2)
X_test_new = np.expand_dims(X_test, axis=2)
```

Xây dựng mạng neural hồi quy đơn giản:

```
model = Sequential()
model.add(Input(shape=(None, 1), dtype="float64"))
model.add(SimpleRNN(200, return_sequences=True, return_state=False,
activation='relu'))
model.add(SimpleRNN(200, return_sequences=False, return_state=False,
activation='relu'))
model.add(Dense(2, activation='sigmoid'))
```

(?) Compile mô hình với các thông số sau:

- Hàm tối ưu: Adam với learning_rate = 0.01.

- Hàm loss: Binary Cross-Entropy.
- Độ đo: Accuracy.

(?) Sử dụng hàm summary để xem cấu trúc mô hình đã xây dựng?

(?) Huấn luyện mô hình với các thông số sau:
- Batch size: 128.
- Số epochs: 5.
- Tỉ lệ tạo tập dev: 10%.

(?) Vẽ đồ thị học với Accuracy và Loss?

(?) Tính độ chính xác đánh giá của mô hình bằng độ đo Accuracy?

3. LSTM, BILSTM VÀ EMBEDDING

Embedding

Kỹ thuật Embedding sẽ chuyển từ thành vector đặc (dense vector). Đây là một trong các kỹ thuật được dùng nhiều trong NLP nhằm biểu diễn ngữ nghĩa cho một từ.

Lớp Embedding trong thư viện Keras có các thông số quan trọng sau:
- **input_dim:** kích thước từ vựng. thường là giá trị: len(word_index) + 1.
- **output_dim:** chiều của vector (đôi khi còn được gọi là embedding_dim).
- **embeddings_initializer:** phương pháp khởi tạo giá trị của embedding.
- **embeddings_regularizer:** chuẩn hoá giá trị của embedding.

Xem thêm về Embedding: https://keras.io/api/layers/core_layers/embedding/.

Lớp LSTM

Xây dựng một lớp LSTM cho mạng neural.

Các thông số quan trọng bao gồm:
- **units:** số lượng units trong layer..
- **activation:** hàm kích hoạt.

- **dropout:** sử dụng dropount hay không.
- **return_sequences:** trả về output là một sequence (đối với Many-to-many) hoặc là một giá trị (Many-to-one).
- **recurrent_initializer**, **kernel_initializer** và **bias_initializer**: khởi tạo giá trị.
- **recurrent_regularizer**, **kernel_regularizer**: chuẩn hoá mạng neural.

Xem thêm: https://keras.io/api/layers/recurrent_layers/lstm/.

```
model = Sequential()
model.add(Input(shape=(None, ), dtype="float64"))
model.add(Embedding(len(word_index), 128))
model.add(LSTM(200, return_sequences=False))
model.add(Dense(2, activation='sigmoid'))
```

(?) Compile mô hình với các thông số sau:

- Hàm tối ưu: Adam với learning_rate = 0.01.
- Hàm loss: Binary Cross-Entropy.
- Độ đo: Accuracy.

(?) Sử dụng hàm summary để xem cấu trúc mô hình đã xây dựng?

(?) Huấn luyện mô hình với các thông số sau:

- Batch size: 128.
- Số epochs: 5.
- Tỉ lệ tạo tập dev: 10%.

(?) Vẽ đồ thị học với Accuracy và Loss?

(?) Đánh giá độ chính xác mô hình bằng độ đo Accuracy?

Lớp Bidirectional

Sử dụng để xây dựng BiLSTM hoặc BiGRU.

Có hai thông số quan trọng gồm:

- layer: Một lớp (layer) của thư viện Keras, có thể là LSTM hoặc GRU.
- merge_mode: phương pháp dùng để merge giá trị forward và backward (xem thêm về LSTM và GRU để rõ hơn), mặc định là "concat".

Xem thêm: https://keras.io/api/layers/recurrent_layers/bidirectional/.

Sử dụng Bidirectional để xây dựng lớp BiLSTM:

```
model = Sequential()
model.add(Input(shape=(None, ), dtype="float64"))
model.add(Embedding(len(word_index), 128))
model.add(Bidirectional(LSTM(200, return_sequences=False)))
model.add(Dense(2, activation='sigmoid'))
```

(?) Compile mô hình với các thông số sau:

- Hàm tối ưu: Adam với learning_rate = 0.01.
- Hàm loss: Binary Cross-Entropy.
- Độ đo: Accuracy.

(?) Sử dụng hàm summary để xem cấu trúc mô hình đã xây dựng?

(?) Huấn luyện mô hình với các thông số sau:

- Batch size: 128.
- Số epochs: 5.
- Tỉ lệ tạo tập dev: 10%.

(?) Vẽ đồ thị học với Accuracy và Loss?

(?) Đánh giá độ chính xác mô hình bằng độ đo Accuracy?

BÀI TẬP

Bài tập 1

Tăng số lượng epochs lên 20 và thực hiện huấn luyện mô hình.

a. Vẽ đồ thị học với Accuracy và Loss.

b. Cho biết độ chính xác đánh giá của mô hình?

Bài tập 2

Thử thêm 1 lớp Simple RNN nữa vào mô hình và cho biết kết quả độ chính xác đánh giá của mô hình?

Ghi chú: Thêm vào trước lớp SimpleRNN hiện tại, đặt return_sequence = True.

Bài tập 3

Xây dựng mô hình gồm 1 lớp Embedding và 1 lớp LSTM kết nối với nhau.

a. Vẽ đồ thị học với Accuracy và Loss.

b. Cho biết độ chính xác đánh giá của mô hình?

Bài tập 4

Xây dựng mô hình gồm 1 lớp Embedding và 2 lớp LSTM kết nối với nhau.

a. Vẽ đồ thị học với Accuracy và Loss.

b. Cho biết độ chính xác đánh giá của mô hình?

Bài tập 5

a. Tìm hiểu GRU và xây dựng mô hình với lớp GRU (thực hiện mô hình tương tự Bài tập 3, nhưng thay lớp LSTM thành GRU).

b. So sánh độ chính xác đánh giá của 2 mô hình.

Hướng dẫn nộp bài

- Các file nộp:
  - File source code: Đặt tên là **MSSV_Lab5.ipynb** (hoặc **.jpynb**) (Chú thích rõ câu hỏi, các bước làm và kết quả).
  - File báo cáo trả lời các câu hỏi trong bài: Đặt tên là

MSSV_Lab5.pdf (Trình bày rõ ràng, ghi lại đề mục và câu hỏi tương ứng).

→ Nén lại và đặt tên **MSSV_Lab5.zip**.

- Nộp bài ở mục Submission **Lab 5** trên website môn học.
- Deadline nộp bài: **3 tuần**.

Chúc các bạn học tốt !