HUẤN LUYỆN MẠNG RNN

SỬ DỤNG BACKPROPAGATION THROUGH TIME VÀ GENETIC ALGORITHM

NHÓM 1:

LƯƠNG HỮU PHÚ LỘC

MAI THIỆN QUANG

BÙI TẤN PHÁT

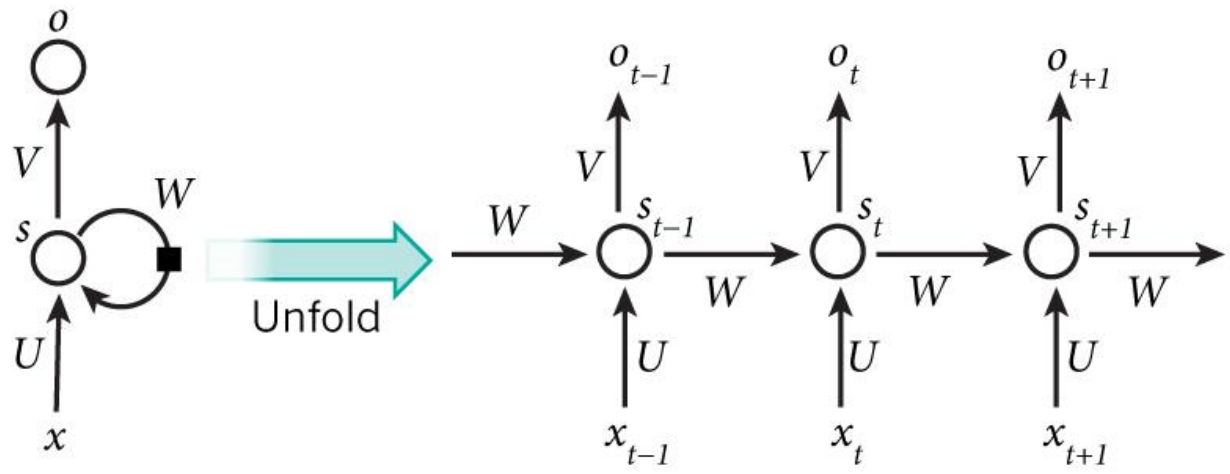
PHẠM NGỌC KHÔI NGUYÊN

NGUYỄN TRỌNG PHÚC

GVHD: TS. PHẠM VIỆT CƯỜNG

1. Mạng Recurrent Neural Network (RNN)

Mạng RNN được sử dụng trong xử lý ngôn ngữ tự nhiên vì sự phụ thuộc của các ngõ vào (input) với nhau, nghĩa là ngõ ra (output) của 1 input phụ thuộc vào 1 input trước đó.



Ở đây, tính hồi quy được thể hiện thông qua việc thực hiện cùng 1 nhiệm vụ tính toán tại mỗi lớp hay các trọng số được sử dụng chung (U, V, W). Tại mỗi time step t:

- x_t là input
- o_t là ouput với hàm kích hoạt $o_t = softmax(Vs_t)$
- s_t là trạng thái ẩn (hidden state) hay "bộ nhớ" của mạng. Trong báo cáo này, s_t được tính $s_t = f(Ux_t + Ws_{t-1})$ với f là hàm $tanh$

Training cho mạng nghĩa là ta cần tìm các ma trận U, V, W hợp với dữ liệu ngõ vào và ra mong muốn. Để train cho mạng nhận diện chữ "hello" ta khởi tạo 2 mảng X,y với X(i) là kí tự đưa vào và y(i) là kí tự kế tiếp. Để dễ dàng tính toán trước hết ta mã hóa mỗi kí tự của chữ "hello" thành các con số như sau:

Kí tự	Số mã hóa tương ứng
' l '	1
' h '	2
' e '	3
' o '	4

Như vậy, Ma trận X sẽ gồm 4 kí tự 'h','e', 'l', 'l' và ma trận y tương ứng sẽ là 'e', 'l', 'l', 'o', sau khi mã hóa ta được mảng X,y như sau:

- X = [2 3 1 1]
- y = [3 1 1 4]

Theo đề cập ở trên, RNN có thể được dùng để dự toán 1 câu dài, có yêu cầu về ngữ nghĩa. Tuy nhiên, trong báo cáo này, dữ liệu sử dụng chỉ vỏn vẹn chữ "hello" nên khá đơn giản cho dự đoán. Chỉ cần ghi nhớ kí tự trước đó để dự đoán kí tự kế tiếp, do đó, s_t chỉ cần lấy s_{t-1} là đủ.

Các thông số quan trọng cần phải chú ý khi khởi tạo mạng và training:

- word dimension (word_dim): số lượng từ vựng khác nhau của dữ liệu
- hidden dimension (hidden_dim): số lớp ẩn tối đa hay số từ tối đa có thể dự đoán.
- Backpropagation truncate (bptt_truncate): giảm bớt việc tính toán BPTT đối với các mẫu huấn luyện lớn. Ở đây, set mặc định bằng 0 vì lượng dữ liệu quá nhỏ

2. Huấn luyện mạng RNN sử dụng Backpropagation through time (BPTT)

a. Khởi tạo mạng RNN

Trước hết, cần khởi tạo các thông số cơ bản như đề cập ở trên cùng với learning rate và số lượng epoch cần lặp.

Sau khi khởi tạo các thông số cơ bản, cần phải khởi tạo trọng số cho mạng (U, V, W). Các trọng số được khởi tạo ngẫu nhiên trong vùng $\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$ với n là số các kết nối từ những lớp trước (nguồn tham khảo bên dưới).

b. Training

Giải thuật train mạng RNN được thực hiện như sau:

- Tại mỗi epoch:
  - Tính lan truyền thuận (forward propagation) để lấy giá trị o_t và s_t (là các ma trận 4x4 thể hiện xác suất cho mỗi kí tự)
  - Tính đạo hàm sai số $\frac{dE}{dU}, \frac{dE}{dV}, \frac{dE}{dW}$ (backpropagation through time)
  - Cập nhật trọng số theo phương pháp Stochastic Gradient Decent (SGD) với learning rate được nhập trước (chọn 0.005)
  - Đánh giá sai số sau mỗi 5 epochs để thay đổi learning rate nếu thấy sai số tăng lên
  - Kết thúc 1 epoch

Sai số được đánh giá dùng hàm Cross-entropy $L(y, o) = -\frac{1}{N}\sum_{n \in N} y_n \log o_n$

Kết thúc quá trình huấn luyện, thu được các trọng số U, V, W

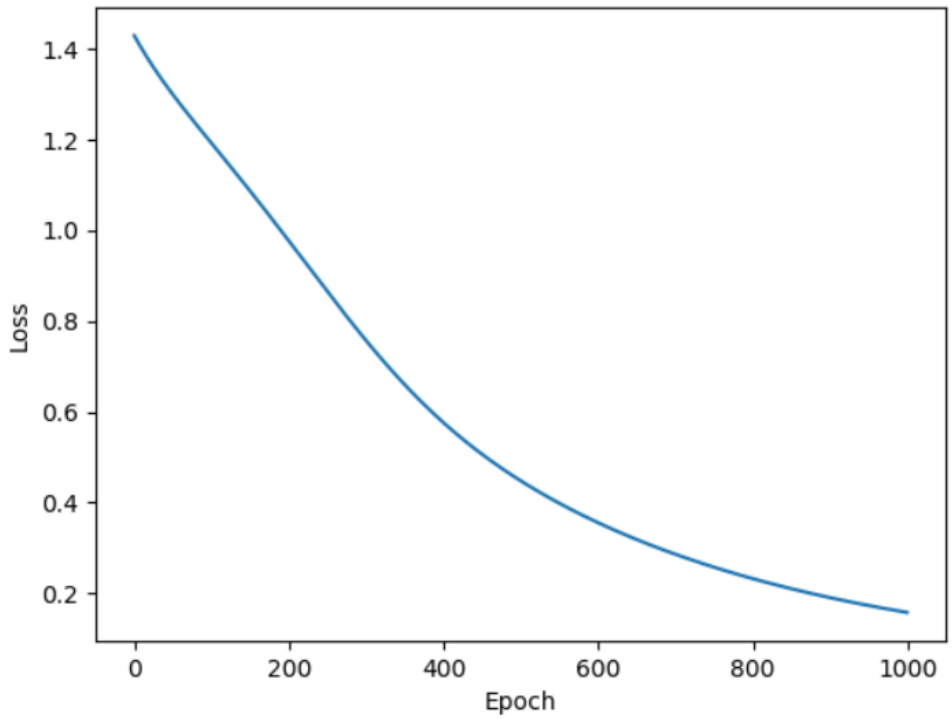
c. Ứng dụng

Sử dụng mạng (quá trình dự đoán):

- Mỗi kí tự nhập vào sẽ được mã hóa thành số để tạo ra input.
- Nhóm tạo 1 hàm “predict” có chức năng tính forward propagation của input sau đó chọn ra giá trị có argument max từ ma trận output o_t.
- Kết quả thu được là số mã hóa, thực hiện chuyển đổi sang chữ sẽ được kết quả mong muốn.
- Tiếp tục sử dụng output dự đoán trên kết hợp với input ban đầu để tạo input tiếp theo
- Quá trình lặp lại khi đã dự đoán đủ kí tự “hello”

d. Kết quả

Với learning rate = 0.005 sau 1000 epoch vẽ được đồ thị sai số như sau:



Sau 1000 epoch, sai số đã giảm từ khoảng 1.422 xuống còn 0.072, như vậy mạng đã được training thành công.

Kết quả dự đoán:

```
print('Input = h --->',generate_seq(model,tokenizer,'h'))
print('Input = he --->',generate_seq(model,tokenizer,'he'))
print('Input = hel --->',generate_seq(model,tokenizer,'hel'))
print('Input = hel --->',generate_seq(model,tokenizer,'hell'))
plt.plot(loss)
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.show()
```

```
Using TensorFlow backend.
Vocabulary Size: 5
Total Sequences: 4
Input = h ---> hello
Input = he ---> hello
Input = hel ---> hello
Input = hel ---> hello
[Finished in 6.4s]
```

3. Huấn luyện mạng RNN sử dụng giải thuật di truyền

(Giải thuật Di truyền nhóm không sử dụng toolbox “ga” của matlab mà được viết lại trên Python)

a. Khởi tạo

Khởi tạo quần thể, ở đây mỗi cá thể của quần thể là tập hợp 3 ma trận U, V, W, nhóm khởi tạo quần thể gồm 10 cá thể ngẫu nhiên theo phương pháp như BPTT

Hàm đánh giá quần thể: ta sẽ đánh giá dựa trên sai số của mạng RNN thông qua cross-entropy $L(y,o) = -\frac{1}{N}\sum_{n\in N} y_n \log o_n$ và gọi đó là “**fitness**”

Chọn lọc: Mạng RNN càng chính xác khi sai số càng nhỏ tức giá trị fitness trên càng nhỏ, dựa vào đó nhóm chọn ra **5 cá thể** có sai số nhỏ nhất để lai ghép, gây đột biến và đưa vào thế hệ sau.

b. Huấn luyện

Nhóm chọn phương pháp lai ghép là lấy 2 hàng đầu của bộ ma trận U, V, W có fitness nhỏ nhất lần lượt ghép với 2 hàng sau của các bộ ma trận khác, còn đột biến nhóm lấy ngẫu nhiên 3 phần tử của mỗi ma trận và cộng trừ một lượng random

Sau khi lai ghép và đột biến ta được thế hệ tiếp theo, tiếp tục lặp lại vòng lặp này chính là quá trình training mạng RNN

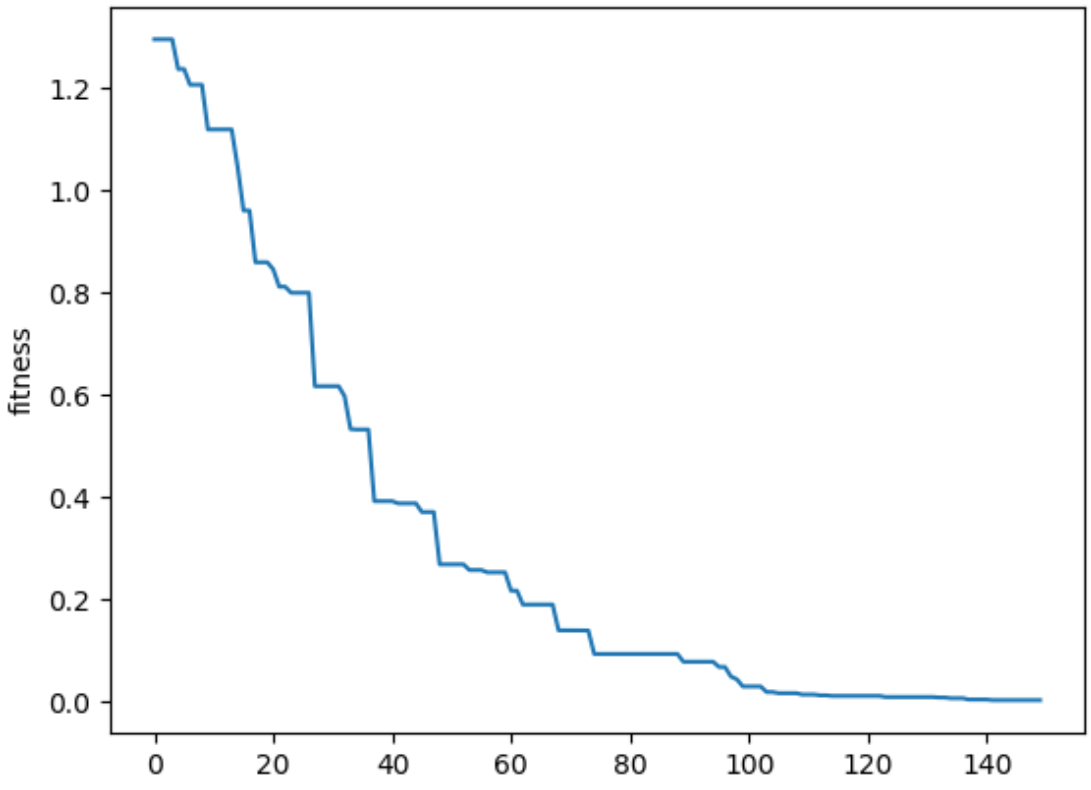
c. Ứng dụng

Sử dụng mạng (quá trình dự đoán):

- Mỗi kí tự nhập vào sẽ được mã hóa thành số để tạo ra input.
- Nhóm tạo 1 hàm “predict” có chức năng tính forward propagation của input sau đó chọn ra giá trị có argument max từ ma trận output o_t.
- Kết quả thu được là số mã hóa, thực hiện chuyển đổi sang chữ sẽ được kết quả mong muốn.
- Tiếp tục sử dụng output dự đoán trên kết hợp với input ban đầu để tạo input tiếp theo
- Quá trình lặp lại khi đã dự đoán đủ kí tự “hello”

d. Kết quả

Kết quả của hàm fitness với số thế hệ là 150



➔ Sau 150 thế hệ hàm fitness giảm xuống rất bé nghĩa là sai số cross entropy rất nhỏ. Như vậy có thể coi mạng RNN training thành công

Kết quả của mạng RNN khi với từng input:

```
64
65
66  model = RNN_GA(4)
67  model.train(X,y,num_of_generations = 150)
68
69  print('Input = h --->',generate_seq(model,tokenizer,'h'))
70  print('Input = he --->',generate_seq(model,tokenizer,'he'))
71  print('Input = hel --->',generate_seq(model,tokenizer,'hel'))
72  plt.plot(model.graph)
73  plt.ylabel('fitness')
74  plt.show()
75
76
77
```

```
Using TensorFlow backend.
Vocabulary Size: 4
Total Sequences: 4
Input = h ---> hello
Input = he ---> hello
Input = hel ---> hello
```

➔ Mạng có thể nhận diện đúng cả 3 loại ngõ vào là 1 chữ h, 2 chữ he, và 3 chữ hel

Tài liệu tham khảo

RNN model and Backpropagation:

[1] http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[2] https://machinelearningmastery.com/develop-word-based-neural-language-models-python-keras/

Genetic Algorithm:

[3] https://medium.com/@phamduychv94/genetic-algorithms-gi%E1%BA%A3i-thu%E1%BA%ADt-di-truy%E1%BB%81n-python-dd531166c385

RNN initial parameter:

[4] http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf