Bài 13: Recurrent neural network

May 25, 2019 | Nttuan8 | Posted in Deep Learning cơ bản

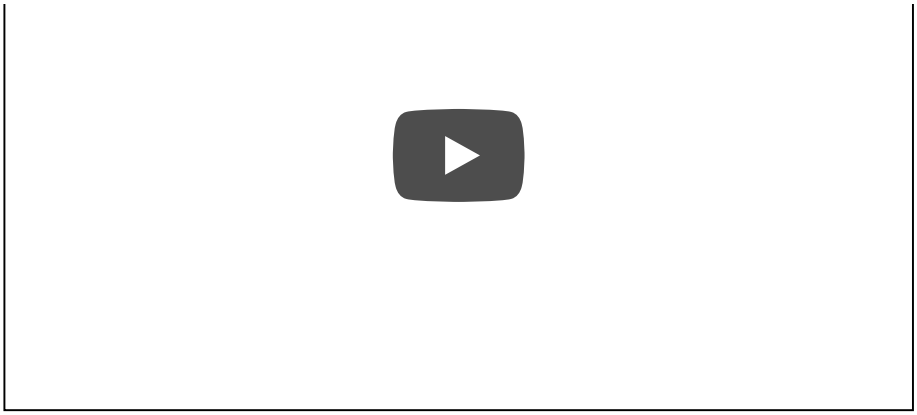
Deep learning có 2 mô hình lớn là Convolutional Neural Network (CNN) cho bài toán có input là ảnh và Recurrent neural network (RNN) cho bài toán dữ liệu dạng chuỗi (sequence). Mình đã giới thiệu về Convolutional Neural Network (CNN) và các ứng dụng của deep learning trong computer vision bao gồm: classification, object detection, segmentation. Có thể nói là tương đối đầy đủ các dạng bài toán liên quan đến CNN. Bài này mình sẽ giới thiệu về RNN.

Nội dung

Recurrent Neural Network là gì?

Bài toán: Cần phân loại hành động của người trong video, input là video 30s, output là phân loại hành động, ví dụ: đứng, ngồi, chạy, đánh nhau, bắn súng,...

Human Action Recognition with Deep Learning



Khi xử lý video ta hay gặp khái niệm FPS (frame per second) tức là bao nhiêu frame (ảnh) mỗi giây. Ví dụ 1 FPS với video 30s tức là lấy ra từ video 30 ảnh, mỗi giây một ảnh để xử lý.

Ta dùng 1 FPS cho video input ở bài toán trên, tức là lấy ra 30 ảnh từ video, ảnh 1 ở giây 1, ảnh 2 ở giây 2,... ảnh 30 ở giây 30. Bây giờ input là 30 ảnh: ảnh 1, ảnh 2,... ảnh 30 và output là phân loại hành động. Nhận xét:

- Các ảnh có **thứ tự** ví dụ ảnh 1 xẩy ra trước ảnh 2, ảnh 2 xẩy ra trước ảnh 3,... Nếu ta đảo lộn các ảnh thì có thể thay đổi nội dung của video. Ví dụ: nội dung video là cảnh bắn nhau, thứ tự đúng là A bắn trúng người B và B chết, nếu ta đảo thứ tự ảnh thành người B chết xong A mới bắn thì rõ ràng bây giờ A không phải là kẻ giết người => nội dung video bị thay đổi.
- Ta có thể dùng CNN để phân loại 1 ảnh trong 30 ảnh trên, nhưng rõ ràng là 1 ảnh không thể mô tả được nội dung của cả video. Ví dụ: Cảnh người cướp điện thoại, nếu ta chỉ dùng 1 ảnh là người đấy cầm điện thoại lúc cướp xong thì ta không thể biết được cả hành động cướp.

=> Cần một mô hình mới có thể giải quyết được bài toán với input là sequence (chuỗi ảnh 1->30) => RNN ra đời.

Dữ liệu dạng sequence

Dữ liệu có thứ tự như các ảnh tách từ video ở trên được gọi là sequence, time-series data.

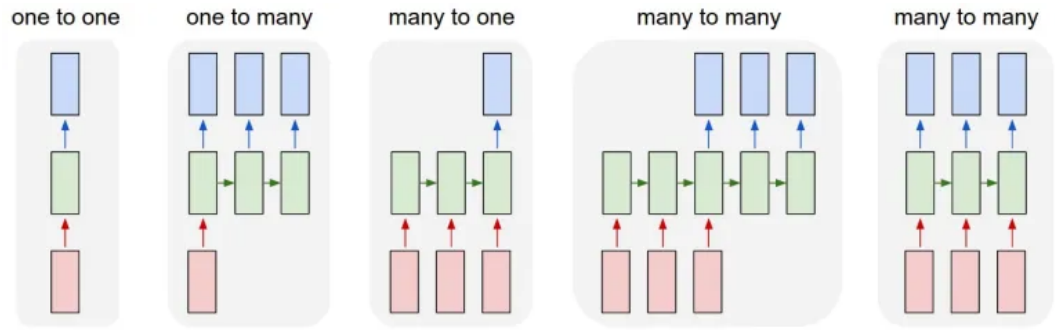
Trong bài toán dự đoán đột quỵ tim cho bệnh nhân bằng các dữ liệu tim mạch khám trước đó. Input là dữ liệu của những lần khám trước đó, ví dụ i1 là lần khám tháng 1, i2 là lần khám tháng 2,... i8 là lần khám tháng 8. (i1,i2,..i8) được gọi là sequence data. RNN sẽ học từ input và dự đoán xem bệnh nhân có bị đột quy tim hay không.

Competition 2021
Bài 6: Lưu và load model trong Pytorch
Bài 5: Transfer learning
Bài 4: Train Neural Network
Bài 3: Neural Network

Mục Bài Viết

Ví dụ khác là trong bài toán dịch tự động với input là 1 câu, ví dụ "tôi yêu Việt Nam" thì vị trí các từ và sự xắp xếp cực kì quan trọng đến nghĩa của câu và dữ liệu input các từ ['tôi', 'yêu', 'việt', 'nam'] được gọi là sequence data. **Trong bài toán xử lý ngôn ngữ (NLP) thì không thể xử lý cả câu được và người ta tách ra từng từ làm input, giống như trong video người ta tách ra các ảnh (frame) làm input.**

Phân loại bài toán RNN



Các dạng bài toán RNN

One to one: mẫu bài toán cho Neural Network (NN) và Convolutional Neural Network (CNN), 1 input và 1 output, ví dụ với CNN input là ảnh và output là ảnh được segment.

One to many: bài toán có 1 input nhưng nhiều output, ví dụ: bài toán caption cho ảnh, input là 1 ảnh nhưng output là nhiều chữ mô tả cho ảnh đấy, dưới dạng một câu.

Nguồn: https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2

Many to one: bài toán có nhiều input nhưng chỉ có 1 output, ví dụ bài toán phân loại hành động trong video, input là nhiều ảnh (frame) tách ra từ video, ouptut là hành động trong video

Many to many: bài toán có nhiều input và nhiều output, ví dụ bài toán dịch từ tiếng anh sang tiếng việt, input là 1 câu gồm nhiều chữ: "I love Vietnam" và output cũng là 1 câu gồm nhiều chữ "Tôi yêu Việt Nam".

Ứng dụng bài toán RNN

Về cơ bản nếu bạn thấy sequence data hay time-series data và bạn muốn áp dụng deep learning thì bạn nghĩ ngay đến RNN. Dưới đây là một số ứng dụng của RNN:

- **Speech to text**: Chuyển giọng nói sang text.
- **Sentiment classification**: phân loại số sao cho các bình luận, ví dụ: input: "ứng dụng tốt", output: 4 sao.
- **Machine translation**: Bài toán dịch tự động giữa các ngôn ngữ.
- **Video recognition**: Nhận diện hành động trong video.
- **Heart attack**: Dự đoán đột quỵ tim.
- ....

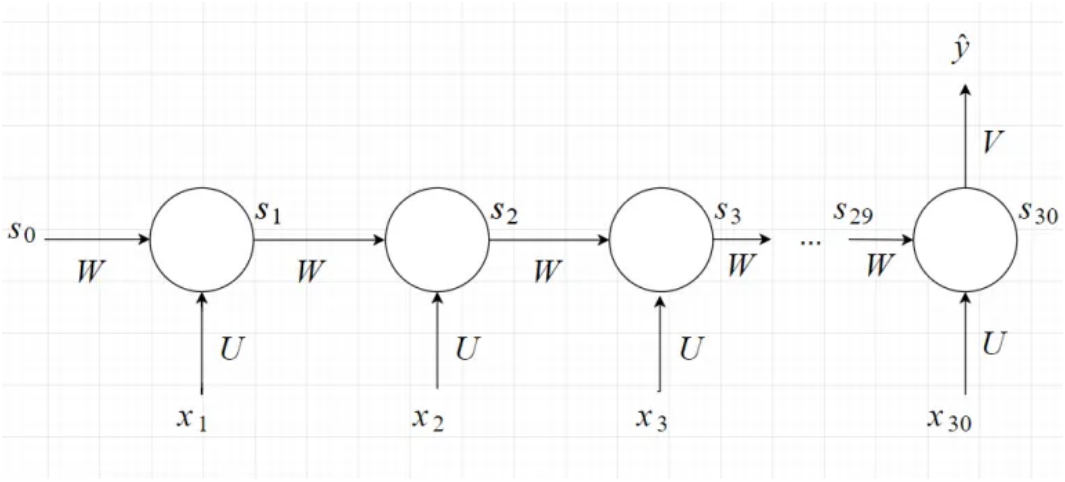
Mô hình bài toán RNN

Mô hình RNN

Bài toán: Nhận diện hành động trong video 30s. Đây là dạng bài toán many to one trong RNN, tức nhiều input và 1 output.

Input ta sẽ tách video thành 30 ảnh ở mỗi giây. Các ảnh sẽ được cho qua model CNN để lấy ra các feature (feature extraction) thành các vector có kích thước n*1. Vector tương ứng với ảnh ở giây thứ i là x_i.

Output là vector có kích thước d*1, softmax function được sử dụng như trong bài phân loại ảnh.

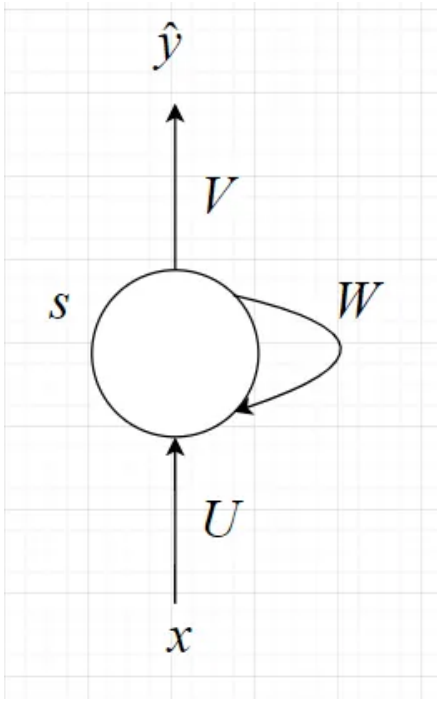


Mô hình RNN cho bài toán.

Ta có:

- Mô hình có 30 input và 1 output, các input được cho vào model đúng với thứ tự ảnh trong video $x_1, x_2, ...x_{30}$.
- Mỗi hình tròn được gọi là 1 state, state t có input là x_t và s_{t-1} (output của state trước); output là $s_t = f(U * x_t + W * s_{t-1})$. f là activation function thường là tanh hoặc ReLU.
- Có thể thấy s_t mang cả thông tin từ state trước (s_{t-1}) và input của state hiện tại => s_t giống như memory nhớ các đặc điểm của các input từ x_1 đến x_t
- s_0 được thêm vào chỉ cho chuẩn công thức nên thường được gán bằng 0 hoặc giá trị ngẫu nhiên. Có thể hiểu là ban đầu chưa có dữ liệu gì để học thì memory rỗng.
- Do ta chỉ có 1 output, nên sẽ được đặt ở state cuối cùng, khi đó s_{30} học được thông tin từ tất cả các input. $\hat{y} = g(V * s_{30})$. g là activation function, trong bài này là bài toán phân loại nên sẽ dùng softmax.

Ta thấy là ở mỗi state các hệ số W, U là giống nhau nên model có thể được viết lại thành:



Mô hình RNN rút gọn

Tóm lại:

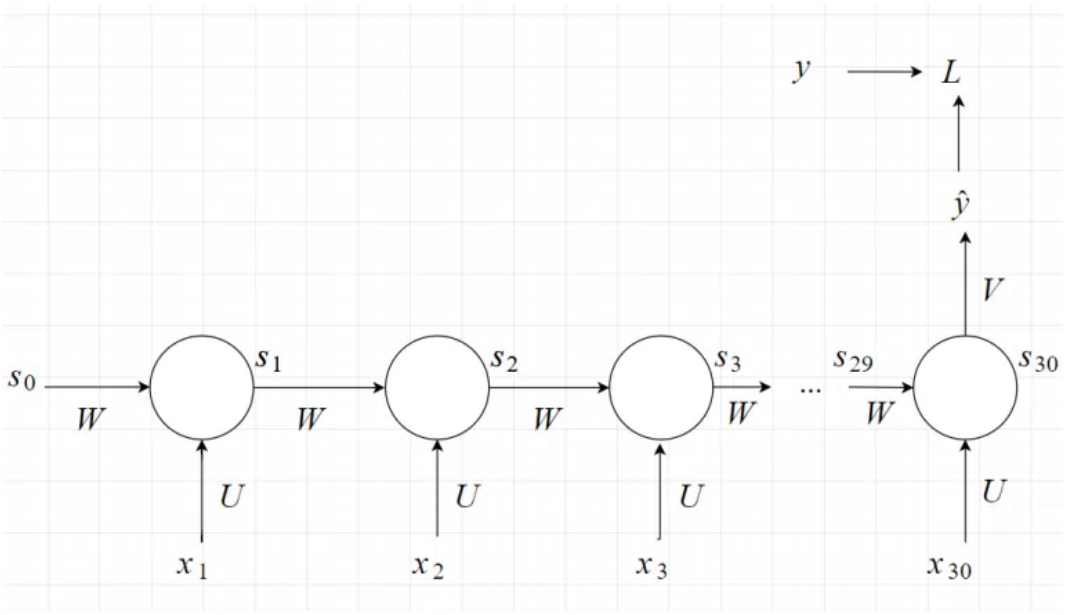
- x_i là vector có kích thước n*1, s_i là vector có kích thước m*1, y_i là vector có kích thước d*1. U là ma trận có kích thước m*n, W là ma trận

có kích thước m*m và V là ma trận có kích thước d*m.

- $s_0 = 0, s_t = f(U * x_t + W * s_{t-1})$ với t >= 1
- $\hat{y} = g(V * s_{30})$

Loss function

Loss function của cả mô hình bằng tổng loss của mỗi output, tuy nhiên ở mô hình trên chỉ có 1 output và là bài toán phân loại nên categorical cross entropy loss sẽ được sử dụng.



Loss function

Backpropagation Through Time (BPTT)

Có 3 tham số ta cần phải tìm là W, U, V. Để thực hiện gradient descent, ta cần tính: $\frac{\partial L}{\partial U}, \frac{\partial L}{\partial V}, \frac{\partial L}{\partial W}$.

Tính đạo hàm với V thì khá đơn giản:

$$\frac{\partial L}{\partial V} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial V}$$

Tuy nhiên với U, W thì lại khác.

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial s_{30}} * \frac{\partial s_{30}}{\partial W}$$

Do $s_{30} = f(W * s_{29} + V * x_{30})$ có s_{29} phụ thuộc vào W. Nên áp dụng công thức hồi cấp 3 bạn học: $(f(x) * g(x))' = f'(x) * g(x) + f(x) * g'(x)$. Ta có

$$L = -\sum_{i=1}^{10} y_i * log(\hat{y}_i)$$

- $s_0 = 0, s_t = f(U * x_t + W * s_{t-1})$ với t >= 1
- $\hat{y} = g(V * s_{30})$

$\frac{\partial s_{30}}{\partial W} = \frac{\partial s'_{30}}{\partial W} + \frac{\partial s_{30}}{\partial s_{29}} * \frac{\partial s_{29}}{\partial W}$, trong đó $\frac{\partial s'_{30}}{\partial W}$ là đạo hàm của s_{30} với W khi coi s_{29} là constant với W.

Tương tự trong biểu thức s_{29} có s_{28} phụ thuộc vào W, s_{28} có s_{27} phụ thuộc vào W ... nên áp dụng công thức trên và chain rule:

$\frac{\partial L}{\partial W} = \sum_{i=0}^{30} \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial s_{30}} * \frac{\partial s_{30}}{\partial s_i} * \frac{\partial s'_i}{\partial W}$, trong đó $\frac{\partial s_{30}}{\partial s_i} = \prod_{j=i}^{29} \frac{\partial s_{j+1}}{\partial s_j}$ và $\frac{\partial s'_i}{\partial W}$ là đạo hàm của s_i với W khi coi s_{i-1} là constant với W.

Nhìn vào công thức tính đạo hàm của L với W ở trên ta có thể thấy hiện tượng vanishing gradient ở các state đầu nên ta cần mô hình tốt hơn để giảm hiện tượng vaninshing gradient => Long short term memory (LSTM) ra đời và sẽ được giới thiệu ở bài sau. Vì trong bài toán thực tế liên quan đến time-series data thì LSTM được sử dụng phổ biến hơn là mô hình RNN thuần nên bài này không có code, bài sau sẽ có code ứng dụng với LSTM.

Bài 14: Long short term memory

Bài 12: Image segmentation với U-net

7 Comments Sort by **Newest**

Add a comment...

Xung Vanvu

em nghĩ là cái công thức đạo hàm của tích (f(x) * g(x))' = f'(x) * g(x) + f(x) * g'(x)
'

phải thay là đạo hàm của hàm hợp f(g)(x) thì đúng hơn ạ.

Like · Reply · 9w

TAGS: Deep Learning Recurrent Neural Netwrk