Recurrent Neural Network (RNN)

Introduction

Những bất lợi trong ANN, CNN:

- Input, output có thể có độ dài khác nhau và khác examples (data training)
- "Không chia sẽ những gì đã được học qua với những đoạn text khác " : Khi training 1 example (input) thì các trọng số xuất hiện cho biết dấu hiệu của đối tượng (output_desired) vừa train được

Recurrent Neural Network là dạng Artificial Neural Network, trong đó kết nối giữa các nút tạo thành một đồ thị có hướng dọc theo một trình tự. Điều này cho phép nó thể hiện hành vi ,động thái, thời gian trong một chuỗi thời gian (time sequence).

Recurrent neural networks được sử dụng trong nhận dạng giọng nói, dịch ngôn ngữ, dự đoán cổ phiếu; Nó thậm chí còn được sử dụng trong nhận dạng hình ảnh để mô tả nội dung trong hình ảnh.

Sequence Data

RNN xử lý tốt sequence data để dự đoán.

Sequence data có nhiều dạng.

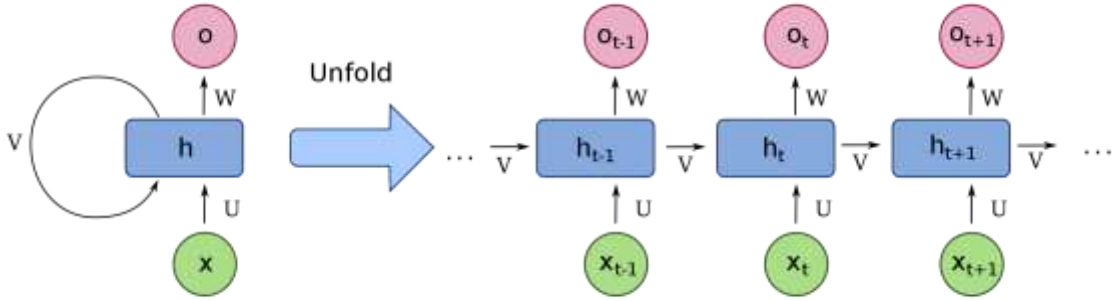
Âm thanh là một chuỗi tự nhiên. Bạn có thể cắt một quang phổ âm thanh thành các phần và nạp vào RNN.

Văn bản là một dạng khác của chuỗi. Bạn có thể ngắt văn bản thành chuỗi ký tự hoặc chuỗi các từ.

RNN (general)

Ý tưởng đằng sau RNNs là sử dụng thông tin tuần tự.

RNNs được gọi là *recurrent* bởi vì nó thực hiện một nhiệm vụ tương tự cho mọi phần tử của một chuỗi, với output phụ thuộc vào các tính toán trước đó. Một cách khác để suy nghĩ về RNN là họ có một "memory" để thu thập thông tin về những gì đã được tính toán trước đó đến hiện tại. Về lý thuyết RNNs có thể sử dụng thông tin trong các chuỗi dài tùy ý, nhưng trong thực tế chúng bị giới hạn chỉ nhìn lại một vài bước. Dưới đây là mô hình một RNN điển hình:



Hình: Recurrent Neural Network tổng quát và mô hình thể hiện diễn biến bên trong.

Với :

- x_t: input tại time step t. Ví dụ, x_1 có thể là *one-hot vector* tương ứng với từ thứ hai của câu
- h_t: hidden state tại time step t. Đây là "memory" của network.

  h_t được tính toán dựa trên hidden state trước đó và input ở bước hiện tại.

  $$h_t = f(U.x_t + W.h_{t-1})$$

  Function f thường là nonlinearity (phi tuyến tính) như tanh hay ReLU.

  $h_{-1} = 0$: yêu cầu khởi tạo để tính toán. ? (0 hay -1) -> nghĩ là 0

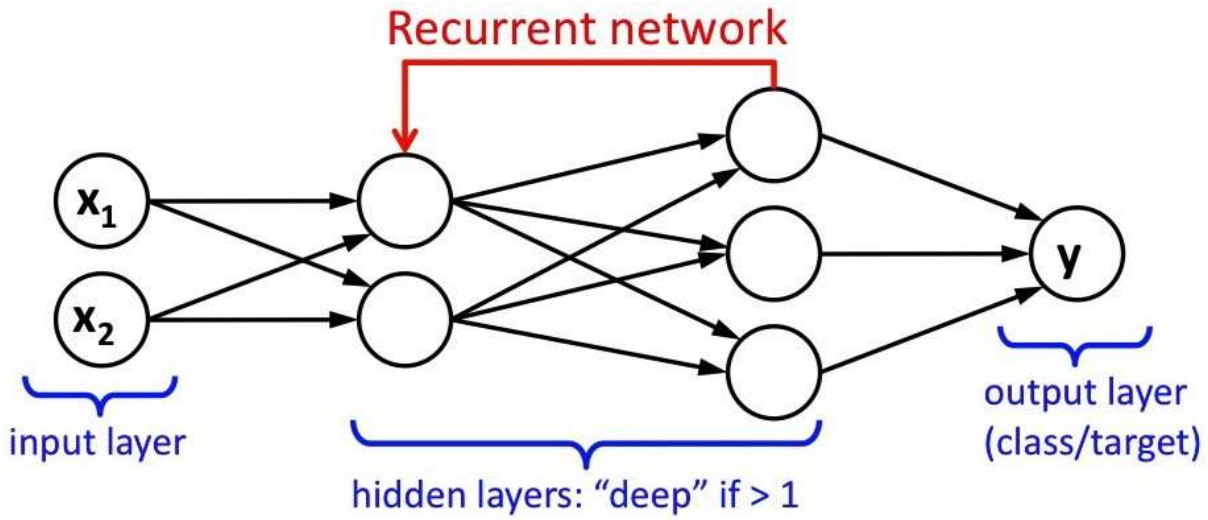
- o_t: output tại step t. Ví dụ, nếu chúng ta muốn dự đoán từ tiếp theo trong một câu, nó sẽ là một vector xác suất trong bộ từ vựng của chúng ta.

  $$o_t = Softmax(V.h_t)$$

- Các tham số U,W,V là các trọng số học được từ data.

Ví dụ, nếu chuỗi mà chúng ta quan tâm là một câu gồm 5 từ, network sẽ được tách ra thành một neural network có 5 layer, một layer cho mỗi từ.

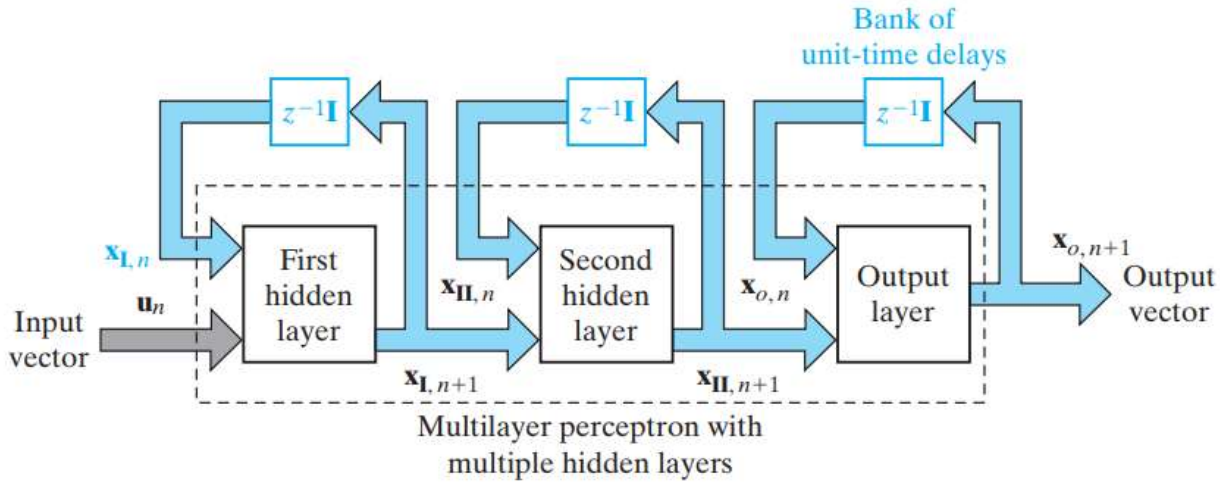
*Trong NLP, **one-hot vector**: là một ma trận 1 × N (vector) được sử dụng để phân biệt từng từ trong một từ vựng với mọi từ khác trong từ vựng. Vectơ bao gồm 0 trong tất cả các ô ngoại trừ một số 1 trong ô được sử dụng duy nhất để nhận dạng từ.



Recurrent Network Architectures

(Các biến thể của Recurrent Network)

Recurrent Multilayer Perceptrons



Hình: Recurrent multilayer perceptron

Ta có:

$$\mathbf{x}_{\mathbf{I},n+1} = \phi_{\mathbf{I}}(\mathbf{x}_{\mathbf{I},n+1}, \mathbf{U}_n)$$

$$\mathbf{x}_{\mathbf{II},n+1} = \phi_{\mathbf{II}}(\mathbf{x}_{\mathbf{II},n+1}, \mathbf{x}_{\mathbf{I},n+1})$$

...

$$\mathbf{x}_{\mathbf{o},n+1} = \phi_{\mathbf{o}}(\mathbf{x}_{\mathbf{o},n+1}, \mathbf{x}_{\mathbf{K},n+1})$$

Với:

- $\mathbf{x}_{\mathbf{I},n}$: biểu thị output của first hidden layer; $\mathbf{x}_{\mathbf{II},n}$ biểu thị output của second hidden layer, tượng tự tiếp tục.
- $\mathbf{x}_{\mathbf{o},n}$: biểu thị output cuối cùng của output layer
- $\mathbf{U}_n$: input vector
- $\phi_{\mathbf{I}}(,)$; $\phi_{\mathbf{II}}(,)$; …; $\phi_{\mathbf{o}}(,)$: biểu thị cho activation function tại first hidden layer, second hidden layer, …, output layer.

Hopfield network (cont)

Hopfield network là một RNN, trong đó tất cả các kết nối đều đối xứng. Nó đòi hỏi đầu vào cố định và do đó không phải là RNN chung, vì nó không xử lý chuỗi các mẫu. Nó đảm bảo rằng nó sẽ hội tụ. Nếu các kết nối được đào tạo bằng cách sử dụng học tập Hebbian thì mạng Hopfield có thể thực hiện như bộ nhớ nội dung mạnh mẽ, chống lại sự thay đổi kết nối.

Second order RNNs (cont)

Gated recurrent unit (cont)

Long short-term memory (cont)

Bi-directional (cont)

Learning Algorithm

Giới thiệu hai chế độ training một recurrent network, được mô tả như sau:

1. Epochwies Training:
   Trong mỗi epoch, recurrent network sử dụng một trình tự thời gian của các cặp phản hồi input–target. Bắt đầu chạy từ một số initial state cho đến khi nó đạt đến một new state, tại thời điểm đó việc đào tạo được dừng lại và network được đặt lại về trạng thái ban đầu cho epoch tiếp theo.
   Trạng thái ban đầu không nhất thiết phải giống nhau cho mỗi giai đoạn đào tạo. Thay vào đó, điều quan trọng là trạng thái ban đầu cho epoch mới khác với trạng thái mà network đạt được vào cuối epoch trước đó.
   Trong epochwise training cho các recurrent network, thuật ngữ "epoch" được sử dụng theo nghĩa khác với một multilayer perceptron thông thường. Mặc dù một epoch trong việc đào tạo một multilayer perceptron liên quan đến toàn bộ mẫu đào tạo của các cặp phản ứng input–target, một epoch trong việc đào tạo recurrent network liên quan đến một single string các cặp phản ứng input–target tạm thời liên tiếp.
2. Continuous training:
   Phương pháp đào tạo thứ hai này phù hợp cho các tình huống không có reset states hoặc on-line learning là bắt buộc (không bắt buộc có 2 tình huống trên).
   Tính năng phân biệt của continuous training là network học trong khi thực hiện xử lý tín hiệu. Nói một cách đơn giản, quá trình học tập không bao giờ dừng lại.
   Hãy xem xét, ví dụ, việc sử dụng một mạng tái phát để mô hình hóa một quá trình không ổn định như tín hiệu lời nói. Trong tình huống này, hoạt động liên tục của mạng không có thời điểm thuận tiện để dừng đào tạo và bắt đầu lại với các giá trị khác nhau cho các thông số tự do của network.

Giới thiệu hai thuật toán học (learning algorithm) cho các recurrent network, tóm tắt như sau:

1. The back-propagation-through-time (BPTT) algorithm
   Hoạt động trên tiền đề: các hoạt động tạm thời của một recurrent network có thể được mở ra (unfolded) thành một multilayer perceptron. Điều kiện này sau đó sẽ mở đường cho việc áp dụng thuật toán back-propagation chuẩn. Thuật toán BPTT có thể được thực hiện trong chế độ epochwise , chế độ continuous (real-time) hoặc kết hợp chúng.
2. The real-time recurrent learning (RTRL) algorithm
   Bắt nguồn từ state-space model được mô tả bằng phương trình ….(cont) (15.10) and (15.11).

Về cơ bản, BPTT và RTRL liên quan đến việc lan truyền các dẫn xuất (derivatives), một cái theo hướng backward và cái còn lại là forward.

Chúng có thể được sử dụng trong bất kỳ quá trình đào tạo nào yêu cầu sử dụng các dẫn xuất (derivatives). BPTT yêu cầu ít tính toán hơn RTRL, nhưng không gian bộ nhớ được yêu cầu bởi BPTT tăng nhanh khi độ dài của một chuỗi các cặp phản hồi input–target liên tục tăng.

Nói chung, BPTT là tốt hơn cho *off-line (batch) training*, và RTRL là phù hợp hơn cho *on-line continuous training*.

Trong mọi trường hợp, hai thuật toán này chia sẻ nhiều tính năng phổ biến:

1. Cả hai đều dựa trên phương pháp gradient descent
2. Cả hai đều tương đối đơn giản để thực hiện, nhưng có thể chậm để hội tụ
3. Chúng có liên quan trong biểu diễn biểu đồ signal-flow (dòng tín hiệu) của thuật toán BPTT, có thể thu được từ sự chuyển đổi biểu diễn biểu đồ signal-flow của một dạng nhất định của thuật toán RTRL.

BACK PROPAGATION THROUGH TIME

Thuật toán back-propagation-through-time (BPTT) để đào tạo một recurrent network là một mở rộng của thuật toán back-propagation tiêu chuẩn.

Giả sử:

- N thể hiện cho một recurrent network cần phải học một nhiệm vụ thời gian (temporal task), bắt đầu từ time n_0 đến tất cả con đường đến time n.
- N^* thể hiện feedforward network (kết quả từ việc mở ra (unfolding) hoạt động tạm thời (temporal operation) của recurrent network N).

Unfolded network N^* có liên quan đến mạng gốc N như sau:

1. Với mỗi time-step trong khoảng thời gian(n_0, n), network có một layer chứa K neurons, trong đó K là số lượng neuron có trong network.
2. Trong mỗi layer của network, có một bản sao của mỗi neuron trong network.
3. Với mỗi time-step $l \in (n_0, n)$, kết nối synap từ neuron i trong lớp l tới neuron j trong lớp $l + 1$ của network là một bản sao của kết nối synap từ neuron i đến neuron j trong network.

Tùy thuộc vào việc sử dụng epochwise training hay continuous (real-time) training mà ta áp dụng,triển khai back propagation through time.

Epochwise Back Propagation Through Time (cont)

Truncated Back Propagation Through Time (cont)

REAL-TIME RECURRENT LEARNING (cont)

VANISHING GRADIENTS IN RECURRENT NETWORKS (cont)

Nguồn

https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

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

https://en.wikipedia.org/wiki/Recurrent_neural_network

https://en.wikipedia.org/wiki/One-hot

Simon O. Haykin (2009),Neural Networks and Learning Machines, Third Edition, Pearson

https://indico.io/blog/sequence-modeling-neuralnets-part1/