Chào bạn! Nếu bạn muốn crawl dữ liệu thời tiết của địa phương mình đang sống, bạn có thể sử dụng các thư viện Python như BeautifulSoup hoặc Scrapy để lấy dữ liệu từ trang web chứa thông tin thời tiết. [Bạn có thể tìm hiểu thêm về cách crawl dữ liệu từ trang web bằng Python tại](https://bing.com/search?q=crawl+d%e1%bb%af+li%e1%bb%87u+th%e1%bb%9di+ti%e1%ba%bft+c%e1%bb%a7a+%c4%91%e1%bb%8ba+ph%c6%b0%c6%a1ng+m%c3%acnh+%c4%91ang+s%e1%bb%91ng) [1](https://bing.com/search?q=crawl+d%E1%BB%AF+li%E1%BB%87u+th%E1%BB%9Di+ti%E1%BA%BFt+c%E1%BB%A7a+%C4%91%E1%BB%8Ba+ph%C6%B0%C6%A1ng+m%C3%ACnh+%C4%91ang+s%E1%BB%91ng). [Nếu bạn muốn sử dụng API để lấy dữ liệu thời tiết, bạn có thể tìm hiểu thêm tại](https://bepos.io/blogs/crawl-data/" \t "_blank) [2](https://bepos.io/blogs/crawl-data/). Chúc bạn thành công!

[1. bing.com](https://bing.com/search?q=crawl+d%e1%bb%af+li%e1%bb%87u+th%e1%bb%9di+ti%e1%ba%bft+c%e1%bb%a7a+%c4%91%e1%bb%8ba+ph%c6%b0%c6%a1ng+m%c3%acnh+%c4%91ang+s%e1%bb%91ng)

[2. bepos.io](https://bepos.io/blogs/crawl-data/)

[3. vi.ldplayer.net](https://vi.ldplayer.net/apps/life-channel-accurate-local-weather-forecast-on-pc.html)

[4. accuweather.com](https://www.accuweather.com/vi/vn/national/current-weather-maps)

[5. vi.weatherspark.com](https://vi.weatherspark.com/history)

Tất nhiên, đây là một ví dụ đơn giản về cách crawl dữ liệu thời tiết bằng Python sử dụng thư viện BeautifulSoup:

import requests

from bs4 import BeautifulSoup

url = 'https://www.weather.com/weather/today/l/21.28,105.50?par=google&temp=c'

page = requests.get(url)

soup = BeautifulSoup(page.content, 'html.parser')

location = soup.find('h1', class\_='CurrentConditions--location--1Ayv3').get\_text().strip()

time = soup.find('div', class\_='CurrentConditions--timestamp--1SWy5').get\_text().strip()

temperature = soup.find('span', class\_='CurrentConditions--tempValue--3KcTQ').get\_text().strip()

description = soup.find('div', class\_='CurrentConditions--phraseValue--2xXSr').get\_text().strip()

print(f'Location: {location}')

print(f'Time: {time}')

print(f'Temperature: {temperature}')

print(f'Description: {description}')

Trong ví dụ này, chúng ta sử dụng thư viện requests để lấy nội dung của trang web và BeautifulSoup để phân tích cú pháp HTML của trang web. Sau đó, chúng ta tìm các phần tử HTML chứa thông tin thời tiết và trích xuất nội dung của chúng. Cuối cùng, chúng ta in ra các giá trị đã trích xuất. [Bạn có thể tìm hiểu thêm về cách crawl dữ liệu từ trang web bằng Python tại](https://ichi.pro/vi/huong-dan-ve-truc-quan-hoa-du-lieu-du-lieu-thoi-tiet-91189164306819" \t "_blank) [1](https://ichi.pro/vi/huong-dan-ve-truc-quan-hoa-du-lieu-du-lieu-thoi-tiet-91189164306819). Chúc bạn thành công!

[1. ichi.pro](https://ichi.pro/vi/huong-dan-ve-truc-quan-hoa-du-lieu-du-lieu-thoi-tiet-91189164306819)

[2. ichi.pro](https://ichi.pro/vi/python-trich-xuat-du-lieu-thoi-tiet-cap-do-ngay-59691880792044)

[3. mcivietnam.com](https://mcivietnam.com/blog-detail/huong-dan-lay-du-lieu-tu-web-bang-python-de-hieu/)

[4. codelearn.io](https://codelearn.io/sharing/web-crawling-voi-beautifulsoup4-python)

[5. youtube.com](https://www.youtube.com/watch?v=A-Usa0w-Nxw)

[6. meteoguru.uk](https://meteoguru.uk/)

[7. ichi.pro](https://ichi.pro/vi/cach-tao-he-thong-canh-bao-thoi-tiet-bang-python-105189306721434)

[8. codelearn.io](https://codelearn.io/sharing/python-crawl-du-lieu-tao-bai-bao-nhanh)

Mở đầu chung

Giới thiệu về mô hình lstm

Giới thiệu về dự báo thời tiết

Giới thiệu về tập dữ liệu? (quan trọng):

* Đánh giá phim
* Dự báo/cảnh báo thời tiết (nhiệt độ, lượng mưa)
* Giá cổ phiếu, chứng khoán
* Mực nước sông, độ mặn,…
* Đoán từ kế tiếp trong văn bản Việt Nam
* Phân tích, phân loại tâm lý, cảm xúc (dựa vào văn bản)
* Máy đọc hiểu văn bản?
* Hành vi ???
* Hành động, cử chỉ con người
* Mức tiêu thụ năng lượng
* Lạm phát ?
* Chatbot
* Chuyển động giao thông ?
* Tóm tắt văn bản tự động
* Trích xuất thông tin từ văn bản

Giới thiệu về công cụ

Giới thiệu về code mô hình

Chạy mô hình và kết quả

Dự đoán và kết luận

(hướng phát triển)

Mạng Bộ nhớ ngắn hạn dài (LSTM) là một loại mạng thần kinh tái phát (RNN) được thiết kế để giải quyết vấn đề độ dốc biến mất được tìm thấy trong RNN truyền thống. Ưu điểm đáng chú ý của nó so với các RNN khác, mô hình Markov ẩn và các phương pháp học trình tự khác nhau nằm ở chỗ nó tương đối không nhạy cảm với độ dài khoảng cách. Mục tiêu chính là cung cấp khả năng bộ nhớ ngắn hạn cho RNN có thể tồn tại qua hàng nghìn bước thời gian, do đó có tên là "bộ nhớ ngắn hạn dài".

LSTM tìm thấy ứng dụng trong các lĩnh vực khác nhau như phân loại, xử lý và dự đoán dữ liệu chuỗi thời gian. Ví dụ bao gồm các ứng dụng trong nhận dạng chữ viết tay, nhận dạng giọng nói, dịch máy, phát hiện hoạt động giọng nói, điều khiển robot, trò chơi điện tử và chăm sóc sức khỏe.

Một đơn vị LSTM thông thường bao gồm một ô, một cổng đầu vào, một cổng đầu ra và một cổng quên. Tế bào ghi nhớ các giá trị trong những khoảng thời gian tùy ý và ba cổng điều chỉnh luồng thông tin vào và ra khỏi tế bào. Cổng quên quyết định loại bỏ thông tin nào khỏi trạng thái trước đó bằng cách gán trạng thái trước đó, so với đầu vào hiện tại, giá trị từ 0 đến 1. Giá trị (làm tròn) là 1 có nghĩa là giữ thông tin và giá trị 0 có nghĩa là vứt bỏ nó. Cổng đầu vào quyết định những thông tin mới nào cần lưu trữ ở trạng thái hiện tại, sử dụng hệ thống tương tự như cổng quên. Cổng đầu ra kiểm soát những phần thông tin nào ở trạng thái hiện tại được đưa ra đầu ra bằng cách gán giá trị từ 0 đến 1 cho thông tin, xem xét trạng thái trước đó và trạng thái hiện tại. Việc xuất thông tin có liên quan từ trạng thái hiện tại một cách có chọn lọc cho phép mạng LSTM duy trì các mối phụ thuộc lâu dài, hữu ích để đưa ra dự đoán, cả trong các bước thời gian hiện tại và tương lai.

Một đơn vị LSTM điển hình bao gồm một ô, một cổng đầu vào, một cổng đầu ra và một cổng quên. Ô đóng vai trò quan trọng trong việc lưu giữ các giá trị trong các khoảng thời gian tùy ý, trong khi ba cổng điều chỉnh luồng thông tin vào và ra trong ô.

Cổng quên có trách nhiệm xác định thông tin nào cần loại bỏ khỏi trạng thái trước đó. Quyết định này được thực hiện bằng cách gán giá trị từ 0 đến 1 cho trạng thái trước đó khi so sánh với đầu vào hiện tại. Giá trị (làm tròn) là 1 biểu thị việc giữ lại thông tin, trong khi giá trị 0 biểu thị việc loại bỏ thông tin đó.

Tương tự, các cổng đầu vào quyết định những phần thông tin mới nào sẽ được lưu trữ ở trạng thái hiện tại, sử dụng cơ chế tương tự như các cổng quên. Mặt khác, cổng đầu ra chi phối việc lựa chọn thông tin từ trạng thái hiện tại đến đầu ra. Quyết định này được thực hiện bằng cách gán một giá trị trong khoảng từ 0 đến 1 cho thông tin, có tính đến cả trạng thái trước đó và hiện tại.

Bằng cách xuất ra có chọn lọc thông tin liên quan từ trạng thái hiện tại, mạng LSTM bảo tồn một cách khéo léo các phụ thuộc lâu dài, có giá trị. Khả năng này cho phép mạng đưa ra dự đoán chính xác ở cả bước thời gian hiện tại và tương lai.

Lịch sử:

Bộ nhớ Dài-Ngắn Hạn (LSTM) là một đột phá đáng chú ý được phát minh bởi Hochreiter và Schmidhuber vào năm 1997, nhanh chóng đạt được sự công nhận và thiết lập các bản ghi chính xác trong nhiều lĩnh vực ứng dụng.

Khoảng năm 2007, LSTM đã chuyển đổi ngành công nghiệp nhận dạng giọng nói, vượt trội so với các mô hình truyền thống trong nhiều ứng dụng giọng nói. Vào năm 2009, mạng LSTM đào tạo bởi Connectionist Temporal Classification (CTC) đã tạo nên lịch sử khi trở thành RNN đầu tiên giành chiến thắng trong các cuộc thi nhận dạng mẫu, đặc biệt là trong cuộc thi nhận dạng chữ viết tay được kết nối. Năm 2014, Baidu của Trung Quốc đã sử dụng RNN do CTC đào tạo để vượt qua tiêu chuẩn bộ dữ liệu nhận dạng giọng nói 2S09 Switchboard Hub5'00 mà không cần sử dụng bất kỳ phương pháp xử lý giọng nói truyền thống nào.

Không chỉ giỏi trong lĩnh vực giọng nói, LSTM còn đã cải thiện đáng kể khả năng nhận dạng giọng nói với lượng từ vựng lớn và tổng hợp văn bản thành giọng nói. Google Android đã tích hợp LSTM vào tính năng nhận dạng giọng nói của mình. Năm 2015, Google đã báo cáo về bước nhảy vọt ấn tượng với hiệu suất tăng đến 49% trong tính năng nhận dạng giọng nói, chủ yếu nhờ sử dụng LSTM được đào tạo bởi CTC.

LSTM tiếp tục phá kỷ lục trong nhiều lĩnh vực khác nhau, bao gồm cải tiến đáng kể trong dịch máy, mô hình hóa ngôn ngữ và xử lý ngôn ngữ đa ngôn ngữ. Sự kết hợp của LSTM với mạng thần kinh tích chập (CNN) đã đẩy mạnh khả năng chú thích hình ảnh tự động, đóng góp quan trọng vào tiến bộ của lĩnh vực này.

Bộ nhớ ngắn hạn dài (LSTM) được Hochreiter và Schmidhuber giới thiệu vào năm 1997, đánh dấu một bước đột phá quan trọng mà từ đó đã đạt được những thành tựu đáng khen ngợi trên nhiều lĩnh vực ứng dụng khác nhau.

Khoảng năm 2007, LSTM đã bắt đầu một giai đoạn chuyển đổi trong nhận dạng giọng nói, vượt qua các mô hình thông thường trong nhiều ứng dụng giọng nói. Năm 2009, mạng LSTM được đào tạo bởi Phân loại tạm thời kết nối (CTC) đã đạt được một cột mốc lịch sử khi trở thành Mạng thần kinh tái phát (RNN) đầu tiên giành chiến thắng trong các cuộc thi nhận dạng mẫu. Chiến thắng này đặc biệt đáng chú ý trong các cuộc thi nhận dạng chữ viết tay có kết nối. Vào năm 2014, công ty Baidu của Trung Quốc đã sử dụng RNN do CTC đào tạo để vượt qua các tiêu chuẩn của bộ dữ liệu nhận dạng giọng nói Hub5'00 của Tổng đài 2S09 mà không cần dùng đến bất kỳ phương pháp xử lý giọng nói truyền thống nào.

Tác động của LSTM mở rộng đến việc nâng cao khả năng nhận dạng giọng nói với vốn từ vựng lớn và tổng hợp chuyển văn bản thành giọng nói, nổi bật trong Google Android. Vào năm 2015, tính năng nhận dạng giọng nói của Google đã chứng kiến bước nhảy vọt về hiệu suất đáng chú ý là 49%, được ghi nhận là nhờ LSTM do CTC đào tạo.

LSTM không chỉ phá vỡ các kỷ lục về đổi mới dịch máy, mô hình hóa ngôn ngữ và xử lý ngôn ngữ đa ngôn ngữ mà còn cho thấy những tiến bộ đáng chú ý trong việc kết hợp với Mạng thần kinh chuyển đổi (CNN). Sự hợp tác này đã cải thiện đáng kể độ chính xác của chú thích hình ảnh tự động, góp phần đáng kể vào tiến bộ của lĩnh vực này.

Động lực

Về lý thuyết, RNN cổ điển (hoặc "vanilla") có thể theo dõi các phụ thuộc dài hạn tùy ý trong chuỗi đầu vào. Vấn đề với RNN vanilla có tính chất tính toán (hoặc thực tế): khi huấn luyện RNN vanilla bằng cách truyền ngược, độ dốc dài hạn được truyền ngược có thể "biến mất" (nghĩa là chúng có thể có xu hướng về 0) hoặc " nổ tung" (nghĩa là chúng có thể tiến tới vô cùng), do các tính toán liên quan đến quá trình này sử dụng các số có độ chính xác hữu hạn. RNN sử dụng các đơn vị LSTM giải quyết một phần vấn đề biến mất độ dốc, bởi vì các đơn vị LSTM cho phép độ dốc cũng không thay đổi. Tuy nhiên, mạng LSTM vẫn có thể gặp phải vấn đề bùng nổ độ dốc.

Trực giác đằng sau kiến trúc LSTM là tạo ra một mô-đun bổ sung trong mạng lưới thần kinh để tìm hiểu khi nào nên nhớ và khi nào nên quên thông tin thích hợp. Nói cách khác, mạng tìm hiểu một cách hiệu quả thông tin nào có thể cần thiết sau này theo trình tự và khi nào thông tin đó không còn cần thiết nữa. Chẳng hạn, trong bối cảnh xử lý ngôn ngữ tự nhiên, mạng có thể học các phụ thuộc ngữ pháp. LSTM có thể xử lý câu "Dave, do những tuyên bố gây tranh cãi của anh ấy, giờ là một kẻ ngang ngược" bằng cách ghi nhớ giới tính và số lượng ngữ pháp (có khả năng thống kê) của chủ đề Dave, lưu ý rằng thông tin này phù hợp với đại từ của anh ấy và lưu ý rằng thông tin này không còn quan trọng sau động từ is.

Động lực:

Động lực đằng sau việc khám phá các mạng Bộ nhớ ngắn hạn dài (LSTM) bắt nguồn từ những thách thức cố hữu mà Mạng thần kinh tái phát (RNN) cổ điển hoặc "vanilla" phải đối mặt trong việc nắm bắt và duy trì một cách hiệu quả các phụ thuộc dài hạn trong chuỗi đầu vào. Trong khi về mặt lý thuyết, RNN vanilla có thể xử lý các phụ thuộc dài hạn tùy ý, thì các vấn đề thực tế lại phát sinh trong quá trình đào tạo với lan truyền ngược. Vấn đề nằm ở bản chất tính toán của RNN vanilla, trong đó độ dốc dài hạn có thể biến mất (gần bằng 0) hoặc bùng nổ (tiến đến vô cực) do các số có độ chính xác hữu hạn được sử dụng trong tính toán.

Để giải quyết thách thức này, các đơn vị LSTM đã được giới thiệu để giảm bớt vấn đề độ dốc biến mất ở một mức độ nào đó, cho phép độ dốc chảy mà không giảm đi. Tuy nhiên, khả năng xảy ra sự cố bùng nổ độ dốc vẫn tồn tại trong mạng LSTM.

Ý tưởng nền tảng đằng sau kiến trúc LSTM là tích hợp một mô-đun bổ sung vào mạng lưới thần kinh để tìm hiểu những khoảnh khắc tối ưu để ghi nhớ và quên thông tin quan trọng. Nói một cách đơn giản hơn, mạng có khả năng nhận biết thông tin nào có liên quan cho các bước trong tương lai và khi một số chi tiết nhất định trở nên lỗi thời. Ví dụ, trong lĩnh vực xử lý ngôn ngữ tự nhiên, LSTM có thể học các phụ thuộc ngữ pháp. Hãy xem xét câu "Dave, do những tuyên bố gây tranh cãi của anh ấy, giờ đây đã trở thành một kẻ bị hạ đẳng." LSTM xử lý câu này có thể nhớ giới tính ngữ pháp và số lượng của chủ ngữ, Dave, có liên quan đến đại từ tiếp theo "của anh ấy". Đồng thời, nó có thể nhận ra rằng thông tin này trở nên không liên quan sau khi động từ "is" xuất hiện. Do đó, mạng LSTM đưa ra giải pháp cho thách thức nắm bắt và quản lý các phần phụ thuộc dài hạn một cách hiệu quả.

Đào tạo

Một RNN sử dụng các đơn vị LSTM có thể được huấn luyện theo kiểu được giám sát trên một tập hợp các chuỗi huấn luyện, sử dụng thuật toán tối ưu hóa như giảm độ dốc kết hợp với lan truyền ngược theo thời gian để tính toán độ dốc cần thiết trong quá trình tối ưu hóa, nhằm thay đổi từng trọng số của LSTM mạng tỷ lệ với đạo hàm của lỗi (ở lớp đầu ra của mạng LSTM) đối với trọng số tương ứng.

Một vấn đề với việc sử dụng độ dốc giảm dần cho RNN tiêu chuẩn là độ dốc lỗi biến mất nhanh chóng theo cấp số nhân với độ trễ thời gian giữa các sự kiện quan trọng. Điều này là do

lim

�

→

∞

�

�

=

0

{\displaystyle \lim \_{n\to \infty }W^{n}=0} nếu bán kính quang phổ của

�

W nhỏ hơn 1.[2][21]

Tuy nhiên, với các đơn vị LSTM, khi các giá trị lỗi được truyền ngược từ lớp đầu ra, lỗi vẫn còn trong ô của đơn vị LSTM. "Băng chuyền lỗi" này liên tục cung cấp lỗi trở lại từng cổng của đơn vị LSTM cho đến khi chúng học cách cắt bỏ giá trị.

Đào tạo:

Huấn luyện Mạng thần kinh tái phát (RNN) bằng cách sử dụng các đơn vị Bộ nhớ ngắn hạn dài (LSTM) thường bao gồm một cách tiếp cận được giám sát trên một tập hợp các trình tự huấn luyện. Quá trình tối ưu hóa sử dụng các thuật toán như giảm độ dốc kết hợp với lan truyền ngược theo thời gian. Điều này đảm bảo tính toán độ dốc cần thiết để điều chỉnh trọng số của mạng LSTM. Việc điều chỉnh tỷ lệ thuận với đạo hàm của sai số ở lớp đầu ra của mạng LSTM liên quan đến các trọng số tương ứng.

Một thách thức đáng kể mà các RNN tiêu chuẩn phải đối mặt trong quá trình huấn luyện là vấn đề biến mất độ dốc. Trong những trường hợp này, độ dốc lỗi biến mất theo cấp số nhân theo độ trễ thời gian giữa các sự kiện quan trọng. Hiện tượng này đặc biệt rõ rệt khi bán kính phổ của ma trận trọng số (W) nhỏ hơn 1.

Tuy nhiên, các đơn vị LSTM đưa ra giải pháp cho vấn đề độ dốc biến mất. Khi các giá trị lỗi được truyền ngược từ lớp đầu ra, các lỗi này vẫn tồn tại trong ô của đơn vị LSTM. Điều này tạo thành một "băng chuyền lỗi" trong đó lỗi liên tục quay trở lại từng cổng của đơn vị LSTM. Thông qua vòng phản hồi liên tục này, các cổng dần dần học cách kiểm soát và cắt bỏ giá trị lỗi một cách thích hợp. Cơ chế này ngăn chặn một cách hiệu quả sự biến mất hoàn toàn của độ dốc lỗi, cho phép mạng LSTM tìm hiểu và duy trì sự phụ thuộc lâu dài trong chuỗi đầu vào.

Chức năng tính điểm CTC

Nhiều ứng dụng sử dụng các ngăn xếp LSTM RNN và huấn luyện chúng bằng cách phân loại thời gian kết nối (CTC) để tìm ma trận trọng số RNN giúp tối đa hóa xác suất của các chuỗi nhãn trong tập huấn luyện, cho các chuỗi đầu vào tương ứng. CTC đạt được cả sự liên kết và sự công nhận.

Lựa chọn thay thế

Đôi khi, việc đào tạo (các bộ phận của) LSTM bằng phương pháp tiến hóa thần kinh hoặc bằng phương pháp gradient chính sách có thể có lợi, đặc biệt khi không có "giáo viên" (nghĩa là nhãn đào tạo).

Thành công

Đã có một số câu chuyện thành công về đào tạo RNN theo kiểu không giám sát với các đơn vị LSTM.

Năm 2018, Bill Gates gọi đây là “cột mốc lớn trong việc thúc đẩy trí tuệ nhân tạo” khi các bot do OpenAI phát triển có thể đánh bại con người trong trò chơi Dota 2. OpenAI Five bao gồm năm mạng lưới thần kinh độc lập nhưng được phối hợp. Mỗi mạng được đào tạo bằng phương pháp gradient chính sách mà không cần giáo viên giám sát và chứa Bộ nhớ ngắn hạn dài hạn 1024 đơn vị, một lớp, có thể nhìn thấy trạng thái trò chơi hiện tại và phát ra các hành động thông qua một số đầu hành động có thể có.

Vào năm 2018, OpenAI cũng đã đào tạo một LSTM tương tự bằng các gradient chính sách để điều khiển một bàn tay robot giống con người, điều khiển các vật thể vật lý với sự khéo léo chưa từng có.

Vào năm 2019, chương trình AlphaStar của DeepMind đã sử dụng lõi LSTM sâu để chơi trò chơi điện tử phức tạp Starcraft II một cách xuất sắc. Đây được coi là bước tiến đáng kể đối với Trí tuệ nhân tạo tổng hợp.

Chức năng điểm CTC:

Trong các ứng dụng khác nhau, việc sử dụng Mạng thần kinh tái phát (LSTM) bộ nhớ ngắn hạn dài (LSTM) xếp chồng bao gồm việc huấn luyện chúng bằng cách sử dụng Phân loại tạm thời kết nối (CTC). CTC nhằm mục đích tìm ra ma trận trọng số RNN tối ưu giúp tối đa hóa xác suất của chuỗi nhãn trong một tập huấn luyện nhất định, tương ứng với chuỗi đầu vào. Cách tiếp cận này đạt được hiệu quả cả nhiệm vụ liên kết và nhận dạng trong bối cảnh học theo trình tự.

Các lựa chọn thay thế:

Mặc dù thực tiễn phổ biến liên quan đến việc đào tạo LSTM (hoặc các phần của LSTM) thông qua phân loại thời gian theo chủ nghĩa kết nối, các phương pháp thay thế như phương pháp tiến hóa thần kinh hoặc phương pháp gradient chính sách có thể có lợi. Những phương pháp thay thế này trở nên đặc biệt hữu ích trong các tình huống không có sẵn "giáo viên", tức là nhãn đào tạo.

Câu chuyện thành công:

Đã có những câu chuyện thành công đáng chú ý trong việc đào tạo Mạng thần kinh tái phát (RNN) với các đơn vị LSTM theo cách không được giám sát:

Năm 2018, OpenAI đã đạt được một cột mốc quan trọng khi các bot do tổ chức có tên OpenAI Five phát triển đã vượt qua người chơi con người trong trò chơi Dota 2. OpenAI Five bao gồm năm mạng lưới thần kinh độc lập nhưng được phối hợp, mỗi mạng được đào tạo bằng phương pháp gradient chính sách mà không có một giáo viên giám sát. Mỗi mạng kết hợp Bộ nhớ ngắn hạn dài 1024 đơn vị, một lớp.

OpenAI tiếp tục chứng minh tính linh hoạt của LSTM bằng cách đào tạo một mạng tương tự thông qua các gradient chính sách để điều khiển bàn tay robot giống con người với sự khéo léo chưa từng có trong việc thao tác các vật thể vật lý.

Năm 2019, chương trình AlphaStar của DeepMind đã thể hiện sức mạnh của lõi LSTM sâu trong việc chơi xuất sắc trò chơi điện tử phức tạp Starcraft II. Thành tựu này được công nhận là một bước tiến đáng kể hướng tới việc hiện thực hóa Trí tuệ nhân tạo tổng hợp.

Chức Năng Điểm CTC:

Trong nhiều ứng dụng khác nhau, việc sử dụng mạng thần kinh tái phát (RNN) với đơn vị bộ nhớ ngắn hạn dài (LSTM) xếp chồng thường đi kèm với việc huấn luyện chúng thông qua phân loại tạm thời kết nối (CTC). CTC nhằm mục đích tìm ra ma trận trọng số RNN tối ưu nhằm tối đa hóa xác suất của chuỗi nhãn trong một tập huấn luyện cụ thể, tương ứng với chuỗi đầu vào. Cách tiếp cận này hiệu quả cho cả nhiệm vụ liên kết và nhận dạng trong bối cảnh học theo trình tự.

Các Lựa Chọn Thay Thế:

Mặc dù việc đào tạo LSTM (hoặc một phần của LSTM) thông qua CTC là phổ biến, nhưng có những phương pháp thay thế như neuroevolution hoặc gradient chính sách có thể mang lại lợi ích. Những phương pháp thay thế này trở nên hữu ích đặc biệt khi không có "giáo viên" (tức là không có nhãn huấn luyện).

Câu Chuyện Thành Công:

Có nhiều câu chuyện thành công đáng chú ý về việc đào tạo mạng thần kinh tái phát (RNN) với đơn vị LSTM một cách không được giám sát:

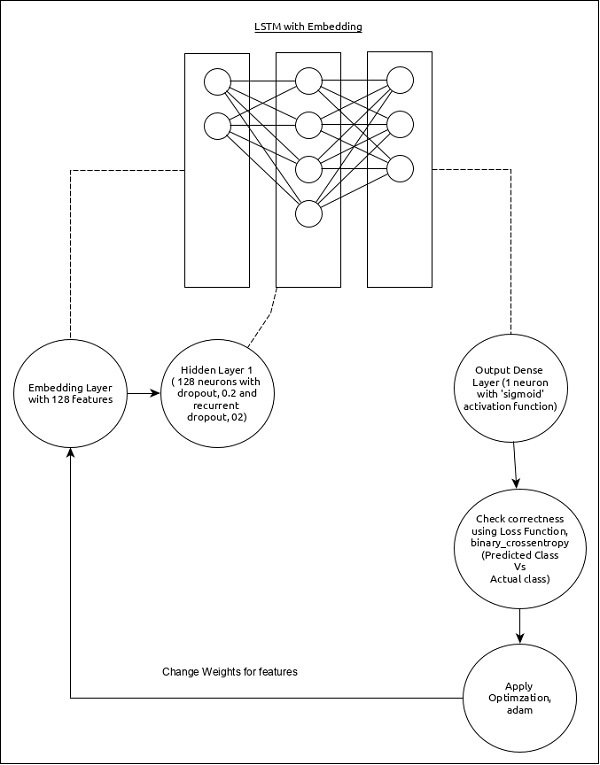
Năm 2018, OpenAI đạt một cột mốc quan trọng khi các bot phát triển bởi tổ chức, được gọi là OpenAI Five, vượt qua người chơi con người trong trò chơi Dota 2. OpenAI Five bao gồm năm mạng lưới thần kinh độc lập nhưng được điều phối, mỗi mạng được đào tạo bằng phương pháp gradient chính sách mà không có giáo viên giám sát. Mỗi mạng sử dụng LSTM với một lớp có 1024 đơn vị.

OpenAI tiếp tục chứng minh tính linh hoạt của LSTM bằng cách đào tạo một mạng tương tự thông qua gradient chính sách để điều khiển một bàn tay robot giống con người, thực hiện việc thao tác các vật thể vật lý một cách khéo léo chưa từng có.

Năm 2019, chương trình AlphaStar của DeepMind đã sử dụng lõi LSTM sâu để xuất sắc trong trò chơi điện tử phức tạp Starcraft II. Thành tựu này được coi là một bước tiến quan trọng đối với Trí Tuệ Nhân Tạo Tổng Hợp.

Mô hình phân tích trình tự có thể được trình bày như sau:

Các tính năng cốt lõi của mô hình như sau:



Lớp đầu vào sử dụng lớp Embedding(Nhúng) với 128 đặc trưng (features).

Lớp đầu tiên, Dense bao gồm 128 đơn vị với số normal dropout và số recurrent dropout liên tục được đặt thành 0,2.

Lớp đầu ra, Dense bao gồm 1 đơn vị và hàm kích hoạt (activation) ‘sigmoid’.

Sử dụng binary\_crossentropy làm hàm mất mát.

Sử dụng adam làm Trình tối ưu hóa.

Sử dụng accuracy (độ chính xác ) làm thước đo.

Sử dụng 32 batch size.

15 epochs.

Danh mục viết tắt

MSE Mean Square Error SSE Sum of Squared Errors LNR Linear

Regression SVR Support Vector Regression RNN Recurrent Neural NetworkBPTT Backpropagation Through Time LSTM Long Short Term MemoryTóm tắt đồ án

Trong bối cảnh cả thế giới đang điên đảo vì thị trường tài chính và chứng

khoán, trí tuệ nhân tạo cũng là một lĩnh vực đang lớn mạnh lên từng ngày.

Tại sao chúng ta không thử kết hợp chúng lại để tạo nên những điều kì diệu.

Trong đồ án môn học này, chúng em tập trung xây dựng các mô hình dự

đoán giá tương lai của thị trường chứng khoán dựa trên dữ liệu của những

năm trước.

Tập dữ liệu được cung cấp gồm dữ liệu lịch sử chứa các bản ghi về giá cổ

phiếu của nhiều cổ phiếu khác nhau như Apple, Tesla, Microsoft, Facebook.

Tập dữ liệu cũng chứa giá cổ phiếu theo ngày với giá mở cửa, đóng cửa,

cao và thấp cùng với khối lượng giao dịch vào ngày đó. Tập dữ liệu được lấy

từ nguồn: https://www.kaggle.com/altruistdelhite04/loanpredictionproblem-dataset

Dữ liệu đầu ra của bài toán

Mục tiêu của đồ án là nghiên cứu về ý tưởng, kỹ thuật cốt lõi của các thuật

toán hồi quy đã được học và áp dụng chúng để giải quyết một vấn đề thực

tiễn trong đời sống. qua đó chúng em đánh giá kết quả thực nghiệm, phân

tích thách thức và đề xuất hướng giải quyết trong tương lai

2.4.3 Long Short-Term Memory (LSTM) Giới thiệu về Recurrent Neural

Network Trước khi đi sâu vào giải thích chi tiết mạng LSTM, nhóm sẽ giới

thiệu sơ qua về mạng nơ-ron hồi quy (Recurrent Neural Network -RNN).

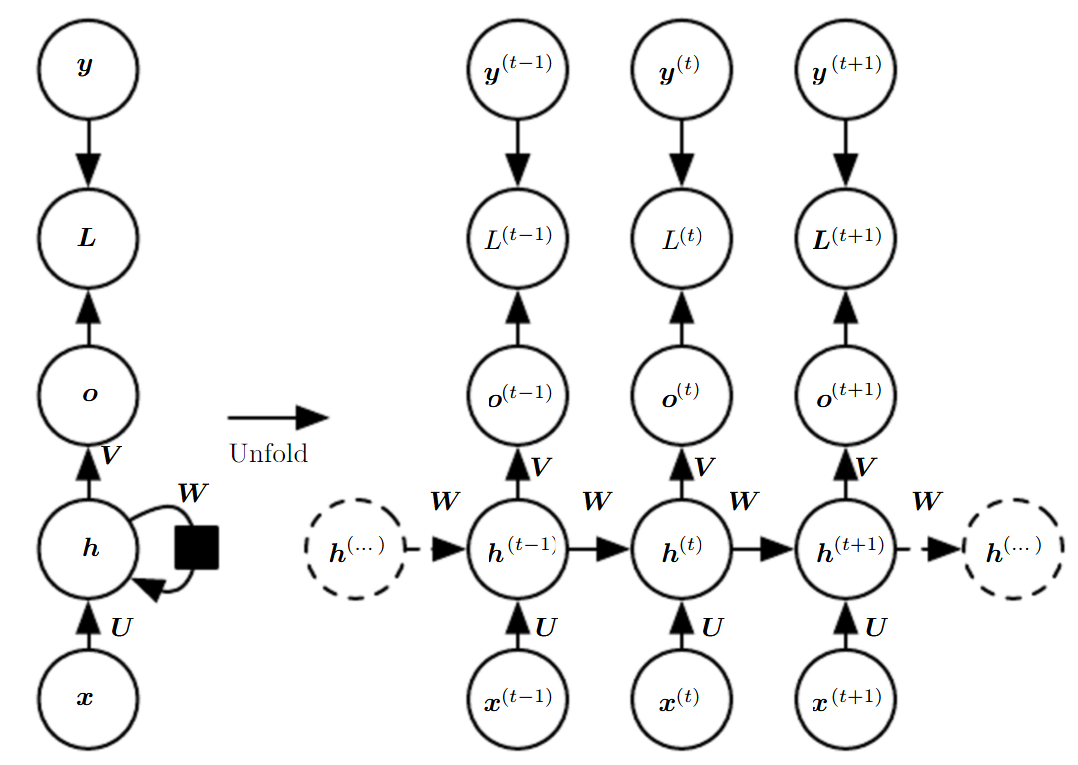
Đây là mạng nơ-ron nhân tạo được thiết kế cho việc xử lý các loại dữ liệu có

dạng chuỗi tuần tự. Trong mạng RNN, trạng thái ẩn tại mỗi bước thời gian sẽ

được tính toán dựa vào dữ liệu đầu vào tại bước thời gian tương ứng và các

thông tin có được từ bước thời gian trước đó, tạo khả năng ghi nhớ các

thông tin đã được tính toán ở những bước thời gian trước cho mạng



Kiến trúc của một mạng RNN cơ bản khi được duỗi ra

Trong Hình 2.1, xét tại mỗi bước thời gian t theo chiều từ dưới lên trên,

x(t) là giá trị đầu vào.

h(t) là trạng thái ẩn

o(t) là giá trị đầu ra.

U, W, V là các ma trận trọng số của mạng RNN.

L là hàm tính mất mát giữa giá trị ra o(t) từ mạng RNN và giá trị đầu ra

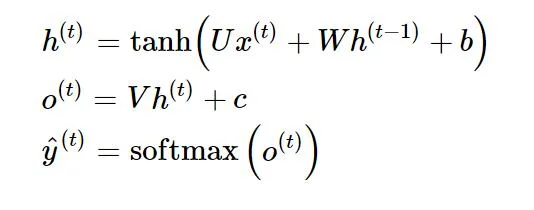
chuẩn y(t) từ tập dữ liệu.

Đi sâu vào kiến thức chi tiết hơn, chúng ta xem các vector x(1) , x(2) , …,

x(τ) đại diện cho các phần tử trong chuỗi dữ liệu đầu vào, tại mỗi bước thời

gian t, mạng RNN nhận lần lượt từng vector x (t) và thực hiện những tính

toán để ánh xạ thành chuỗi đầu ra được mô tả bởi các phương trình sau:



Trong đó: • x (t): Giá trị đầu vào tại bước thời gian t • h (t): Trạng thái ẩn

bước tại thời gian t • (t): Giá trị đầu ra tại bước thời gian t • y (t): Vector xác

xuất đã chuẩn hóa qua hàm softmax tại bước thời gian t • U, V, W: Các ma

trận trong số mạng RNN tương ứng với các kết nối theo chiều lần lượt là từ

dầu vào đến trạng thái ẩn, từ trạng thái ẩn đến đầu ra và từ trạng thái ẩn

đến trạng thái ẩn. • b,c: Độ lệch (bias)

A diagram of a mathematical equation

Description automatically generated

Kiến trúc chi tiết của một mạng RNN tại mỗi bước thời gian.

22

Các vấn đề về gradient trong quá trình huấn luyện

Gradient biến mất (Vanishing Gradient Problem) và gradient bùng nổ (Exploding Grandient Problem) là những vấn đề gặp phải khi sử dụng các kỹ

thuật tối ưu hóa trọng số dựa trên gradient để huấn luyện mạng nơ-ron. Các

vấn đề này thường gặp phải là do việc lựa chọn các hàm kích hoạt không

hợp lí hoặc số lượng các lớp ẩn của mạng quá lớn. Đặc biệt, các vấn đề này

thường hay xuất hiện trong quá trình huấn luyện các mạng nơ-ron hồi quy

Trong thuật toán BPTT. Khi chúng ta càng quay lùi về các bước thời gian

trước đó thì các giá trị gradient càng giảm dần, điều này làm giảm tốc độ

hội tụ của các trọng số do sự thay đổi hầu như rất nhỏ. Trong một số trường

hợp khác, các gradient có giá trị rất lớn khiến cho quá trình cập nhật các

trọng số bị phân kỳ và vấn đề này được gọi là gradient bùng nổ. Các vấn đề

về gradient biến mất thường được quan tâm hơn vấn đề gradient bùng nổ do vấn đề gradient biến mất khó có thể được nhận biết trong khi gradient

bùng nổ có thể dễ dàng quan sát và nhận biết hơn. Có nhiều nghiên cứu đề

xuất các giải pháp để giải quyết những vấn đề này như lựa chọn hàm kích

hoạt hợp lý, thiết lập các kích thước cho mạng hợp lý hoặc khởi tạo các

trọng số ban đầu phù hợp khi huấn luyện. Một trong các giải pháp cụ thể có

thể chỉ ra là thuật toán Truncated BPTT, một biến thể cái tiến của BPTT

được áp dụng trong quá trình huấn luyện mạng nơ-ron hồi quy trên các

chuỗi dài.

Ngoài ra, cơ chế của mạng LSTM được đề xuất đã khắc phục được các vấn

đề này sẽ được giới thiệu trong phần tiếp theo.

Cơ chế hoạt động của mạng LSTM LSTM là một phiên bản mở rộng của

mạng RNN, được đề xuất vào năm 1997 bởi Sepp Hochreiter và Jurgen

Schmidhuber. LSTM được thiết kế để giải quyết các bài toán về phụ thuộc

xa (long-term dependencies) trong mạng RNN do bị ảnh hưởng bởi vấn đề

gradient biến mất.

Có thể hiểu một cách đơn giản là mạng RNN cơ bản trong thực tế không có

khảnăng ghi nhớ thông tin từ các bước có khoảng cách xa và do đó những

phần tử đầu tiên trong chuỗi đầu vào không có nhiều ảnh hưởng đến các

kết quả tính toán dự đoán phần tử cho chuỗi đầu ra trong các bước sau.

A diagram of a memory cell

Description automatically generated

Hình 2.8. Sơ đồ biểu diễn kiến trúc bên trong của một tế bào LSTM.

Mạng LSTM có thể bao gồm nhiều tế bào LSTM (LSTM memory cell) liên

kết với nhau và kiến trúc cụ thể của mỗi tế bào được biểu diễn như trong

Hình 2.6.

Ý tưởng của LSTM là bổ sung thêm trạng thái bên trong tế bào (cell internal

state) st và ba cổng sàng lọc các thông tin đầu vào và đầu ra cho tế bào

bao gồm: forget gate ƒt, input gate it và output gate ot.

Tại mỗi bước thời gian t, các cổng đều lần lượt nhận giá trị đầu vào xt (đại

diện cho một phần tử trong chuỗi đầu vào) và giá trị ht-1 có được từ đầu ra

của memory cell từ bước thời gian trước đó t - 1. Các cổng đều đóng vai trò

có nhiệm vụ sàng lọc thông tin với mỗi mục đích khác nhau:

. Forget gate:

Có nhiệm vụ loại bỏ những thông tin không cần thiết nhận được khỏi cell

internal state • Input gate: Có nhiệm vụ chọn lọc những thông tin cần thiết

nào được thêm vào cell internal state

• Output gate: Có nhiệm vụ xác định những thông tin nào từ cell internal state được sử dụng như đầu ra Trước khi trình bày các phương trình mô tả cơ chế hoạt động bên trong của một tế bào LSTM, chúng ta sẽ thống nhất quy ước một số ký hiệu được sử dụng sau đây: • xt là vector đầu vào tại mỗi bước thời gian t • Wf,x, Wf.h, W�� ̃,��, W�� ̃,��, Wi,x, Wi,h, Wo,x, Wo,h là các ma trận trọng số trong mỗi tế bào LSTM. • bf, b�� ̃, bi, bo là các vector bias.

• ƒt, it, ot lần lượt chứa các giá trị kích hoạt lần lượt cho các cổng forget gate, input gate và output gate tương ứng. • st, �� ̃lần lượt là các vector đại diện cho cell internal state và candidate value. • ht là giá trị đầu ra của tế bào LSTM. Trong quá trình lan truyền xuôi (forward pass), cell internal state st và giá trị đầu ra ht được tính như sau: Ở bước đầu tiên, tế bào LSTM quyết định những thông tin nào cần được loại bỏ từ cell internal state ở bước thời gian trước đó St-1. Activation value ƒt của forget gate tại bước thời gian tđược tính dựa trên giá trị đầu vào hiện tại xt giá trị đầu ra ht-1 từ tế bào LSTM ở bước trước đó và bias bf của forget gate. Hàm sigmoid function biến đổi tất cả activation value về miền có giá trị trong khoảng từ 0 (hoàn toàn quên) và 1 (hoàn toàn ghi nhớ):

A black text with a plus and a white background

Description automatically generated

Ở bước thứ hai, tế bào LSTM quyết định những thông tin nào cần được thêm vào cell internal state st. Bước này bao gồm hai quá trình tính toán đối với �� ̃và ƒt. Candidate value �� ̃t biểu diễn những thông tin tiềm năng cần được thêm vào cell internal state được tính như sau:

A math symbols on a white background

Description automatically generated

Activation value it của input gate theo đó cũng được tính như sau:

A math equations and a plus

Description automatically generated

Ở bước thứ ba, giá lới của cell internal state st được tính dựa trên kết quả tính toán thu được từ các bước trước với phép nhân Hadamard theo từng phần tử được ký hiệu bằng o:

A black text with black letters

Description automatically generated with medium confidence

Ở bước cuối cùng, giá trị đầu ra ht của tế bào LSTM được tính toán dựa theo hai phương trình sau:

A math equations on a white background

Description automatically generated

Ý tưởng chính của RNN (Recurrent Neural Network) là sử dụng chuỗi các thông tin. Trong các mạng nơ-ron truyền thống tất cả các đầu vào và cả đầu ra là độc lập với nhau. Tức là chúng không liên kết thành chuỗi với nhau. Nhưng các mô hình này không phù hợp trong rất nhiều bài toán. Ví dụ, nếu muốn đoán từ tiếp theo có thể xuất hiện trong một câu thì ta cũng cần biết các từ trước đó xuất hiện lần lượt thế nào chứ nhỉ? RNN được gọi là hồi quy (Recurrent) bởi lẽ chúng thực hiện cùng một tác vụ cho tất cả các phần tử của một chuỗi với đầu ra phụ thuộc vào cả các phép tính trước đó. Nói cách khác, RNN có khả năng nhớ các thông tin được tính toán trước đó. Trên lý thuyết, RNN có thể sử dụng được thông tin của một văn bản rất dài, tuy nhiên thực tế thì nó chỉ có thể nhớ được một vài bước trước đó (ta cùng bàn cụ thể vấn đề này sau) mà thôi. Về cơ bản một mạng RNN có dạng như sau:



Mô hình trên mô tả phép triển khai nội dung của một RNN. Triển khai ở đây có thể hiểu đơn giản là ta vẽ ra một mạng nơ-ron chuỗi tuần tự. Ví dụ ta có một câu gồm 5 chữ “Đẹp trai lắm gái theo”, thì mạng nơ-ron được triển khai sẽ gồm 5 tầng nơ-ron tương ứng với mỗi chữ một tầng. Lúc đó việc tính toán bên trong RNN được thực hiện như sau:

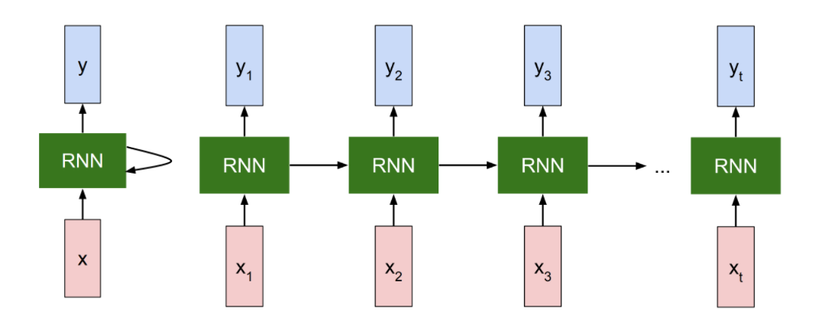
5.3. Mạng LSTM

Gần đây, mạng LSTM mà ta có đề cập một chút phía trên được chú ý và sử dụng khá phổ biến. Về cơ bản mô hình của LSTM không khác mô hình truyền thống của RNN, nhưng chúng sử dụng hàm tính toán khác ở các trạng thái ẩn. Bộ nhớ của LSTM được gọi là tế bào (Cell) và bạn có thể tưởng tượng rằng chúng là các hộp đen nhận đầu vào là trạng thái phía trước ht-1 và đầu vào hiện tại xt . Bên trong hộp đen này sẽ tự quyết định cái gì cần phải nhớ và cái gì sẽ xoá đi. Sau đó, chúng sẽ kết hợp với trạng thái phía trước, nhớ hiện tại và đầu vào hiện tại. Vì vậy mà ta ta có thể truy xuất được quan hệ của các từ phụ thuộc xa nhau rất hiệu quả. Có thể khi mới làm quen với LSTM thì chúng hơi khó hiểu đôi chút, nhưng nếu bạn có hứng thú thì hãy xem bài viết xuất sắc này (bản dịch tại đây).

Dự báo thời tiết là một bài toán hết sức quan trọng trong cuộc sống hằng ngày. Đầu vào của bài toán sẽ là số liệu về mây, mực nước, lượng mưa, nhiệt độ, độ ẩm, tốc độ gió,... theo ngày của một khu vực nhất định và đầu ra sẽ là các thông số tương ứng của ngày tiếp theo. Trong thực tế việc dự đoán gặp nhiều sai số, khó khăn khi có những ngày thời tiết thất thường, không theo quy luật hoặc thiên tai làm cho dữ liệu không còn đúng. Trong phạm vi bài tập lớn này, mô hình sẽ chỉ sử dụng các thông số khí tượng là nhiệt độ, độ ẩm, lượng mưa và sử dụng thuật toán RNN và LSTM để dự đoán các thông số nhiệt độ, độ ẩm, có mưahay không cho ngày hôm sau

Hướng giải quyết bài toánDữ liệu được lấy từ năm 2014 đến năm 2021 của trạm khí tượng tại một vùng núi phía Bắc. Dữ liệu ban đầu có một số chỗ bị NULL và một số chỗ dữ liệu thất thường (do thiên tai), vậy nên ta cần có bước tiền xử lý để bổ sung những chỗ dữ liệu bị thiếu và chuẩn hóa những chỗ dữ liệu thất thường. Ta sẽ trực quan hóa dữ liệu trên biểu đồ để kiểm tra sự tuần hoàn.Sau khi làm sạch dữ liệu, ta sẽ lựa chọn Feature Vector chứa các số liệu khí tượng và chia thành 3 tập: training set, optimal set và test set.Cuối cùng, chúng ta sẽ tiến hành xây dựng 2 mô hình máy học là RNN và LSTM để dự đoán và so sánh kết quả của 2 mô hình này

Cơ sở lý thuyết2.1 Recurrent Neural NetworkRecurrent Neural Network (RNN) là một thuật toán cực kì quan trọng có khả năng xử lý thông tin dạng chuỗi. RNN có rất nhiều ứng dụng trong thực tế như: Machine Translation, mô hình hóa ngôn ngữ và sinh văn bản, nhận dạng giọng nói, mô tả hình ảnh: RNN kết hợp cùng CNN để sinh ra mô tả cho hình ảnh chưa được gán nhãn.Với mạng Nơ-ron thông thường, dữ liệu sẽ được cho vào cùng một lúc. Nhưng trong thực tế sẽ có rất nhiều bài toán, dữ liệu của chúng ta mang ý nghĩa trình tự, nếu thay đổi trình tự kết quả sẽ khác. Mạng Nơ-ron RNN có khả năng giải quyết bài toán này.Hình ảnh dưới đây sẽ mô tả RNN hoạt động như thế nào



Mỗi block RNN sẽ lấy thông tin từ các block trước và input hiện tại. Các x ở đây đại diện cho dữ liệu đầu vào lần lượt (được chia theo time step). đại diện cho time step thứ t, và là output của một step

Mạng thần kinh hiện đang được ứng dụng rất phổ biến trong các ngành khoa học nhằm cải thiện năng suất và chất lượng của cuộc sống. Một trong những ứng dụng phổ biến là dự đoán kết quả của quá trình sản xuất dựa vào dữ liệu được thu thập trong thời gian trước đó. Bài báo sử dụng dữ liệu sản lượng sữa được cập nhật hàng tháng của một nhà máy sản xuất sữa và dựa vào đó, dự đoán sản lượng sữa trong khoảng thời gian trong tương lại. Dữ liệu được chia thành 2 phần: Một phần dùng cho việc tập huấn dữ liệu (training data) và một phần dùng để kiểm thử (testing data), dữ liệu được sắp xếp theo thứ tự thời gian nên testing data sẽ bao gồm các mốc thời gian trong tương lai so với training data. Tác giả sau đó sử dụng mạng thần kinh LSTM, được hỗ trợ bởi gói sklearn và Keras – bộ kit rất nổi tiếng trong việc hỗ trợ các thuật toán liên quan đến học máy – để dự đoán sản lượng của testing data. Hiệu quả của việc phỏng đoán được đánh giá dựa vào hàm tính sai số RMSE. Sai sô càng thấp thì dự đoán càng chính xác. Để việc so sánh trở nên dễ nhìn, tác giả đã mô phỏng kết quả thành dạng biểu đồ so sánh giá trị thực và giá trị phỏng đoán. Kết quả cho thấy giá trị thực và giá trị phỏng đoán xấp xỉ bằng nhau. Chứng tỏ việc sử dụng LSTM cho dữ liệu dạng chuỗi sắp xếp theo thứ tự thời gian là hiệu quả và là tiền đề tốt để phát triển các bài toán tương tự mang tính ứng dụng

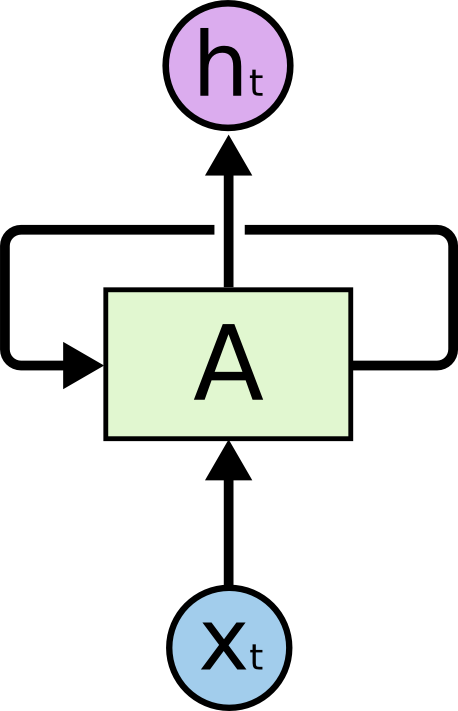
cao hơn.

1. Mạng hồi quy RNN

Con người không bắt đầu suy nghĩ của họ từ đầu tại tất cả các thời điểm. Cũng như bạn đang đọc bài viết này, bạn hiểu mỗi chữ ở đây dựa vào từ bạn đã hiểu các chữ trước đó chứ không phải là đọc tới đâu ném hết đi tới đó, rồi lại bắt đầu suy nghĩ lại từ đầu tới chữ bạn đang đọc. Tức là tư duy đã có một bộ nhớ để lưu lại những gì diễn ra trước đó.

Tuy nhiên các mô hình mạng nơ-ron truyền thống thì không thể làm được việc đó, đó có thể coi là một khuyết điểm chính của mạng nơ-ron truyền thống. Ví dụ, bạn muốn phân loại các bối cảnh xảy ra ở tất cả các thời điểm trong một bộ phim, thì đúng là không rõ làm thế nào để có thể hiểu được một tình huống trong phim mà lại phụ thuộc vào các tình huống trước đó nếusử dụng các mạng nơ-ron truyền thống.

Mạng nơ-ron hồi quy (Recurrent Neural Network) sinh ra để giải quyết vấn đề đó. Mạng này chứa các vòng lặp bên trong cho phép thông tin có thể lưu lại được.



Hình vẽ trên mô tả một đoạn của mạng nơ-ron hồi quy A với đầu vào là xt​ và đầu ra là ht​. Một vòng lặp cho phép thông tin có thể được truyền từ bước này qua bước này qua bước khác của mạng nơ-ron.

Các vòng lặp này khiến cho mạng nơ-ron hồi quy trông có vẻ khó hiểu. Tuy nhiên, nếu bạn để ý một chút thì nó không khác mấy so với các mạng nơ-ron thuần. Một mạng nơ-ron hồi quy có thể được coi là nhiều bản sao chép của cùng một mạng, trong đó mỗi đầu ra của mạng này là đầu vào của một mạng sao chép khác. Nói thì hơi khó hiểu, nhưng bạn hãy xem hình mô tả sau:

A group of letters and numbers on a black background

Description automatically generated

Chuỗi lặp lại các mạng này chính là phân giải của mạng nơ-ron hồi quy, các vòng lặp khiến chúng tạo thành một chuỗi danh sách các mạng sao chép nhau. Bạn có thấy nó khác gì một mạng nơ-ron thuần không? Không khác gì phải không? Các nút của mạng vẫn nhận đầu vào và có đầu ra hệt như mạng nơ-ron thuần.

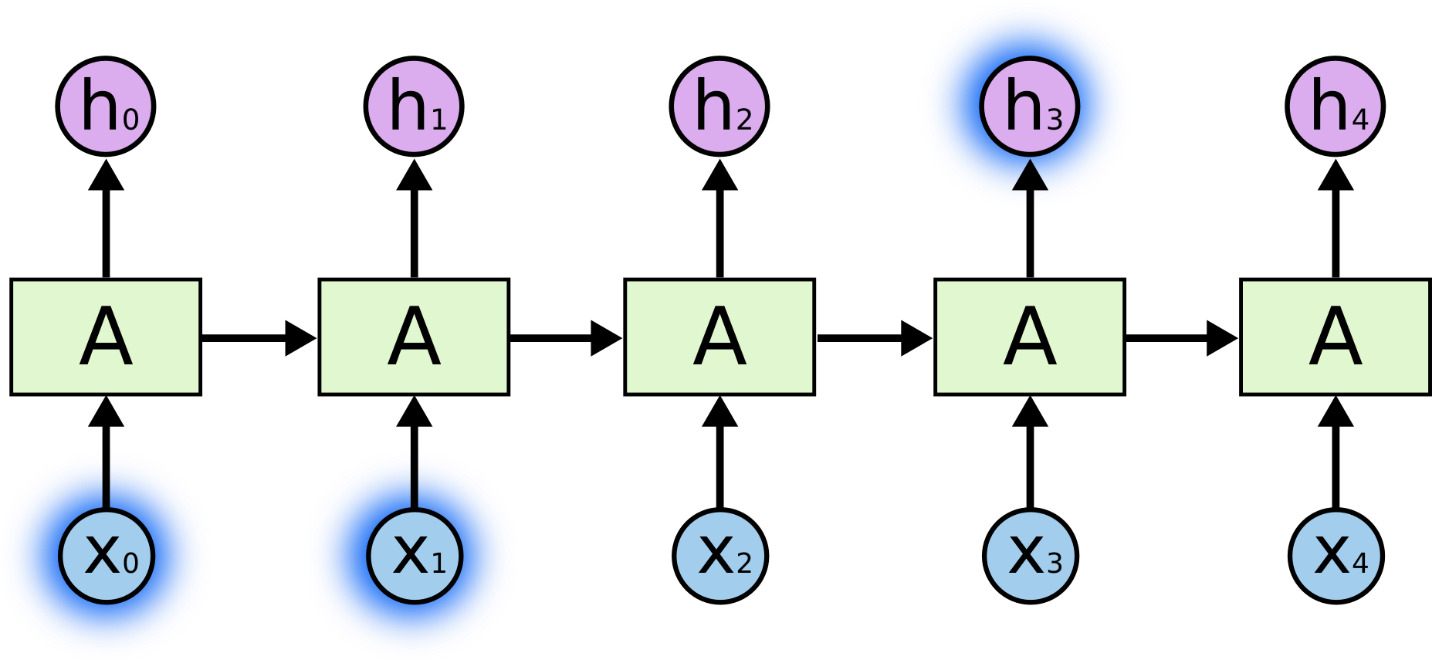
Trong vài năm gần đây, việc ứng dụng RNN đã đưa ra được nhiều kết quả không thể tin nổi trong nhiều lĩnh vực: nhận dạng giọng nói, mô hình hóa ngôn ngữ, dịch máy, mô tả ảnh,… Danh sách vẫn còn đang được mở rộng tiếp. Anh Andrej Karpathy đã đề cập tới một số kêt quả mà RNN mang lại tại bài viết này, nên tôi sẽ không bàn luận thêm nữa. Nhưng tôi vẫn muốn thốt lên rằng chúng thật là quá tuyệt vời.

Đằng sau sự thành công này chính là sự đóng góp của LSTM. LSTM là một dạng đặc biệt của mạng nơ-ron hồi quy, với nhiều bài toán thì nó tốt hơn mạng hồi quy thuần. Hầu hết các kết quả thú vị thu được từ mạng RNN là được sử dụng với LSTM. Trong bài viết này, ta sẽ cùng khám phá xem mạng LSTM là cái gì nhé.

2. Vấn đề phụ thuộc xa

Một điểm nổi bật của RNN chính là ý tưởng kết nối các thông tin phía trước để dự đoán cho hiện tại. Việc này tương tự như ta sử dụng các cảnh trước của bộ phim để hiểu được cảnh hiện thời. Nếu mà RNN có thể làm được việc đó thì chúng sẽ cực kì hữu dụng, tuy nhiên liệu chúng có thể làm được không? Câu trả lời là còn tùy.

Đôi lúc ta chỉ cần xem lại thông tin vừa có thôi là đủ để biết được tình huống hiện tại. Ví dụ, ta có câu: “các đám may trên bầu trời” thì ta chỉ cần đọc tới “các đám may trên bầu” là đủ biết được chữ tiếp theo là “trời” rồi. Trong tình huống này, khoảng cách tới thông tin có được cần để dự đoán là nhỏ, nên RNN hoàn toàn có thể học được.



Nhưng trong nhiều tình huống ta buộc phải sử dụng nhiều ngữ cảnh hơn để suy luận. Ví dụ, dự đoán chữ cuối cùng trong đoạn: “I grew up in France… I speak fluent French.”. Rõ ràng là các thông tin gần (”I speak fluent”) chỉ có phép ta biết được đằng sau nó sẽ là tên của một ngôn ngữ nào đó, còn không thể nào biết được đó là tiếng gì. Muốn biết là tiếng gì, thì ta cần phải có thêm ngữ cảnh “I grew up in France” nữa mới có thể suy luận được. Rõ ràng là khoảng cách thông tin lúc này có thể đã khá xa rồi.

Thật không may là với khoảng cách càng lớn dần thì RNN bắt đầu không thể nhớ và học được nữa.

A black background with different colored circles and letters

Description automatically generated

Về mặt lý thuyết, rõ ràng là RNN có khả năng xử lý các phụ thuộc xa (long-term dependencies). Chúng ta có thể xem xét và cài đặt các tham số sao cho khéo là có thể giải quyết được vấn đề này. Tuy nhiên, đáng tiếc trong thực tế RNN có vẻ không thể học được các tham số đó. Vấn đề này đã được khám phá khá sâu bởi Hochreiter (1991) [tiếng Đức] và Bengio, et al. (1994), trong các bài báo của mình, họ đã tìm được nhưng lý do căn bản để giải thích tại sao RNN không thể học được.

Tuy nhiên, rất cám ơn là LSTM không vấp phải vấn đề đó!

3. Mạng LSTM

Mạng bộ nhớ dài-ngắn (Long Short Term Memory networks), thường được gọi là LSTM - là một dạng đặc biệt của RNN, nó có khả năng học được các phụ thuộc xa. LSTM được giới thiệu bởi Hochreiter & Schmidhuber (1997), và sau đó đã được cải tiến và phổ biến bởi rất nhiều người trong ngành. Chúng hoạt động cực kì hiệu quả trên nhiều bài toán khác nhau nên dần đã trở nên phổ biến như hiện nay.

LSTM được thiết kế để tránh được vấn đề phụ thuộc xa (long-term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, chứ ta không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kì can thiệp nào.

Mọi mạng hồi quy đều có dạng là một chuỗi các mô-đun lặp đi lặp lại của mạng nơ-ron. Với mạng RNN chuẩn, các mô-dun này có cấu trúc rất đơn giản, thường là một tầng tanh.

A screenshot of a computer

Description automatically generated

LSTM cũng có kiến trúc dạng chuỗi như vậy, nhưng các mô-đun trong nó có cấu trúc khác với mạng RNN chuẩn. Thay vì chỉ có một tầng mạng nơ-ron, chúng có tới 4 tầng tương tác với nhau một cách rất đặc biệt.

A screenshot of a computer

Description automatically generated

Giờ thì đừng hoang mang về chi tiết bên trong chúng ngay, chúng ta sẽ khám phá chúng chi tiết chúng ở bước sau. Điều bạn cần làm bây giờ là làm hãy làm quen với các kí hiệu mà ta sẽ sử dụng ở dưới đây:

A black arrow pointing to the left

Description automatically generated

Ở sơ đồ trên, mỗi một đường mang một véc-tơ từ đầu ra của một nút tới đầu vào của một nút khác. Các hình trong màu hồng biểu diễn các phép toán như phép cộng véc-tơ chẳng hạn, còn các ô màu vàng được sử dụng để học trong các từng mạng nơ-ron. Các đường hợp nhau kí hiệu việc kết hợp, còn các đường rẽ nhánh ám chỉ nội dung của nó được sao chép và chuyển tới các nơi khác nhau.

3.1. Ý tưởng cốt lõi của LSTM

Chìa khóa của LSTM là trạng thái tế bào (cell state) - chính đường chạy thông ngang phía trên của sơ đồ hình vẽ.

Trạng thái tế bào là một dạng giống như băng truyền. Nó chạy xuyên suốt tất cả các mắt xích (các nút mạng) và chỉ tương tác tuyến tính đôi chút. Vì vậy mà các thông tin có thể dễ dàng truyền đi thông suốt mà không sợ bị thay đổi.

A diagram of a diagram

Description automatically generated

LSTM có khả năng bỏ đi hoặc thêm vào các thông tin cần thiết cho trạng thái tế báo, chúng được điều chỉnh cẩn thận bởi các nhóm được gọi là cổng (gate).

Các cổng là nơi sàng lọc thông tin đi qua nó, chúng được kết hợp bởi một tầng mạng sigmoid và một phép nhân.

A black background with a yellow and pink rectangle with a black letter

Description automatically generated

Tầng sigmoid sẽ cho đầu ra là một số trong khoản

[

0

,

1

]

[0,1], mô tả có bao nhiêu thông tin có thể được thông qua. Khi đầu ra là

0

0 thì có nghĩa là không cho thông tin nào qua cả, còn khi là

1

1 thì có nghĩa là cho tất cả các thông tin đi qua nó.

Một LSTM gồm có 3 cổng như vậy để duy trì và điều hành trạng thái của tế bào.

3.2. Bên trong LSTM

Bước đầu tiên của LSTM là quyết định xem thông tin nào cần bỏ đi từ trạng thái tế bào. Quyết định này được đưa ra bởi tầng sigmoid - gọi là “tầng cổng quên” (forget gate layer). Nó sẽ lấy đầu vào là

ℎ

�

−

1

h

t−1

​

và

�

�

x

t

​

rồi đưa ra kết quả là một số trong khoảng

[

0

,

1

]

[0,1] cho mỗi số trong trạng thái tế bào

�

�

−

1

C

t−1

​

. Đẩu ra là

1

1 thể hiện rằng nó giữ toàn bộ thông tin lại, còn

0

0 chỉ rằng taonf bộ thông tin sẽ bị bỏ đi.

Quay trở lại với ví dụ mô hình ngôn ngữ dự đoán từ tiếp theo dựa trên tất cả các từ trước đó, với những bài toán như vậy, thì trạng thái tế bào có thể sẽ mang thông tin về giới tính của một nhân vật nào đó giúp ta sử dụng được đại từ nhân xưng chuẩn xác. Tuy nhiên, khi đề cập tới một người khác thì ta sẽ không muốn nhớ tới giới tính của nhân vật nữa, vì nó không còn tác dụng gì với chủ thế mới này.

A diagram of a rectangular object with arrows pointing to the side

Description automatically generated

Bước tiếp theo là quyết định xem thông tin mới nào ta sẽ lưu vào trạng thái tế bào. Việc này gồm 2 phần. Đầu tiên là sử dụng một tầng sigmoid được gọi là “tầng cổng vào” (input gate layer) để quyết định giá trị nào ta sẽ cập nhập. Tiếp theo là một tầng

�

�

�

ℎ

tanh tạo ra một véc-tơ cho giá trị mới

�

�

~

C

t

​

~

​

nhằm thêm vào cho trạng thái. Trong bước tiếp theo, ta sẽ kết hợp 2 giá trị đó lại để tạo ra một cập nhập cho trạng thái.

Chẳng hạn với ví dụ mô hình ngôn ngữ của ta, ta sẽ muốn thêm giới tính của nhân vật mới này vào trạng thái tế bào và thay thế giới tính của nhân vật trước đó.

A diagram of a rectangular object with arrows pointing to the side

Description automatically generated

Giờ là lúc cập nhập trạng thái tế bào cũ

�

�

−

1

C

t−1

​

thành trạng thái mới

�

�

C

t

​

. Ở các bước trước đó đã quyết định những việc cần làm, nên giờ ta chỉ cần thực hiện là xong.

Ta sẽ nhân trạng thái cũ với

�

�

f

t

​

để bỏ đi những thông tin ta quyết định quên lúc trước. Sau đó cộng thêm

�

�

∗

�

�

~

i

t

​

∗

C

t

​

~

​

. Trạng thái mơi thu được này phụ thuộc vào việc ta quyết định cập nhập mỗi giá trị trạng thái ra sao.

Với bài toàn mô hình ngôn ngữ, chính là việc ta bỏ đi thông tin về giới tính của nhân vật cũ, và thêm thông tin về giới tính của nhân vật mới như ta đã quyết định ở các bước trước đó.

A diagram of a rectangular object with arrows pointing up

Description automatically generated

Cuối cùng, ta cần quyết định xem ta muốn đầu ra là gì. Giá trị đầu ra sẽ dựa vào trạng thái tế bào, nhưng sẽ được tiếp tục sàng lọc. Đầu tiên, ta chạy một tầng sigmoid để quyết định phần nào của trạng thái tế bào ta muốn xuất ra. Sau đó, ta đưa nó trạng thái tế bảo qua một hàm

�

�

�

ℎ

tanh để co giá trị nó về khoảng

[

−

1

,

1

]

[−1,1], và nhân nó với đầu ra của cổng sigmoid để được giá trị đầu ra ta mong muốn.

Với ví dụ về mô hình ngôn ngữ, chỉ cần xem chủ thể mà ta có thể đưa ra thông tin về một trạng từ đi sau đó. Ví dụ, nếu đầu ra của chủ thể là số ít hoặc số nhiều thì ta có thể biết được dạng của trạng từ đi theo sau nó phải như thế nào.

A diagram of a graphing diagram

Description automatically generated with medium confidence

4. Các biến thể của bộ nhớ dài hạn

Những thứ ta vừa mô tả ở trên là một LSTM khá bình thường. Nhưng không phải tất cả các LTSM đều giống như vậy. Thực tế, các bài báo về LTSM đều sử dụng một phiên bản hơi khác so với mô hình LTSM chuẩn. Sự khác nhau không lớn, nhưng chúng giúp giải quyết phần nào đó trong cấu trúc của LTSM.

Một dạng LTSM phổ biến được giới thiệu bởi Gers & Schmidhuber (2000) được thêm các đường kết nối “peephole connections”, làm cho các tầng cổng nhận được giá trị đầu vào là trạng thái tế bào.

A rectangular object with arrows pointing to the center

Description automatically generated

Hình trên mô tả các đường được thêm vào mọi cổng, nhưng cũng có những bài báo chỉ thêm cho một vài cổng mà thôi.

Một biến thể khác là nối 2 cổng loại trừ và đầu vào với nhau. Thay vì phân tách các quyết định thông tin loại trừ và thông tin mới thêm vào, ta sẽ quyết định chúng cùng với nhau luôn. Ta chỉ bỏ đi thông tin khi mà ta thay thế nó bằng thông tin mới đưa vào. Ta chỉ đưa thông tin mới vào khi ta bỏ thông tin cũ nào đó đi.

A diagram of a rectangular object with arrows

Description automatically generated

Một biến thể khá thú vị khác của LSTM là Gated Recurrent Unit, hay GRU được giới thiệu bởi Cho, et al. (2014). Nó kết hợp các cổng loại trừ và đầu vào thành một cổng “cổng cập nhập” (update gate). Nó cũng hợp trạng thái tế bào và trạng thái ẩn với nhau tạo ra một thay đổi khác. Kết quả là mô hình của ta sẽ đơn giản hơn mô hình LSTM chuẩn và ngày càng trở nên phổ biến.

A close-up of a sign

Description automatically generated

Trên đây chỉ là một vài biến thế được chú ý nhiều nhất thôi, thực tế có rất nhiều các biến thể khác nhau của LSTM như Depth Gated RNNs của Yao, et al. (2015). Cũng có những biến thể mà chiến lực xử lý phụ thuộc xa hoàn toàn khác như Clockwork RNNs của Koutnik, et al. (2014).

Nếu bạn muốn tìm hiểu xem biến thể nào là tốt nhất và chúng khác nhau thế nào, thì có thể đọc bài so sánh khá hay này của Greff, et al. (2015). Ngoài ra thì Jozefowicz, et al. (2015) thậm chí còn thử hàng chục nghìn kiến trúc RNN khác nhau và tìm ra một vài mô hình hoạt động tốt hơn cả LSTM ở một số bài toán.

5. Kết luận

Như từ đầu tôi đã đề cập tới các kết quả khả quan mà người ta thu được với RNN. Đằng sau các thành quả đó là việc sử dụng LSTM. Chúng hoạt động thực sự tốt hơn nhiều cho hầu hết các bài toán!

Viết ra một tập các công thức, khiến cho LSTM trở nên khá khó hiểu. Nhưng hi vọng là thông qua các bước phân tích trong bày này có thể giúp bạn hình dung được phần nào chiến lược của LSTM thế nào.

LSTM là một bước lớn trong việc sử dụng RNN. Ý tưởng của nó giúp cho tất cả các bước của RNN có thể truy vấn được thông tin từ một tập thông tin lớn hơn. Ví dụ, nếu bạn sử dụng RNN để tạo mô tả cho một bức ảnh, nó có thể lấy một phần ảnh để dự đoán mô tả từ tất cả các từ đầu vào. Bằng chứng là Xu, et al. (2015) đã thực hiện được chính xác việc này. Hiện nay cũng đã có nhiều kết qua thực sự rất thú vị được chú ý và dường như có nhiều kết quả hơn chúng ta vẫn biết.

Sự chú ý không chỉ gói gọn trong nhóm nghiên cứ RNN. Ví dụ Grid LSTMs của Kalchbrenner, et al. (2015) có vẻ như cũng rất tiềm năng. Cũng có người sử dụng RNN trong các mô hình sinh như Gregor, et al. (2015), Chung, et al. (2015), hay Bayer & Osendorfer (2015) cũng rất thú vị. Mấy năm gần đây là quãng thời gian rất sôi nổi của mạng nơ-ron hồi quy, và chúng còn được kì vọng nhiều hơn nữa trong tương lai.