**Câu hỏi**

**Tại sao phải scaler dữ liệu sang dạng (0,1) ?**

Khi xử lý dữ liệu thô, chúng ta sẽ gặp rất nhiều các dạng dữ liệu lạ nên đôi khi rất khó để phân tích và tìm ra insights.

Một cách tiếp cận đơn giản là biến đổi dữ liệu, mục tiêu là đưa dữ liệu về một phân phối chuẩn (normal (Gaussian) distribution),

ví dụ như bell curve. Dữ liệu có dạng phân phối chuẩn giúp mô hình học được tốt hơn và đưa ra dự đoán chính xác và sát với kết quả ta mong muốn hơn.

**Tại sao phải đưa vào units = 50, tại sao phải dropout(0.3), tại sao không bỏ thì có nguy cơ bị overfitting?**

**Lớp dense() ?**

**Tại sao phải inverse từ (0,1) sang bth?**

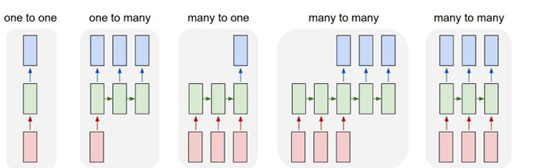
**Evaluate(), Predict() dùng để làm gì?**

RNN và LSTM là gì?

RNN: Recurrent neural network

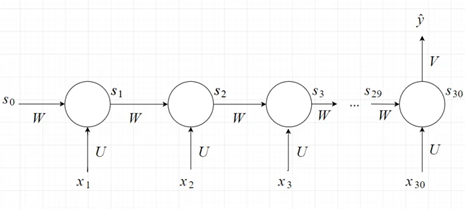
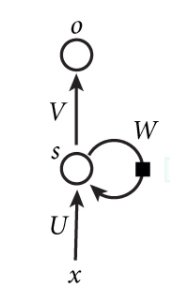
I/ Giới thiệu:

* 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.
* Với đầu vào là các chuỗi thông tin(time series data) một chuỗi các điểm dữ liệu, thường bao gồm các phép đo liên tiếp được thực hiện từ cùng một nguồn trong một khoảng thời gian. (video, đoạn văn, giá cả thị trường,…)
  + time series data là dữ liệu dạng chuỗi có thứ tự(tính liên tục).
* Đầu ra ra là vecto Y^ chứa các thông tin từ các state trước
* **Phân loại bài toán RNN:**
  + One to one: Mạng nơ-rơn thông thường.
  + One to many: Sinh nhạc, đoạn văn.
  + Many to one: phân loại cảm xúc.
  + Many to many: named-entity recognition(NER) dịch máy



II/ Mô hình RNN:

1. Mô hình RNN:



Bài toán: 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: Mô hình trên có 30 input tương ứng với x1,x2,.., x3 theo đúng thứ tự trong dữ liệu
* Output: Một vecto Y^ tại vecto cuối cùng

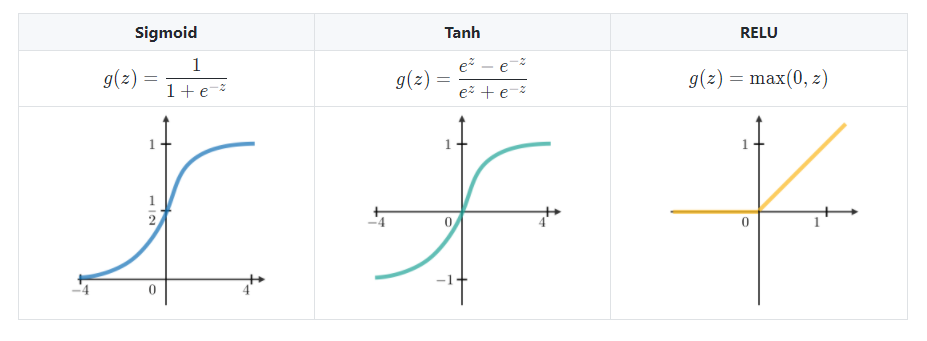
State:

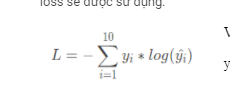
* Tại mỗi hình tròn được gọi là 1 state, nó giúp lưu trữ thông tin từ state phía trước

o Input: xt và st-1 (output của state trước);

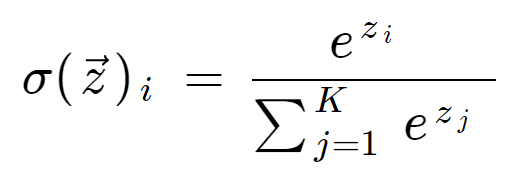
(One-hot encoding: là một quá trình mà các biến phân loại (label) được chuyển đổi thành mã nhị phân nhằm tránh lỗi trong quá trình triển khai thuật toán, trong bài này cái vecto x1,x2 đc xem là 1 vecto one hot)

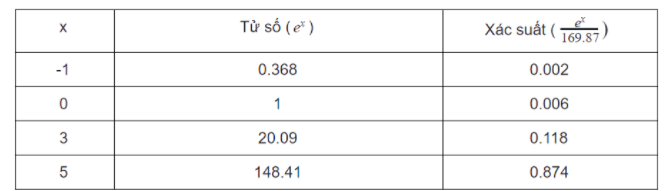
o output là *st*=*f*(*U*∗*xt*+*W*∗*st*−1). f là activation function thường là sigmod, tanh hoặc ReLU.



* Có thể thấy *st*​ mang cả thông tin từ state trước ( *st*−1​) và input của state hiện tại => *st*​ giống như memory nhớ các đặc điểm của các input từ *x*1​ đến *xt*​
* *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.
* W, U, V tối ưu để loss function nhỏ nhất
* 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.  *y*^​=softmax(*V*∗*s*30​).
* 
* 
* L/U = L/y^\*Y^/s30\*s30/U

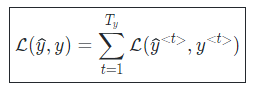
**V là gì?**

**softmax là gì?**

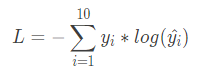
**(softmax(hàm trung bình mũ):** tính toán xác suất xảy ra của một sự kiện. Cụ thể, hàm softmax biến vector k chiều có các giá trị thực bất kỳ thành vector k chiều có giá trị thực có tổng bằng 1. Giá trị nhập có thể dương, âm, bằng 0 hoặc lớn hơn 1, nhưng hàm softmax sẽ luôn biến chúng thành một giá trị nằm trong khoảng (0:1].)

2. Loss function

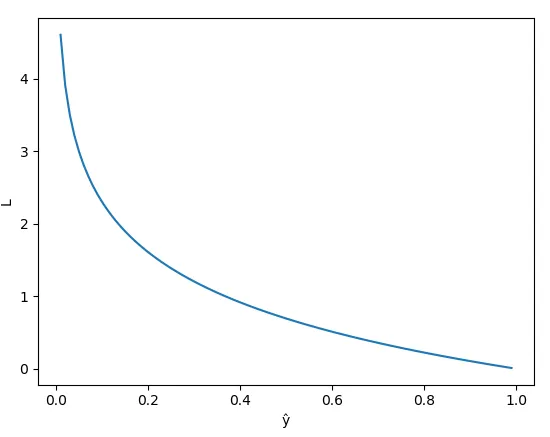
Trong trường hợp của mạng neural hồi quy, hàm mất mát L của tất cả các bước thời gian được định nghĩa dựa theo mất mát ở mọi thời điểm như sau:



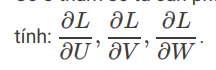
Ở mô hình trên chỉ có 1 output và là bài toán phân loại nên categorical cross entropy loss sẽ được sử dụng.

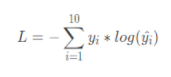
Với yi là giá trị thực tế

y^i là giá trị dự đoán

Đối với trường hợp activation là sigmod thì th y=1 thì L càng về 0.

3. Đạo hàm Backpropagation Through Time (BPTT)

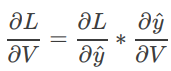
Tính đạo hàm 

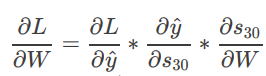


Xét hàm Loss function:



Xét hàm Y^:

Tính đạo hàm của V:



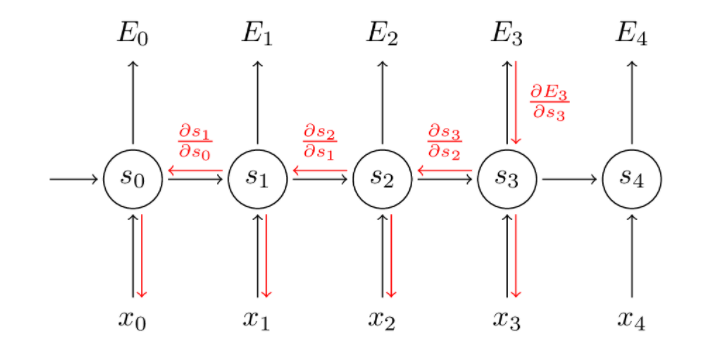
Tính đạo hàm của W:

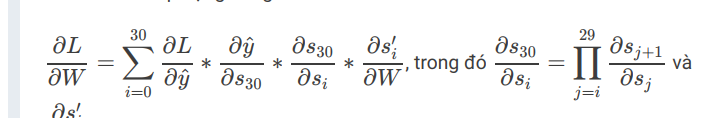


Tuy nhiên đạo hàm riêng của W và U khác với V do:

S\_29 phụ thuộc vào W:

W∂/E30=(∂E3/∂y30^))\*( ∂y3^/∂s3)\*( ∂s3/\*∂sk)\*( ∂sk/ ∂W)



Vậy 

Tương tự vs U ta cũng dược …

Tính

III/ Ứng dụng:

* xử lý ngôn ngữ tự nhiên (NLP - Natural Language Processing)

· Mô hình hóa ngôn ngữ: cho phép ta dự đoán được xác xuất của một từ nào đó xuất hiện sau một chuỗi các từ đi liền trước nó

· tự sinh từ cho phép máy tính có thể tự tạo ra các văn bản mới từ tập mẫu từ đó tạo ra nhiều mẫu văn bản khác nhau.

· Dịch máy (Machine Translation): google dịch, … (Nhưng đầu vào là toàn bộ chuỗi)

· Nhận dạng giọng nói**:** Đưa vào một chuỗi các tín hiệu âm thanh, ta có thể dự đoán được chuỗi các đoạn ngữ âm đi kèm với xác suất của chúng.

· Video tagging: tự động tạo mô tả cho các video chưa được gán nhãn.

IV/ Những hạn chế của RNN dẫn đến mô hình mới LSTM:

1. **Hạn chế:**

RNN có thể xử lý thông tin dạng chuỗi (sequence/ time-series), RNN có thể mang thông tin của frame (ảnh) từ state trước tới các state sau, rồi ở state cuối là sự kết hợp của tất cả các ảnh để dự đoán hành động trong video.

**Vanishing/exploding gradient:** Hiện tượng vanishing và exploding gradient thường gặp trong ngữ cảnh của RNNs. Lí do tại sao chúng thường xảy ra đó là khó để có được sự phụ thuộc dài hạn vì multiplicative gradient có thể tăng/giảm theo hàm mũ tương ứng với số lượng các tầng.

* Các state càng xa ở trước đó thì càng bị vanishing gradient và các hệ số không được update với các frame ở xa. Hay nói cách khác là RNN không học được từ các thông tin ở trước đó xa do vanishing gradient.
* Vậy thực tế là thông tin chỉ mang được qua một số lượng state nhất định, sau đó thì sẽ bị vanishing gradient, hay nói cách khác là model chỉ học được từ các state gần nó => **short term memory**.

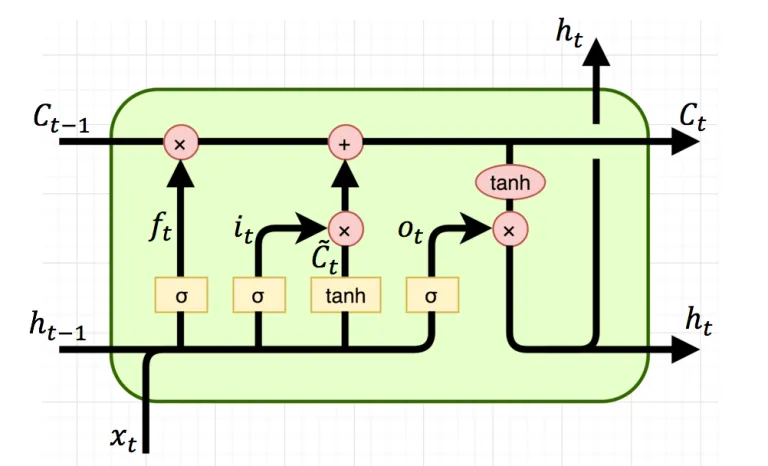
1. Thuật toán LSTM:
2. Giới thiệu:

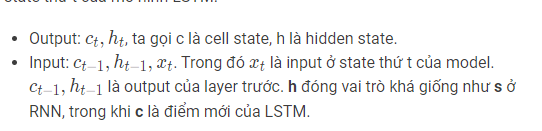
RNN có thể xử lý thông tin dạng chuỗi (sequence/ time-series), RNN có thể mang thông tin của frame (ảnh) từ state trước tới các state sau, rồi ở state cuối là sự kết hợp của tất cả các ảnh để dự đoán hành động trong video.

các state càng xa ở trước đó thì càng bị vanishing gradient và các hệ số không được update với các frame ở xa. Hay nói cách khác là RNN không học được từ các thông tin ở trước đó xa do vanishing gradient.

1. Mô hình:

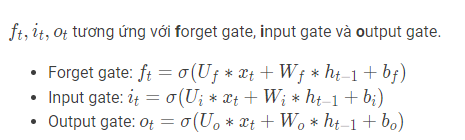
* Kiến trúc của LSTM:





Ta thấy kí hiệu sigmod và tanh là hai hàm activation function. Phép nhân là phép nhân ma trận như sau:





V/ Ứng dụng của LSTM trong bài toán dự đoán giá cổ phiếu:

dự đoán trong một tháng tới lên xuống như nào?

đoán dúng bn sai bn? áp dụng thực tế thu được gì?