**LSTM**

Long short term memory- LSTM

* phát triển lstm bằng python sử dụng thư viện học sâu: keras
  + Keras cung cấp các hàm số cần thiết với cú pháp đơn giản.
  + Keras: có 2 models là Sequential model và Function API
  + dùng để huấn luyện mô hình.
  + Keras khá giống với scikit-learn ở chỗ cùng huấn luyện các mô hình bằng phương thức .fit()
* duy trì trạng thái bộ nhớ trên các chuỗi dài.

LSTM: là một sửa đổi đối với lớp ẩn RNN. LSTM đã cho phép các RNN ghi nhớ các đầu vào của nó trong một khoảng thời gian dài

Diagram, schematic

Description automatically generated

* Nó có thể có bộ nhớ về các đầu vào trước đó trong khoảng thời gian kéo dài.
* 1 khối có:
  + các thành phần làm cho nó ‘thông minh’ hơn
    - Các cổng quản lý trạng thái và đầu ra của khối
      * mỗi cổng sử dụng đơn vị kích hoạt sigmoid-đc kích hoạt hay không( đc xem như điểu kiện làm cho việc thay đổi trạng thái và bổ sung thông tin đi qua khối)
    - Có 3 loại cổng trong 1 khối:
      * Forget Gate: quyết định 1 cách có điều kiện thông tin nào cần loại bỏ khỏi khối
      * Input Gate: quyết định có điều kiện giá trị nào từ đầu vào để cập nhật lại trạng thái bộ nhớ
      * Output Gate: quyết định đầu ra có điều kiện dựa trên đầu vào và bộ nhớ của khối.
  + Bộ nhớ cho các chuỗi gần đây

Ứng dụng

* Dự đoán gì đó tại giờ t với các trường trong quá khứ ( thời gian trước đó )
* Dự đoán giờ tiếp theo dựa vào 24h trước đó.

**Tiền xử lý:**

* Nếu là các giá trị NaN:
  + xóa các hàng nếu nó liên tục nhiều hàng
  + Cho các giá trị này = 0, nếu nó nằm rải rác.
* Nếu tập dữ liệu ban đầu có ngày tháng năm tách thành nhiều cột thì có thể gom lại thành 1 cột và có thể đem làm chỉ mục.
* Chuyển các kiểu chữ → số nguyên.

### scale các data(0,1) ← rất cần thiết cho mô hình mạng nơ ron.

* nếu không scale thì các thuộc tính cá phạm vi lớn sẽ ảnh hưởng lớn trong việc tính toán khoảng cách.
  + Ví dụ như: 1 cột có giá trị khoảng 0→ 1000 và 1 cột có giá trị khoảng 0→1
  + Nếu không scale X thì quá trình huấn luyện sẽ không ổn định
  + hông scalNếu ke y → exploping gradient → không chạy được.

|  |  |  |  |
| --- | --- | --- | --- |
| **standardisation** | **Nomaralisation** | **Min-Max Scaler** | **Robust Scaler** |
| xử lý tốt đc các ngoại lệ | Độ lệch chuẩn sẽ bé hơn | thu hẹp phạm vi về trong khoảng 0→1 | tương tự như Min-Max Scaler |
| Tập trung các giá trị xung quanh giá trị trung bình với độ lệch chuẩn đơn vị. | Scale đc các cột về cùng tỷ lệ nhất, nhưng không xử lý tốt các ngoại lệ. | Nhạy cảm với nhiễu. | Mạnh mẽ đối với các ngoại lệ. |
| Standard Scaler | Min-Max Scaler & Robust Scaler |  |  |
| có thể sẽ hữu ích hơn trong trường hợp dữ liệu tuân theo phân phối Gaussian | sử dụng khi biết phân phối của dữ liệu không tuân theo phân phối Gaussian. |  |  |

* + Phân phối Gausse(chuẩn) có dạng hình chuôn, dl tập trung ở phân trung tâm và giảm dần khi đi xa trung tâm hơn.
    - Dữ liệu gần giá trị trung bình sẽ xuất hiện nhiều hơn so với dữ liệu xa giá trị trung bình.
    - Trong phân phối chuẩn, giá trị trung bình là 0 và độ lệch chuẩn là 1.
* convert dữ liệu thành mô hình học có giám sát: series\_to\_supervised(scaled) đc xây dựng từ shift()
  + nghĩa là thêm các cột thuộc tính ở bước thời gian trong quá khứ như
    - var1(t-1), var2(t-1), var3(t-1), var4(t-1) ⇒ var1(t)
  + lúc này có thể sẽ sinh ra các cột thuộc tính ở hiện tại (t):
    - sẽ xóa đi vì các cột này không cần predict.
* Chia tập dữ liệu ra làm tập train và tập test
* Chuyển X(dl đầu vào) về dạng 3D(không gian 3 chiều [samples, timesteps, features] mà hàm LSTM có thể xử lý được.
  + (1234,1,2): train\_X.shape
  + (1234, ): train\_y.shape
  + (3456, 1, 2): train\_X.shape
  + (3456, ): test\_y.shape
* Thiết kế mạng.
  + Keras có 2 mô hình là Sequential và API function
  + Sequential: RNN là 1 thuật toán đc sử dụng phổ biến trong mô hình này.
    - đc dùng cho mô hình tuần tự ( vd: chuỗi thời gian)
    - model.add(LSTM(50, input\_shape=(1, 8)))
    - Hàm compile: hàm này sd để training models
      * optimizer: dùng để chọn thuật toán training
        + adam, SGD, RMSprop.
  + Model.fit
    - epochs: số lần duyệt qua hết số lượng mẫu trong tập huấn luyện
    - Batch-size: số lượng mẫu mà mini-Batch GD sử dụng cho mỗi lần cập nhật trọng số.

# ví dụ ta có 55000 mẫu

* + - * batch-size=55 → mỗi lần cập nhật trọng số ta dùng 55 mẫu → lúc đó ta mất 55000/55 = 1000 lần lặp để duyệt qua hết tập dữ liệu ( lũ này sẽ hoàn thành 1 epochs)

# Lý do: vì data có thể quá lớn, chúng ta không thể đưa cả tập dl vào train đc → chia nhỏ data ra thành nhiều batch nhỏ.

* + - Verbose = 2: nghĩa là sẽ có thêm giá trị thời gian cho 1 vòng lặp epochs
    - Shuffle: xáo trộn data = True
* Dự đoán cho tập test\_X bằng model đã huấn luyện ở trên.
* Sau đó chuyển về data về 2D
  + test\_X = test\_X.reshape((test\_X.shape[0], test\_X.shape[2]))
* trả về tỷ lệ như lúc đầu: inverse\_transform
  + inv\_y = scaler.inverse\_transform(inv\_y)
  + scaler: nó đã ghi nhớ lại các chỉ số để chuyển đổi nên hàm trên sẽ nghịch đảo về tỷ lệ như ban đầu bằng hàm inverse\_transform