Chương 2:

Học máy và Hệ thống thông minh

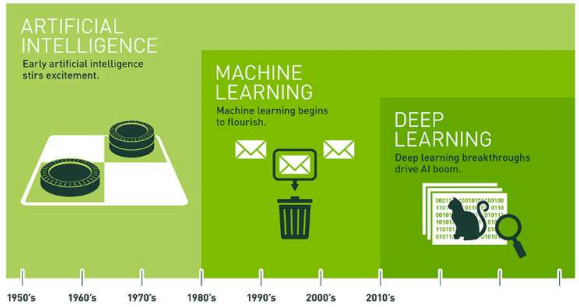
1

Nội dung

- Học máy cổ điển
- Học sâu
- Đánh giá mô hình
- Hệ thống dựa trên học máy
- Notebook và thư viện Scikit-learn

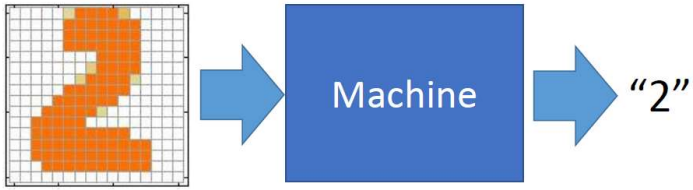
2

Học sâu (deep learning)



3

Học sâu (deep learning)

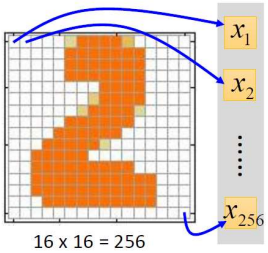


Hung-yi Lee - National Taiwan University

4

Học sâu (deep learning)

Input



Output

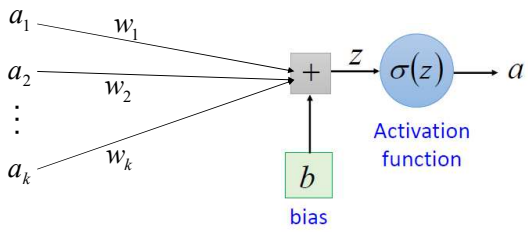
Hình số "2"

- Ô trống: 0
- Ố không trống: 1

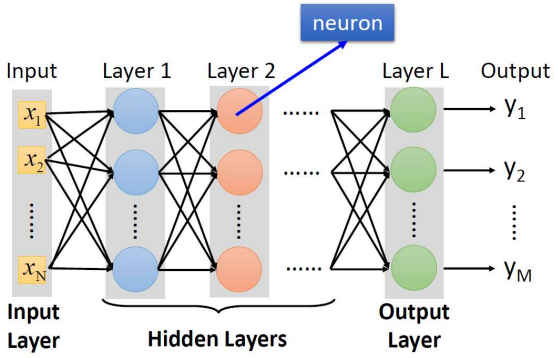
Học sâu (deep learning)

Neuron $f: R^K \rightarrow R$

$$z = a_1 w_1 + a_2 w_2 + \cdots + a_K w_K + b$$

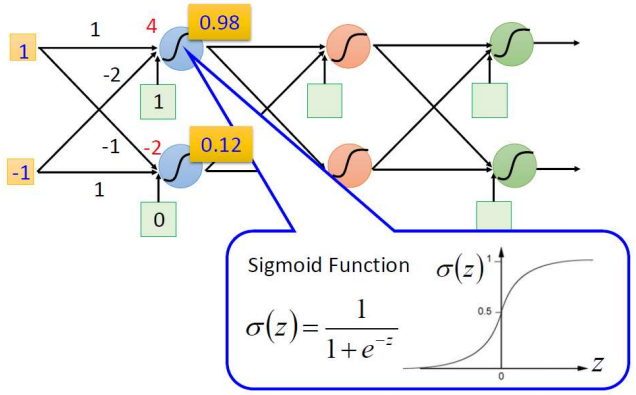


Học sâu (deep learning)



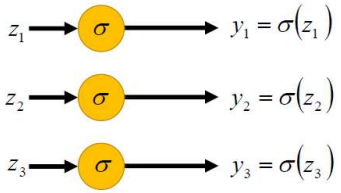
❑ Nhiều lớp ẩn: sâu

Học sâu (deep learning)



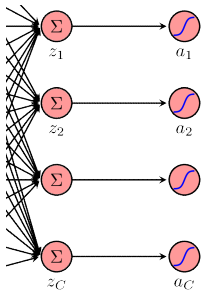
Học sâu (deep learning)

- Hàm softmax
  - Thường dùng ở tầng output để tính xác suất cho mỗi nhãn



Học sâu (deep learning)

- Hàm softmax
  - Thường dùng ở tầng output để tính xác suất cho mỗi nhãn



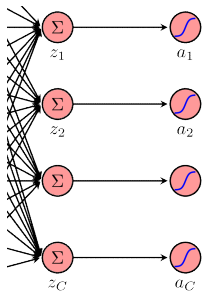
✓ Dùng hàm sigmoid:

$$a_i = \text{sigmoid}(z_i) = \text{sigmoid}(\mathbf{w}_i^T \mathbf{x})$$

✓ Các giá trị a_i không có mối quan hệ nào với nhau.

Học sâu (deep learning)

- Hàm softmax
  - Thường dùng ở tầng output để tính xác suất cho mỗi nhãn



✓ Tạo nên mối quan hệ cho các giá trị a_i để xác định nhãn ở đầu ra:

$$a_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)}, \quad \forall i = 1, 2, \ldots, C$$

$$\Rightarrow \sum_{1}^{i} a_i = 1$$

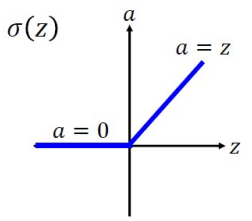
$$\mathbf{a} = \text{softmax}(\mathbf{z}) \in \mathbb{R}^C$$

✓ Tính xác suất cho mỗi a_i

Học sâu (deep learning)

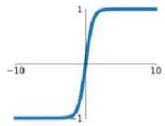
- ReLU (Rectified Linear Unit)
  - Tính toán nhanh

$$f(z) = max(0, z)$$



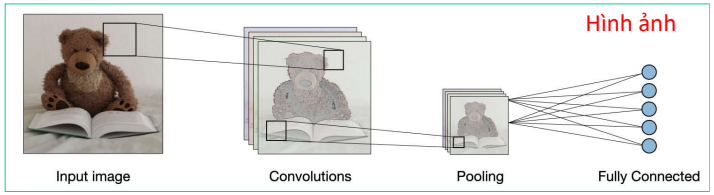
- tanh

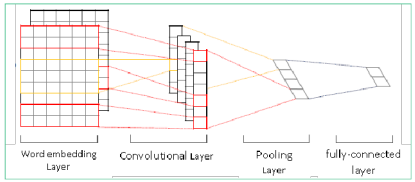
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Học sâu (deep learning)

- Convolutional Neural Network (CNN)





Chuỗi ký tự

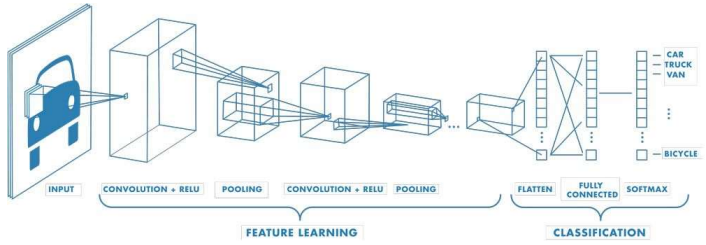
https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

Học sâu (deep learning)

- Convolutional Neural Network (CNN)
  - Lớp input (Word embedding, input image) gồm các ma trận kích thước *n* x *k*, biểu diễn dữ liệu thành *n* vector, mỗi vector có *k* chiều.
  - Lớp convolutional sử dụng phép tích chập để xử lý dữ liệu bằng cách trượt cửa sổ trượt (slide windows) có kích thước cố định (gọi là kernel) trên ma trận dữ liệu đầu vào để thu được kết quả đã được tinh chỉnh.
  - Lớp pooling tổng hợp các vector kết quả của lớp convolutional và giữ lại những vector quan trọng nhất.
  - Lớp Fully-connected sử dụng những vector còn lại ở các lớp trên tạo ra kết quả cuối cùng thông qua quá trình huấn luyện.

Học sâu (deep learning)

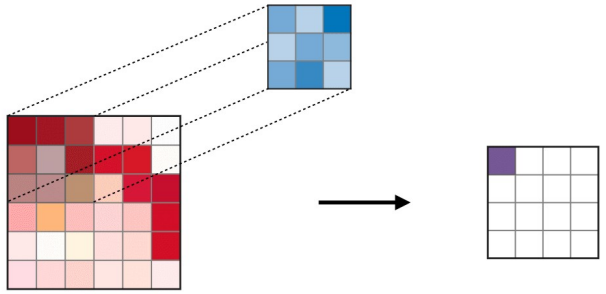
- Convolutional Neural Network (CNN)



https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html

Học sâu (deep learning)

- Convolutional Neural Network (CNN)
  - Convolutional:



https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

Học sâu (deep learning)

- Convolutional Neural Network (CNN)
  - Ví dụ:

```
sequence_length = X_train.shape[1]
inputs = Input(shape=(sequence_length,))
embedding = embedding_layer(inputs)
reshape = Reshape((sequence_length,EMBEDDING_DIM,1))(embedding)

conv_0 = Conv2D(num_filters, (filter_sizes[0], EMBEDDING_DIM),activation='sigmoid',
                kernel_regularizer=regularizers.l2(L2))(reshape)
conv_1 = Conv2D(num_filters, (filter_sizes[1], EMBEDDING_DIM),activation='sigmoid',
                kernel_regularizer=regularizers.l2(L2))(reshape)
conv_2 = Conv2D(num_filters, (filter_sizes[2], EMBEDDING_DIM),activation='sigmoid',
                kernel_regularizer=regularizers.l2(L2))(reshape)

maxpool_0 = MaxPooling2D((sequence_length - filter_sizes[0] + 1, 1), strides=(1,1))(conv_0)
maxpool_1 = MaxPooling2D((sequence_length - filter_sizes[1] + 1, 1), strides=(1,1))(conv_1)
maxpool_2 = MaxPooling2D((sequence_length - filter_sizes[2] + 1, 1), strides=(1,1))(conv_2)

merged_tensor = concatenate([maxpool_0, maxpool_1, maxpool_2], axis=1)
flatten = Flatten()(merged_tensor)
reshape = Reshape((3*num_filters,))(flatten)
dropout = Dropout(drop)(flatten)
output = Dense(units=2, activation='softmax',kernel_regularizer=regularizers.l2(L2))(dropout)
model = Model(inputs, output)
adam = Adam(lr=1e-3)
model.compile(loss='mse', optimizer=adam, metrics=['acc'])
```

17

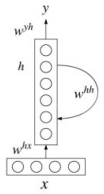
Học sâu (deep learning)

- Recurrent Neural Networks (RNN)
  - Là một lớp các mạng neural có các kết nối giữa các neural tạo thành dạng có hướng có tính chu kỳ.
  - RNN sử dụng bộ nhớ nội bộ của mình để xử lý một chuỗi các đầu vào.
  - Bộ nhớ nội bộ của RNN thực hiện cùng một nhiệm vụ cho mọi phần tử của chuỗi.
  - Các thông tin đầu ra phụ thuộc vào các tính toán của các phần tử trước nó => có sự kế thừa thông tin trong quá trình xử lý.

18

Học sâu (deep learning)

- Recurrent Neural Networks (RNN)

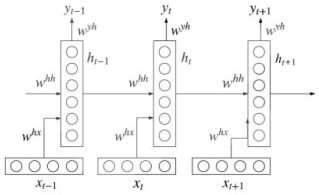


Mạng có tính hồi quy

$$h_t = f(w^{hh}h_{t-1} + w^{hx}x_t) \quad (1)$$

$$y_t = softmax(w^{yh}h_t) \quad (2)$$

Mạng có tính tuần tự



19

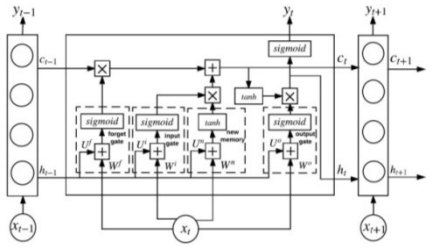
Học sâu (deep learning)

- Long-short term memory (LSTM)
  - Là một loại RNN đặc biệt, có khả năng học các phụ thuộc dài.
  - Thay vì có một tầng neural thì có bốn lớp tương tác theo một cách đặc biệt.
  - Có hai trạng thái: trạng thái ẩn và trạng thái tế bào (cell state)

20

Học sâu (deep learning)

- Long-short term memory (LSTM)



$$f_t = \sigma(W^f x_t + U^f h_{t-1}) \quad (3)$$

$$i_t = \sigma(W^i x_t + U^i h_{t-1}) \quad (4)$$

$$\tilde{C}_t = \tanh(W^n x_t + U^n h_{t-1}) \quad (5)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (6)$$

$$o_t = \sigma(W^0 x_t + U^0 h_{t-1}) \quad (7)$$

$$h_t = o_t * \tanh(C_t) \quad (8)$$

Học sâu (deep learning)

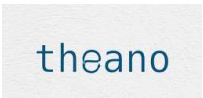
- Long-short term memory (LSTM)
  - Tại thời điểm bước *t*, LSTM trước tiên quyết định thông tin nào sẽ được đổ vào trạng thái tế bào bởi hàm sigmoid hoặc tầng σ, gọi là cổng quên (forget gate).
  - Hàm lấy h_{t-1} (đầu ra từ lớp ẩn trước đó) và x_t (đầu vào hiện tại) và xuất ra một số trong [0, 1], công thức (3):
    - ✓ 1: *giữ hoàn toàn*
    - ✓ 0: *bỏ qua hoàn toàn*.
  - Sau đó LSTM quyết định những thông tin mới sẽ lưu trữ trong trạng thái tế bào.

Học sâu (deep learning)

- Long-short term memory (LSTM)
  - Hàm sigmoid ở (4) quyết định giá trị nào LSTM sẽ cập nhật.
  - Hàm tanh ở (5) tạo ra vector các giá trị ứng viên mới $\tilde{C}$.
  - Cập nhật trạng thái tế bào cũ C_{t-1} và trạng thái tế bào mới C_t theo (6).
  - LSTM quyết định phần nào của trạng thài tế bào sẽ xuất ra dựa theo hàm sigmoid ở (7).
  - Trạng thái tế bào qua hàm tanh và nhân với đầu ra của cổng sigmoid để tạo giá trị ngõ ra theo (8)

Học sâu (deep learning)

- Một số thư viện và công cụ phát triển mô hình học sâu:



Học sâu (deep learning)

- Long-short term memory (LSTM)
  - Ví dụ:

```
model = Sequential()
model.add(embedding_layer)
model.add(LSTM(units=300))
model.add(Dropout(0.3))
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(3, activation='sigmoid'))
model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
```

25

Đánh giá mô hình

- Việc đánh giá hiệu năng hệ thống học máy thường được thực hiện dựa trên thực nghiệm (experimentally), hơn là dựa trên phân tích (analytically)
- Các đánh giá phân tích (analytical evaluation) nhằm chứng minh một hệ thống là đúng đắn (correct) và hoàn chỉnh.
- Không thể xây dựng một đặc tả (định nghĩa) hình thức của vấn đề mà một hệ thống học máy giải quyết

26

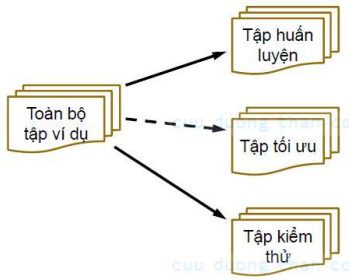
Đánh giá mô hình

- Phương pháp đánh giá:
  - Làm sao có được một đánh giá đáng tin cậy về hiệu năng của hệ thống?
- Tiêu chí đánh giá:
  - Làm sao để đo được hiệu năng của hệ thống?
  - Đo hiệu năng một cách tự động.
  - Không cần sự can thiệp của người dùng

27

Đánh giá mô hình

- Dữ liệu dùng cho đánh giá:



28

Đánh giá mô hình

- Một số phương pháp đánh giá:
  - Hold-out
  - Stratified sampling
  - Repeated hold-out
  - Cross-validation
    - ✓ *k*-fold
    - ✓ Leave-one-out
  - Bootstrap sampling

29

Đánh giá mô hình

- Hold-out (Splitting):
  - Toàn bộ tập mẫu ***D*** được chia thành 2 tập con **không giao nhau:**
    - Tập huấn luyện *train* : để huấn luyện hệ thống
    - Tập kiểm thử *test*: để đánh giá hiệu năng của hệ thống đã học.
    - ***D*** = *train* ∪ *test*
    - |*train*| >> |*test*|
  - Bất kỳ mẫu nào được sử dụng trong giai đoạn huấn luyện hệ thống (thuộc vào *train*) đều không được sử dụng trong giai đoạn đánh giá hệ thống.

30

Đánh giá mô hình

- Stratified sampling (lấy mẫu phân tầng):
  - Là phương pháp dùng cho mục đích cân bằng về phân bố lớp, đảm bảo tỷ lệ phân bố lớp trong tập *train* và *test* là xấp xỉ nhau.
  - Không áp dụng được cho bài toán học máy dự đoán/hồi quy (vì giá trị đầu ra của hệ thống là một giá trị số, không phải là một nhãn lớp).
  - Ví dụ: nếu có tổng số n mẫu, *m* trong số đó là *yes* và *f* *no* (*m* + *f* = *n*), thì kích thước tương đối của hai mẫu là:
    - x_1 = *m* / *n* (yes),
    - x_2 = *f* / *n* (no).

31

Đánh giá mô hình

- Repeated hold-out (hold-out nhiều lần):
  - Áp dụng phương pháp Hold-out nhiều lần để sinh ra và sử dụng các tập huấn luyện và thử nghiệm khác nhau.
    - Trong mỗi bước lặp, một tỷ lệ nhất định của tập ***D*** được lựa chọn ngẫu nhiên để tạo tập huấn luyện.
    - Các giá trị lỗi ghi nhận được trong các bước lặp này được lấy trung bình cộng (averaged) để xác định giá trị lỗi tổng thể
  - Hạn chế:
    - Mỗi bước lặp sử dụng một tập *test* khác nhau.
    - Có một số mẫu trùng lặp trong các lần *test*.

32

Đánh giá mô hình

- Cross-validation:
  - Tránh được sự trùng lặp giữa các tập kiểm thử (một số ví dụ cùng xuất hiện trong các tập kiểm thử khác nhau).
  - k-fold cross-validation
    - Tập mẫu ***D*** được chia thành *k* tập con không giao nhau (*fold*), có kích thước xấp xỉ nhau.
    - Mỗi lần (trong số *k* lần) lặp, một tập con được sử dụng làm tập kiểm thử, và (*k-1*) tập con còn lại được dùng làm tập huấn luyện.
    - *k* giá trị lỗi tương ứng với mỗi *fold* được tính trung bình cộng để thu được giá trị lỗi tổng thể.

33

Đánh giá mô hình

- Cross-validation:
  - Tránh được sự trùng lặp giữa các tập kiểm thử (một số ví dụ cùng xuất hiện trong các tập kiểm thử khác nhau).
  - Các giá trị thường được chọn cho *k*: 10, 5.
  - Phù hợp với trường hợp tập mẫu vừa và nhỏ.

34

Đánh giá mô hình

- Leave-one-out cross-validation:
  - Có thể xem đây là trường hợp riêng của Cross-validation:
    - Số lượng nhóm (các folds) bằng kích thước của tập dữ liệu ($k = |D|$).
    - Mỗi nhóm (*fold*) chỉ bao gồm một mẫu
  - Khai thác tối đa tập mẫu ban đầu
  - Chi phí tính toán rất cao
  - Phù hợp với trường hợp tập mẫu ***D*** rất nhỏ

35

Đánh giá mô hình

- Bootstrap sampling:
  - Phương pháp Bootstrap sampling sử dụng việc lấy mẫu có lặp lại (sampling with replacement) để tạo nên tập huấn luyện:
    - Lấy ra ngẫu nhiên một mẫu $x \in D$ (nhưng không loại bỏ x khỏi tập ***D***)
    - Cho mẫu x vào tập huấn luyện: $train = train \cup x$
    - Lặp lại 2 bước trên n lần (n là số mẫu của ***D***).
  - Sử dụng tập *train* để huấn luyện hệ thống.
  - Sử dụng tất cả các mẫu thuộc ***D*** nhưng không thuộc *train* để tạo nên tập thử nghiệm: $test = \{z \in D; z \notin train\}$

36

Đánh giá mô hình

- Bootstrap sampling:
  - Trong mỗi bước lặp, một mẫu có xác xuất $\left(1-\frac{1}{n}\right)$ không được lựa chọn vào tập *train*.
  - Suy ra, xác suất để một mẫu được đưa vào tập test là:

$$\left(1-\frac{1}{n}\right)^n \approx e^{-1} \approx 0.368$$

  - Như vậy, tập test bao gồm xấp xỉ 36.8% các mẫu trong tập ***D***.
  - Phương pháp này phù hợp với tập dữ liệu ***D*** có kích thước rất nhỏ.

37

Đánh giá mô hình

- Tập tối ưu (validation set):
  - Chia tập mẫu ***D*** thành 3 tập con không giao nhau: tập huấn luyện, tập tối ưu, và tập kiểm thử.
  - Tập kiểm thử không được tham gia vào quá trình huấn luyện.
  - Tập tối ưu (validation set) được sử dụng để tối ưu giá trị các tham số trong giải thuật học máy được sử dụng.
    - Đối với một tham số, giá trị tối ưu là giá trị giúp sinh ra hiệu năng cực đại đối với tập tối ưu.

38