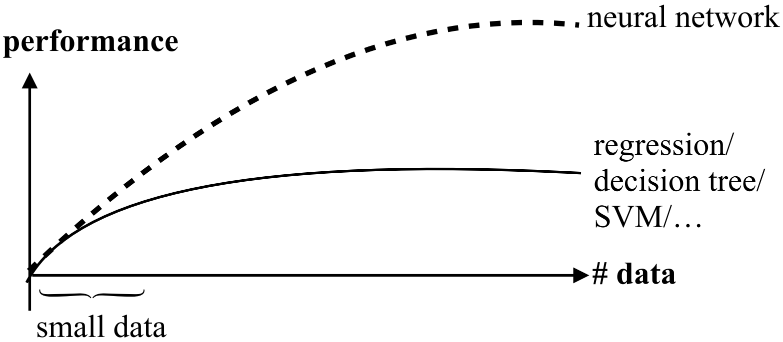
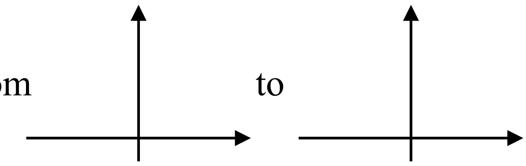
SC201 Lecture 9

Deep Learning

Why is it taking off ?



- Digitalization era makes data booming
- ________ enables complex algorithm
- Better math function

(from  to)

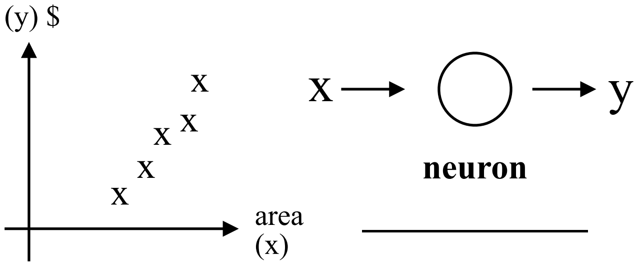
sigmoid ________

(______________________)

- ______________________

replace for loops

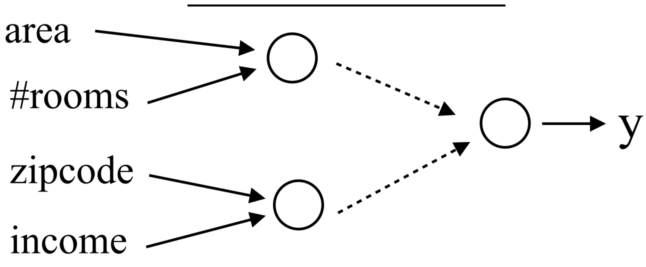
Neuron



____

Neural Network

____



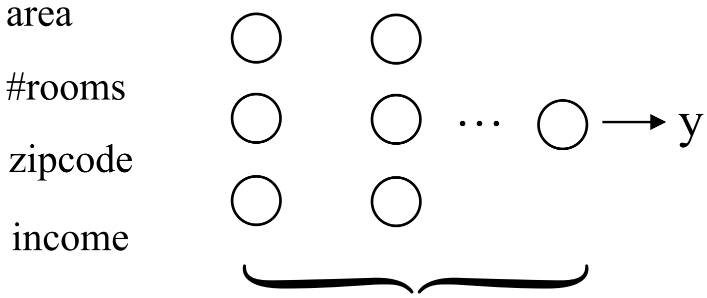
____

① ________________ feature extractor

② 2低維非線性組合出超高維model

+ =

____



The ________ it goes, the more ______________________ it can extract.

→ | |
|---|

Matrix for Machine Learning from Python dict → ________________

① < ______________________________ >

```
for epoch in range(num_epochs):
    for x, y in trainigExamples:
        for i in range(num_features):

            ______________________________
```

② $\varnothing(X_1) = \begin{bmatrix} 3 \\ 27 \\ 29.4 \\ \vdots \\ 0 \end{bmatrix} \begin{array}{l} \rightarrow \text{Pclass} \\ \rightarrow \text{Age} \\ \rightarrow \text{Fare} \\ \\ \rightarrow \text{Embarked} \end{array}$

$W = \begin{bmatrix} W\text{Pclass} \\ W\text{Age} \\ W\text{Fare} \\ \vdots \\ W\text{Embarked} \end{bmatrix}$

③ $\varnothing(X) = \begin{bmatrix} \vdots & \vdots & & \vdots \\ X_1 & X_2 & \cdots & X_m \\ \vdots & \vdots & & \vdots \end{bmatrix} = \begin{bmatrix} 3 & 1 & & 2 \\ 27 & 18 & & 60 \\ 29.4 & 100 & \cdots & 129.9 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & 1 \end{bmatrix}$

$W = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_{nf} \end{bmatrix}$

$Y = [y_1, y_2 \ldots y_m] = [0\ 1 \ldots 1]$

__________ (especially __________) is good at matrix operations!

Basic Matrix Operation in Python

W = [0.1, 0.2, 0.3, 0.4]   # 1D ____________ / __________________________

2D ____________ / _____________

$X_1 = \begin{bmatrix} 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 \end{bmatrix}$ 4x4

$X_2 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0.1 & 0.2 & 0.3 & 0.4 \\ 0.01 & 0.02 & 0.03 & 0.04 \end{bmatrix}$ 3x4

< dot Product >

$X_1 \bullet X_2 \rightarrow$ ________________

$X_2 \bullet X_1 \rightarrow$ ________________

((3x4) x (4x3))

$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0.1 & 0.2 & 0.3 & 0.4 \\ 0.01 & 0.02 & 0.03 & 0.04 \end{bmatrix} \begin{bmatrix} 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} \quad & \quad & \quad \\ \\ \\ \end{bmatrix}$

$$X_1 = \begin{bmatrix} 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 \end{bmatrix} \text{ 4x4} \qquad X_2 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0.1 & 0.2 & 0.3 & 0.4 \\ 0.01 & 0.02 & 0.03 & 0.04 \end{bmatrix} \text{ 3x4}$$

< Transpose>

X_2.shape == (3, 4)

X_2.__.shape == (__, __)

$$\begin{bmatrix} \quad & \quad & \quad \\ \quad & \quad & \quad \\ \quad & \quad & \quad \end{bmatrix}$$

< rank 1 array>

- 1D vector ____________________ transpose operation
- Use ____________________ to change it to 2D array ________________

```
np.random.rand(d1, d2)
reshape(__________)
np.zeros(__________)
```

< Element-wise> (*, +, −, /, //)

$$X_3 = \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}$$

$X_1 * X_3 \rightarrow$ ________________

$X_2 * X_3 \rightarrow$ ________________

((3x4) x (3x4))

$$\begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0.1 & 0.2 & 0.3 & 0.4 \\ 0.01 & 0.02 & 0.03 & 0.04 \end{bmatrix} = \begin{bmatrix} \quad & \quad & \quad & \quad \\ \quad & \quad & \quad & \quad \\ \quad & \quad & \quad & \quad \end{bmatrix}$$

< Vectorization>

Take advantage of __________________ on CPU/GPU,
we can speed up Python code by matrix operations.

$$Z = \begin{bmatrix} Z_0 \\ Z_1 \\ \vdots \\ Z_{nf} \end{bmatrix} \Rightarrow \begin{bmatrix} e^{Z_0} \\ e^{Z_1} \\ \vdots \\ e^{Z_{nf}} \end{bmatrix}$$

__________ = __________________

__________ = __________________

__________ = __________________

__________ = __________________

__________ = __________________

__________ = __________________

__________ = __________________

__________ = __________________

< broadcasting> for ________________ array (or ___________)

(not ________________)

$$\begin{bmatrix} \quad\quad\quad\quad \end{bmatrix} + \begin{bmatrix} \quad \end{bmatrix} = \begin{bmatrix} \quad\quad\quad\quad \end{bmatrix}$$

$$\begin{bmatrix} \quad\quad\quad\quad \end{bmatrix} / \begin{bmatrix} \quad\quad\quad \end{bmatrix} = \begin{bmatrix} \quad\quad\quad\quad \end{bmatrix}$$

Vecterizing Logistic Regression

- **Fowardprop** (______________________)

$$X = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x_1 & x_2 & \cdots & x_m \\ \vdots & \vdots & & \vdots \end{bmatrix} \quad W = \begin{bmatrix} \mathbf{W_1} \\ \mathbf{W_2} \\ \vdots \\ \mathbf{W_{nf}} \end{bmatrix} \quad Y = [0\ 1\ \dots\ 1]$$

K = ____________

H = 1/ (1+np.exp(-k))

L = ________________________________

J = _______________

Backprop

$$W = W - \alpha \frac{dJ}{dW}$$

$$\frac{dJ}{dW} =$$

$$\frac{dJ}{dH} =$$

$$\frac{dH}{dK} = \qquad\qquad \frac{dK}{dW} =$$