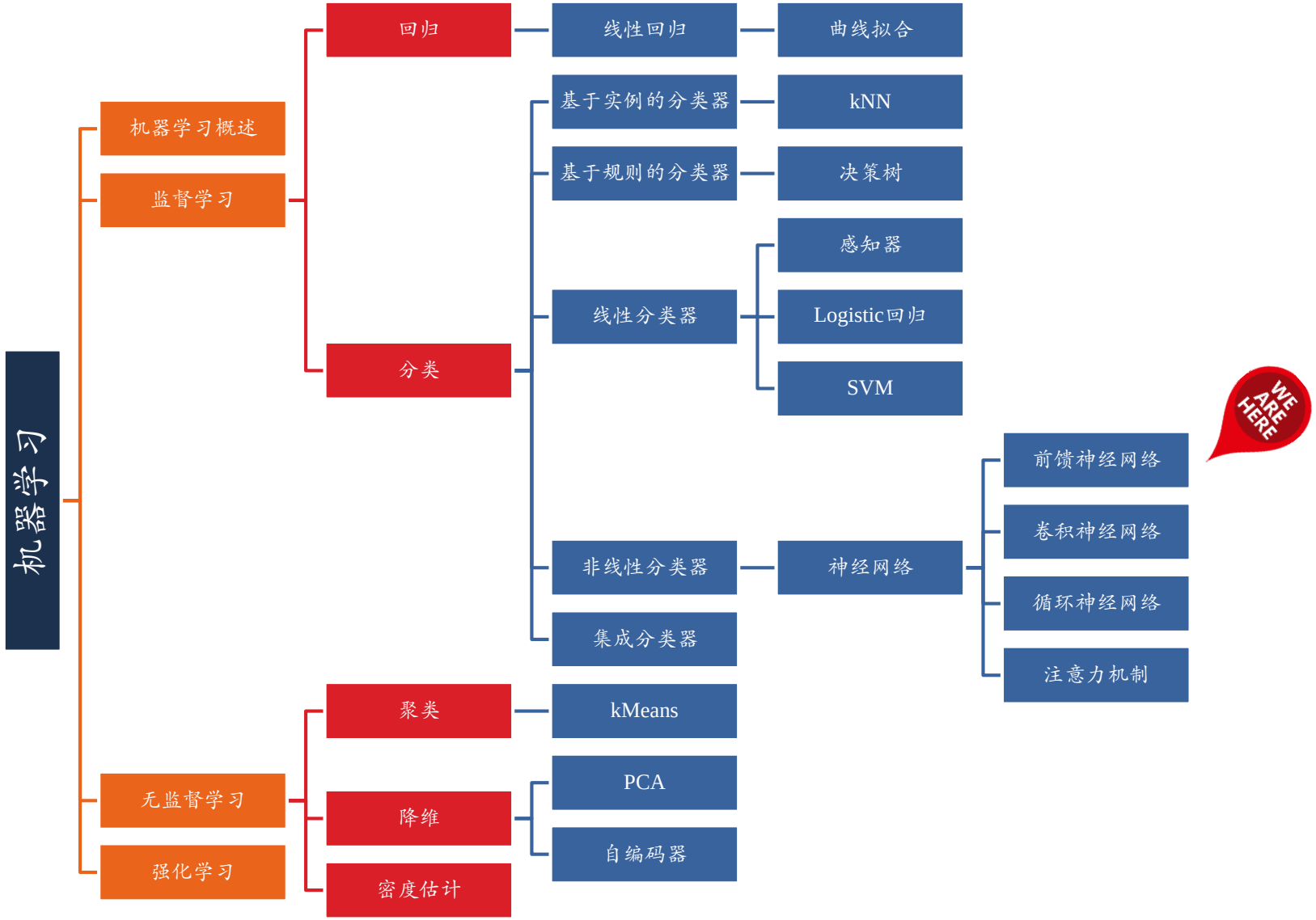
“AI+X微专业”基础类课程

华五教学协同中心
Consortium for the Advancement of Teaching and Learning at E5

# 《模式识别与机器学习》第五讲：前馈神经网络

邱锡鹏
复旦大学 计算机学院
xpqiu@fudan.edu.cn

# 路线图


机器学习
机器学习概述
监督学习
回归
线性回归
曲线拟合
分类
基于实例的分类器
kNN
基于规则的分类器
决策树
线性分类器
感知器
Logistic回归
SVM
非线性分类器
神经网络
前馈神经网络
卷积神经网络
循环神经网络
注意力机制
集成分类器
WE ARE HERE
无监督学习
聚类
kMeans
降维
PCA
自编码器
密度估计
强化学习

# 内容

- 神经网络
  - 神经元
  - 网络结构
- 前馈神经网络
  - 网络结构
- 神经网络的参数学习
  - 计算图与自动微分
- 神经网络与深度学习的关系

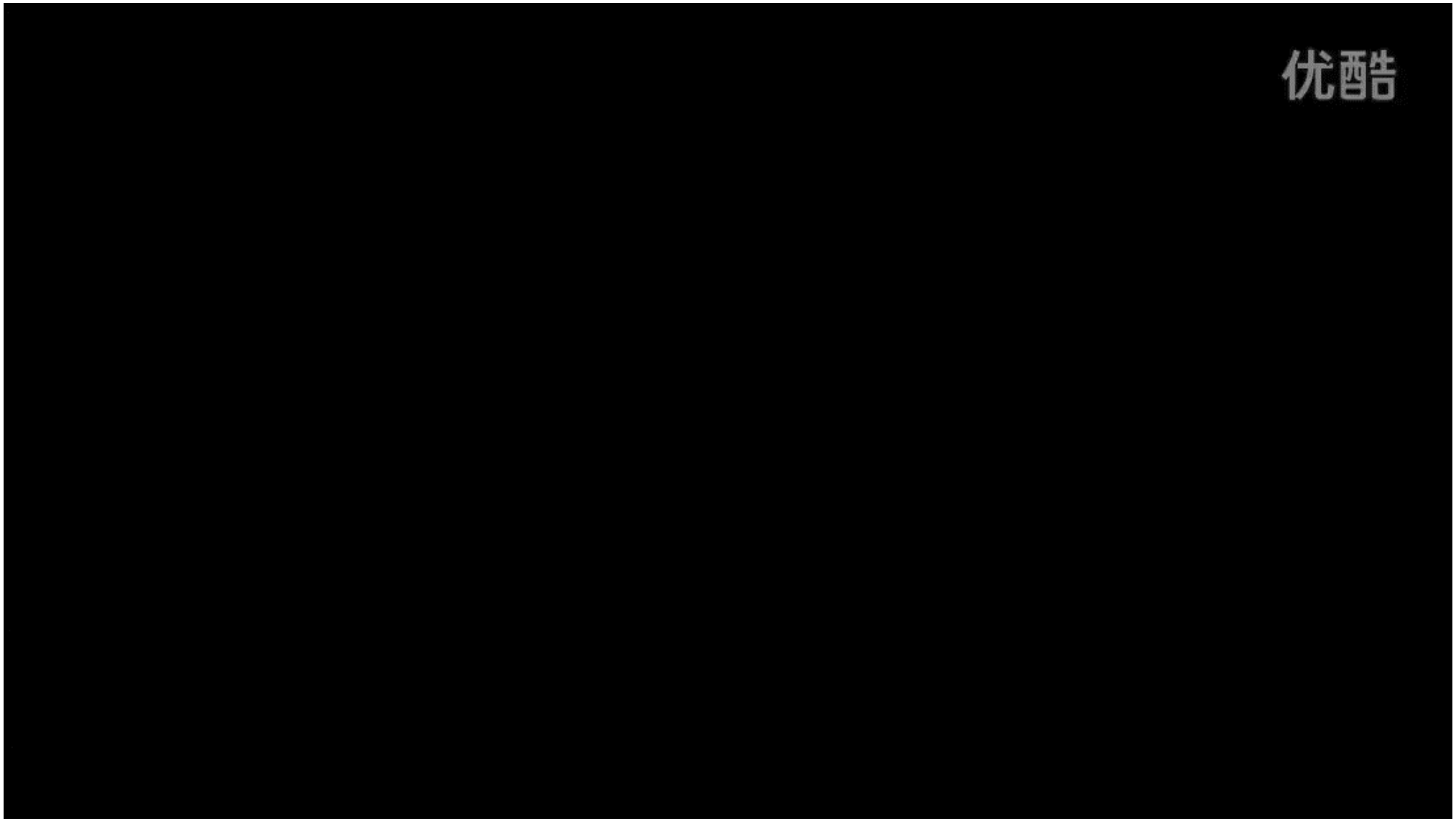
# 本讲内容主要来自

- “Pattern Recognition and Machine Learning” by Bishop
  - Ch. 5.1, 5.2, 5.3

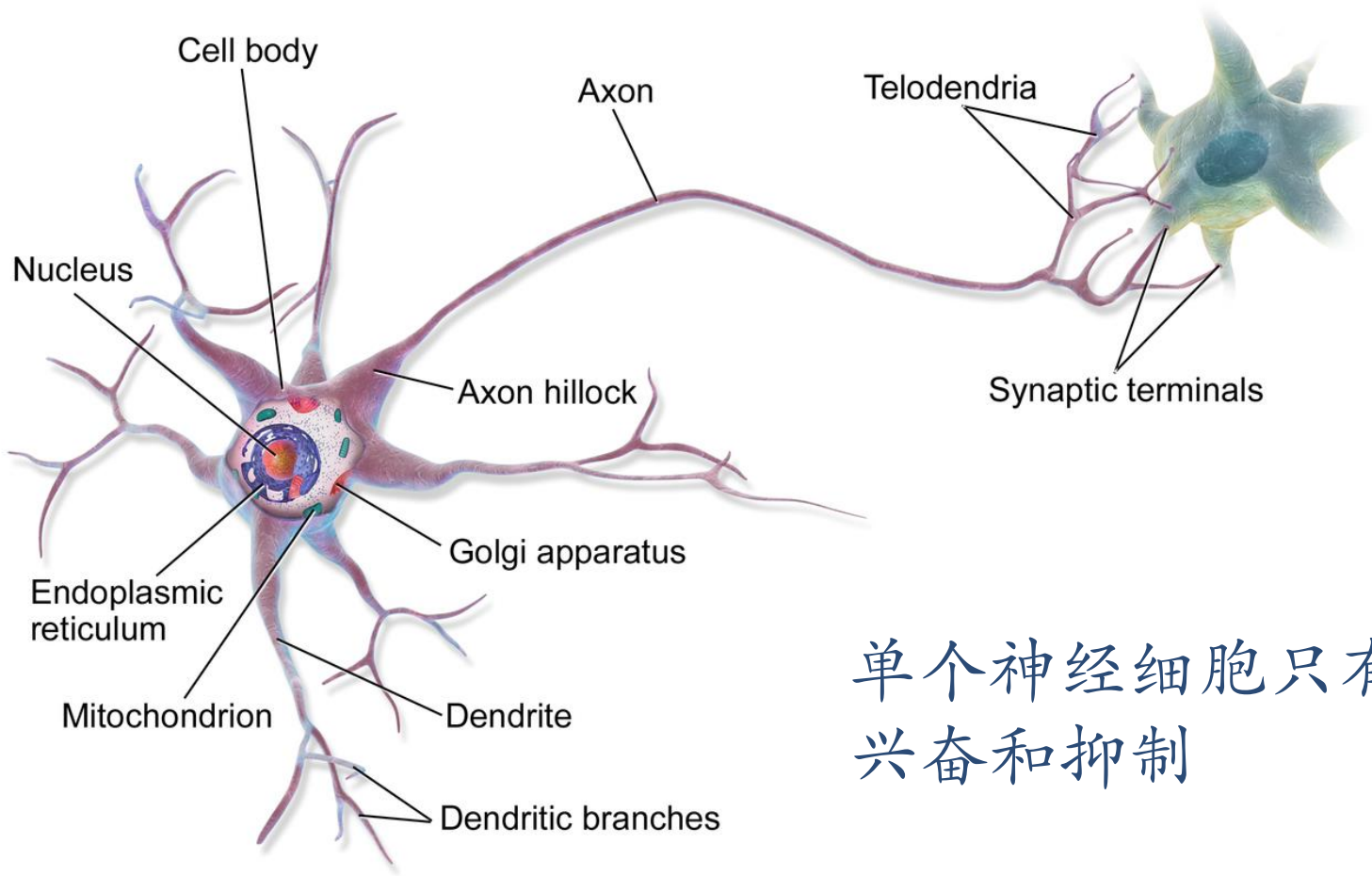
- 《神经网络与深度学习》邱锡鹏
  - 第1章 绪论
  - 第4章 前馈神经网络

# （人工）神经网络

# 生物神经网络介绍

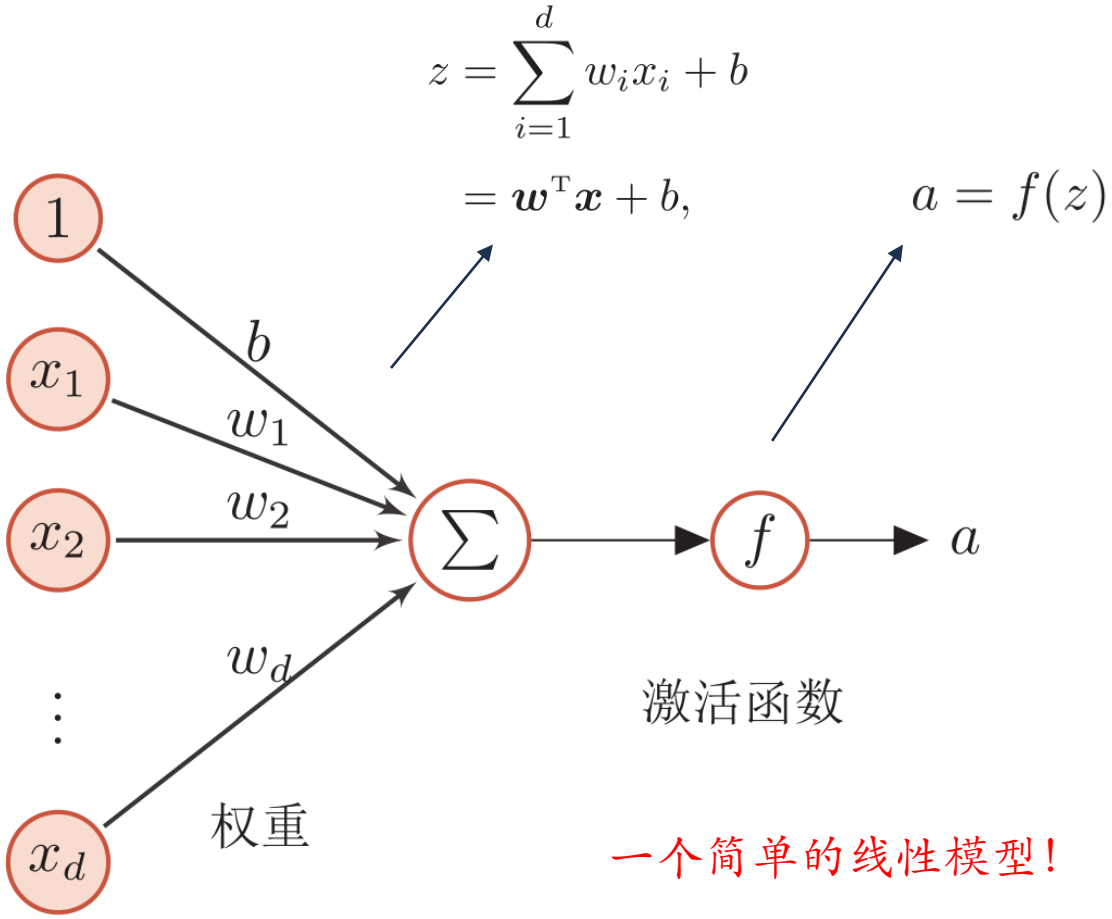


# 生物神经元




单个神经细胞只有两种状态：兴奋和抑制

# 人工神经元


$z = \sum_{i=1}^{d} w_i x_i + b$
$= \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b,$
$a = f(z)$
1
$x_1$
$x_2$
$x_d$
$b$
$w_1$
$w_2$
$w_d$
$\sum$
$f$
$a$
激活函数
权重
一个简单的线性模型!

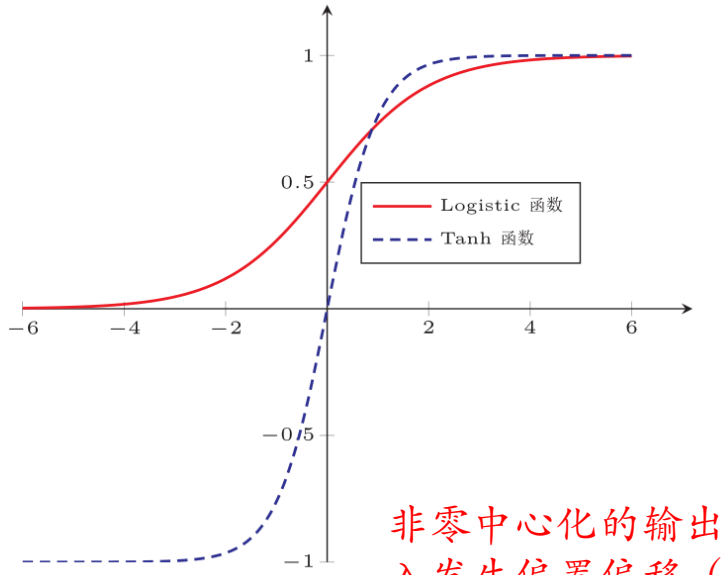
# 如何选择激活函数$f(z)$?

- 连续并可导（允许少数点上不可导）的非线性函数。
  - 可导的激活函数可以直接利用数值优化的方法来学习网络参数。
- 激活函数及其导函数要尽可能的简单
  - 有利于提高网络计算效率。
- 激活函数的导函数的值域要在一个合适的区间内
  - 不能太大也不能太小，否则会影响训练的效率和稳定性。
- 单调递增
  - ???

# 常见激活函数：Sigmoid型函数

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

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




非零中心化的输出会使得其后一层的神经元的输入发生偏置偏移（bias shift），并进一步使得梯度下降的收敛速度变慢。

性质：

饱和函数

Tanh函数是零中心化的，而logistic函数的输出恒大于0

# 常见激活函数：修正线性单元（Rectified Linear Unit）

$$\text{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

$$= \max(0, x).$$

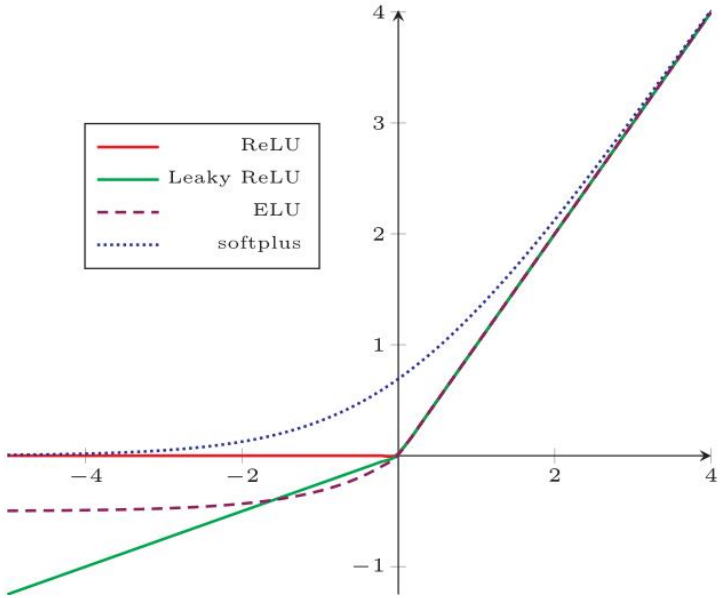
$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma x & \text{if } x \leq 0 \end{cases}$$

$$\text{PReLU}_i(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma_i x & \text{if } x \leq 0 \end{cases}$$

$$\text{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

$$= \max(0, x) + \min(0, \gamma(\exp(x) - 1))$$

$$\text{softplus}(x) = \log(1 + \exp(x))$$




- 计算上更加高效
- 生物学合理性
  - 单侧抑制、宽兴奋边界
- 在一定程度上缓解梯度消失问题

死亡ReLU问题（Dying ReLU Problem）

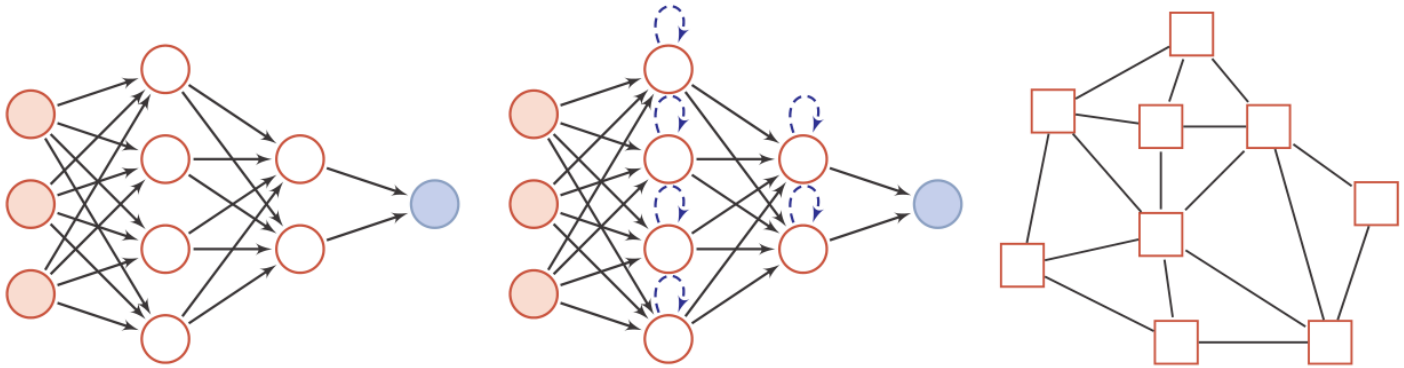
# 人工神经网络

- 人工神经网络主要由大量的神经元以及它们之间的有向连接构成。因此考虑三方面：

- 神经元的激活规则
  - 主要是指神经元输入到输出之间的映射关系，一般为非线性函数。
- 网络的拓扑结构
  - 不同神经元之间的连接关系。
- 学习算法
  - 通过训练数据来学习神经网络的参数。

# 网络结构

- 人工神经网络由神经元模型构成，这种由许多神经元组成的信息处理网络具有并行分布结构。



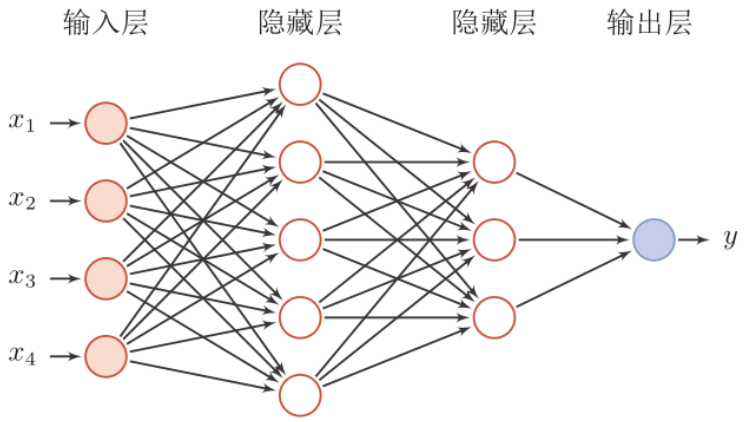
(a) 前馈网络　(b) 记忆网络　(c) 图网络

圆形节点表示一个神经元，方形节点表示一组神经元。

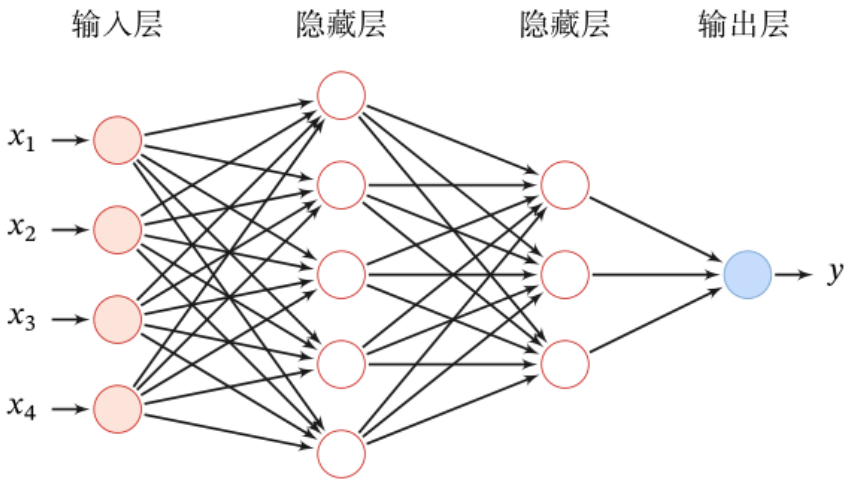
# 前馈神经网络

# 网络结构

- 前馈神经网络（全连接神经网络、多层感知器）
  - 各神经元分别属于不同的层，层内无连接。
  - 相邻两层之间的神经元全部两两连接。
  - 整个网络中无反馈，信号从输入层向输出层单向传播，可用一个有向无环图表示。



# 前馈网络




给定一个前馈神经网络，用下面的记号来描述这样网络：

| 记号 | 含义 |
| --- | --- |
| $L$ | 神经网络的层数 |
| $M_l$ | 第 $l$ 层神经元的个数 |
| $f_l(\cdot)$ | 第 $l$ 层神经元的激活函数 |
| $\boldsymbol{W}^{(l)} \in \mathbb{R}^{M_l \times M_{l-1}}$ | 第 $l-1$ 层到第 $l$ 层的权重矩阵 |
| $\boldsymbol{b}^{(l)} \in \mathbb{R}^{M_l}$ | 第 $l-1$ 层到第 $l$ 层的偏置 |
| $\boldsymbol{z}^{(l)} \in \mathbb{R}^{M_l}$ | 第 $l$ 层神经元的净输入（净活性值） |
| $\boldsymbol{a}^{(l)} \in \mathbb{R}^{M_l}$ | 第 $l$ 层神经元的输出（活性值） |

# 信息传递过程

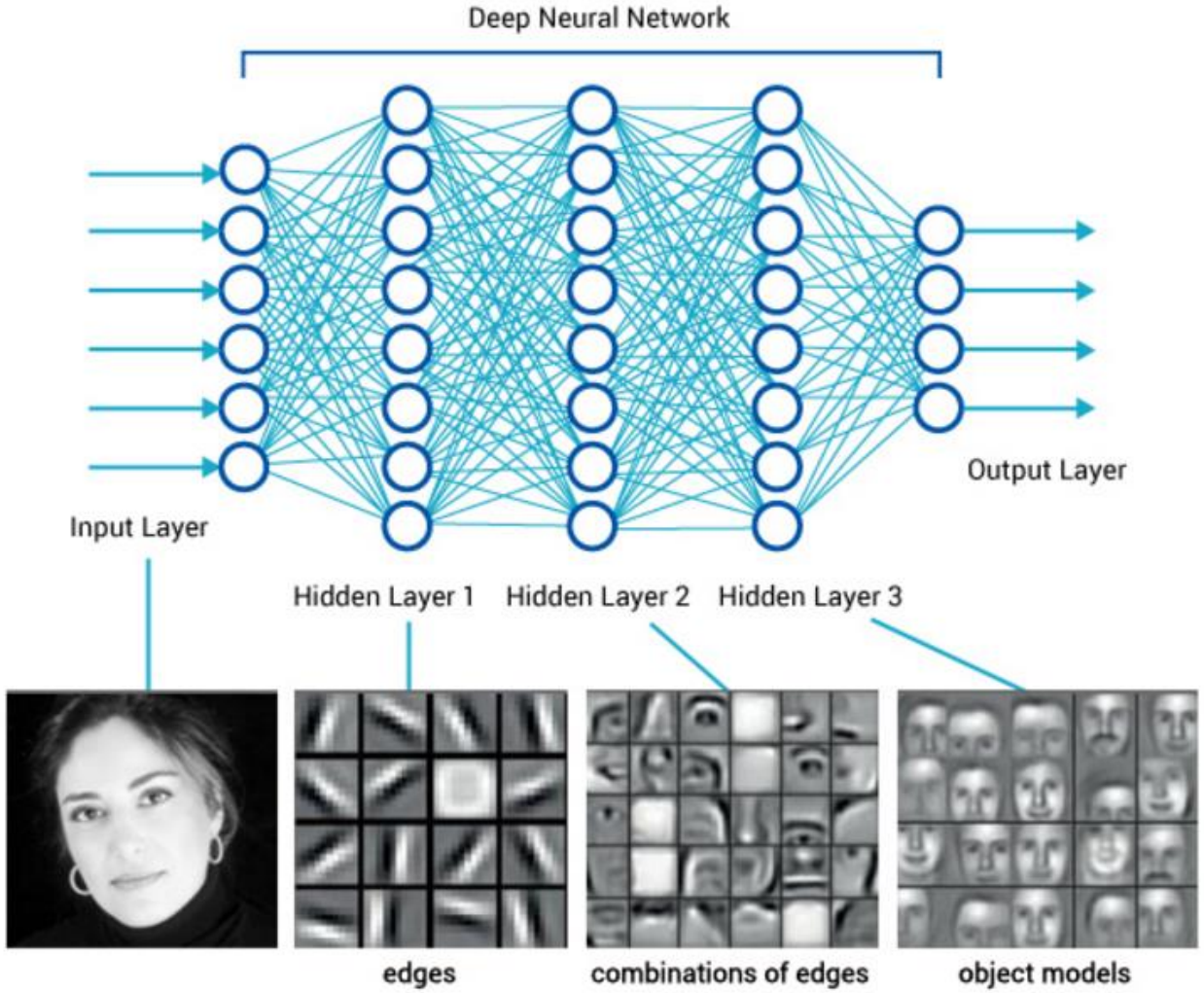
- 前馈神经网络通过下面公式进行信息传播。

$$\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)},$$
$$\boldsymbol{a}^{(l)} = f_l(\boldsymbol{z}^{(l)}).$$

- 前馈计算：

$$\boldsymbol{x} = \boldsymbol{a}^{(0)} \to \boldsymbol{z}^{(1)} \to \boldsymbol{a}^{(1)} \to \boldsymbol{z}^{(2)} \to \cdots \to \boldsymbol{a}^{(L-1)} \to \boldsymbol{z}^{(L)} \to \boldsymbol{a}^{(L)} = \phi(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}))$$

# 深层前馈神经网络


Deep Neural Network
Input Layer
Hidden Layer 1
Hidden Layer 2
Hidden Layer 3
Output Layer
edges
combinations of edges
object models

# 通用近似定理

**定理 4.1 – 通用近似定理（Universal Approximation Theorem）[Cybenko, 1989, Hornik et al., 1989]：** 令 $\varphi(\cdot)$ 是一个非常数、有界、单调递增的连续函数，$\mathcal{I}_d$ 是一个 $d$ 维的单位超立方体 $[0,1]^d$，$C(\mathcal{I}_d)$ 是定义在 $\mathcal{I}_d$ 上的连续函数集合。对于任何一个函数 $f \in C(\mathcal{I}_d)$，存在一个整数 $m$，和一组实数 $v_i, b_i \in \mathbb{R}$ 以及实数向量 $\mathbf{w}_i \in \mathbb{R}^d$，$i = 1, \cdots, m$，以至于我们可以定义函数

$$F(\mathbf{x}) = \sum_{i=1}^{m} v_i \varphi(\mathbf{w}_i^{\mathrm{T}} \mathbf{x} + b_i), \tag{4.33}$$

作为函数 $f$ 的近似实现，即

$$|F(\mathbf{x}) - f(\mathbf{x})| < \epsilon, \forall \mathbf{x} \in \mathcal{I}_d. \tag{4.34}$$

其中 $\epsilon > 0$ 是一个很小的正数。

根据通用近似定理，对于具有线性输出层和至少一个使用“挤压”性质的激活函数的隐藏层组成的前馈神经网络，只要其隐藏层神经元的数量足够，它可以以任意的精度来近似任何从一个定义在实数空间中的有界闭集函数。

# 应用到机器学习

- 回顾：分类器

$$\hat{y} = g(\varphi(\mathbf{x}), \theta)$$

分类器　　神经网络

- 神经网络可以作为一个“万能”函数来使用，可以用来进行复杂的特征转换，或逼近一个复杂的条件分布。
  - 如果$g(\cdot)$为Logistic回归，那么Logistic回归分类器可以看成神经网络的最后一层。

# 参数学习

# 应用到机器学习

- 对于多分类问题
  - 如果使用Softmax回归分类器，相当于网络最后一层设置C 个神经元，其输出经过Softmax函数进行归一化后可以作为每个类的条件概率。

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z}^{(L)})$$

  - 采用交叉熵损失函数，对于样本(x,y)，其损失函数为

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\mathbf{y}^{\mathrm{T}} \log \hat{\mathbf{y}}$$

# 参数学习

- 给定训练集为$D = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$，将每个样本$\boldsymbol{x}^{(n)}$输入给前馈神经网络，得到网络输出为$\hat{y}^{(n)}$，其在数据集D上的结构化风险函数为：

$$\mathcal{R}(W, \mathbf{b}) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(\mathbf{y}^{(n)}, \hat{\mathbf{y}}^{(n)}) + \frac{1}{2}\lambda\|W\|_F^2$$

- 梯度下降

$$W^{(l)} \leftarrow W^{(l)} - \alpha\frac{\partial\mathcal{R}(W, \mathbf{b})}{\partial W^{(l)}}$$

$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \alpha\frac{\partial\mathcal{R}(W, \mathbf{b})}{\partial \mathbf{b}^{(l)}}$$

# 如何计算梯度?

- 神经网络为一个复杂的复合函数
  - 链式法则

$$y = f^5(f^4(f^3(f^2(f^1(x))))) \rightarrow \frac{\partial y}{\partial x} = \frac{\partial f^1}{\partial \mathrm{x}} \frac{\partial f^2}{\partial f^1} \frac{\partial f^3}{\partial f^2} \frac{\partial f^4}{\partial f^3} \frac{\partial f^5}{\partial f^4}$$

- 反向传播算法
  - 根据前馈网络的特点而设计的高效方法

- 一个更加通用的计算方法
  - 自动微分（Automatic Differentiation，AD）

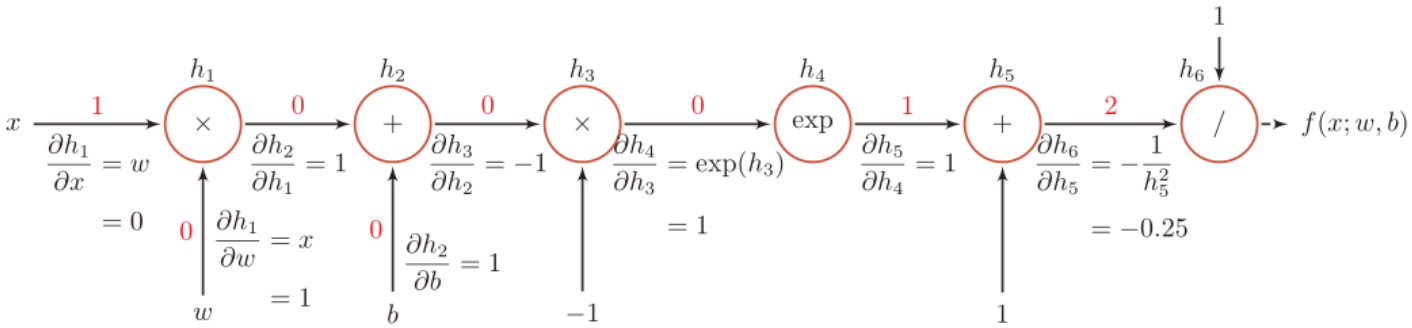
# 计算图与自动微分

# 计算图与自动微分

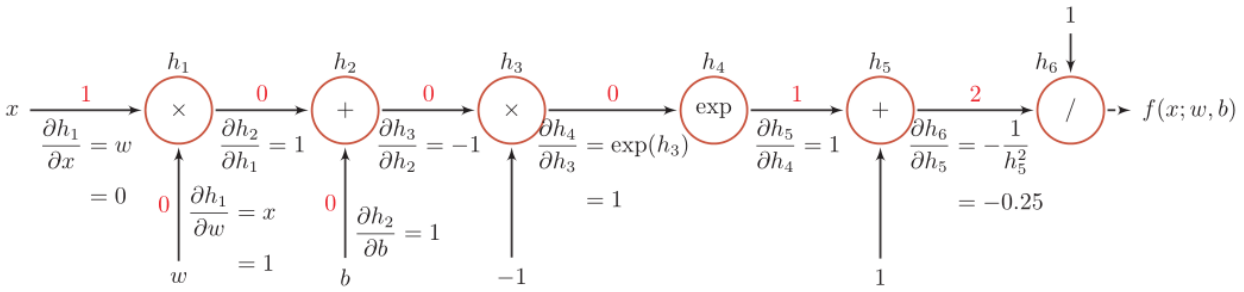
- 自动微分是利用链式法则来自动计算一个复合函数的梯度。

$$f(x;w,b)=\frac{1}{\exp(-(wx+b))+1}.$$

- 计算图



# 计算图




| 函数 | 导数 | |
|---|---|---|
| $h_1 = x \times w$ | $\frac{\partial h_1}{\partial w} = x$ | $\frac{\partial h_1}{\partial x} = w$ |
| $h_2 = h_1 + b$ | $\frac{\partial h_2}{\partial h_1} = 1$ | $\frac{\partial h_2}{\partial b} = 1$ |
| $h_3 = h_2 \times -1$ | $\frac{\partial h_3}{\partial h_2} = -1$ | |
| $h_4 = \exp(h_3)$ | $\frac{\partial h_4}{\partial h_3} = \exp(h_3)$ | |
| $h_5 = h_4 + 1$ | $\frac{\partial h_5}{\partial h_4} = 1$ | |
| $h_6 = 1/h_5$ | $\frac{\partial h_6}{\partial h_5} = -\frac{1}{h_5^2}$ | |

当x = 1,w = 0,b = 0时，可以得到

$$\begin{aligned}\frac{\partial f(x;w,b)}{\partial w}|_{x=1,w=0,b=0} &= \frac{\partial f(x;w,b)}{\partial h_6}\frac{\partial h_6}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial w} \\ &= 1 \times -0.25 \times 1 \times 1 \times -1 \times 1 \times 1 \\ &= 0.25.\end{aligned}$$

# 自动微分

- 前向模式和反向模式
  - 反向模式和反向传播的计算梯度的方式相同

- 如果函数和参数之间有多条路径，可以将这多条路径上的导数再进行相加，得到最终的梯度。

# 反向传播算法 (自动微分的反向模式)

- 前馈神经网络的训练过程可以分为以下三步
  - 前向计算每一层的状态和激活值，直到最后一层
  - 反向计算每一层的参数的偏导数
  - 更新参数

# 静态计算图和动态计算图

- 静态计算图是在编译时构建计算图，计算图构建好之后在程序运行时不能改变。
  - Theano和Tensorflow1.0
- 动态计算图是在程序运行时动态构建。两种构建方式各有优缺点。
  - DyNet，Chainer和PyTorch

- 静态计算图在构建时可以进行优化，并行能力强，但灵活性比较差低。动态计算图则不容易优化，当不同输入的网络结构不一致时，难以并行计算，但是灵活性比较高。

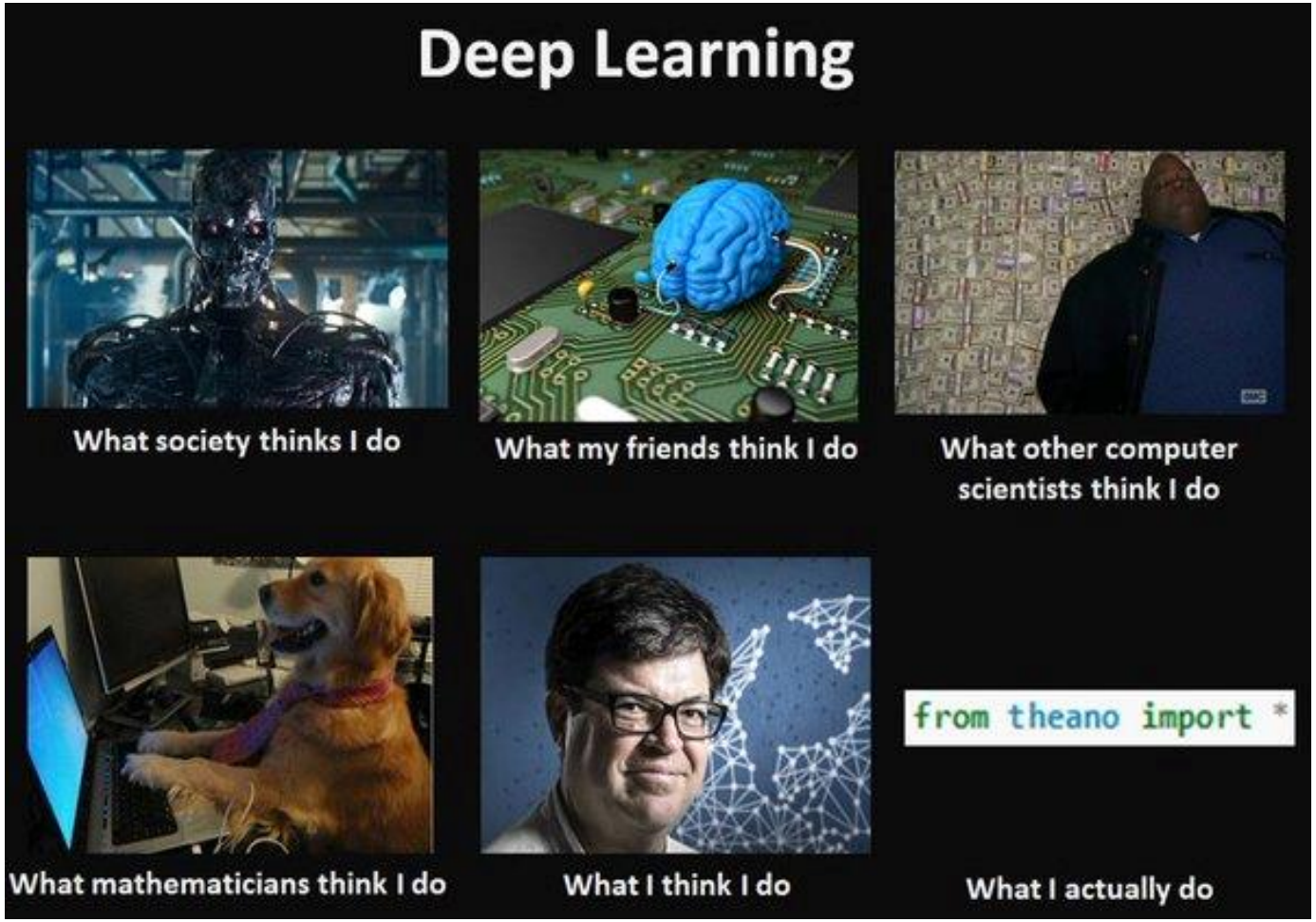
# 深度学习的三个步骤


Deep Learning is so simple ……

# Getting started: 30 seconds to Keras

```
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.optimizers import SGD

model = Sequential()
model.add(Dense(output_dim=64, input_dim=100))
model.add(Activation("relu"))
model.add(Dense(output_dim=10))
model.add(Activation("softmax"))

model.compile(loss='categorical_crossentropy',
   optimizer='sgd', metrics=['accuracy'])

model.fit(X_train, Y_train, nb_epoch=5, batch_size=32)

loss = model.evaluate(X_test, Y_test, batch_size=32)
```

# 如何实现?

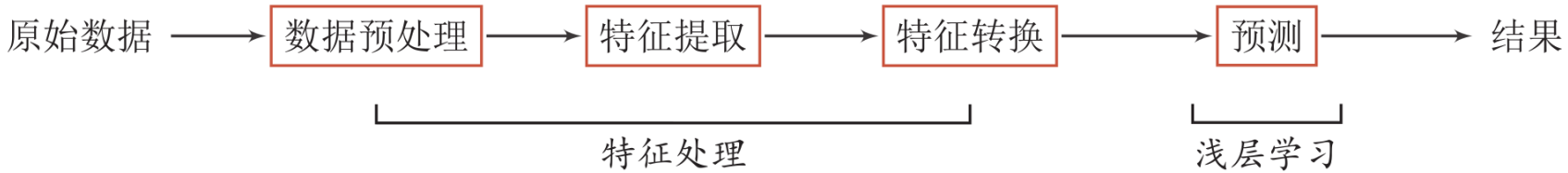

Deep Learning
What society thinks I do
What my friends think I do
What other computer scientists think I do
What mathematicians think I do
What I think I do
from theano import *
What I actually do

# 深度学习
# Deep Learning

# 机器学习

- 当我们用机器学习来解决一些模式识别任务时，一般的流程包含以下几个步骤：

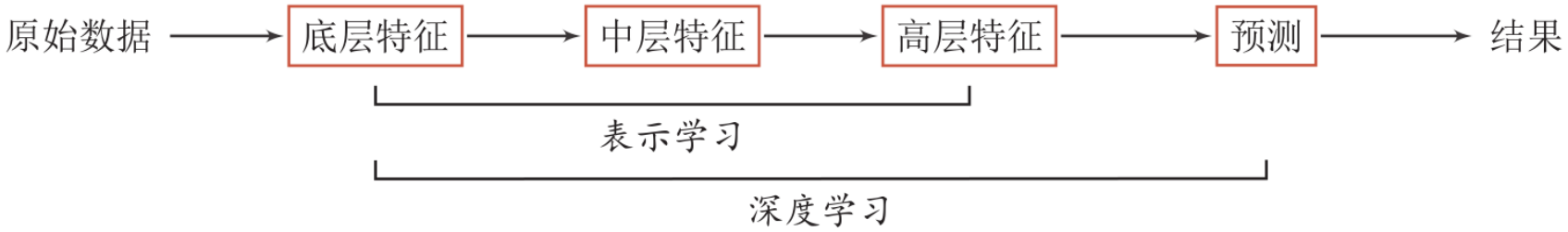
特征工程（Feature Engineering）




- 浅层学习（Shallow Learning）：不涉及特征学习，其特征主要靠人工经验或特征转换方法来抽取。

# 深度学习

- 通过构建具有一定“深度”的模型，可以让模型来自动学习好的特征表示（从底层特征，到中层特征，再到高层特征），从而最终提升预测或识别的准确性。




难点：贡献度分配问题

# 深度学习的数学描述

浅层学习

$$y = f(\mathbf{x})$$

$$y = f^{(2)}(f^{(1)}(\mathbf{x}))$$ 可学习的基函数

深度学习

$$y = f^{(5)}(f^{(4)}(f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))))$$

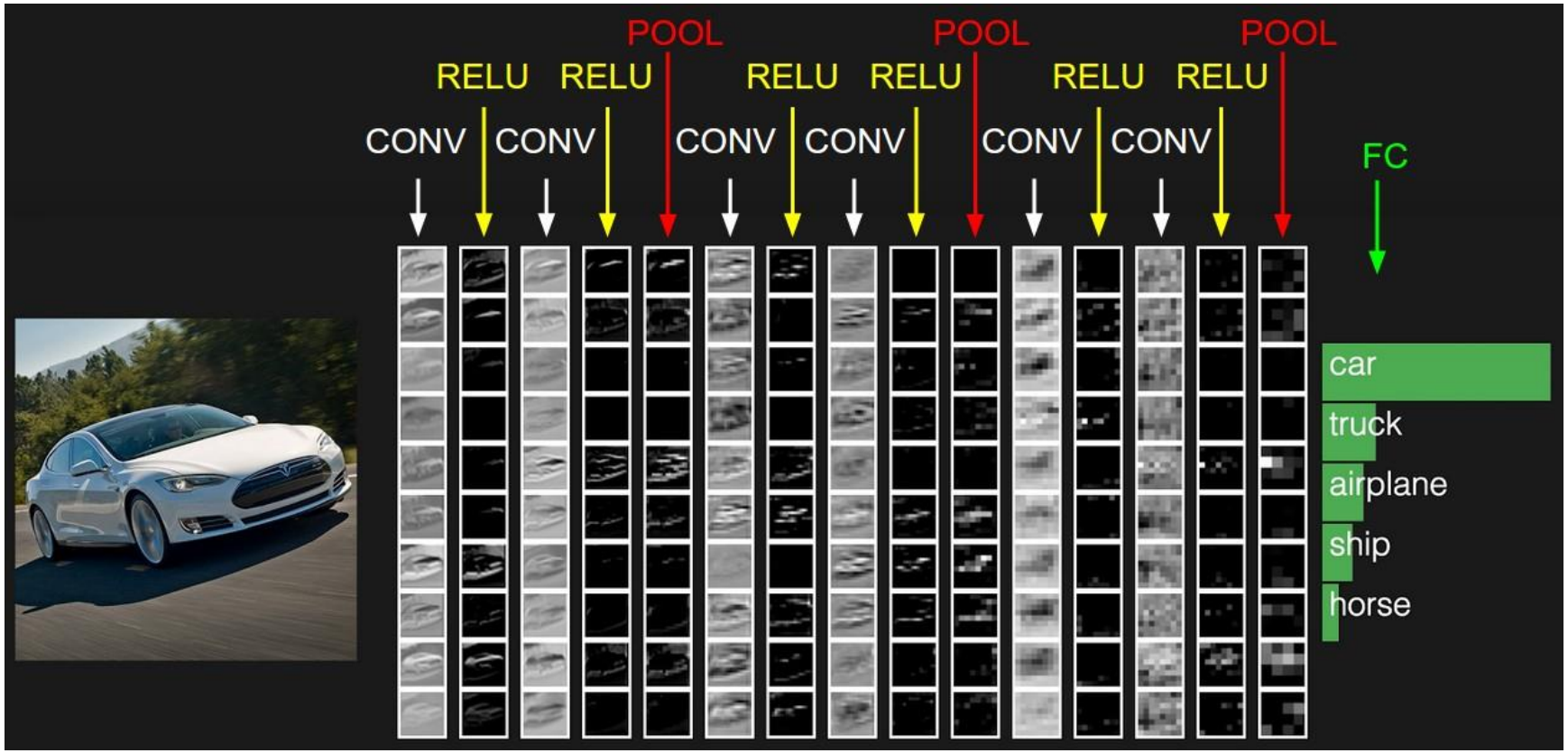
---

$f^{(l)}(x)$ 非线性函数，不一定连续。

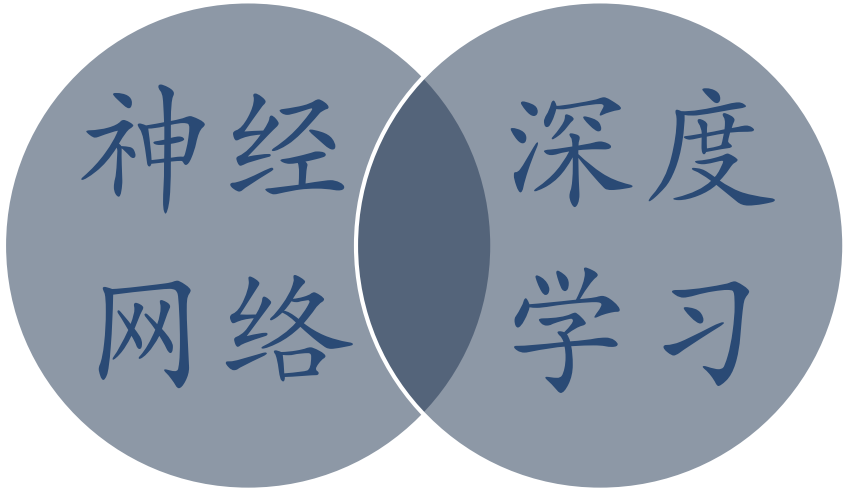
当$f^{l}(x)$连续时，个复合函数称为神经网络。

$$f^{l}(x) = \sigma\left(W^{(l)} f^{(l-1)}(x)\right)$$

# 表示学习与深度学习


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

# 神经网络与深度学习的关系




神经网络：一种以（人工）神经元为基本单元的模型

深度学习：一类机器学习问题，主要解决贡献度分配问题。

# 课后练习

- 知识点
  - 激活函数
  - 误差反向传播
  - 自动微分与计算图
- 编程练习
  - 使用PyTorch、Paddle或Mindspore实现前馈神经网络