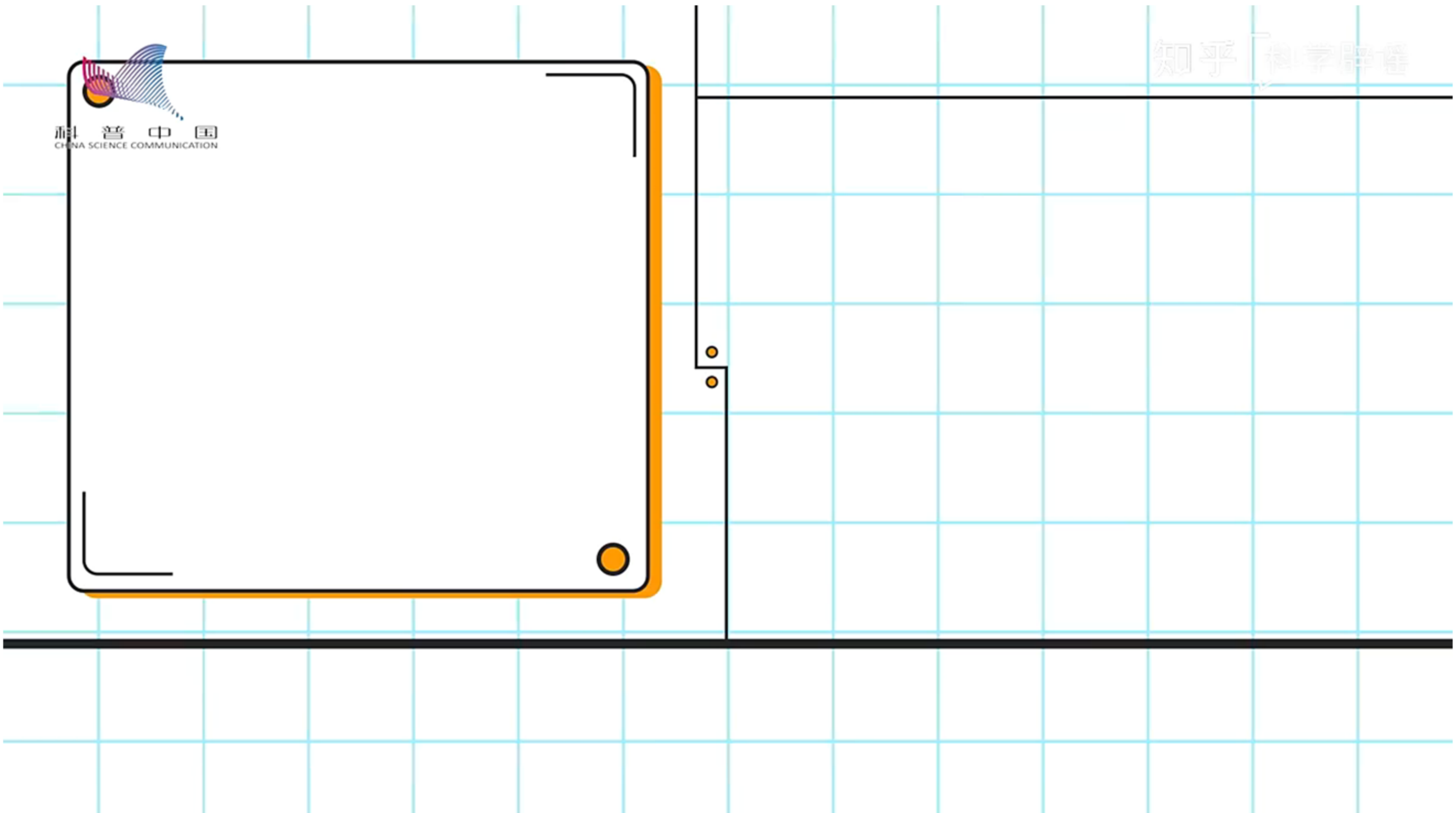
中国科学院南京分院
Nanjing Branch of Chinese Academy of Sciences

人工智能原理与算法
8. 前向神经网络

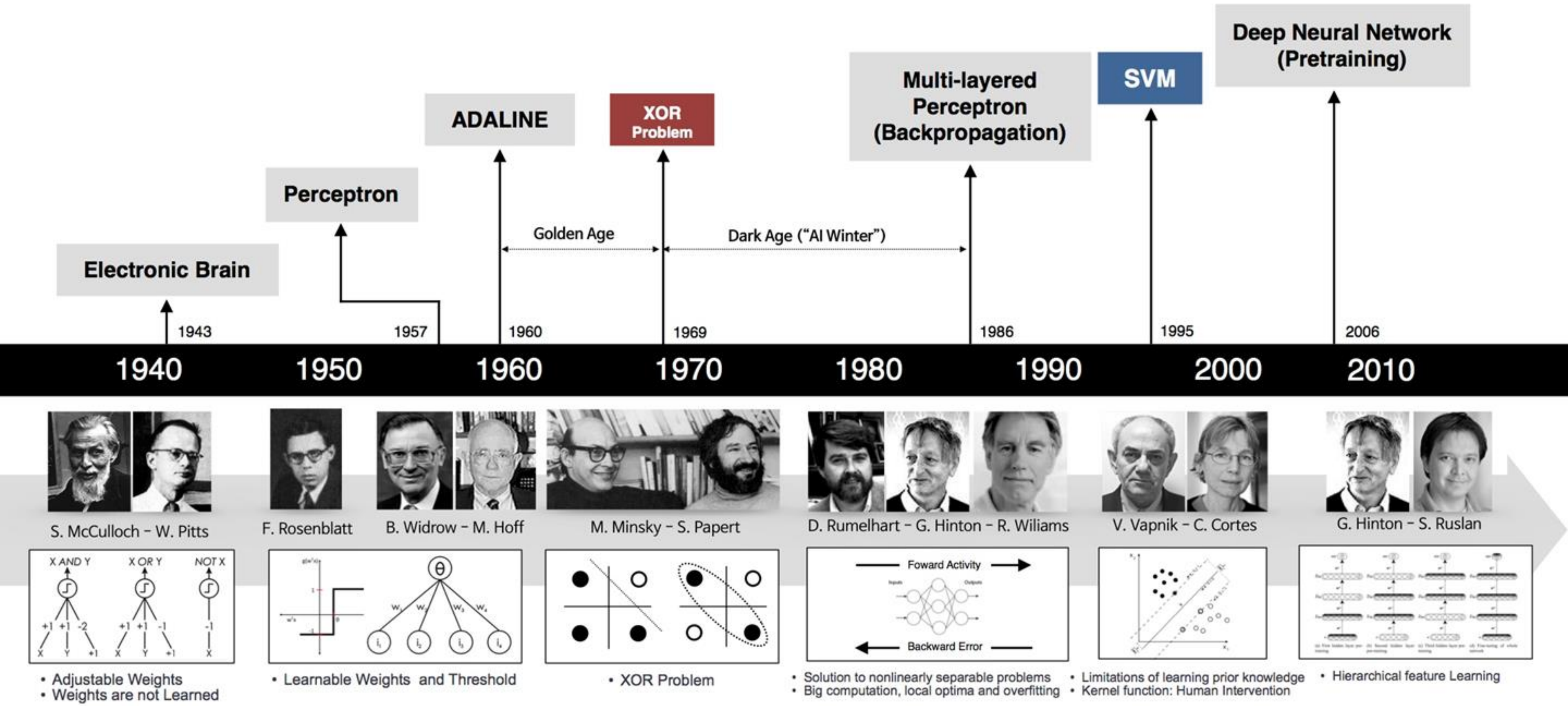
夏睿

2023.3.29

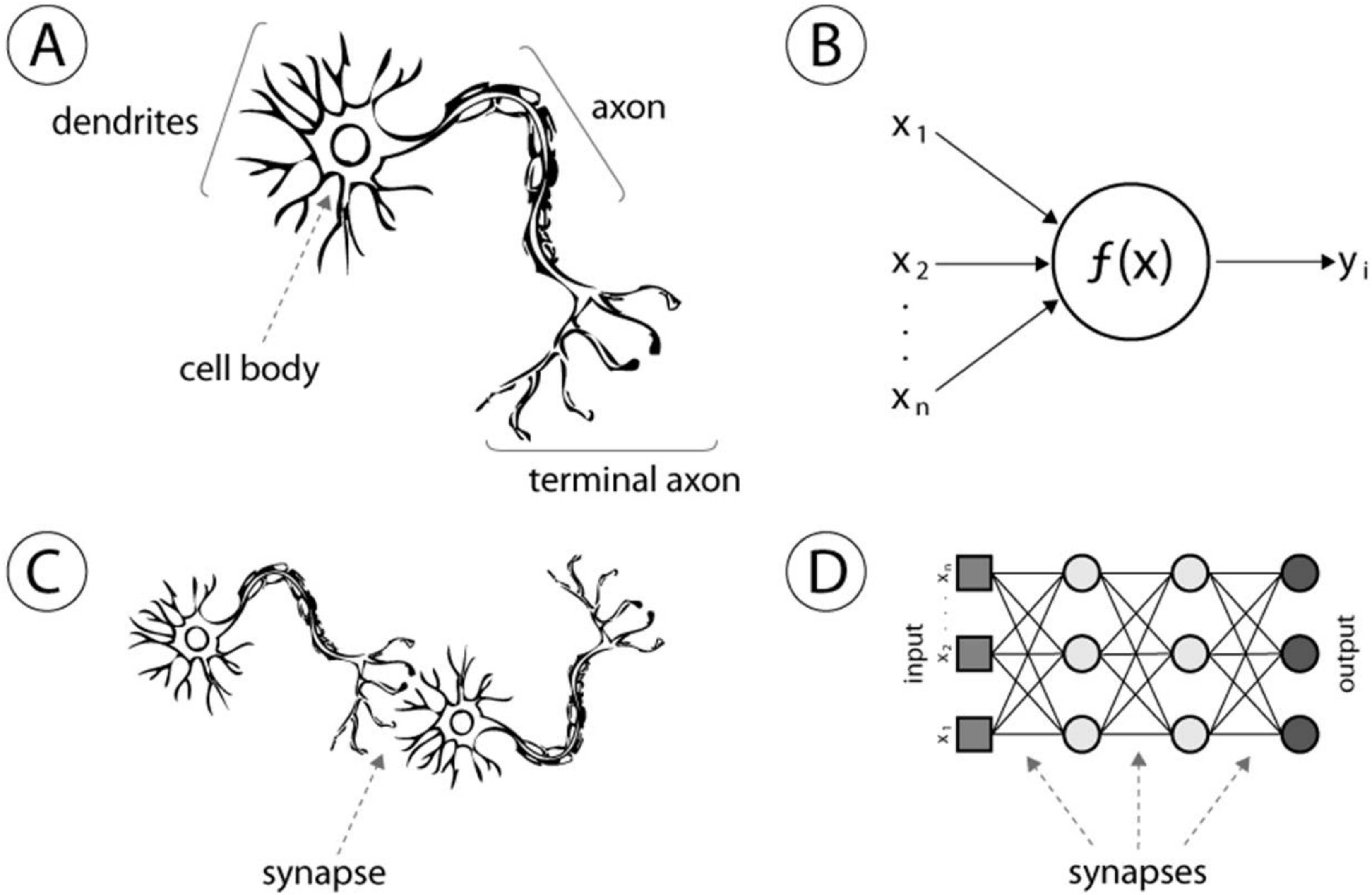
人类大脑与神经网络



人工神经网络的发展历史



启发于神经网络


A
dendrites
axon
cell body
terminal axon
B
x 1
x 2
x n
f(x)
y i
C
synapse
D
input
x 1
x 2
x n
output
synapses

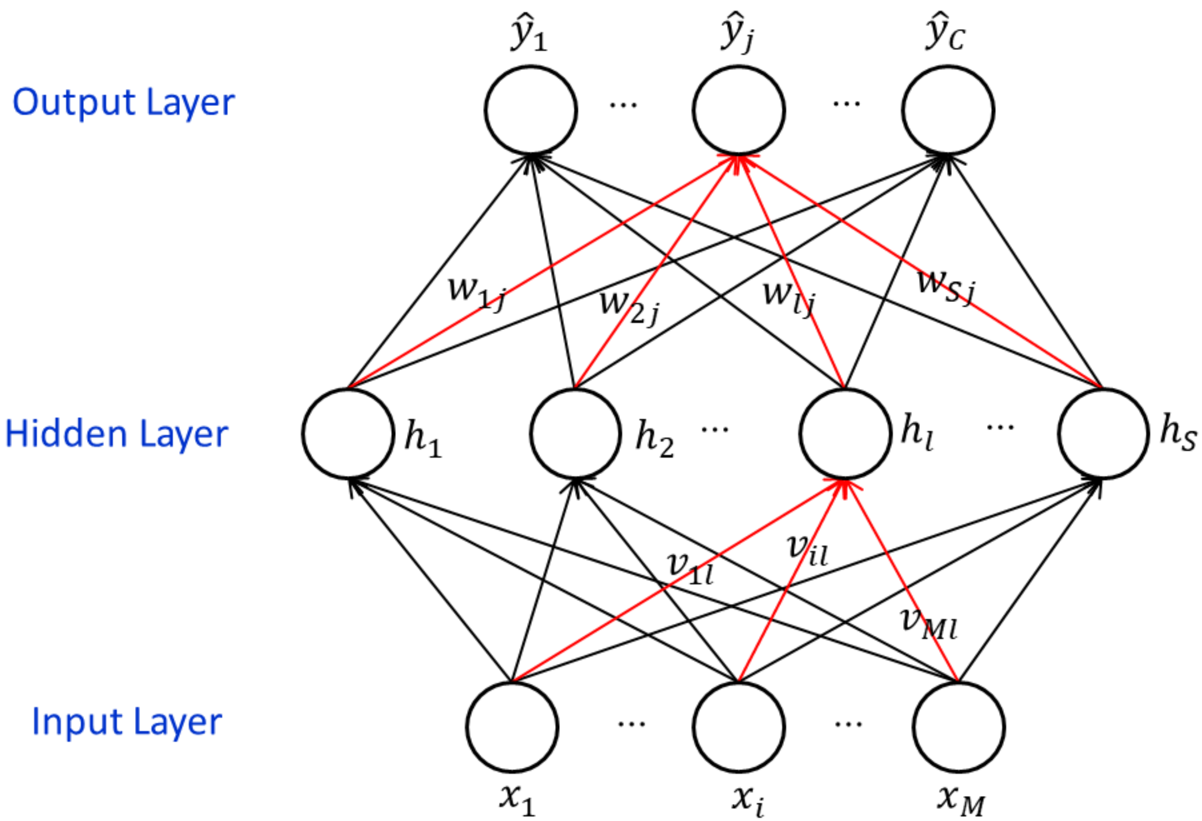
单层神经网络

- 单层神经网络回顾（线性模型）
  - 线性回归
  - Logistic/Softmax回归
  - 感知机/多类感知机

- 假设
  - 反映输入到输出的数学模型（包含未知参数）
- 学习
  - 学习准则（损失函数）
  - 最优化方法
- 预测
  - 模型假设作为预测函数

三层前向神经网络

- 网络结构图



- 模型假设

$$\hat{y}_j = \delta(\beta_j + \theta_j)$$

$$\beta_j = \sum_{l=1}^{S} w_{lj} h_l$$

$$h_l = \delta(\alpha_l + \gamma_l)$$

$$\alpha_l = \sum_{i=1}^{M} v_{il} x_i$$

学习算法

- 训练集

$$D = \{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), (\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}), \ldots, (\boldsymbol{x}^{(N)}, \boldsymbol{y}^{(N)})\}, \boldsymbol{x}^{(k)} \in \mathbb{R}^M, \boldsymbol{y}^{(k)} \in \mathbb{R}^C$$

$\boldsymbol{y}^{(k)}$为类别独热向量

- 损失函数

$$E^{(k)} = \frac{1}{2}\sum_{j=1}^{C}\left(\hat{y}_j^{(k)} - y_j^{(k)}\right)^2$$

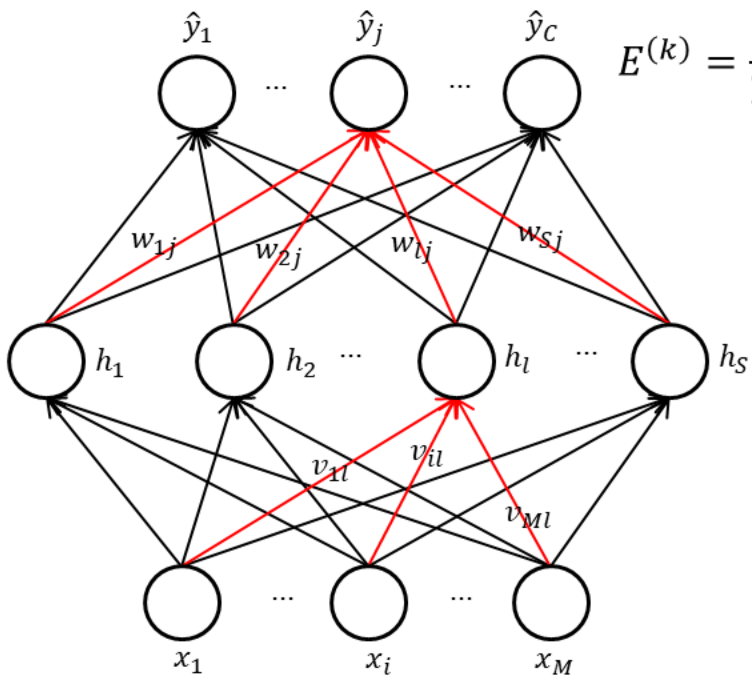
- 参数

$$\boldsymbol{v} \in \mathbb{R}^{M*S}, \boldsymbol{\gamma} \in \mathbb{R}^S, \boldsymbol{w} \in \mathbb{R}^{S*C}, \boldsymbol{\theta} \in \mathbb{R}^C$$

- 梯度

$$\frac{\partial E^{(k)}}{\partial v_{il}}, \frac{\partial E^{(k)}}{\partial \gamma_l}, \frac{\partial E^{(k)}}{\partial w_{lj}}, \frac{\partial E^{(k)}}{\partial \theta_j}$$

梯度计算



$$E^{(k)} = \frac{1}{2}\sum_{j=1}^{C}\left(\hat{y}_j^{(k)} - y_j^{(k)}\right)^2$$

$$\hat{y}_j = \delta(\beta_j + \theta_j)$$

$$\beta_j = \sum_{l=1}^{S} w_{lj} h_l$$

$$h_l = \delta(\alpha_l + \gamma_l)$$

$$\alpha_l = \sum_{i=1}^{M} v_{il} x_i$$

误差

$$\frac{\partial E^{(k)}}{\partial \hat{y}_j^{(k)}} = \boxed{\left(\hat{y}_j^{(k)} - y_j^{(k)}\right) = error_j}$$

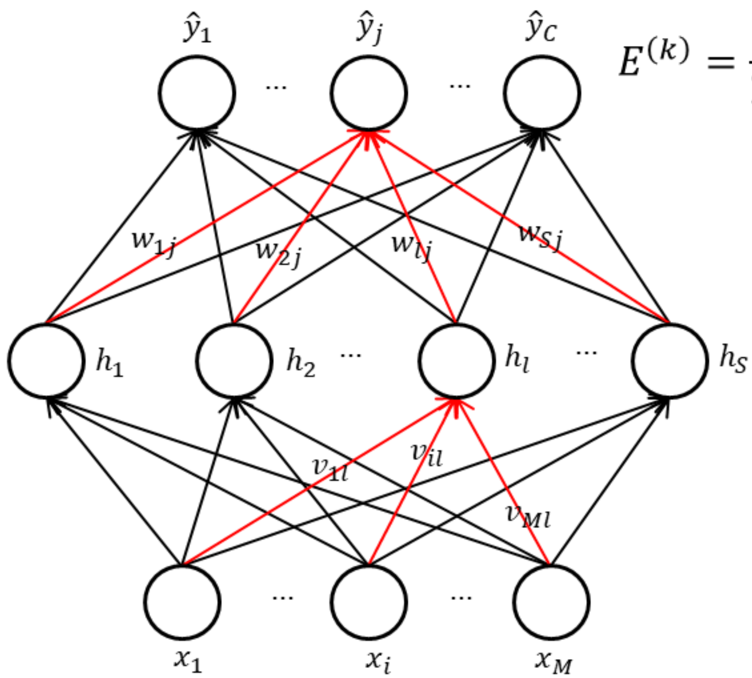
$$\frac{\partial \hat{y}_j^{(k)}}{\partial(\beta_j + \theta_j)} = \hat{y}_j^{(k)} \cdot \left(1 - \hat{y}_j^{(k)}\right)$$

$$\frac{\partial(\beta_j + \theta_j)}{\partial w_{lj}} = h_l$$

$$\begin{aligned}\frac{\partial E^{(k)}}{\partial w_{lj}} &= \frac{\partial E^{(k)}}{\partial \hat{y}_j^{(k)}} \cdot \frac{\partial \hat{y}_j^{(k)}}{\partial(\beta_j + \theta_j)} \cdot \frac{\partial(\beta_j + \theta_j)}{\partial w_{lj}} \\ &= \left(\hat{y}_j^{(k)} - y_j^{(k)}\right) \cdot \hat{y}_j^{(k)} \cdot \left(1 - \hat{y}_j^{(k)}\right) h_l \\ &= \boxed{error_j^{OutputLayer}} h_l\end{aligned}$$

广义误差 · 输入

梯度计算



$$E^{(k)} = \frac{1}{2}\sum_{j=1}^{C}\left(\hat{y}_j^{(k)} - y_j^{(k)}\right)^2$$

$$\hat{y}_j = \delta(\beta_j + \theta_j)$$

$$\beta_j = \sum_{l=1}^{S} w_{lj} h_l$$

$$h_l = \delta(\alpha_l + \gamma_l)$$

$$\alpha_l = \sum_{i=1}^{M} v_{il} x_i$$

误差

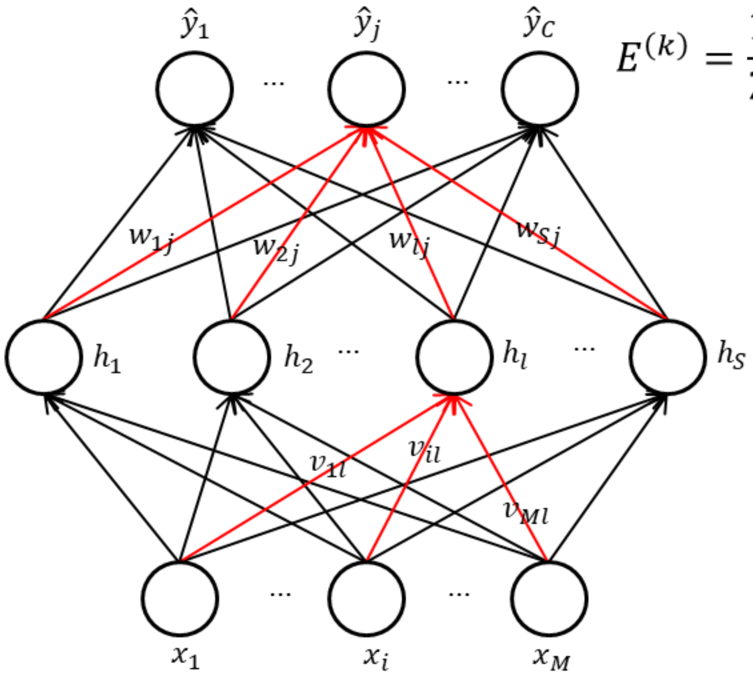
$$\frac{\partial E^{(k)}}{\partial \hat{y}_j^{(k)}} = \boxed{\left(\hat{y}_j^{(k)} - y_j^{(k)}\right) = error_j}$$

$$\frac{\partial \hat{y}_j^{(k)}}{\partial(\beta_j + \theta_j)} = \hat{y}_j^{(k)} \cdot \left(1 - \hat{y}_j^{(k)}\right)$$

$$\frac{\partial(\beta_j + \theta_j)}{\partial w_{lj}} = h_l$$

$$\begin{aligned}\frac{\partial E^{(k)}}{\partial \theta_j} &= \frac{\partial E^{(k)}}{\partial \hat{y}_j^{(k)}} \cdot \frac{\partial \hat{y}_j^{(k)}}{\partial(\beta_j + \theta_j)} \cdot \frac{\partial(\beta_j + \theta_j)}{\partial \theta_j} \\ &= \left(\hat{y}_j^{(k)} - y_j^{(k)}\right) \cdot \hat{y}_j^{(k)} \cdot \left(1 - \hat{y}_j^{(k)}\right) \\ &= error_j^{OutputLayer} \cdot 1\end{aligned}$$

梯度计算



$$E^{(k)} = \frac{1}{2}\sum_{j=1}^{C}\left(\hat{y}_j^{(k)} - y_j^{(k)}\right)^2$$

$$\hat{y}_j = \delta(\beta_j + \theta_j)$$

$$\beta_j = \sum_{l=1}^{S} w_{lj} h_l$$

$$h_l = \delta(\alpha_l + \gamma_l)$$

$$\alpha_l = \sum_{i=1}^{M} v_{il} x_i$$

$$\frac{\partial E^{(k)}}{\partial(\beta_j + \theta_j)} = error_j^{OutputLayer}$$

$$\frac{\partial(\beta_j + \theta_j)}{\partial h_l} = w_{lj}$$

$$\frac{\partial h_l}{\partial(\alpha_l + \gamma_l)} = h_l \cdot (1 - h_l)$$

$$\frac{\partial(\alpha_l + \gamma_l)}{\partial v_{il}} = x_i^{(k)}$$

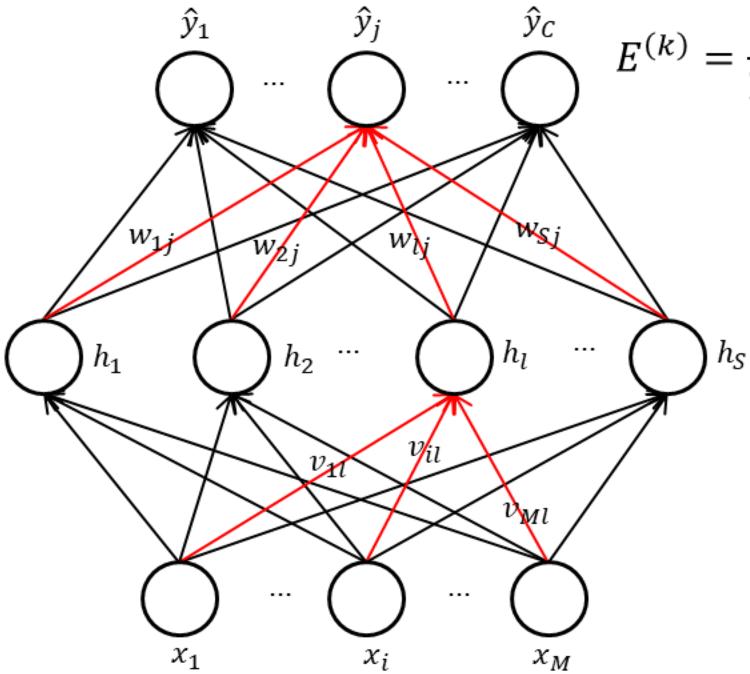
$$\frac{\partial E^{(k)}}{\partial v_{il}} = \sum_{j=1}^{C} \frac{\partial E^{(k)}}{\partial \hat{y}_j^{(k)}} \cdot \frac{\partial \hat{y}_j^{(k)}}{\partial(\beta_j + \theta_j)} \cdot \frac{\partial(\beta_j + \theta_j)}{\partial h_l} \cdot \frac{\partial h_l}{\partial(\alpha_l + \gamma_l)} \cdot \frac{\partial(\alpha_l + \gamma_l)}{\partial v_{il}}$$

$$= \sum_{j=1}^{C} error_j^{OutputLayer} \cdot w_{lj} \cdot h_l \cdot (1 - h_l) \cdot x_i^{(k)}$$

$$= \boxed{error_l^{HiddenLayer} \cdot x_i^{(k)}}$$

广义误差 · 输入

梯度计算



$$E^{(k)} = \frac{1}{2}\sum_{j=1}^{C}\left(\hat{y}_j^{(k)} - y_j^{(k)}\right)^2$$

$$\hat{y}_j = \delta(\beta_j + \theta_j)$$

$$\beta_j = \sum_{l=1}^{S} w_{lj} h_l$$

$$h_l = \delta(\alpha_l + \gamma_l)$$

$$\alpha_l = \sum_{i=1}^{M} v_{il} x_i$$

$$\frac{\partial E^{(k)}}{\partial(\beta_j + \theta_j)} = error_j^{OutputLayer}$$

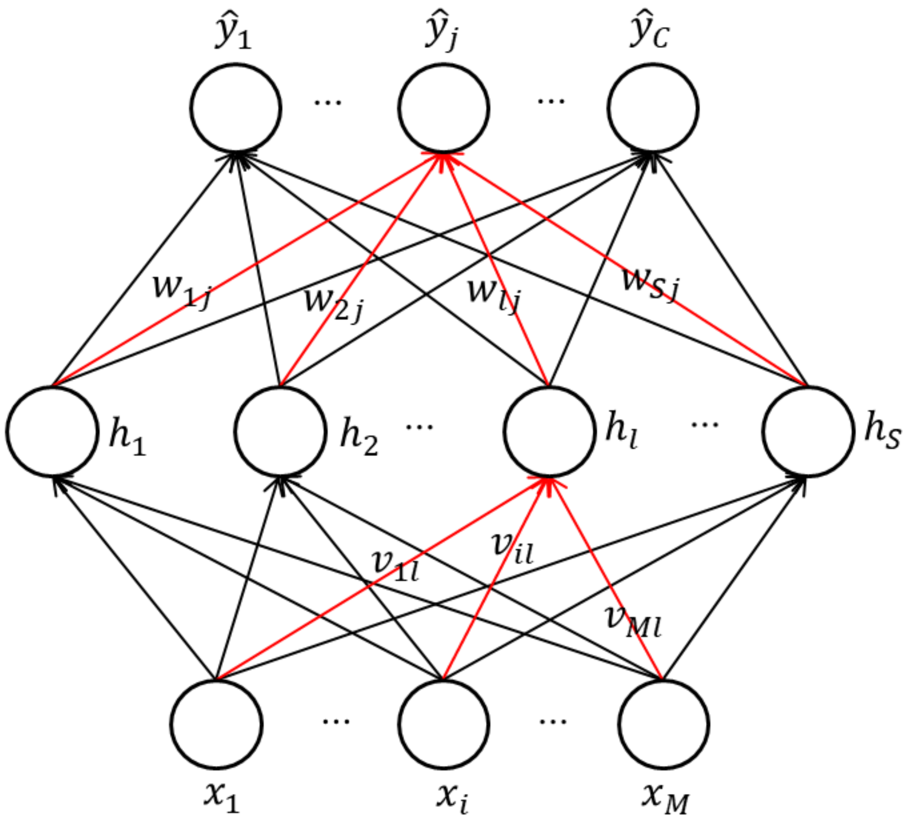
$$\frac{\partial(\beta_j + \theta_j)}{\partial h_l} = w_{lj}$$

$$\frac{\partial h_l}{\partial(\alpha_l + \gamma_l)} = h_l \cdot (1 - h_l)$$

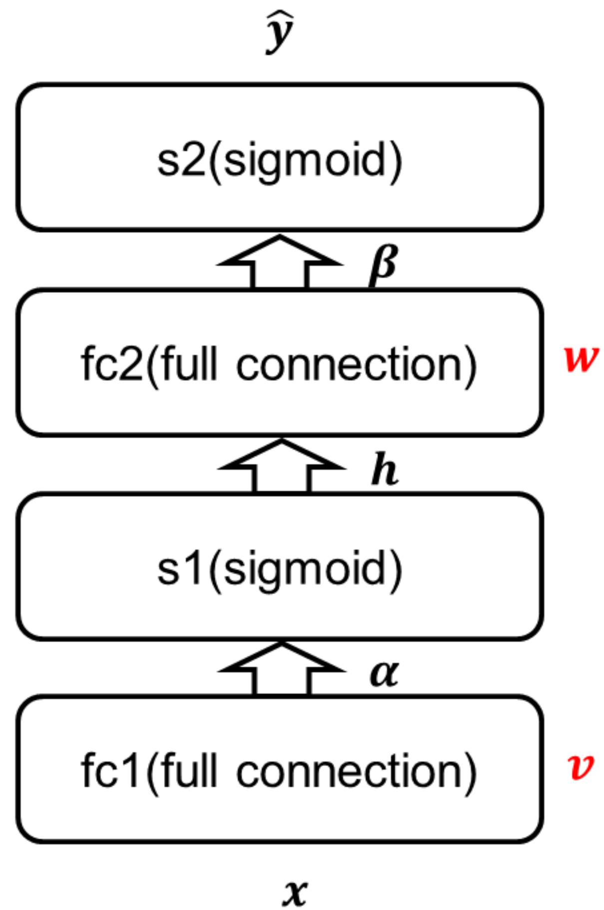
$$\frac{\partial(\alpha_l + \gamma_l)}{\partial v_{il}} = x_i^{(k)}$$

$$\begin{aligned}\frac{\partial E^{(k)}}{\partial \gamma_l} &= \sum_{j=1}^{C} \frac{\partial E^{(k)}}{\partial \hat{y}_j^{(k)}} \cdot \frac{\partial \hat{y}_j^{(k)}}{\partial(\beta_j + \theta_j)} \cdot \frac{\partial(\beta_j + \theta_j)}{\partial h_l} \cdot \frac{\partial h_l}{\partial(\alpha_l + \gamma_l)} \cdot \frac{\partial(\alpha_l + \gamma_l)}{\partial \gamma_l} \\ &= \sum_{j=1}^{C} error_j^{OutputLayer} \cdot w_{lj} \cdot h_l \cdot (1 - h_l) \cdot 1 \\ &= error_l^{HiddenLayer} \cdot 1\end{aligned}$$

前向运算



$\hat{y}_1$ ... $\hat{y}_j$ ... $\hat{y}_C$
w_{1j} w_{2j} w_{lj} w_{Sj}
h_1 h_2 ... h_l ... h_S
v_{1l} v_{il} v_{Ml}
x_1 ... x_i ... x_M



$\hat{\boldsymbol{y}}$
s2(sigmoid)
$\boldsymbol{\beta}$
fc2(full connection)
$\boldsymbol{w}$
$\boldsymbol{h}$
s1(sigmoid)
$\boldsymbol{\alpha}$
fc1(full connection)
$\boldsymbol{v}$
$\boldsymbol{x}$

误差反向传播（Error Back-Propagation）

参数求导的链式法则

$$\frac{\partial E}{\partial v_{il}} = \sum_{j=1}^{C} \underline{\frac{\partial E}{\partial \hat{y}_j}} \cdot \frac{\partial \hat{y}_j}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial h_l} \cdot \frac{\partial h_l}{\partial \alpha_l} \cdot \frac{\partial \alpha_l}{\partial v_{il}}$$

error · s2.grad

sum (error · fc2.grad)

error · s1.grad

error · fc1.grad

当前误差 = 前序误差 · 当前单元梯度，

从顶层向底层不断累积，反向传播

算法流程

- 梯度更新公式

$$w_{lj} \leftarrow w_{lj} - \eta \cdot \frac{\partial E^{(k)}}{\partial \omega_{lj}}$$

$$\theta_j \leftarrow \theta_j - \eta \cdot \frac{\partial E^{(k)}}{\partial \theta_j}$$

$$v_{il} \leftarrow v_{il} - \eta \cdot \frac{\partial E^{(k)}}{\partial v_{il}}$$

$$\gamma_l \leftarrow \gamma_l - \eta \cdot \frac{\partial E^{(k)}}{\partial \gamma_l}$$

其中 η 是学习率

- 伪代码

Input: training set: $\mathcal{D} = \{(\boldsymbol{x}^{(k)}, \boldsymbol{y}^{(k)})\}_{k=1}^{N}$
learning rate η
batch size b
maximum epoch E

Steps：
1: initialize all parameters within $(0,1)$
2: for $step$ from 1 to $N/b * E$:
3: randomly a batch of b samples $\mathcal{D}_b$ from $\mathcal{D}$
4: for each sample $(\boldsymbol{x}^{(k)}, \boldsymbol{y}^{(k)})$ in $\mathcal{D}_b$:
5: calculate $\widehat{\boldsymbol{y}}^{(k)}$ // Forward
6: for each sample $(\boldsymbol{x}^{(k)}, \boldsymbol{y}^{(k)})$ in $\mathcal{D}_b$:
7: update $\boldsymbol{\omega}$, $\boldsymbol{\theta}$, $\boldsymbol{v}$ and $\boldsymbol{\gamma}$ accumulatively //Backward
Output: trained FNN

不同的激活函数

Commonly Used Activation Functions

1. step function : $f(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$

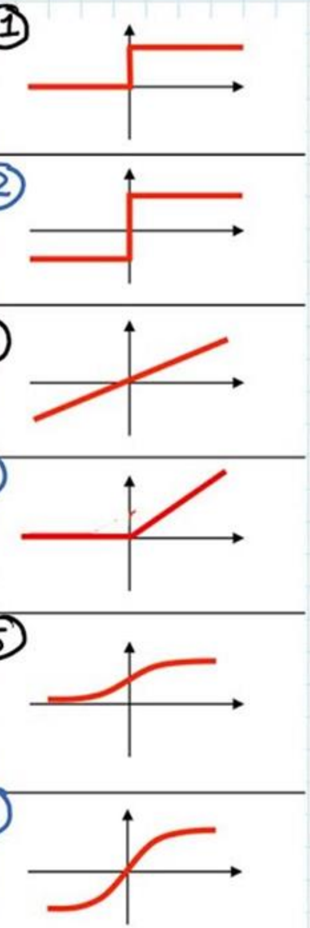
2. Signum function: $f(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$

3. Linear function: $f(z) = x$

4. ReLU function : $f(z) = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}$

5. Sigmoid function: $f(z) = \dfrac{e^x}{1+e^x}$

6. Hyperbolic tan : $\tanh(z) = \dfrac{e^z - e^{-z}}{e^z + e^{-z}}$

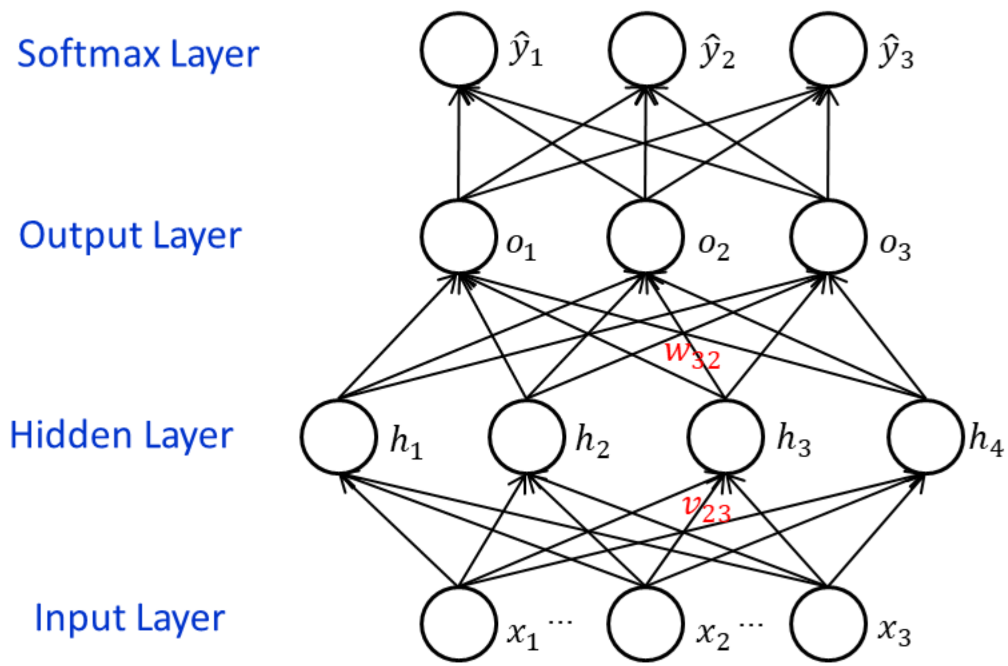


	Range
①	$\{0, 1\}$
②	$\{-1, 1\}$
③	$(-\infty, \infty)$
④	$(0, \infty)$
⑤	$(0, 1)$
⑥	$(-1, 1)$

by Dr. Pankaj Kumar Porwal (BTech - IIT Mumbai, PhD - Cornell University) : Principal, Techno India NJR Institute of Technology, Udaipur

课堂习题

- 基于三层前馈神经网络进行三分类问题建模，其中 $\boldsymbol{x} \in \mathbb{R}^3$、$\hat{\boldsymbol{y}} \in \mathbb{R}^3$、$\boldsymbol{h} \in \mathbb{R}^4$ 分别是输入、输出和隐藏层的向量表示。假设输入层到隐藏层为全连接，并以Sigmoid函数激活，隐藏层到输出层为全连接，并送入softmax函数得到各类后验概率输出。



$$\hat{y}_j = \frac{\exp o_j}{\sum_{j'=1}^{3} \exp o_{j'}}$$

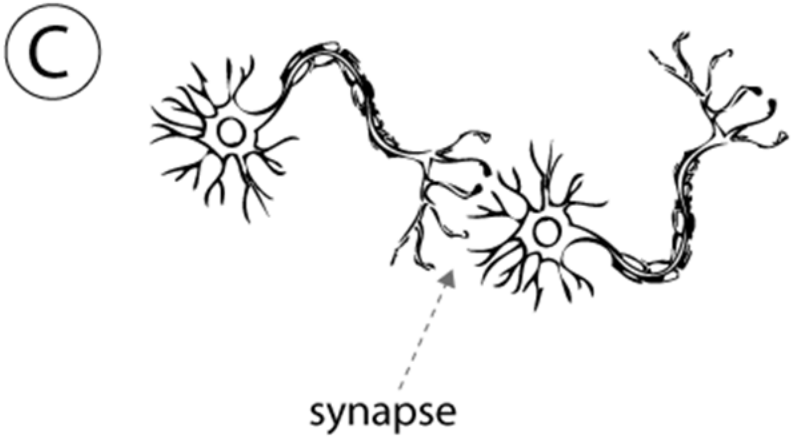
$$o_j = \sum_{l=1}^{4} w_{lj} h_l$$

$$h_l = \delta(a_l)$$

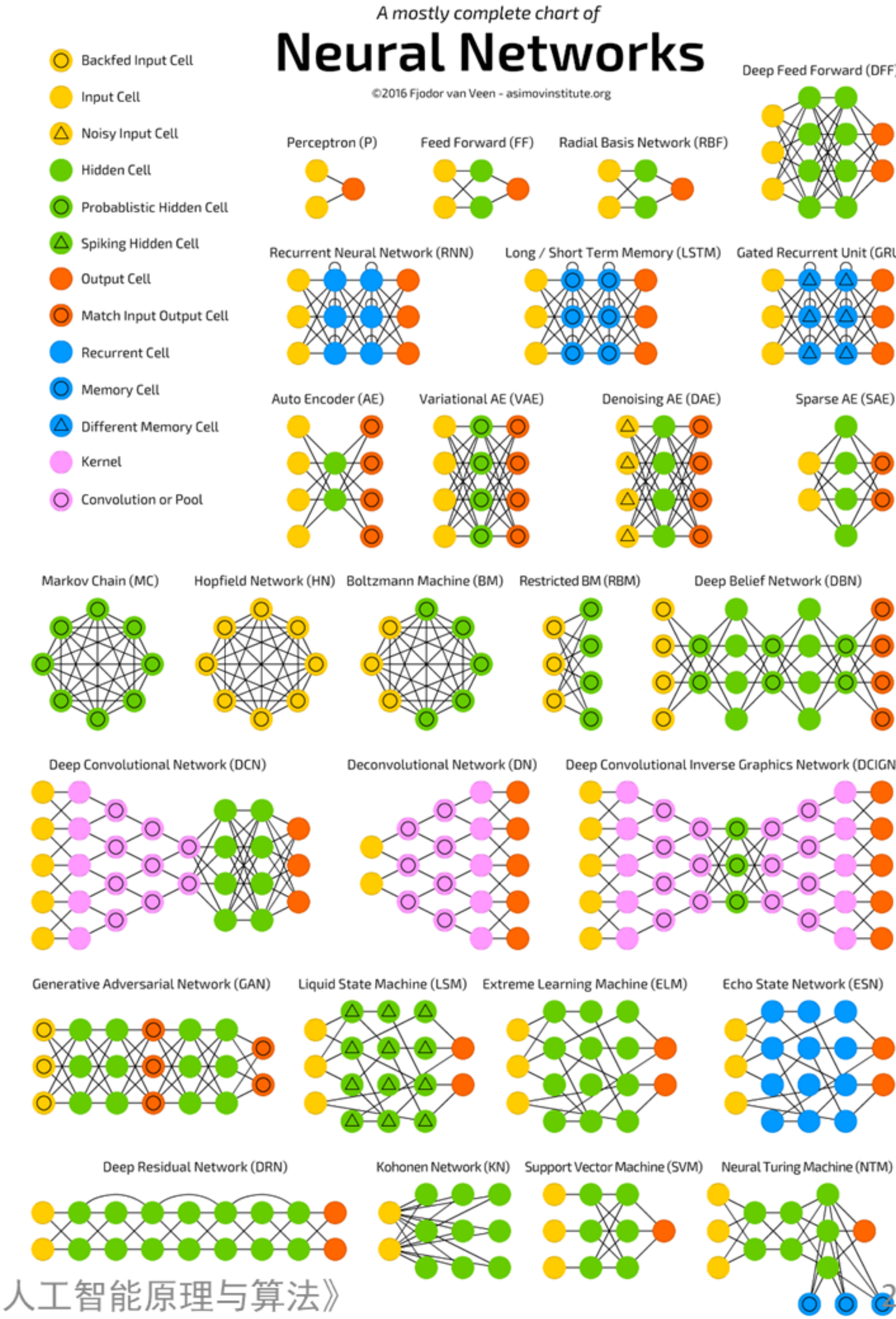
$$a_l = \sum_{i=1}^{3} v_{il} x_i$$

问题：试基于BP算法，推导交叉熵损失下的参数w_{32}和v_{23}的更新规则。

人工神经网络的发展

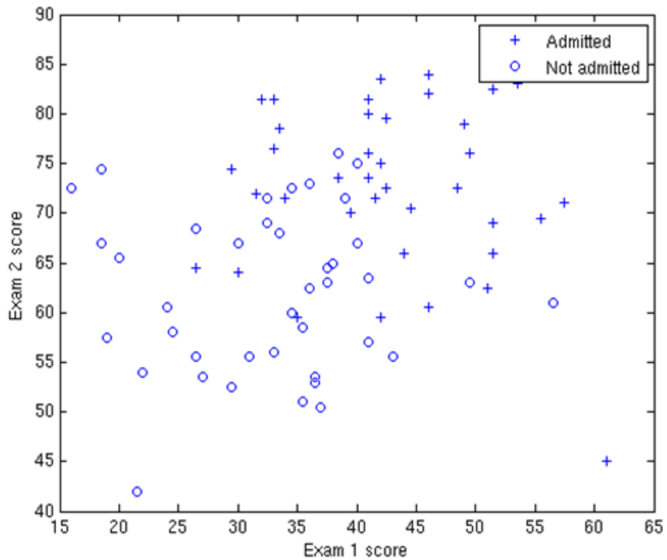


- 前向神经网络
- 卷积神经网络
- 递归神经网络
- 循环神经网络
- Transformer
- 预训练模型



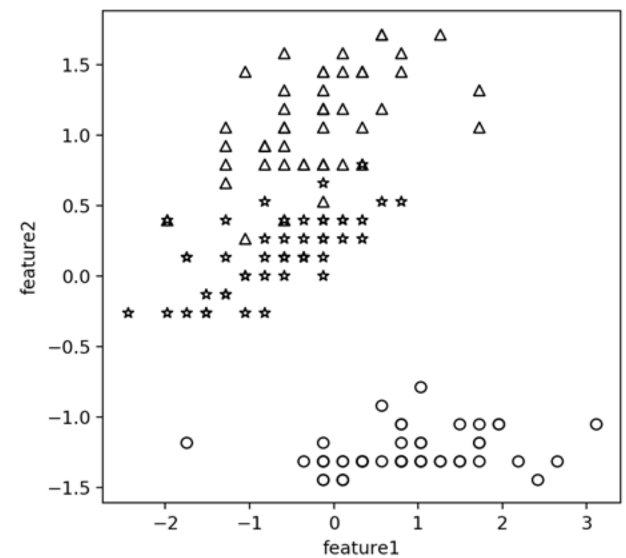
作业#5

(1) Binary Exam Dataset



http://www.nustm.cn/member/rxia/ml/data/Exam.zip

(2) Multi-class Iris Dataset



http://www.nustm.cn/member/rxia/ml/data/Iris.zip

- 分别针对数据集(1)和(2)，编程实现多层前向神经网络FNN，支持可变的网络层数和每层节点数，绘制损失函数下降曲线和动态分类边界，报告分类正确率；调节模型参数（隐层节点数、隐层数、激活函数、正则项、Softmax层及交叉熵损失等），观察实验结果变化。
- 基于深度学习框架（pytorch或tensorflow）实现上述任务，并与自己的编程结果进行比较。

本讲结束 欢迎提问