第6章 非线性规划

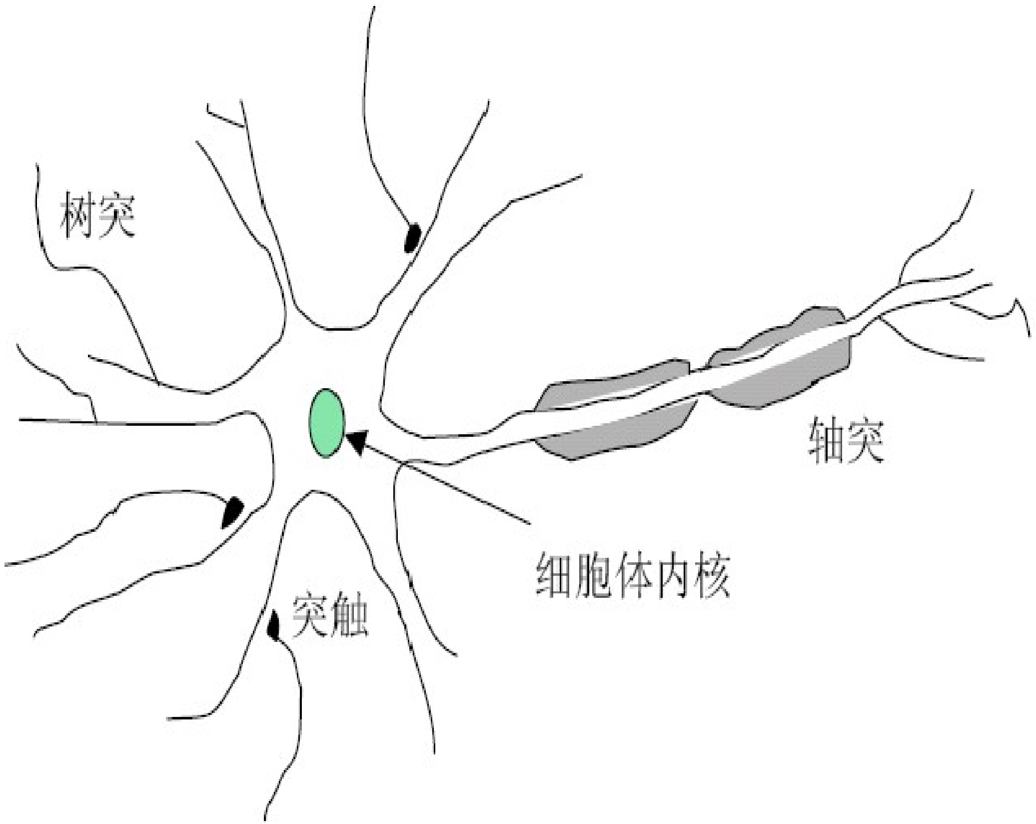
- 智能计算
  - 神经网络
  - 遗传算法

神经网络的概念

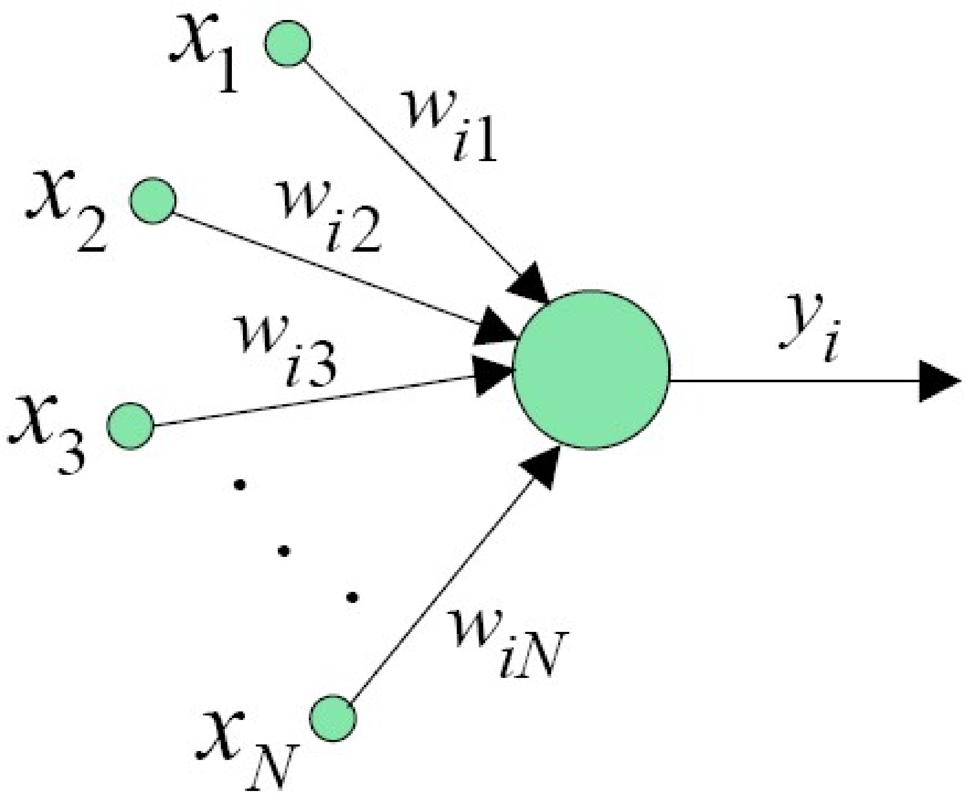
- 什么是神经网络？

- 模拟生物神经系统结构，实现特定功能的人工系统。
  - 数学模型
  - 计算机程序
  - 电子线路、集成芯片

神经元数学模型



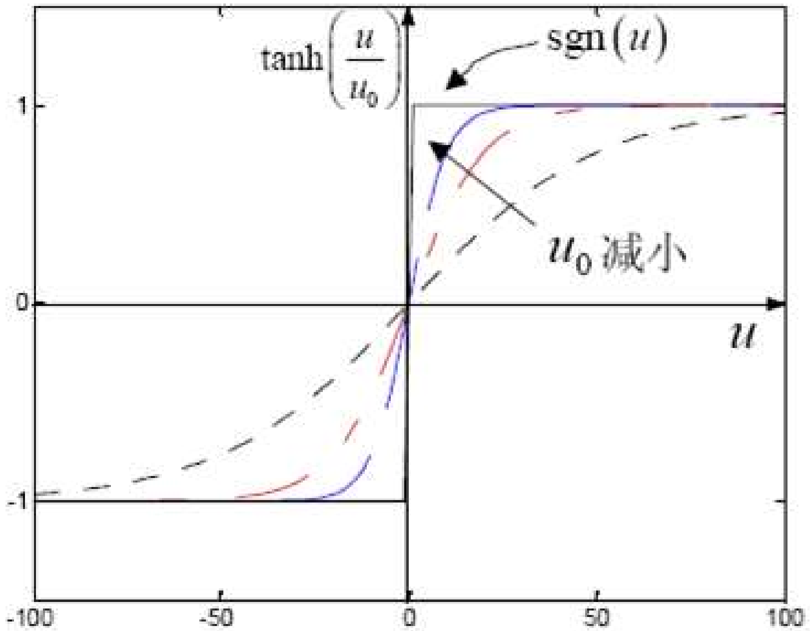
神经元结构



$$y_i = \Gamma(\sum_{j=1}^{N} w_{ij} x_j - \theta_i)$$

McCulloch-Pitts 模型

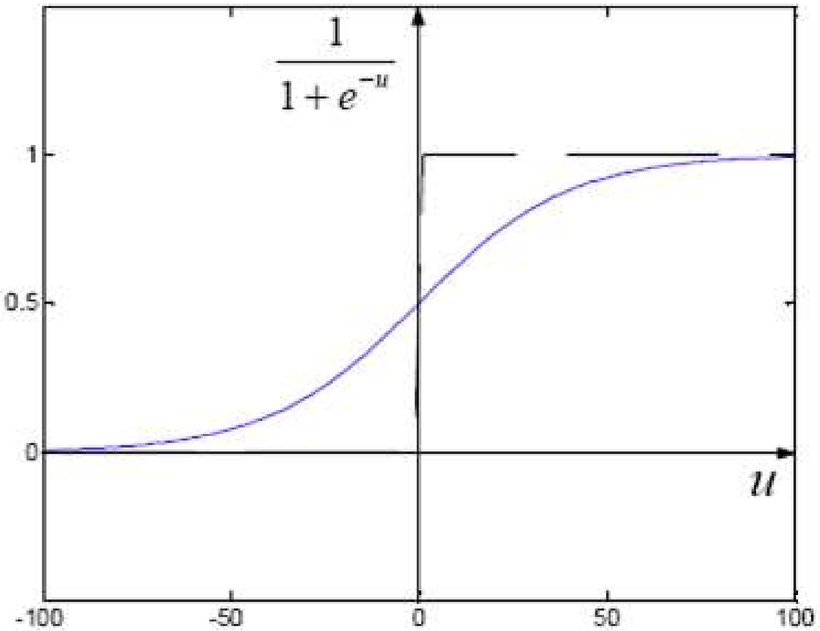
激励函数(Activation Function)



(a)

双曲正切函数

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



(b)

Sigmoid函数

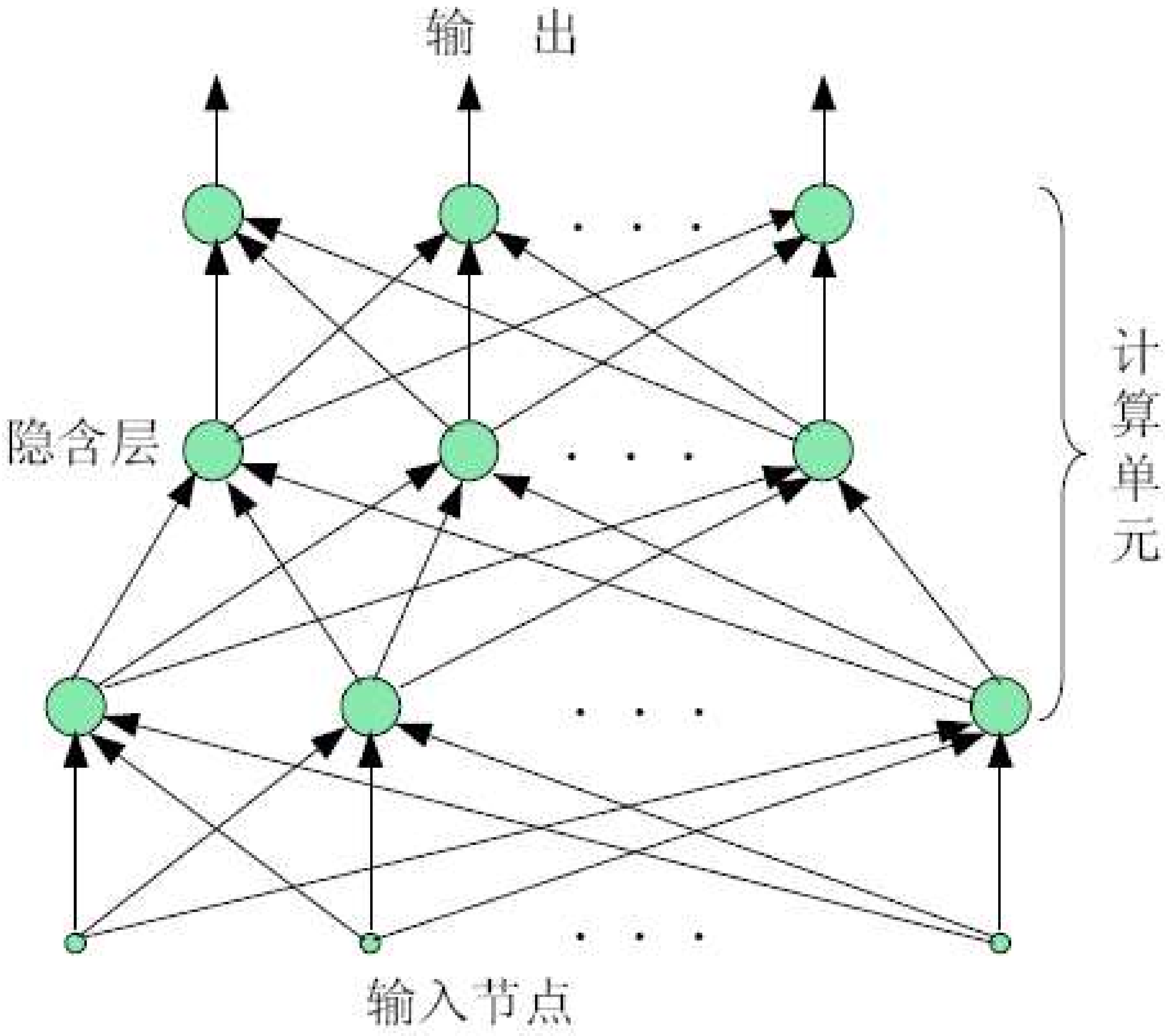
$$y(u) = \frac{1}{1+e^{u}}$$

神经网络的连接类型

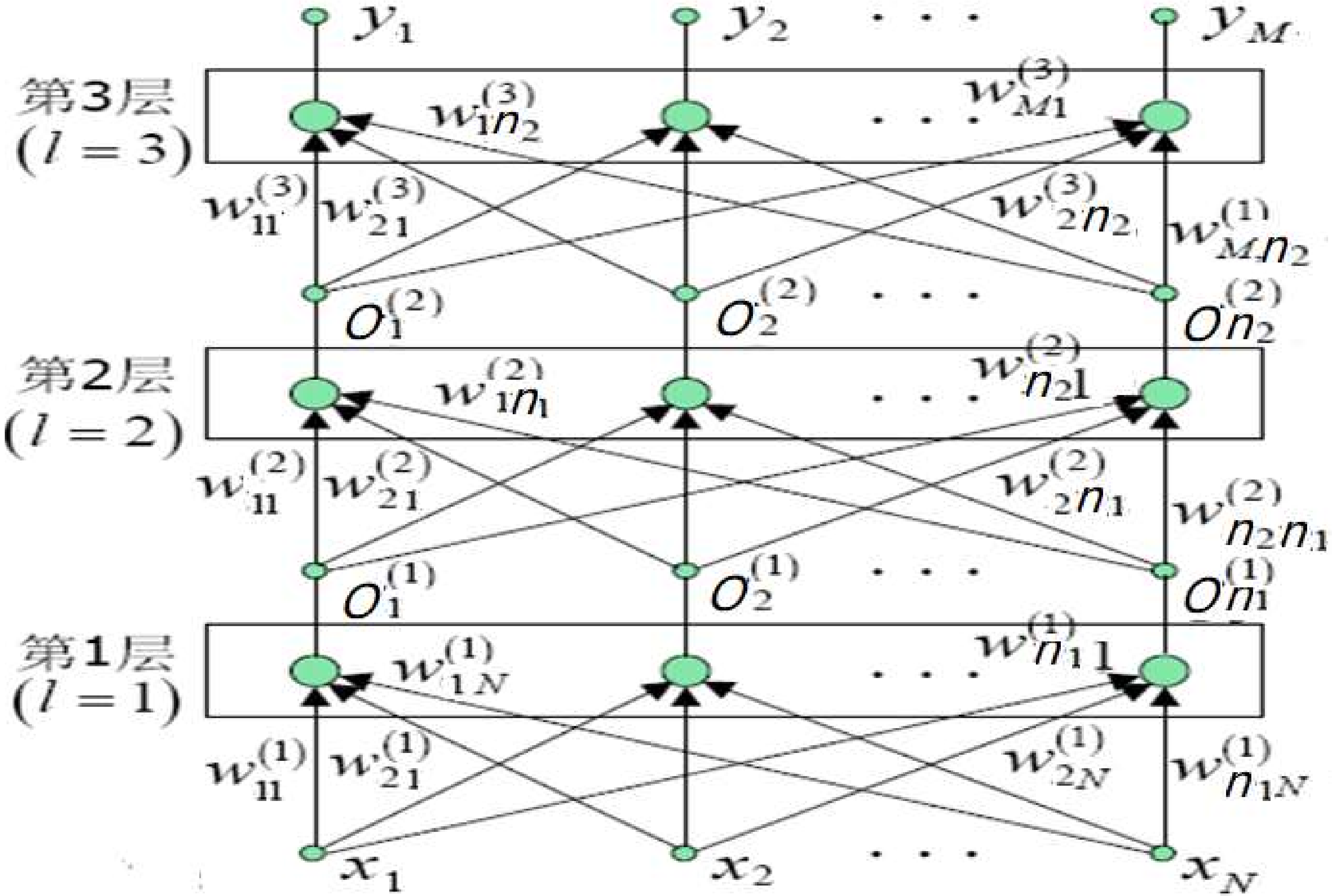
- 前馈网络：MLP(Multi-Layer Perceptron) 、BP NN
- 反馈网络：Hopfield NN，Recurrent NN
- 混合网络：ART、Kohonen映射

前馈网络结构

- 参数
  - 层数
  - 每层神经元的个数
  - 神经元的激励函数



具体结构



y_1 y_2 y_M
第3层 $(l=3)$
第2层 $(l=2)$
第1层 $(l=1)$
$w_{1n_2}^{(3)}$ $w_{M1}^{(3)}$ $w_{11}^{(3)}$ $w_{21}^{(3)}$ $w_{2n_2}^{(3)}$ $w_{Mn_2}^{(1)}$
$O_1^{(2)}$ $O_2^{(2)}$ $O_{n_2}^{(2)}$
$w_{1n_1}^{(2)}$ $w_{n_21}^{(2)}$ $w_{11}^{(2)}$ $w_{21}^{(2)}$ $w_{2n_1}^{(2)}$ $w_{n_2n_1}^{(2)}$
$O_1^{(1)}$ $O_2^{(1)}$ $O_{n_1}^{(1)}$
$w_{1N}^{(1)}$ $w_{n_11}^{(1)}$ $w_{11}^{(1)}$ $w_{21}^{(1)}$ $w_{2N}^{(1)}$ $w_{n_1N}^{(1)}$
x_1 x_2 x_N

网络模型

- 输入层

$$o_j^{(1)} = \Gamma_1(u_j^{(1)}) = \Gamma_1(\sum_{k=1}^{N} w_{jk}^{(1)} x_k) \qquad j = 1, 2 \ldots n_1$$

- 第l＋1个隐含层

$$u_j^{(l+1)} = \sum_{k=1}^{n_l} w_{jk}^{(l+1)} o_k^{(l)}$$

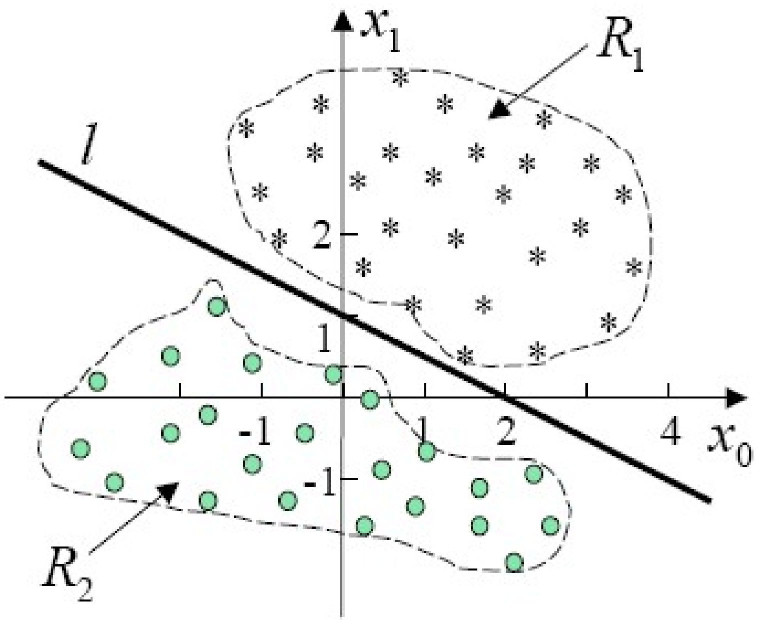
$$o_j^{(l+1)} = \Gamma_{l+1}(u_j^{(l+1)}) \qquad j = 1, 2 \ldots n_{l+1} \qquad l = 1, 2 \ldots L-1$$

- 输出层

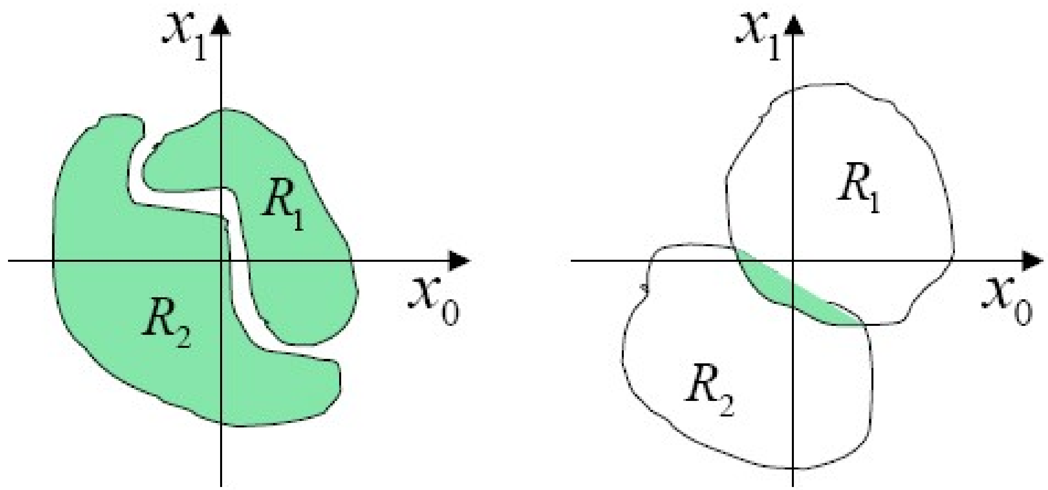
$$y_j = \Gamma_L(u_j^{(L)}) = \Gamma_L(\sum_{i=1}^{n_{L-1}} w_{ji}^{(L)} o_i^{(L-1)}) \qquad j = 1, 2 \ldots M$$

应用1：分类

- 分类

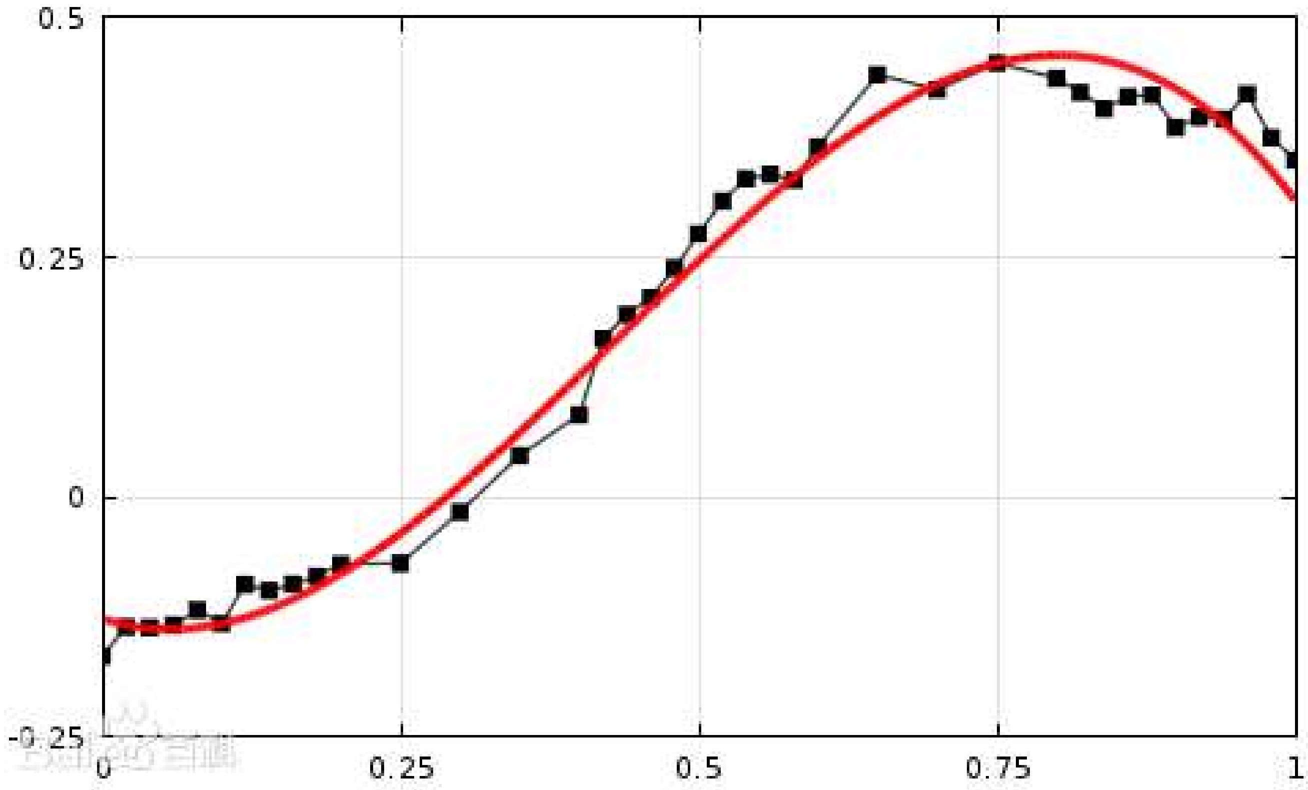


二维空间中线性可分类的示例



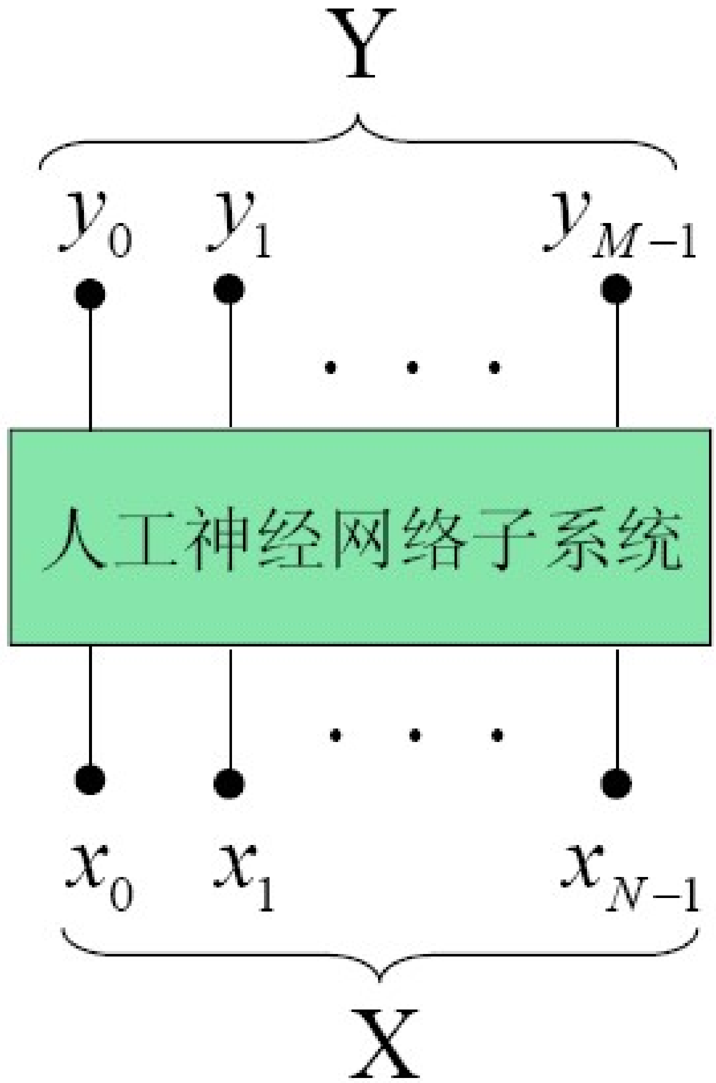
二维空间中非线性可分类的示例

应用2：函数拟合



有导师学习

- 目标：实现一个映射



- 有导师学习
  - 样本集
  - 测试集

- 应用阶段

目标函数/损失函数

- 批量拟合平方误差函数

$$E = \sum_{p=1}^{P} E_p$$

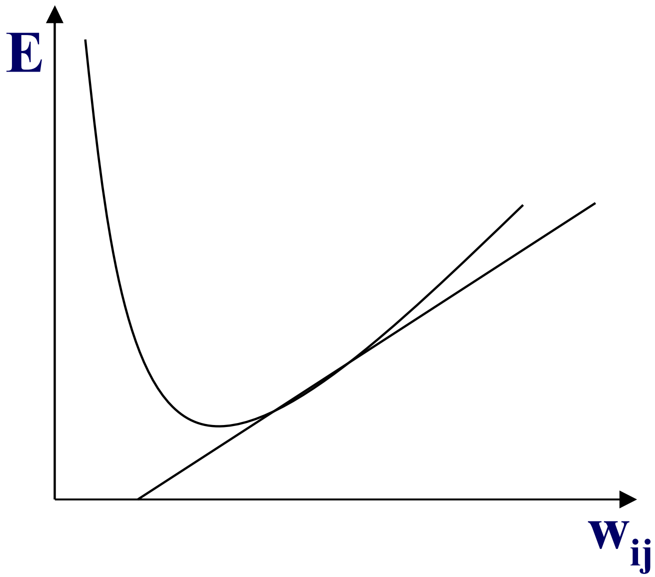
P为样本总数

- 单次拟合平方误差函数

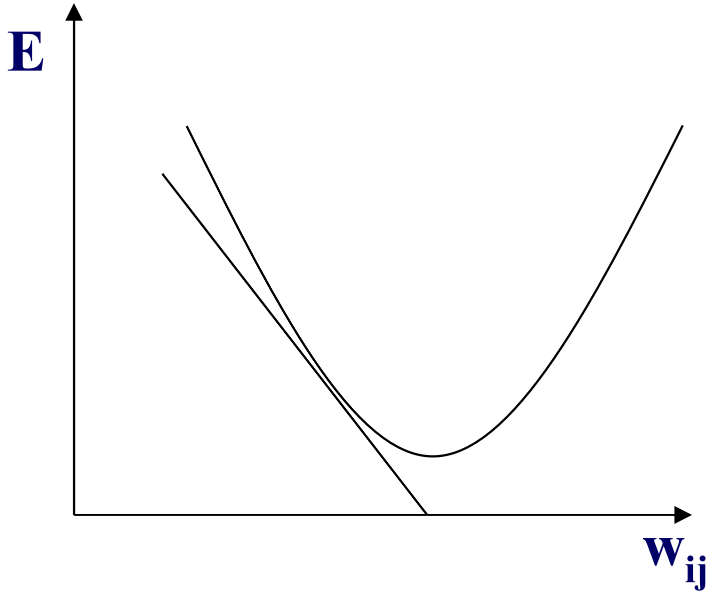
$$E_p = \frac{1}{2}\sum_{j=1}^{M}(t_{pj} - y_{pj})^2$$

t为教师值

梯度下降法



$\frac{\partial E}{\partial w_{ij}} > 0$，此时$\Delta w_{ij} < 0$



$\frac{\partial E}{\partial w_{ij}} < 0$, 此时$\Delta w_{ij} > 0$

$$w_{ji}^{(l)}(k+1) = w_{ji}^{(l)}(k) + \Delta_p w_{ji}^{(l)}(k) \qquad \Delta_p w_{ji}^{(l)} = -\eta \frac{\partial E_p}{\partial w_{ji}^{(l)}}$$

BP(Back Propagation)学习算法

- 采用梯度法：$\Delta_p w_{ji}^{(l)} = -\eta \dfrac{\partial E_p}{\partial w_{ji}^{(l)}}$　　$l = 1, 2, \ldots, L$
- 其中：

$$\frac{\partial E_p}{\partial w_{ji}^{(l)}} = \frac{\partial E_p}{\partial u_{pj}^{(l)}} \frac{\partial u_{pj}^{(l)}}{\partial w_{ji}^{(l)}}$$

$$\frac{\partial u_{pj}^{(l)}}{\partial w_{ji}^{(l)}} = \frac{\partial}{\partial w_{ji}^{(l)}} \sum_k w_{jk}^{(l)} o_{pk}^{(l-1)} = o_{pi}^{(l-1)}$$

- 定义广义误差：$\delta_{pj}^{(l)} = -\dfrac{\partial E_p}{\partial u_{pj}^{(l)}}$
- 可得：$\Delta_p w_{ji}^{(l)} = \eta \delta_{pj}^{(l)} o_{pi}^{(l-1)}$

反向误差传播

➢隐含层时，有：

$$\delta_{pj}^{(l)} = -\frac{\partial E_p}{\partial u_{pj}^{(l)}} = -\frac{\partial E_p}{\partial o_{pj}^{(l)}} \frac{\partial o_{pj}^{(l)}}{\partial u_{pj}^{(l)}}$$

$$= \left(\sum_k -\frac{\partial E_p}{\partial u_{pk}^{(l+1)}} \frac{\partial u_{pk}^{(l+1)}}{\partial o_{pj}^{(l)}} \right) \Gamma_l'(u_{pj}^{(l)})$$

$$= \sum_k (\delta_{pk}^{(l+1)} \cdot w_{kj}^{(l+1)}) \cdot \Gamma_l'(u_{pj}^{(l)}) \qquad l = 1, 2, \ldots, L-1$$

➢输出层时，有：

$$\delta_{pj}^{(L)} = -\frac{\partial E_p}{\partial u_{pj}^{(L)}} = -\frac{\partial E_p}{\partial y_{pj}} \frac{\partial y_{pj}}{\partial u_{pj}^{(L)}} = (t_{pj} - y_{pj}) \Gamma_L'(u_{pj}^{(L)})$$

权值更新

$$\Delta_p w_{ji}^{(l)} = \eta \delta_{pj}^{(l)} o_{pi}^{(l-1)} \qquad l = 1, 2, \ldots, L$$

η为学习率

$$w_{ji}^{(l)}(k+1) = w_{ji}^{(l)}(k) + \Delta_p w_{ji}^{(l)}(k)$$

单样本学习

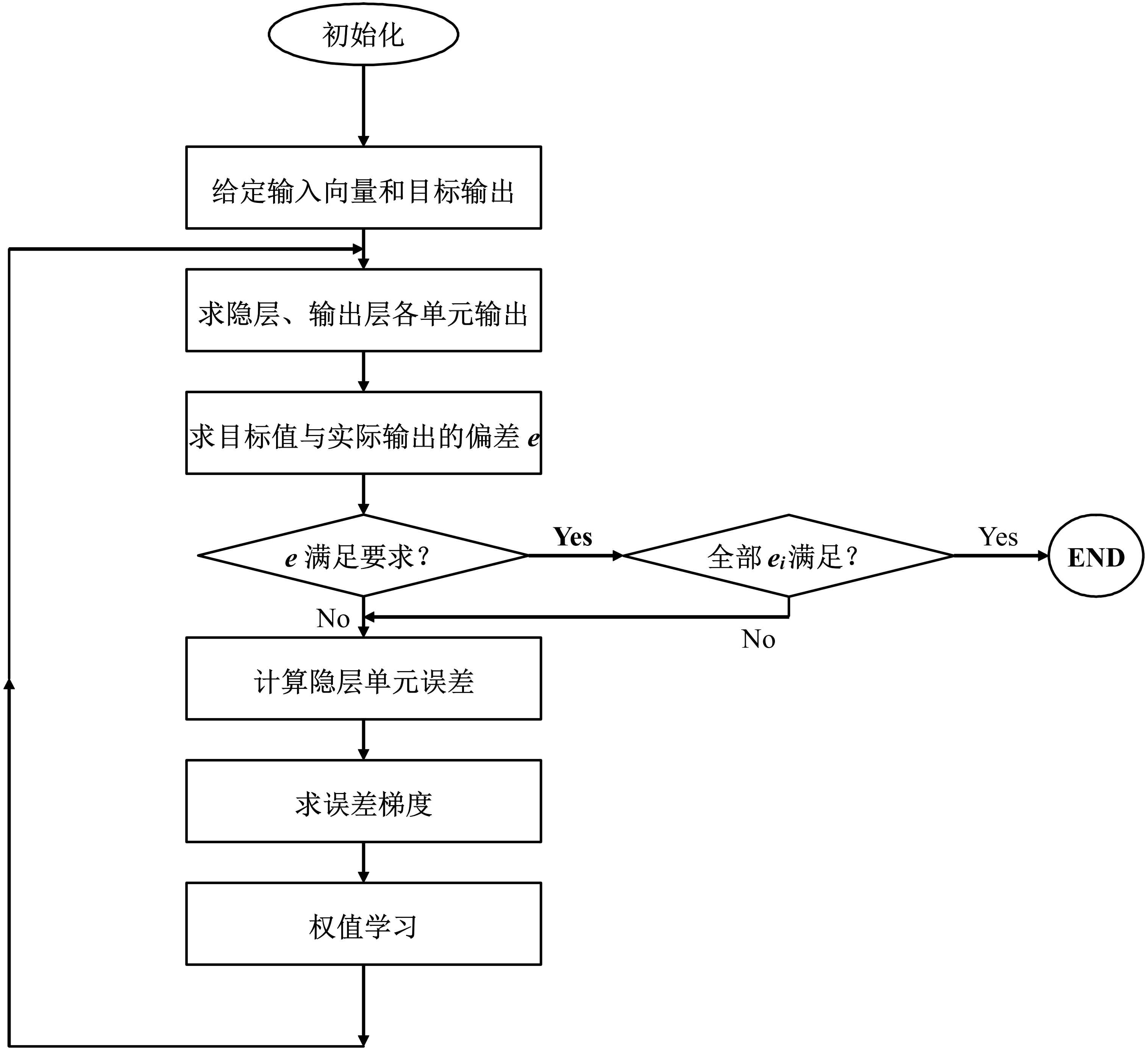
$$w_{ji}^{(l)}(k+1) = w_{ji}^{(l)}(k) + \sum_{p=1}^{P} \Delta_p w_{ji}^{(l)}(k)$$

批量样本学习

P为批量大小

训练方法

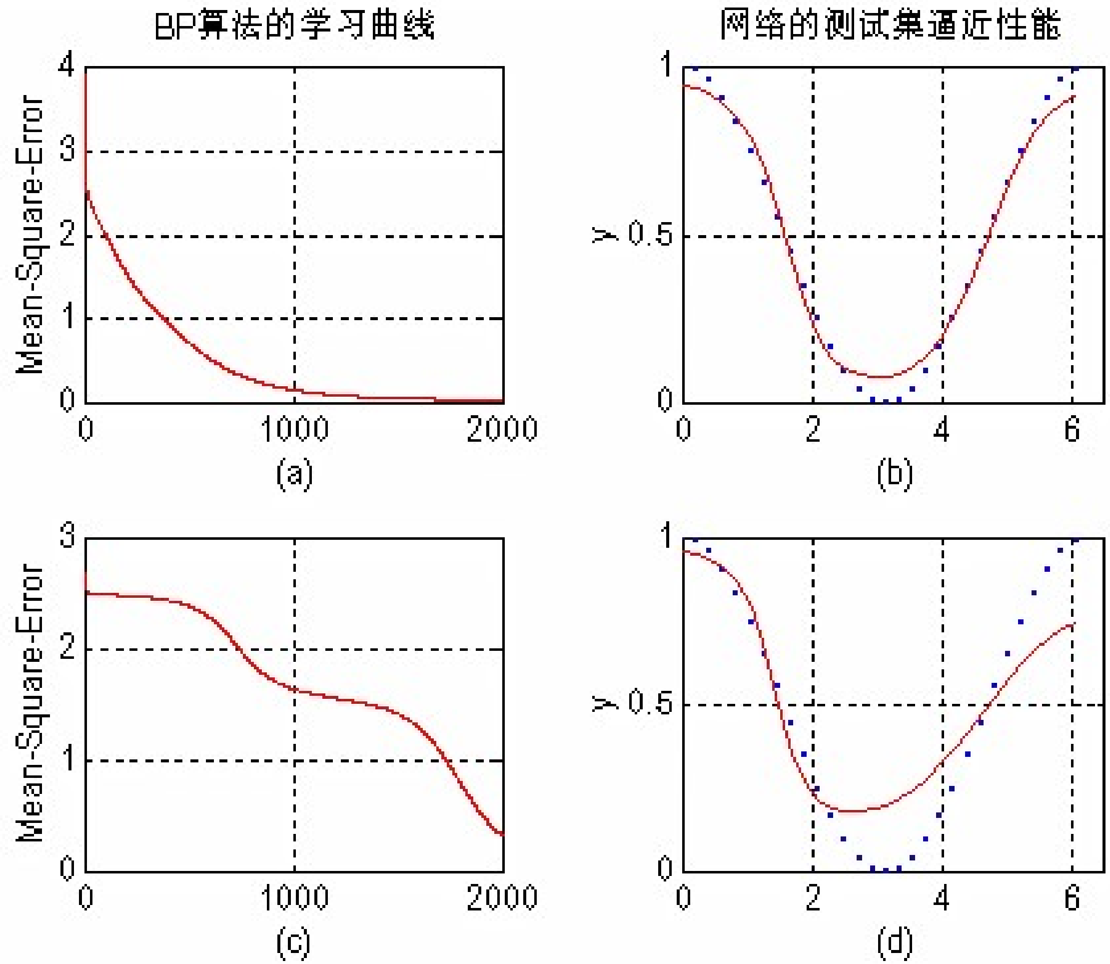
- 批量梯度下降法(Batch Gradient Decent, BGD)，*P*为所有样本数。学习精度最高，容易过拟合，速度慢，不适合大批量训练。
- 随机梯度下降法（Stochastic Gradient Decent, SGD），P=1，每次只使用1个随机选出的样本，也称为增量梯度下降法。在目标函数为严格凸函数时，多轮训练的情况下，可证明在期望意义下是收敛的。
- 小批量梯度下降法（mini-Batch Gradient Decent，MBGD），P为总样本中随机选出的1个到数百个样本，前两种方法的折中，是目前大规模网络学习的主流算法。



初始化
给定输入向量和目标输出
求隐层、输出层各单元输出
求目标值与实际输出的偏差 e
e 满足要求？
Yes
全部 e_i满足？
Yes
END
No
No
计算隐层单元误差
求误差梯度
权值学习

BP网络的逼近能力

- 三层网络可以任意精度逼近连续非线性函数



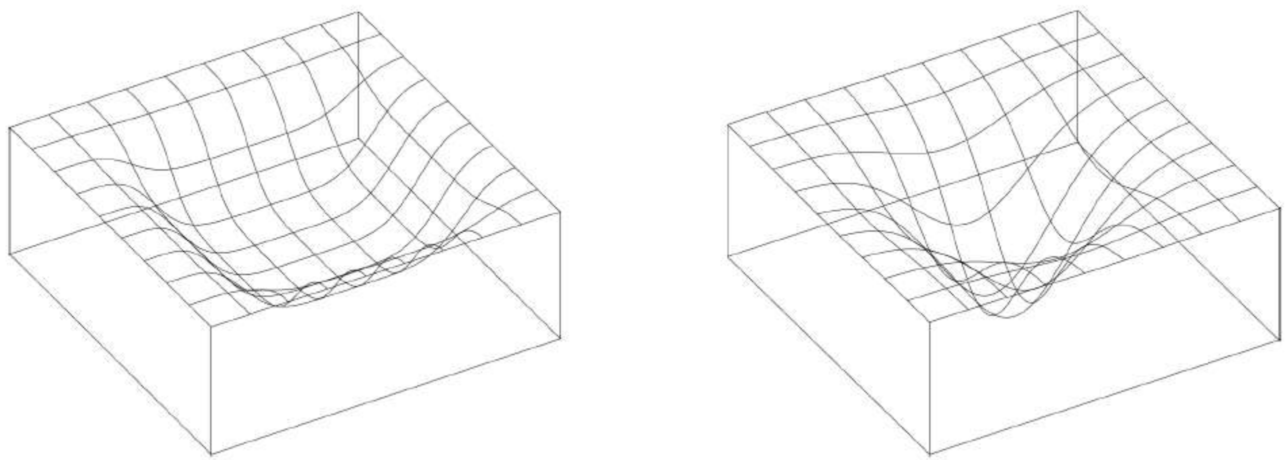
单隐层

双隐层

y=0.5*(1+cos(x))

神经网络的优化问题

- 目标函数存在大量驻点
- 大规模网络，不同极值点从建模角度等价
- 尖锐的极小值、全局最优解的网络性能反而不一定好。（过拟合现象）



(a) 平坦最小值　　(b) 尖锐最小值

梯度校正

- 梯度振荡： $$\Delta_p w_{ji}^{(l)} = -\eta \frac{\partial E_p}{\partial w_{ji}^{(l)}} = \eta \delta_{pj}^{(l)} o_{pi}^{(l-1)}$$
- 对策：增加惯量矩（Momentum)：减小振荡，加速学习

$$\Delta w_{ji}^{(l)}(k) = \eta \delta_{pj}^{(l)} o_{pi}^{(l-1)} + \alpha \Delta w_{ji}^{(l)}(k-1)$$

- 梯度消失和梯度爆炸：随着层数增加，前面隐含层容易出现梯度过小或过大的现象，导致学习缓慢或网络不稳定。
- 对策：改变激励函数（ReLU）、权值正则化（损失函数中增加权值大小的惩罚项）、梯度截断（设置梯度阈值）、数据归一化…

$$\delta_{pj}^{(l)} = -\frac{\partial E_p}{\partial u_{pj}^{(l)}} = \sum_k (\delta_{pk}^{(l+1)} \cdot w_{kj}^{(l+1)}) \cdot \Gamma_l'(u_{pj}^{(l)}) \qquad o_j^{(l-1)} = \Gamma_{l-1}(u_j^{(l-1)})$$

学习率设计原则

- 学习速率越大，收敛越快，但容易产生振荡；而学习速率越小，收敛越慢。 可以采用随时间衰减的学习率。
- 小批量学习时，因为学习开始时的梯度比较大且不稳定，一般先选择比较小的学习率，等梯度比较稳定后，再改用正常的学习率。称为学习率预热。
- 周期性地增大学习率，走出当前局部极小值，探索其他区域。
- 梯度累积值很大时，采用较小的学习率，防止在一个方向走得太远。

常见算法

- AdaGrad（Adaptive Gradient）

  目的：自适应调整学习率，适合凸函数下的寻优

$$\Delta \boldsymbol{w}(k) = -\frac{\eta}{\sqrt{n_k + \varepsilon}} \boldsymbol{g}_k \qquad \boldsymbol{g}_k = \frac{1}{P}\sum_{p=1}^{P} \frac{\partial E_p}{\partial \boldsymbol{w}} \qquad n_k = n_{k-1} + \boldsymbol{g}_k \odot \boldsymbol{g}_k$$

$$\varepsilon = 10^{-7} \qquad n_0 = 0$$

累积平方梯度

- RMSProp (Root Mean Square propagation)

  目的：防止Adagrad算法训练后期步长过小，适合非凸情形

$$n_k = \rho n_{k-1} + (1-\rho)\boldsymbol{g}_k \odot \boldsymbol{g}_k \qquad \rho = 0.9$$

梯度二阶矩的指数加权平均估计

- Adam: Momentum+RMSProp +偏差修正

  目的：修正指数加权平均开始阶段值偏小的情况 $n_k' = \dfrac{n_k}{1-\rho^k}$

应用举例

- 计算机视觉CV
- 自然语言处理NLP
- 系统建模
- 直觉推理
- 知识表达

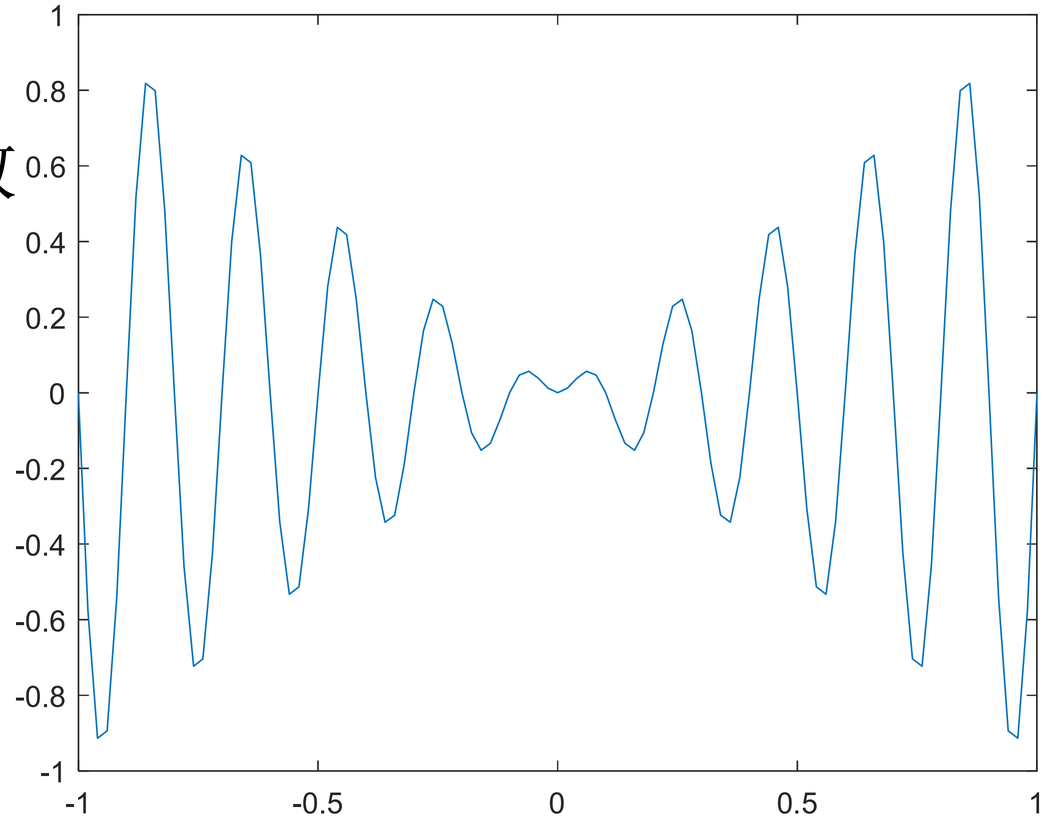
遗传算法

- 美国的J. Holland教授于1975年提出

- 模拟生物染色体的运作（复制、交叉、变异），提出的一种随机化搜索算法

目的

- 解决一个优化问题
- 描述方法：适应度函数
- 例：求函数极值



$$f(x) = x \cdot \sin(10\pi \cdot x)$$

求解方法

- 随机化搜索
- 种群搜索
- 编码（常见的为二进制）

  例：若前面的函数极值问题要精确到6位小数

  $$2^{21} < 3 \times 10^{6} < 2^{22}$$

几个术语

个体（染色体）

➢基因型：1000101110110101000111

基因

解码

编码

- 表现型：0.637197

染色体的运算

- 复制
- 交叉
- 变异

复制的选择概率

- 轮盘赌选择，它的基本思想是：各个个体被选中的概率与其适应度函数值大小成正比。设群体大小为n，个体i 的适应度为 F_i，则个体i 被选中遗传到下一代群体的概率为：

$$P_i = F_i / \sum_{i=1}^{n} F_i$$

交叉

交叉点

- 交叉前：
- 00000|011100000000010000
- 11100|000001111111000101
- 交叉后：
- 00000|000001111111000101
- 11100|011100000000010000

变异

变异点

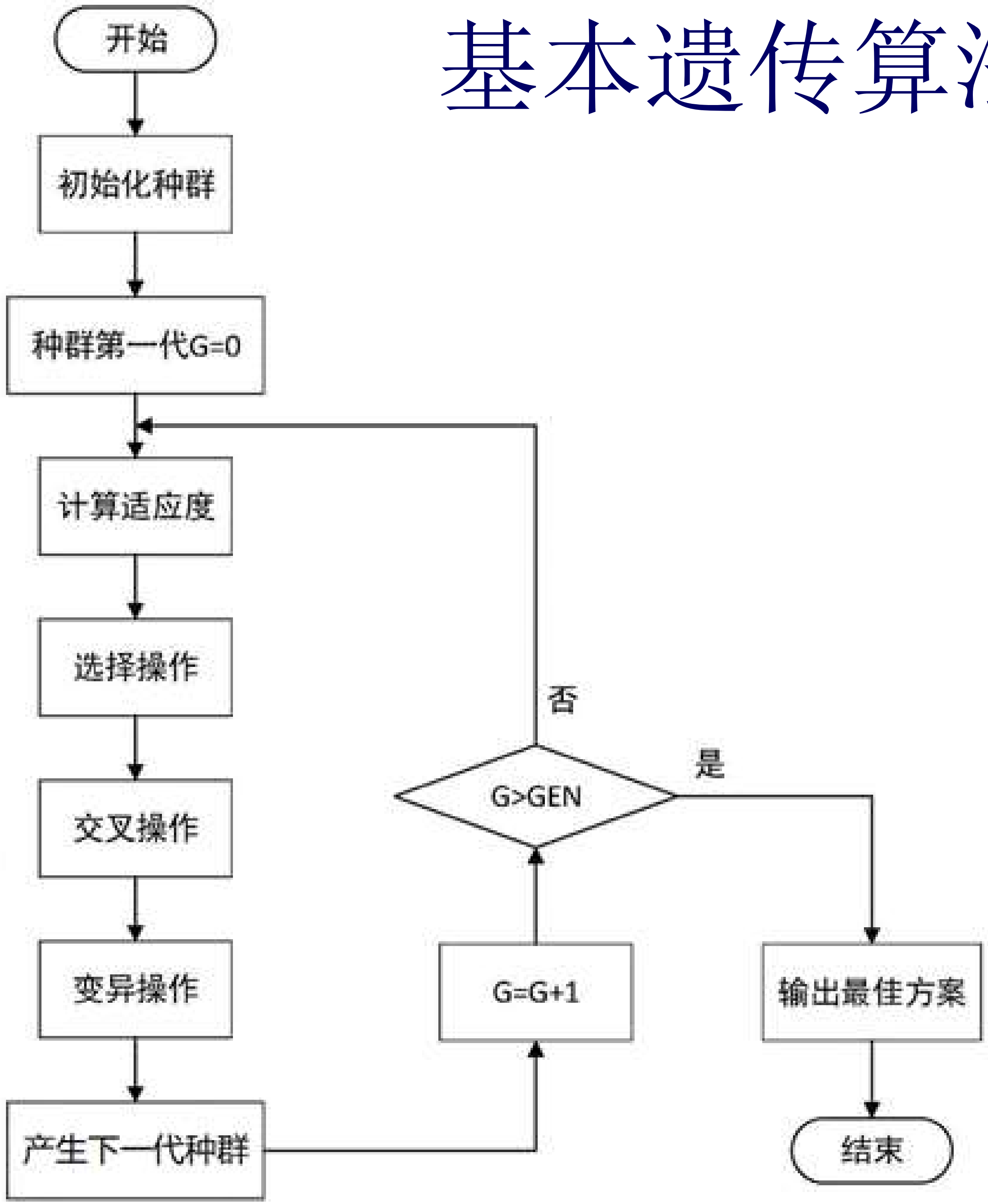
- 变异前：
- 000001110000<u>0</u>000010000
- 变异后：
- 000001110000<u>1</u>000010000

具体参数

- P_c： 交叉概率
- P_m： 变异概率

- M ： 种群规模
- T ： 遗传运算的终止进化代数

基本遗传算法流程



开始
初始化种群
种群第一代G=0
计算适应度
选择操作
交叉操作
变异操作
产生下一代种群
G=G+1
G>GEN
否
是
输出最佳方案
结束

遗传算法的特点

- 群体搜索，易于并行化处理；
- 不是盲目穷举，而是启发式搜索；
- 适应度函数不受连续、可微等条件的约束，适用范围很广。

- 问题：找到的是否是全局最优解？

Exploration and Exploitation

- 引入扰动的概率选择
  - $P_c = 0.4\text{--}0.9$，
  - $P_m = 0.001\text{-}0.01$

- 改进算法
  - 保护已寻得的最好解

收敛性证明

- 模式定理：具有低阶、短定义距以及平均适应度高于种群平均适应度的模式在子代中呈指数增长。
  - 阶：确定基因的个数，例： O(10**1)=3
  - 定义距：确定基因间的最大距离，例：δ(10**1)=4

- Markov链分析

类似的算法

- 蚁群算法（Ant Colony Optimization， ACO）
- 粒子群算法（Particle Swarm Optimization， PSO）
- 免疫算法（Immune Algorithm， IA）
- 。。。