2020.2.18 - 2020.6.2 9:50-12:15@6C102雨课堂+腾讯会议
《模式识别与机器学习》

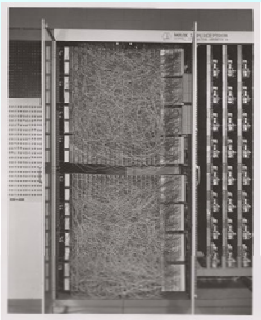
第四章
人工神经网络

2020.3.3

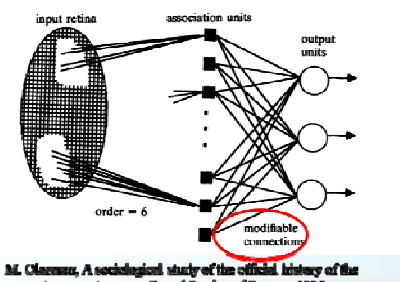
感知器

- 为什么把它叫做学习机器？
  - ① 因为它是一台机器
  - ② 因为它会学习！
  - －它不是编好程序的冯诺依曼计算机，是一台根据训练数据自我调整的学习机器

$$y = \mathrm{sgn}\left(\sum_{i=1}^{n} w_i x_i + w_0\right)$$



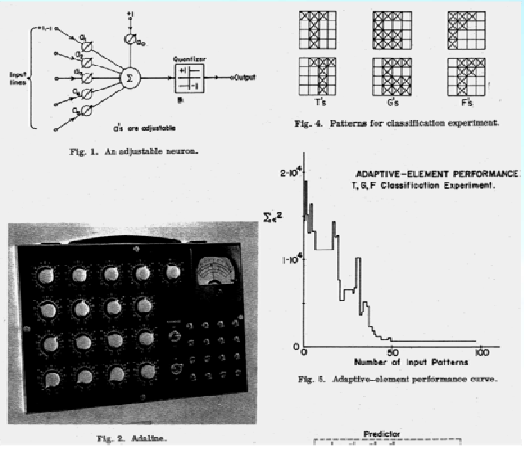
https://en.wikipedia.org/wiki/Perceptron



M. Olazaran, A sociological study of the official history of the perceptron controversy, *Social Studies of Science*, 1996

ADALINE

Widrow & Hoff, Adaptive switching circuits, *1960 IRE Western Electric Show and Convention Record, Part 4*, pp.96-104, Aug, 1960



Widrow & Hoff, Adaptive switching circuits, *1960 IRE Western Electric Show and Convention Record, Part 4*, pp.96-104, Aug, 1960

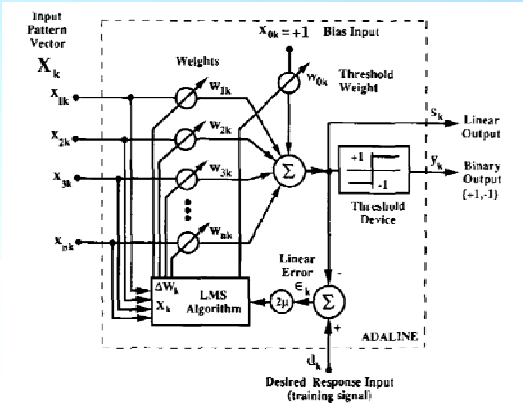
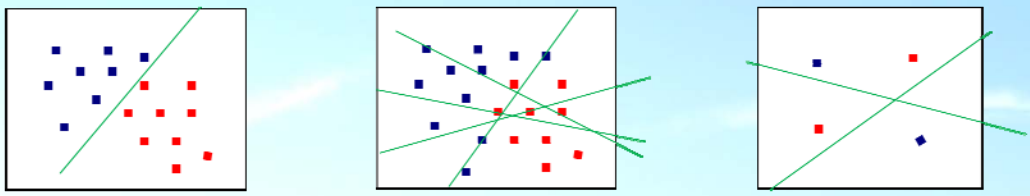


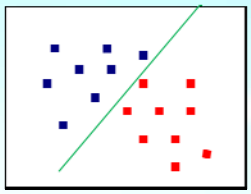
Fig. 2. Adaptive linear element (Adaline).

Widrow & Lehr, 30 years of adaptive neural networks: Perceptron, Madaline, and Backpropagation, *Proceedings of the IEEE*, 78(9): 1415-1442, 1990

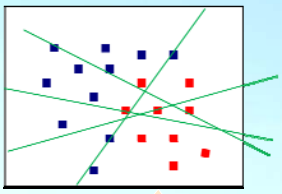
线性不可分情况怎么办？



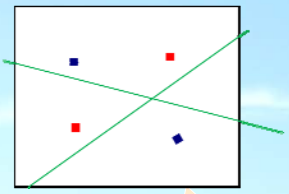
线性不可分情况怎么办？



线性可分



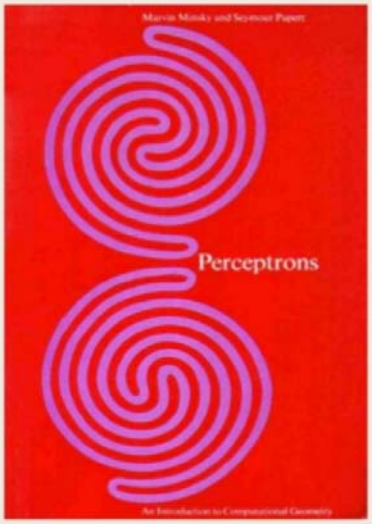
线性不可分
但错误可容忍



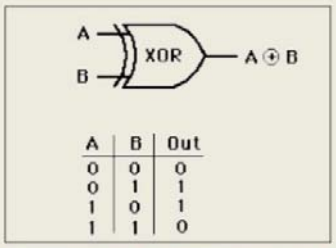
线性不可分
且无调和余地

对感知器（乃至整个机器学习）的质疑

1969: Perceptrons can't do XOR!



http://www.i-programmer.info/images/stories/BabBag/AI/book.jpg



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

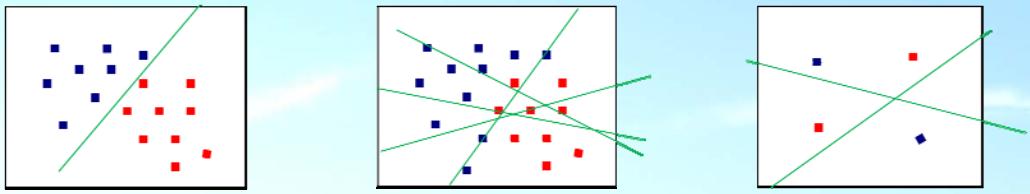
http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/ietron/xor.gif

Minsky & Papert

https://constructingkids.files.wordpress.com/2013/05/minsky-papert-71-csolomon-x640.jpg

https://pmirla.github.io/2016/08/16/AI-Winter.html

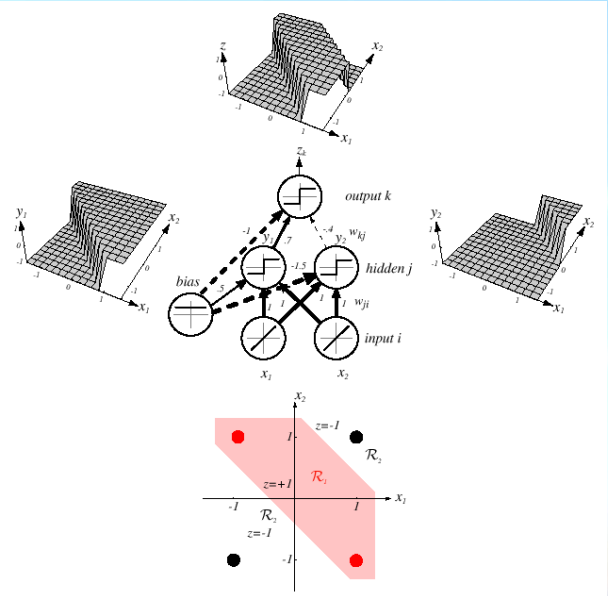
线性不可分情况怎么办？



有什么办法？ D

请弹幕回答

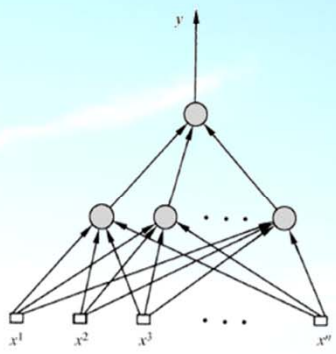
用多个感知器构造非线性分类器

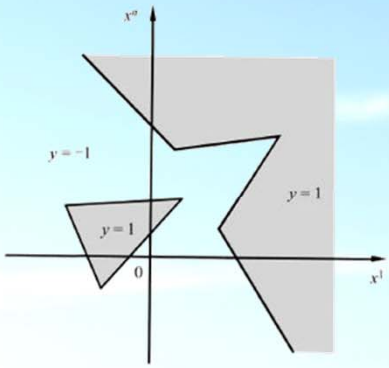


分段线性分类的思想

Duda, Hart & Stork, Pattern Classification (2nd edition), John Wiley & Sons, 2001

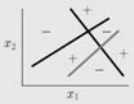
早期的多层感知器（神经网络）





用感知器布尔分解非线性函数

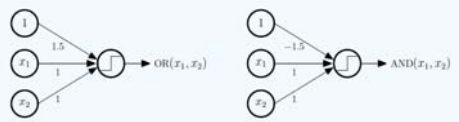
Consider a target function f whose '+' and '−' regions are illustrated below.

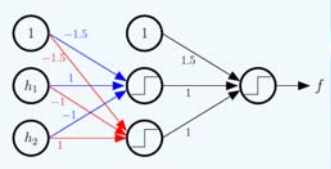


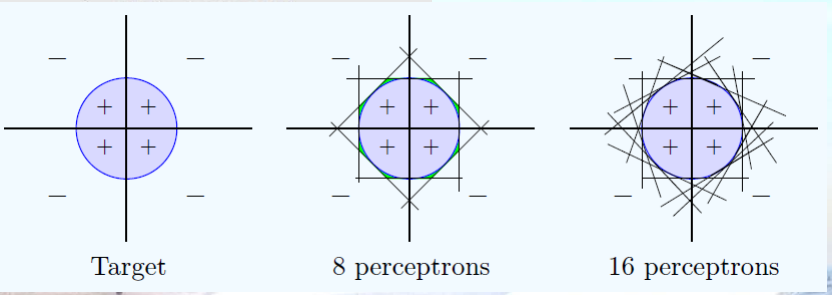
The target f has three perceptron components h_1, h_2, h_3:

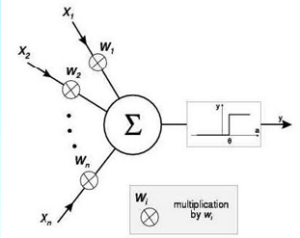
Show that

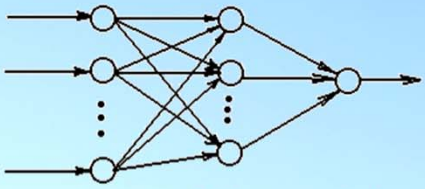
$$f = \overline{h_1}h_2h_3 + h_1\overline{h_2}h_3 + h_1h_2\overline{h_3}.$$











input layer 输入层　hidden layer 隐层　output layer 输出层

非线性判别函数

$$g_k(\mathbf{x}) \equiv y_k = \mathrm{Sign}\left(\sum_j w_{kj}\mathrm{Sign}\left(\sum_i w_{ji}x_i + w_{j0}\right) + w_{k0}\right)$$

机器学习的基本要素：感知器版

- 怎样造一个学习机器？
  - 它需要老师
    - → 我们设计它（特征和模型） $y = \mathrm{sgn}(\sum_{i=1}^{d} w_i x_i + w_0)$
  - 它需要训练/学习材料
    - → 训练数据 $\{(x_1, y_1), \ldots, (x_N, y_N)\}$; $x_j \in R^{d+1}, y_j \in \{-1, 1\}$
  - 我们需要为它树立学习的目标
    - → 目标函数、学习准则 $\min J_P(\alpha) = \sum_{y_j \alpha^T y_j < 0}(-\alpha^T y_j)$
  - 我们需要告诉它怎样学
    - → 学习/训练算法 $\alpha(k+1) = \alpha(k) - \rho_k \nabla J = \alpha(k) + \rho_k \sum_{y_j \in Y^k} y_j$

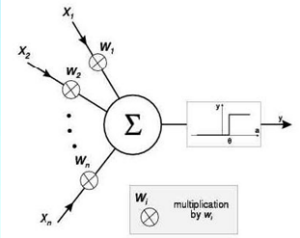
机器学习的基本要素：罗杰斯特回归版

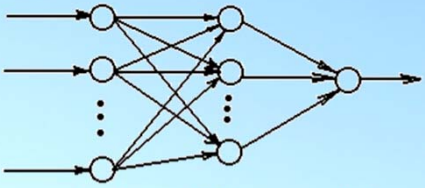
- 怎样造一个学习机器？
  - 它需要老师
    - → 我们设计它（特征和模型） $h(x) = \theta(w^T x)$
  - 它需要训练/学习材料
    - → 训练数据 $\{(x_1, y_1), \ldots, (x_N, y_N)\}$; $x_j \in R^{d+1}, y_j \in \{-1, 1\}$
  - 我们需要为它树立学习的目标
    - → 目标函数、学习准则 $\min E(w) = -\frac{1}{N}\sum_{j=1}^{N} \ln\left(1 + e^{-y_j w^T x_j}\right)$
  - 我们需要告诉它怎样学
    - → 学习/训练算法 $w(k+1) = w(k) - \eta \nabla E$

机器学习的基本要素：线性回归版

- 怎样造一个学习机器？
  - 它需要老师
    - → 我们设计它（特征和模型） $f(x) = \sum_{i=0}^{d} w_i x_i = w^T x$
  - 它需要训练/学习材料
    - → 训练数据 $\{(x_1, y_1), \ldots, (x_N, y_N)\}$; $x_j \in R^{d+1}, y_j \in R$
  - 我们需要为它树立学习的目标
    - → 目标函数、学习准则 $\min E = \frac{1}{N}\sum_{j=1}^{N}\left(f(x_j) - y_j\right)^2$
  - 我们需要告诉它怎样学
    - → 学习/训练算法 $w(k+1) = w(k) - \rho_k \nabla E$

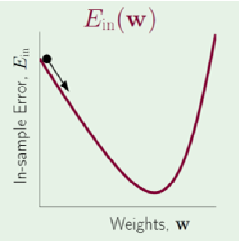
多层感知器版？ D





非线性判别函数

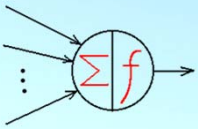
$$g_k(\mathbf{x}) \equiv y_k = \text{Sign}\left(\sum_j w_{kj}\text{Sign}\left(\sum_i w_{ji}x_i + w_{j0}\right) + w_{k0}\right)$$

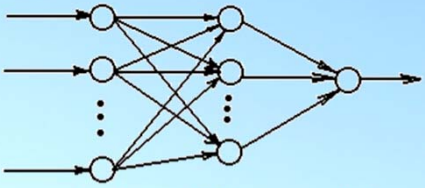


- 梯度下降，老套路还行吗?
  - 无法对中间权值求梯度

D

怎么办?



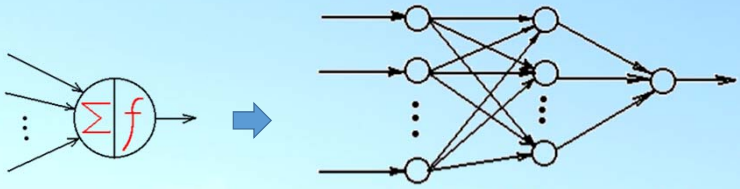


f: 激活函数/响应函数

非线性判别函数

$$g_k(\mathbf{x}) \equiv y_k = f\left(\sum_j w_{kj}f\left(\sum_i w_{ji}x_i + w_{j0}\right) + w_{k0}\right)$$

怎么办？



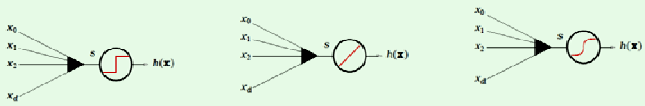
f: 激活函数/响应函数

非线性判别函数

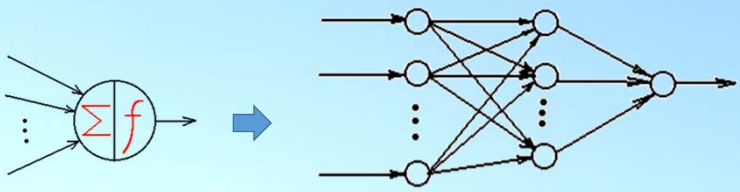
$$g_k(\mathbf{x}) \equiv y_k = f\left(\sum_j w_{kj} f\left(\sum_i w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$$

$$s = \sum_{i=0}^{d} w_i x_i$$

linear classification	linear regression	logistic regression
$h(\mathbf{x}) = \text{sign}(s)$	$h(\mathbf{x}) = s$	$h(\mathbf{x}) = \theta(s)$



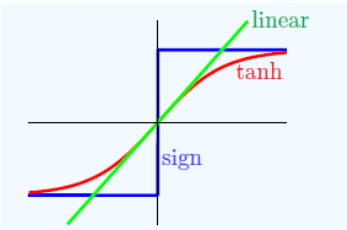
怎么办？



f: 激活函数/响应函数

非线性判别函数

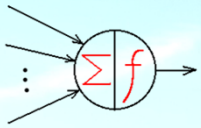
$$g_k(\mathbf{x}) \equiv y_k = f\left(\sum_j w_{kj} f\left(\sum_i w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$$

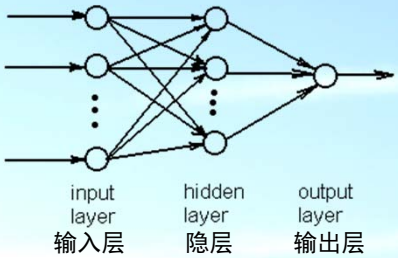


- 阶跃函数不行
- 线性激活函数行不行？

Sec 6.2 Problem 1, Duda et al, Pattern Classification, p. 335

多层感知器MLP (Multi-layer Perceptron)
--- 最经典的人工神经网络Artificial Neural Network (ANN)

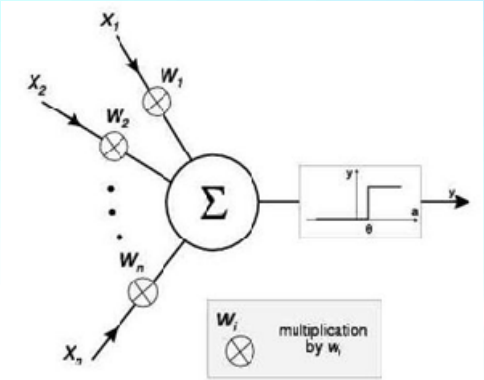


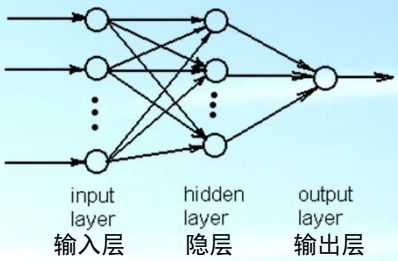


a 3-layer MLP

$$g_k(\mathbf{x}) \equiv y_k = f\left(\sum_j w_{kj} f\left(\sum_i w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$$

多层感知器MLP (Multi-layer Perceptron)
--- 最经典的人工神经网络Artificial Neural Network (ANN)

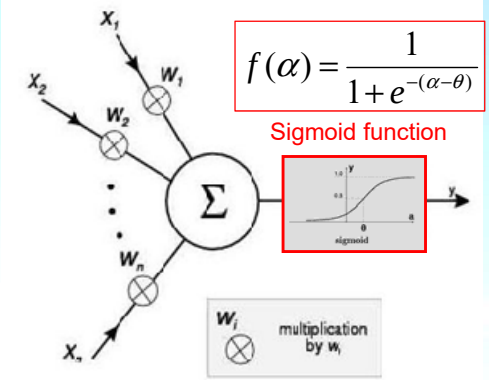




a 3-layer MLP

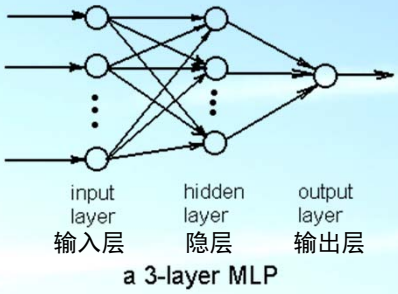
$$g_k(\mathbf{x}) \equiv y_k = f\left(\sum_j w_{kj} f\left(\sum_i w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$$

多层感知器MLP (Multi-layer Perceptron)
--- 最经典的人工神经网络Artificial Neural Network (ANN)



$$f(\alpha) = \frac{1}{1+e^{-(\alpha-\theta)}}$$

Sigmoid function



a 3-layer MLP

$$g_k(\mathbf{x}) \equiv y_k = f\left(\sum_j w_{kj} f\left(\sum_i w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$$

多层感知器实现非线性的直观理解

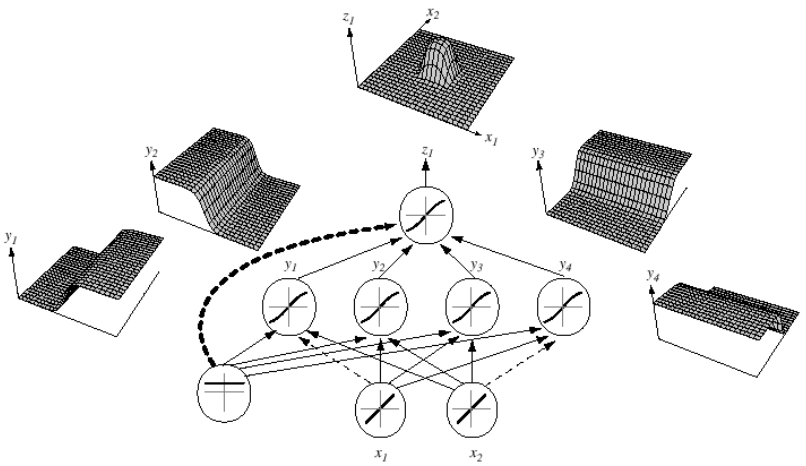
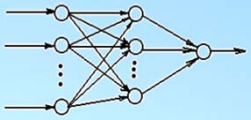


FIGURE 6.2. A 2-4-1 network (with bias) along with the response functions at different units; each hidden output unit has sigmoidal activation function $f(\cdot)$. In the case shown, the hidden unit outputs are paired in opposition thereby producing a "bump" at the output unit. Given a sufficiently large number of hidden units, any continuous function from input to output can be approximated arbitrarily well by such a network. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Duda, Hart & Stork, Pattern Classification (2nd edition), John Wiley & Sons, 2001

多层感知器的表示能力

$$g_k(\mathbf{x}) \equiv y_k = f\left(\sum_j w_{kj} f\left(\sum_i w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$$



理论结果：

多层感知器能够实现任意连续的输入输出函数，只要有充分数目的隐层节点、合适的激活函数和权值。

- From the Kolmogorov theorem:

  Any continuous function $g(\mathbf{x})$ defined on the unit hypercube I^n (I=[0,1] and n≥2) can be represented in the form

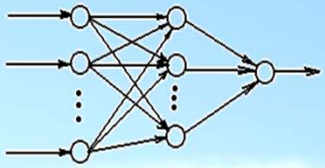
$$g(\mathbf{x}) = \sum_{j=1}^{2n+1} \Xi_j \left(\sum_{i=1}^{d} \Psi_{ij}(x_i)\right)$$

  for properly chosen functions Ξ_j and Ψ_{ij}.

Duda, Hart & Stork, Pattern Classification, John Wiley & Sons, 2001, p.287

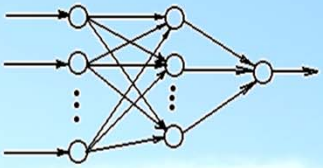
怎样设计用作模式识别的多层感知器？

- 设计一个合适的网络
  - 编码：
    - 两类 0 vs. 1
    - 多类 1-of-C (e.g.,1000, 0100, 0010, 0001) one-hot编码
  - 结构：
    - 三层网（一个隐层）可实现空间内任意的凸形区域的划分
    - 四层网（两个隐层）可实现任意形状（连续或不连续）区域划分



怎样设计用作模式识别的多层感知器？

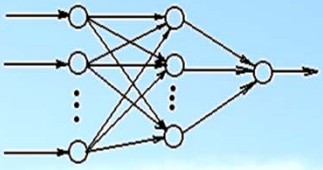
- 设计一个合适的网络
  - 编码：
    - 两类 0 vs. 1
    - 多类 1-of-C (e.g.,1000, 0100, 0010, 0001)
  - 结构：
    - 三层网（一个隐层）可实现空间内任意的凸形成区域的划分
    - 四层网（两个隐层）可实现任意形状（连续或不连续）区域划分

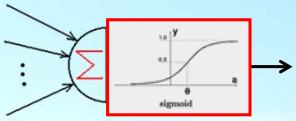


但是，
多少个隐节点？

怎样设计用作模式识别的多层感知器？

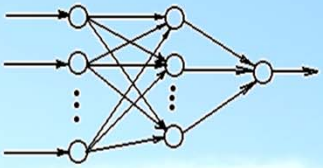
- 设计一个合适的网络
  - 编码：
    - 两类 0 vs. 1
    - 多类 1-of-C (e.g.,1000, 0100, 0010, 0001)
  - 结构：
    - 三层网（一个隐层）可实现空间内任意的凸形成区域的划分
    - 四层网（两个隐层）可实现任意形状（连续或不连续）区域划分
- 选择合适的结点函数
  - Sigmoid函数 或tanh函数 双曲正切函数
  - ReLU …

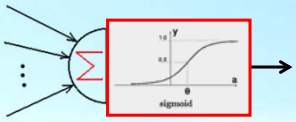


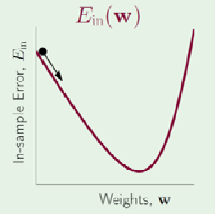


怎样设计用作模式识别的多层感知器？

- 设计一个合适的网络
  - 编码：
    - 两类 0 vs. 1
    - 多类 1-of-C (e.g.,1000, 0100, 0010, 0001)
  - 结构：
    - 三层网（一个隐层）可实现空间内任意的凸形成区域的划分
    - 四层网（两个隐层）可实现任意形状（连续或不连续）区域划分
- 选择合适的结点函数
  - Sigmoid函数 或tanh函数
  - ReLU …
- 训练神经网络参数
  - 用什么训练？→训练样本
  - 怎样训练？





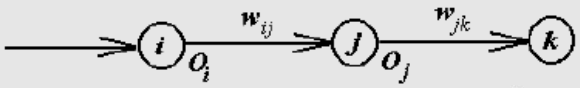


➔ 问题：当很多人集体完成一件事而出错的时候，怎样把错误追究到每个人？

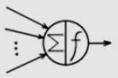
休息1分钟

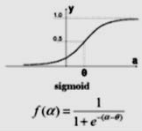
误差反向传播 Back-Propagation (BP) of the Error

LeCun, 1986; Rumelhart, Hinton & Williams, 1986; Parker, 1985



节点响应 for node j, $net_j = \sum_i w_{ij} O_i$, $O_j = f(net_j)$





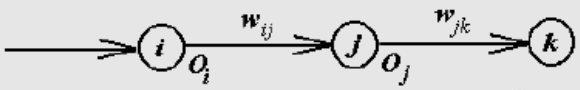
实际输出 期望输出 for output node: $\hat{y}_j = O_j$ is the actual output, y_j is the desired output, error

误差 error: $$E = \frac{1}{2}\sum_j (y_j - \hat{y}_j)^2$$

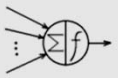
误差梯度 gradients: $$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} = \delta_j O_i, \quad \delta_j \equiv \frac{\partial E}{\partial net_j}$$

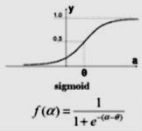
误差反向传播 Back-Propagation (BP) of the Error

LeCun, 1986; Rumelhart, Hinton & Williams, 1986; Parker, 1985



节点响应 for node j, $net_j = \sum_i w_{ij} O_i$, $O_j = f(net_j)$



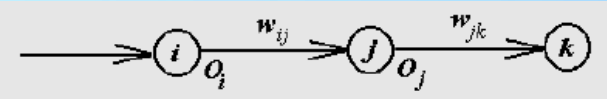


实际输出 期望输出 for output node: $\hat{y}_j = O_j$ is the actual output, y_j is the desired output, error

误差 error: $$E = \frac{1}{2}\sum_j (y_j - \hat{y}_j)^2$$

目标函数：平方误差

误差梯度 gradients: $$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} = \delta_j O_i, \quad \delta_j \equiv \frac{\partial E}{\partial net_j}$$



误差梯度 gradients: $\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} = \delta_j O_i, \quad \delta_j \equiv \frac{\partial E}{\partial net_j}$

对输出节点 for output nodes: $O_j = \hat{y}_j, \quad \delta_j = \frac{\partial E}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial net_j} = -(y_j - \hat{y}_j) f'(net_j)$

对隐层节点 for hidden nodes (back-propagation):

$$\delta_j = \frac{\partial E}{\partial net_j} = \sum_k \frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial O_j} \cdot \frac{\partial O_j}{\partial net_j} = \sum_k \delta_k w_{jk} f'(net_j)$$

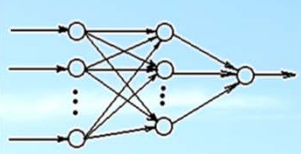
权值学习 adaption (at iteration t):

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t), \qquad \Delta w_{ij}(t) = -\eta \delta_j(t) O_i(t)$$

学习步长 where: η - step length (learning rate)

误差反向传播（BP)算法

LeCun, 1986; Rumelhart, Hinton & Williams, 1986; Parker, 1985

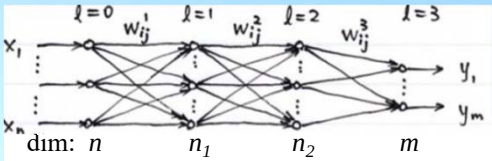


- Purpose: to train MLP weights with data
- Goal: to minimize the error (MSE)
- Algorithm: *gradient descent*
- Basic steps:
  - forward: compute the output with certain input 信号正向传播
    input nodes >>> hidden nodes >>> output nodes
  - compare: discrepancy between output and desired 计算误差
  - backward: propagate the error back through the network 误差反向传播
    input nodes <<< hidden nodes <<< output nodes
  - adaptation: adapt the weights according to the errors 梯度下降学习

Sigmoid激活函数下BP算法的伪代码

1º. Initialize weights (with small random values), t=0

2º. Apply a training sample $\mathbf{x} = [x_1, \cdots, x_n]^T \in R^n$ with desired output $D = [d_1, \cdots, d_m]^T \in R^m$



3º. Forward calculation: $Y = [y_1, \cdots, y_m]^T \in R^m$,

$$y_l = f\left(\sum_{j=1}^{n_2} w_{jl} f\left(\sum_{j=1}^{n_1} w_{jk} f\left(\sum_{i=1}^{n} w_{ij} x_i\right)\right)\right),\ l = 1, \cdots, m$$

4º. Adjust weights from the output layer. For layer l,

$$w_{ij}^l(t+1) = w_{ij}^l(t) + \eta \delta_j^l x_i^{l-1},\ j = 1, \cdots, n_l, i = 1, \cdots, n_{l-1}$$

Where for the output layer

$$\delta_j^l = y_j(1 - y_j)(d_j - y_j), j = 1, \cdots, m$$

$f'(\alpha) = f(\alpha)(1 - f(\alpha))$
for $f(\alpha) = 1/(1 + e^{-\alpha})$

and for hidden layers

$$\delta_j^l = x_j^l(1 - x_j^l) \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} w_{jk}^{l+1}(t), j = 1, \cdots, n_l$$

5º. Loop: if stop criterion not met, set t=t+1 and go to 2º with another sample.

机器学习的基本要素：多层感知器版

- 怎样造一个学习机器？
  - 它需要老师
    - → 我们设计它（特征和模型）$g_k(\mathbf{x}) = f\left(\sum_j w_{kj} f\left(\sum_i w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$
  - 它需要训练/学习材料
    - → 训练数据 $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\},\ \boldsymbol{x}_j \in R^{d+1}, y_j \in \{0,1\}$ or $[0,1]$
  - 我们需要为它树立学习的目标
    - → 目标函数、学习准则 $\min E(\boldsymbol{w}) = -\frac{1}{2}\sum_{j=1}^{N}(y - \hat{y})^2$
  - 我们需要告诉它怎样学
    - → 学习/训练算法 BP算法 $w_{ij}^l(t+1) = w_{ij}^l(t) + \eta \delta_j^l x_i^{l-1}$

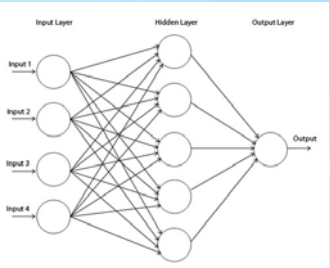
常见训练策略

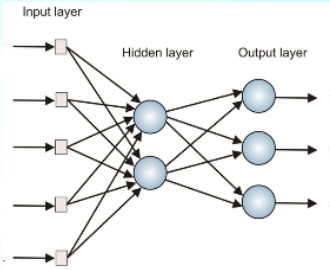
- 逐一训练
  - 每个样本对网络逐一训练
- 随机训练
  - 从训练集中随机抽取样本进行训练，每次抽取一个或一组
- 整批训练
  - 所有样本一起训练（即把所有样本都输入一遍后再用累计误差进行训练）
- 查询式训练
  - 依据神经网络的输出挑选训练样本
    → “主动学习（Active Learning）”

Duda, Hart & Stork, Pattern Classification, John Wiley & Sons, 2001, p.293-295, 480-481

多层感知器能学习干什么？

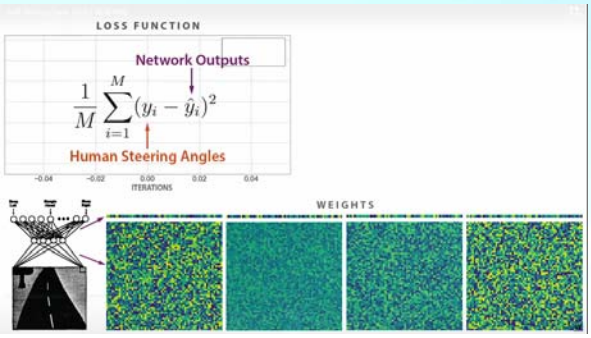
- Almost Everything!
  - 模式识别
    - 把类别编码称输出
  - 回归/函数估计/预测
    - 输出为实数
  - 数据压缩表示
    - 自己学自己：把输入当作输出





视频：早年自动驾驶实例
ALVINN and Autonomous Navigation in 1990s

从ALV到ALVNN



Self driving cars [S1E2 ALVINN].mp4 ~11:12

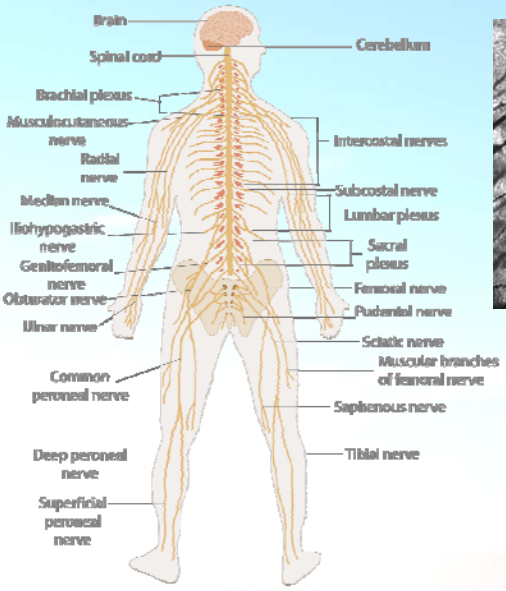
基于神经网络的自动驾驶

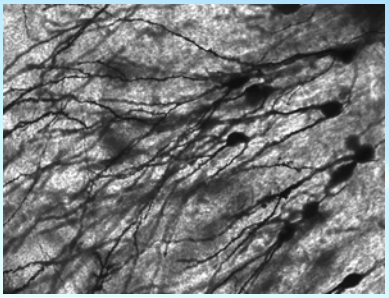
Stanford CS229 Lecture 2, 3:21-8:40

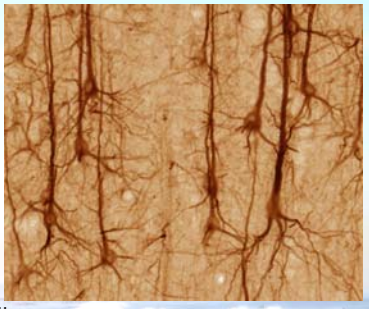
休息1分钟

为什么称作神经网络？

Neurons 神经元

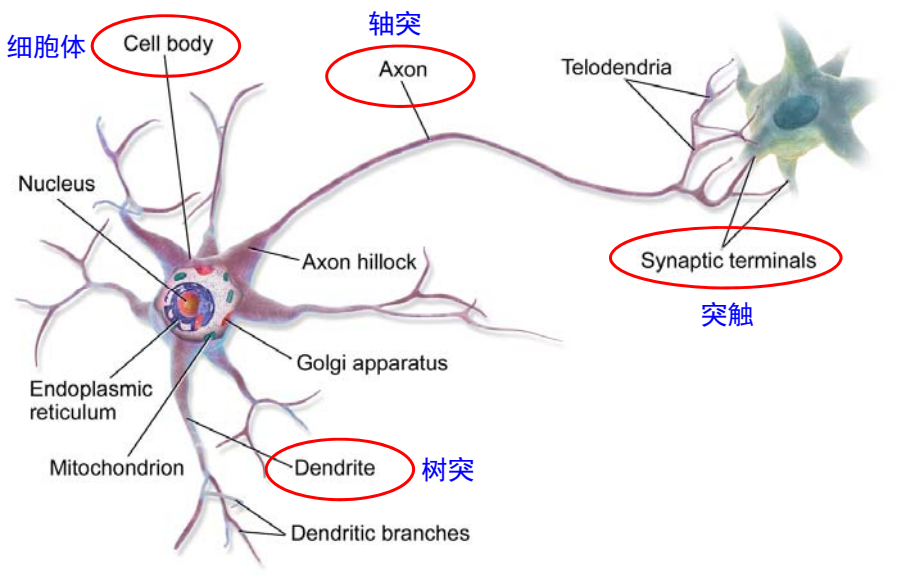






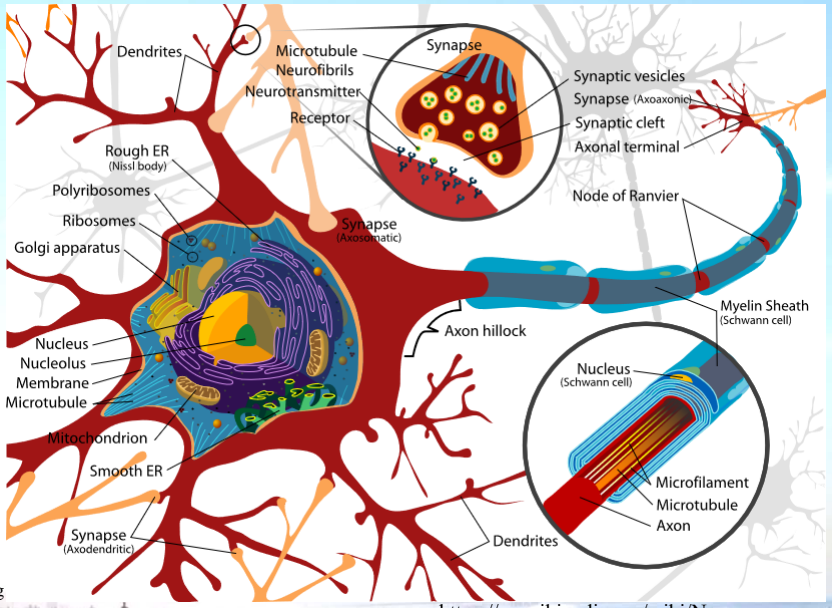
-- Wikipedia

Neurons 神经元



https://en.wikipedia.org/wiki/Neuron

Neurons 神经元

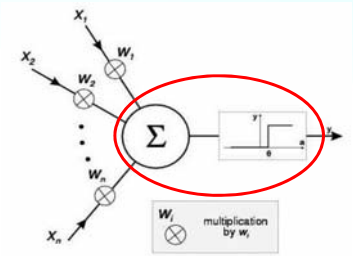


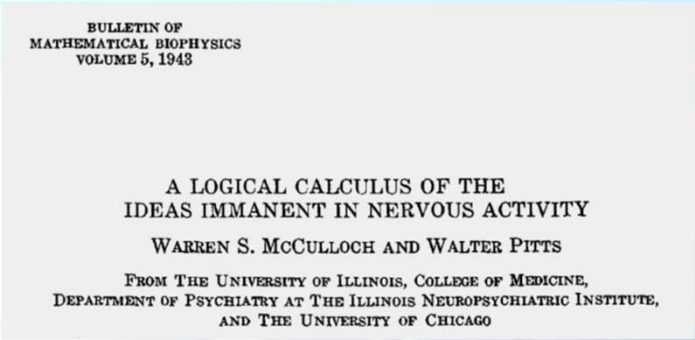
https://en.wikipedia.org/wiki/Neuron

人工神经元

- 对自然神经元的数学模型
  - 阈值逻辑单元(TLU- Threshold Logic Unit)模型 （McCulloch and Pitts in 1943）

$$y = \mathrm{sgn}\left(\sum_{i=1}^{n} w_i x_i + w_0\right), \qquad y = \begin{cases} +1 & \Rightarrow \text{class } A \\ -1 & \Rightarrow \text{class } B \end{cases}$$



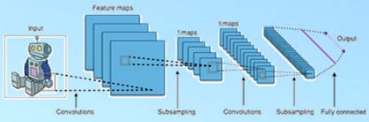

BULLETIN OF
MATHEMATICAL BIOPHYSICS
VOLUME 5, 1943

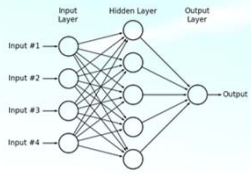
A LOGICAL CALCULUS OF THE
IDEAS IMMANENT IN NERVOUS ACTIVITY

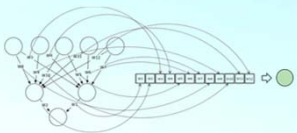
WARREN S. MCCULLOCH AND WALTER PITTS

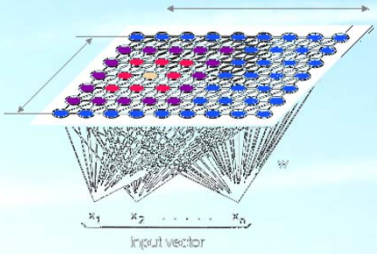
FROM THE UNIVERSITY OF ILLINOIS, COLLEGE OF MEDICINE,
DEPARTMENT OF PSYCHIATRY AT THE ILLINOIS NEUROPSYCHIATRIC INSTITUTE,
AND THE UNIVERSITY OF CHICAGO

- 人工神经网络（ANN）
  - 大量简单计算节点单元（神经元）
  - 经过一系列权值连接成一个网络
  - 权值根据数据按某种算法进行学习

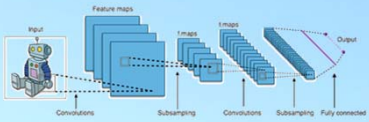


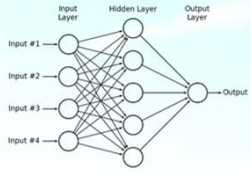


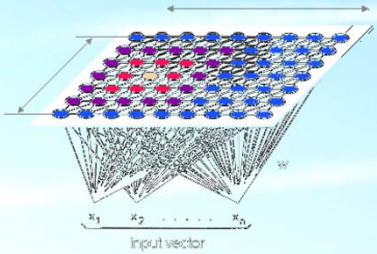




- 人工神经网络（ANN）
  - 大量简单计算节点单元（神经元）
  - 经过一系列权值连接成一个网络
  - 权值根据数据按某种算法进行学习





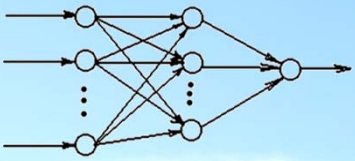


- 神经网络的三要素：
  - 网络结构
  - 节点激活函数/计算 特性
  - 学习算法

这些要素的不同构成了不同类型的神经网络

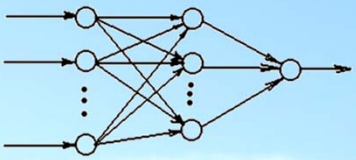
1980年代三种主要类型的神经网络

- 前馈型神经网络 Feedforward NN
  - 代表性方法：多层感知器

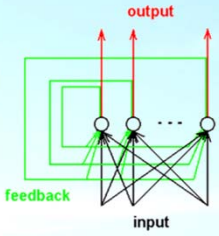


1980年代三种主要类型的神经网络

- 前馈型神经网络 Feedforward NN
  - 代表性方法：多层感知器

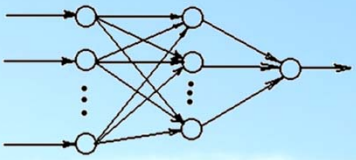


- 反馈型神经网络 Feedback NN
  - 代表性方法 Hopfield NN

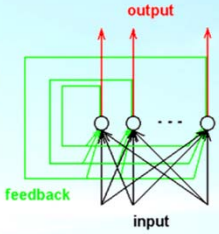


1980年代三种主要类型的神经网络

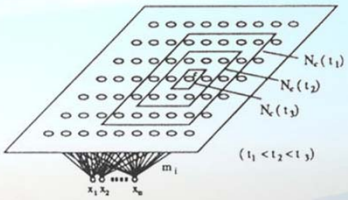
- 前馈型神经网络 Feedforward NN
  - 代表性方法：多层感知器



- 反馈型神经网络 Feedback NN
  - 代表性方法 Hopfield NN



- 竞争学习神经网络 Competitive Learning NN
  - 代表性方法：自组织映射 Self-organizing map



再看神经网络的训练

学习曲线

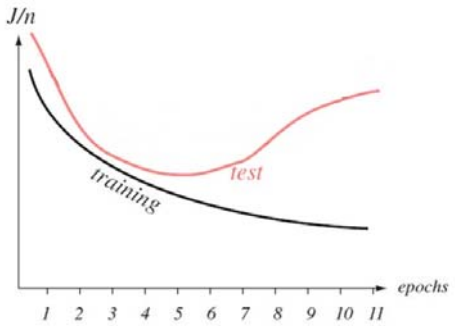


FIGURE 6.6. A learning curve shows the criterion function as a function of the amount of training, typically indicated by the number of epochs or presentations of the full training set. We plot the average error per pattern, that is, $1/n \sum_{p=1}^{n} J_p$. The validation error and the test or generalization error per pattern are virtually always higher than the training error. In some protocols, training is stopped at the first minimum of the validation set. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

学习曲线

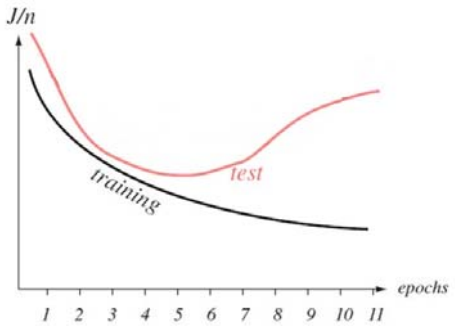


FIGURE 6.6. A learning curve shows the criterion function as a function of the amount of training, typically indicated by the number of epochs or presentations of the full training set. We plot the average error per pattern, that is, $1/n \sum_{p=1}^{n} J_p$. The validation error and the test or generalization error per pattern are virtually always higher than the training error. In some protocols, training is stopped at the first minimum of the validation set. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

观察：

- 训练错误率并非总是单调下降
- 并非训练错误率越小测试错误率也越小
  - 过学习现象：
    训练错误率减小，测试错误率反而增大

学习曲线

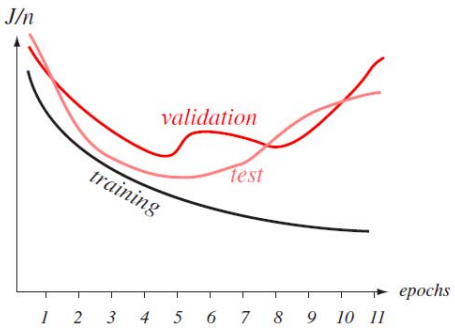


FIGURE 6.6. A learning curve shows the criterion function as a function of the amount of training, typically indicated by the number of epochs or presentations of the full training set. We plot the average error per pattern, that is, $1/n \sum_{p=1}^{n} J_p$. The validation error and the test or generalization error per pattern are virtually always higher than the training error. In some protocols, training is stopped at the first minimum of the validation set. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

建议：

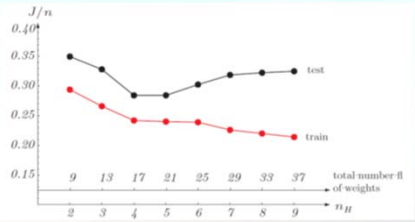
- 密切关注训练过程中的误差变化
- 从训练集中分出一个内部验证集
  - 用验证集上的表现推断是否过学习

BP算法训练中可能遇到的问题

- 收敛慢
- 学习曲线震荡
- 过学习：推广能力差
- 可能的原因：
  - 网络结构或激活函数问题
  - 训练样本不充分或不合适
  - 初值不合适，学习率不合适，…
  - 缺乏恰当的预处理（比如归一化）
  - 需要更好的训练策略
  - 运气问题
  - 问题提出的有问题，比如$\boldsymbol{x}$和y没有关系
  - …

改进BP算法的可能策略

- 激活函数
  - Sigmoid, tanh, ReLU, …
- 特征尺度调整或归一化
  - 有效值范围，相对重要性等
- 目标值调整
  - 比如在两类分类问题中，用0.1对0.9替代0对1
- 引入“伪样本”增大训练样本集
  - 如加噪、引入不变性特征的样本，平衡两类样本等
- 隐层节点数、层数
  - 与样本规模和/或问题复杂度相适应
  - ~n/10?
  - 网络剪枝



Duda, Hart & Stork, Pattern Classification, John Wiley & Sons, 2001, p.306-318

改进BP算法的可能策略

- 初始化
  - 探索不同的初始化策略，以是算法有机会搜索到整个空间
- 学习率（步长）
  - 比如对平方误差采用 $\eta_{opt} = (\partial^2 J/\partial w^2)^{-1}$
  - 根据二阶导数动态调整
  - 对不同权值采用不同的学习率

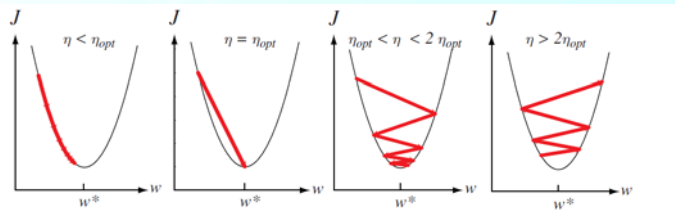


FIGURE 6.16. Gradient descent in a one-dimensional quadratic criterion with different learning rates. If $\eta < \eta_{opt}$, convergence is assured, but training can be needlessly slow. If $\eta = \eta_{opt}$, a single learning step suffices to find the error minimum. If $\eta_{opt} < \eta < 2\eta_{opt}$, the system will oscillate but nevertheless converge, but training is needlessly slow. If $\eta > 2\eta_{opt}$, the system diverges. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

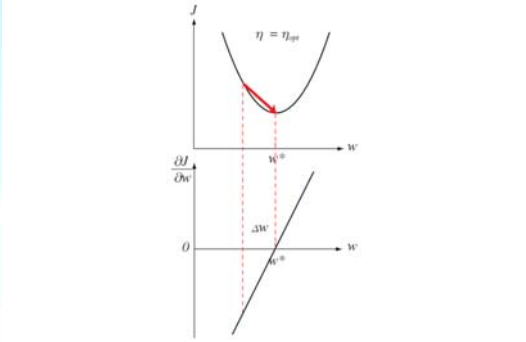


FIGURE 6.17. If the criterion function is quadratic (above), its derivative is linear (below). The optimal learning rate η_{opt} ensures that the weight value yielding minimum error, w^*, is found in a single learning step. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

改进BP算法的可能策略

- 引入动量（Momentum）
  $$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + (1-\alpha)\Delta\boldsymbol{w}_{bp}(t) + \alpha(\boldsymbol{w}(t) - \boldsymbol{w}(t-1))$$
- 权值衰减（Weight decay） $\boldsymbol{w}^{new} = \boldsymbol{w}^{old}(1-\epsilon)$
  - 等价于定义了新的正则化误差 $J_{ef}(\boldsymbol{w}) = J(\boldsymbol{w}) + \frac{2\epsilon}{\eta}\boldsymbol{w}^T\boldsymbol{w}$
- 引入提示输出（Hints）
  - 加入比较容易学习的扩增目标
- 终止条件
  - 用验证集决定是否提前终止

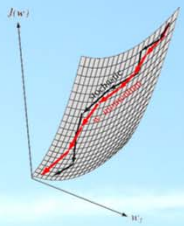


FIGURE 6.18. The incorporation of momentum into stochastic gradient descent by Eq. 37 (red arrows) reduces the variation in overall gradient directions and speeds learning. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

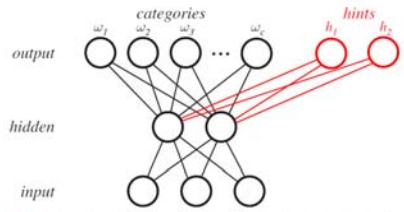


FIGURE 6.19. In learning with hints, the output layer of a standard network having c category units is augmented with hint units. During training, the target vectors are also augmented with signals for the hint units. In this way the input-to-hidden weights learn improved feature groupings. The hint units are not used during classification, and thus they and their hidden-to-output weights are removed from the trained network. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

本章知识点

- 多层感知器原理
- BP算法
- 人工神经网络的基本概念

作业

- 第二、四章
  - 2.5 MSE与Fisher线性判别
  - 2.6 罗杰斯特函数
  - 2.7 罗杰斯特回归的梯度下降算法
  - 4.1 线性多层感知器实现的函数映射
  - 4.2 采用tanh()函数的BP算法
- 截止日期：3月9日
- 第三、四章
  - 3.3 Softmax小实验
  - 4.3 多层感知器小实验
- 截止日期：3月16日


下课，下周见！
Xuegong Zhang
57