目录

- 机器学习概述
- 机器学习的主要策略和基本结构
- 常见的几种学习方法
- 基于神经网络的学习

5.1 机器学习概述

- 学习的基本概念

学习是人类具有的一种重要的智能行为。

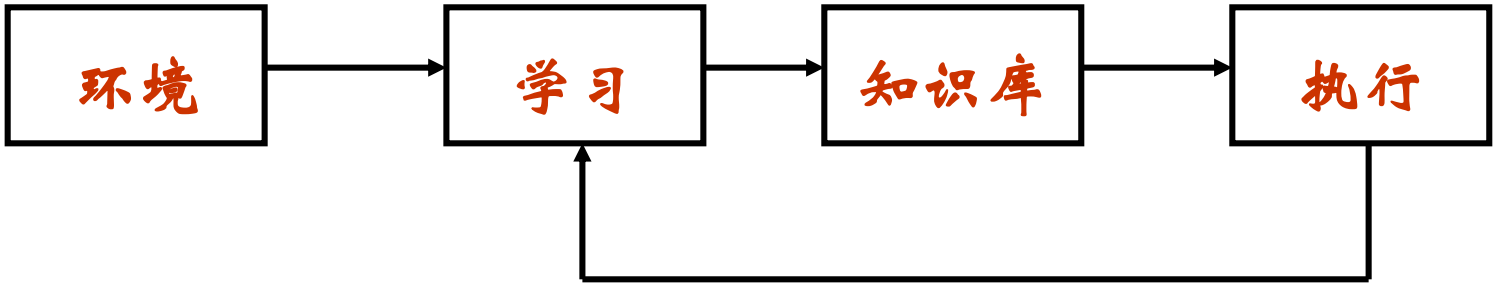
西蒙的观点：学习就是系统在不断重复的工作中对本身能力的增强或者改进，使得系统在下一次执行同样任务或类似任务时，会比现在做得更好或效率更高。

- 机器学习的定义

机器学习是研究如何使用机器来模拟人类学习活动的一门学科。稍为严格的提法是：机器学习是一门研究机器获取新知识和新技能，并识别现有知识的学问。

5.2 机器学习的主要策略和基本结构

机器学习系统的基本结构



- 环境向系统的学习部分提供某些信息
- 学习部分利用这些信息修改知识库，以增进系统执行部分完成任务的效能
- 执行部分根据知识库完成任务，同时把获得的信息反馈给学习部分
- 在具体的应用中，环境，知识库和执行部分决定了具体的工作内容，学习部分所需要解决的问题完全由上述3部分确定

5.2 机器学习的主要策略和基本结构

机器学习的主要策略

学习过程与推理过程是紧密相连的，按照学习中使用推理的多少，机器学习所采用的策略大体上可包括如下几种，学习中所用的推理越多，系统的能力越强。

- 机械学习
- 示教学习
- 类比学习
- 示例学习

5.3 常见的几种学习方法

机械学习

1. 机械学习的模式

机械学习是最简单的机器学习方法。机械学习又称记忆学习，即把新的知识存储起来，供需要时检索调用，而不需要计算和推理。机械学习又是最基本的学习过程。任何学习系统都必须记住它们获取的知识。在机械学习系统中，知识的获取是以较为稳定和直接的方式进行的，不需要系统进行过多的加工

5.3 常见的几种学习方法

基于解释的学习

解释学习（**explanation** 最初是由美国**Illinois** 大学的**DeJong** 于**1983** 年提出来的。在经验学习的基础上，运用领域知识对单个例子的问题求解作出解释，这是一种关于知识间因果关系的推理分析，可产生一般的控制策略。

解释学习根据任务所在领域知识和正在学习的概念知识，对当前实例进行分析和求解，得出一个表征求解过程的因果解释树，以获取新的知识。在获取新的知识过程中，通过对属性、表征现象和内在关系等进行解释而学习到新的知识。

5.3 常见的几种学习方法

基于事例的学习

当无法建立好的模型时，可通过记录事例进行学习。

采用基于事例的学习：首先，任何时候都可以应用相容启发方法，把某个预先观察过的事物的特性赋予一个从未见过的新事物；其次，学会如何用于迅速找到特征空间内的最近邻物体。

相容启发：无论何时要猜测某事物的特性，除了提供一套参考事例外不知道其他情况；通过测量其他事物的已知特性，找到最相近的事例，该事例的特性特性是已知的。作为猜测：所求未知特性是与最相似事例的已知特性一样的。

5.3 常见的几种学习方法

强化学习

强化学习的概念

强化学习（**reinforcement learning**）：智能系统从环境到行为映射的学习，以使奖励信号（强化信号）函数值最大。

连接主义的学习分类：

监督学习（**supervised learning)**

—从其输入/输出的实例中学习一个函数　如分类

无监督学习（**unsupervised learning**）

—在未提供明确的输出值情况下，学习输入的模式 如聚类

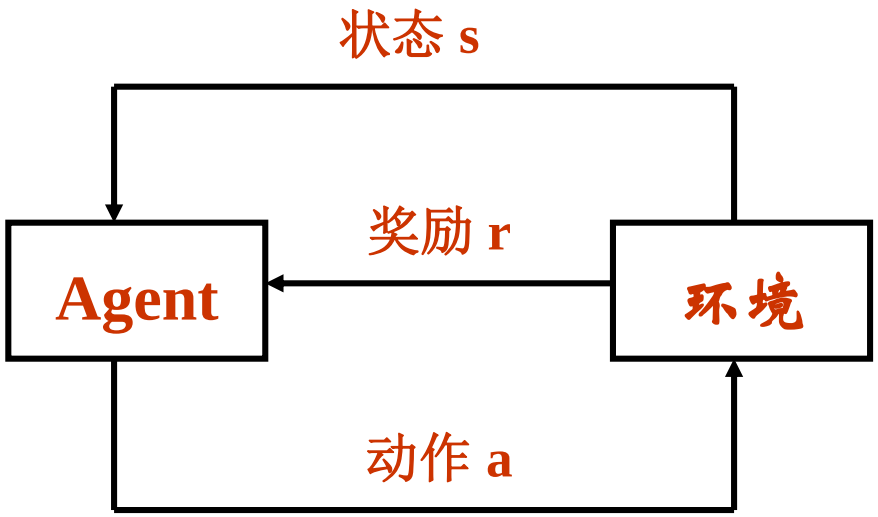
强化学习

5.3 常见的几种学习方法

强化学习

基本原理：如果系统某个动作导致环境正的奖励，那么系统以后产生这个动作的趋势便会加强。反之系统产生这个动作的趋势便减弱。这和生理学中的条件反射原理是接近的。

基本模型：



5.3 常见的几种学习方法

强化学习

- 强化学习中由环境提供的强化信号对智能体（Agent）产生动作的好坏做一种评价
- 不是直接告诉智能体要做什么或者要采取哪个动作,而是智能体通过看哪个动作得到了最多的奖励来自己发现
- 智能体的动作的影响不只是立即得到的奖励，而且还影响接下来的动作和最终的奖励。

试错搜索(trial-and-error search)和延期强化(delayed reinforcement)这两个特性是强化学习中两个最重要的特性。

5.3 常见的几种学习方法

强化学习

强化学习系统接受环境状态的输入s，根据内部的推理机制，系统输出相应的行为动作a。环境在系统动作作用a下，变迁到新的状态s'。系统接受环境新状态的输入，同时得到环境对于系统的瞬时奖惩反馈r。对于强化学习系统来讲，其目标是学一个行为策略π：$S \rightarrow A$，使系统选择的动作能够获得环境奖励的累计值最大。

5.4 基于神经网络的学习

- 人工神经网络的应用？
- https://www.bilibili.com/video/av19184124?from=search&seid=15202434440337473404

- https://www.bilibili.com/video/av26004653?from=search&seid=18132351520209860073

本章内容

- 人工神经网络简介
- 人工神经元结构
  - 输入信号
  - 激活函数
  - 几何构型
  - 学习规则

生物神经系统

- 人脑：复杂的、非线性的、并行的“计算机”。
  - 具有模式识别、感知和电机控制人物的能力。

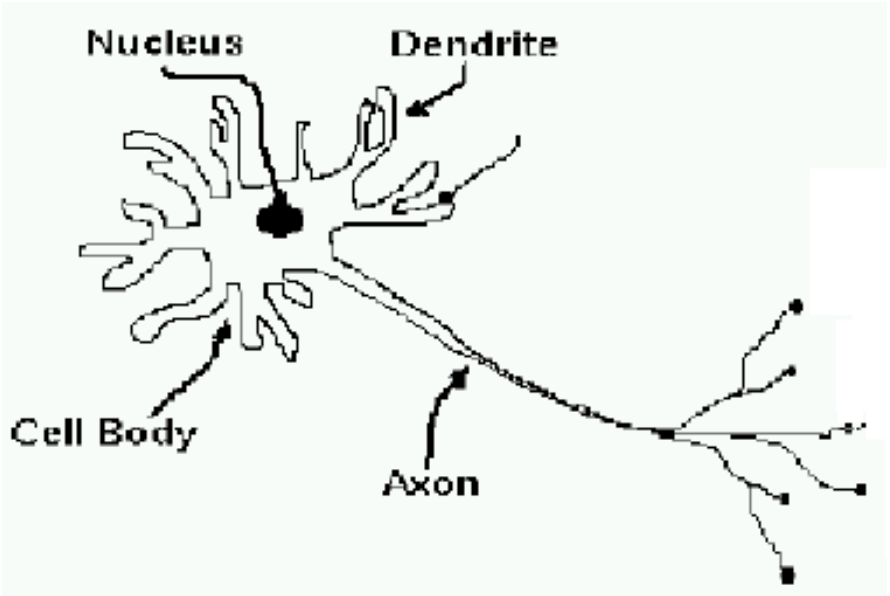
- 据估计，人类脑皮层中有100亿~5000亿数量级的神经元和60万亿的突触。这些神经元排列成1000个主模块，每个有大约500个神经网络。

- 是否有可能对人脑建模？
  - 现在还不行。当前的神经建模仅仅是针对解决特定问题所建立的小规模的人工神经网络。

生物神经系统

- 生物神经系统的基本构成单元是神经细胞，称为神经元。

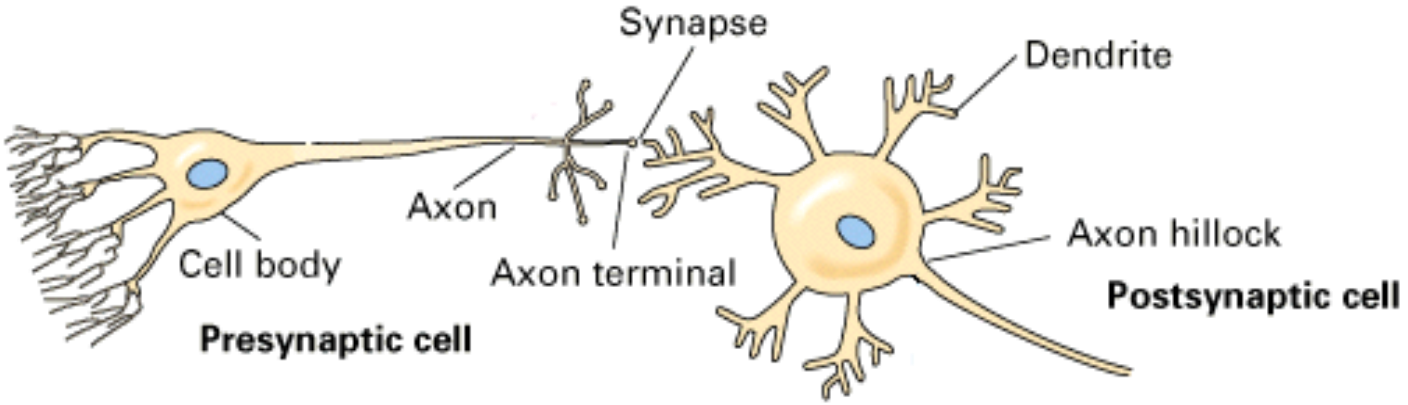
- 一个神经元由一个细胞体、大量树突和一个轴突组成。



- **Nucleus**：神经元
- **Cell Body**：细胞体
- **Dendrite**：树突
- **Axon**：轴突

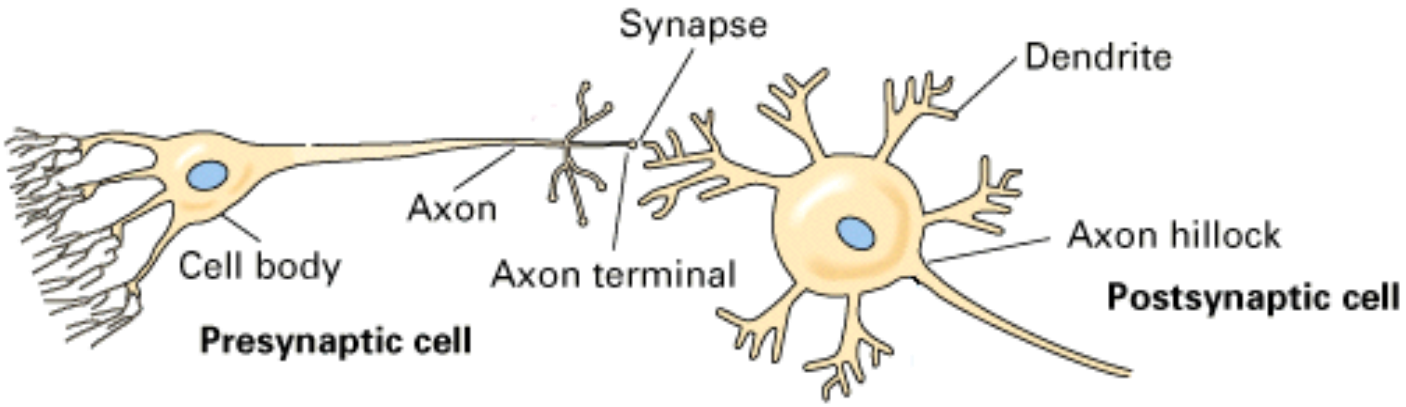
生物神经元

- 神经元之间互相广泛地连接，这种连接是由一个神经元的轴突和另一个神经元的树突相连接来实现的。



- **Synapse**：突触
- **Axon terminal**：轴突终端
- **Axon hillock** ：轴丘
- 信号输出端：轴突
- 信号输入端：树突

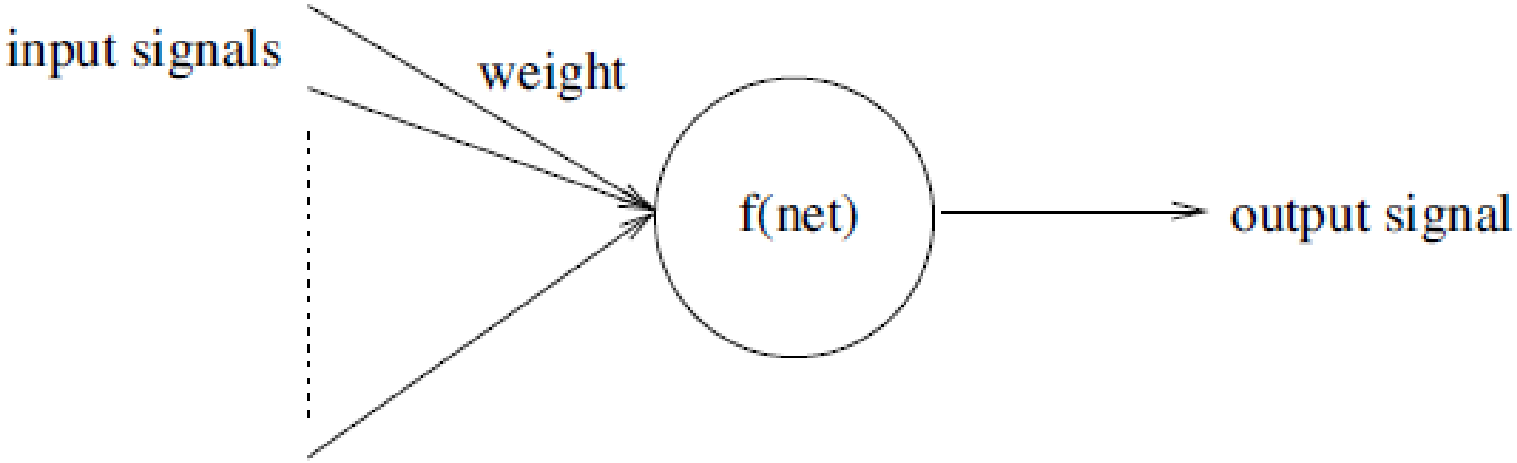
生物神经元



- 只有当细胞“激发”，信号才能被传播到神经元的轴突。
- 一个神经元既可以抑制、也可以激活一个信号。

人工神经元的结构

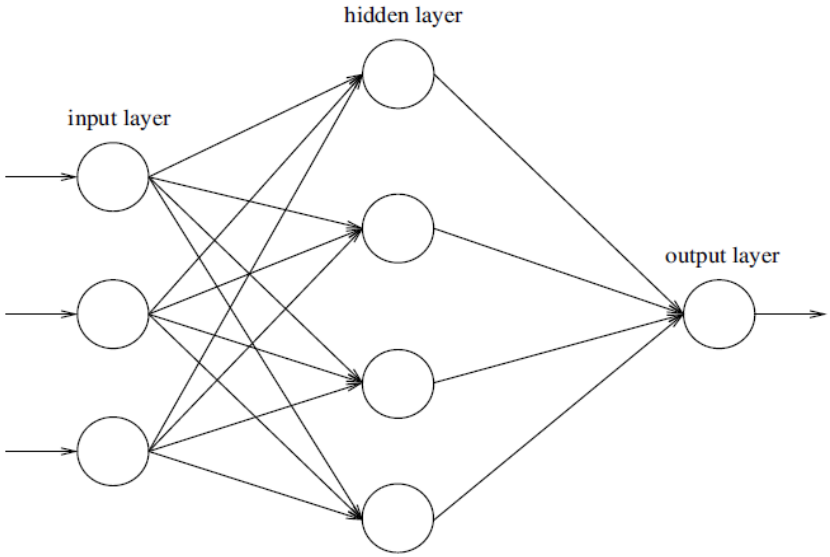
- 每一个神经元从外部环境或其他人工神经元接收信号，当它被激发后就将信号传递给所有与之相连的人工神经元。



- ***f*(*net*)：激活函数。**
- 对一个神经元的激发或抑制信号的强度，都是由激活函数来控制的。

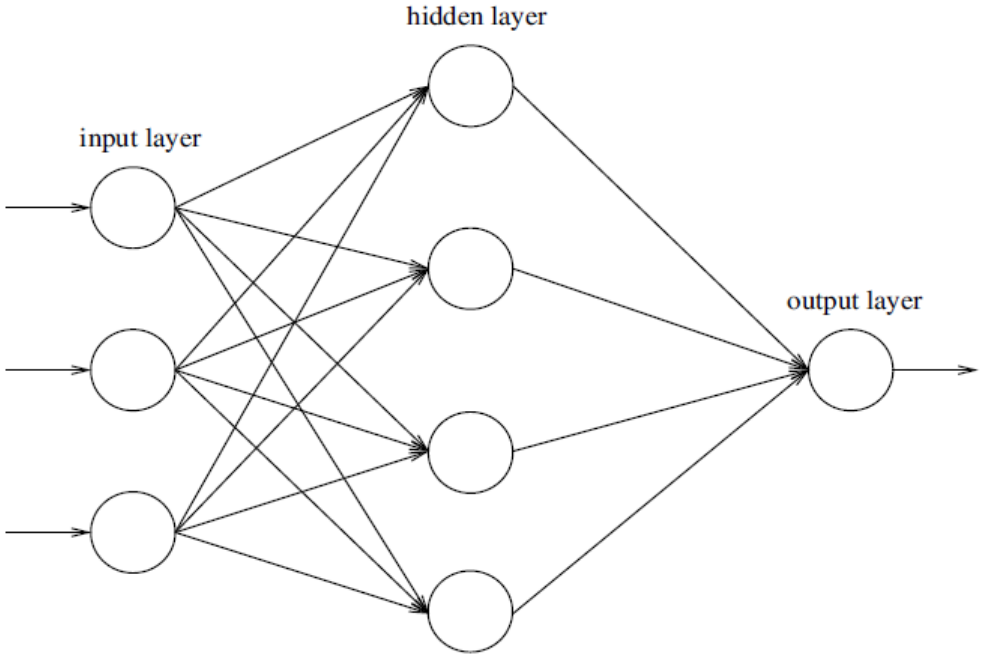
人工神经网络

- 人工神经网络（**Artificial Neural Network**）是一个由人工神经元（**Artificial Neuron**）组成的分层网络。



- 通常，一个人工神经网络可以由一个输入层、多个隐层和一个输出层组成。
  - 一个上层的神经元与下一层的神经元是彼此全连接或部分连接的。
  - 有时，也可能与前一层的反馈连接。

人工神经网络



- 本质上，神经网络是一个从$\mathbf{R}^{\mathbf{I}}$到$\mathbf{R}^{\mathbf{K}}$的非线性映射的实现，即：

$$f_{NN}: \ R^I \rightarrow R^K$$

  - 其中，I和K分别是输入空间和目标（期望输出）空间的维数。

人工神经网络的类型

- 单层神经网络，例如**Hopfield**网络。
- 多层前馈神经网络，例如标准反向传播（**standard backpropagation**）、函数连接（**functional link**）、乘积单元网络（**product unit networks**）。
- 时间神经网络，例如**simple recurrent networks**，**time-delay neural networks**。
- 自组织神经网络，例如**Kohonen**自组织特征映射和学习向量量化器。
- 监督和非监督神经网络的结合，例如某些径向基函数神经网络。

人工神经网络的应用领域

- 分类：预测输入向量的类别
- 模式匹配：产生一个与给定输入向量最关联的模式
- 模式完善：补全一个给定输入向量缺失的部分
- 优化：在一个优化问题中找出参数的最优值
- 控制：在给定一个输入向量的情况下，给出一个恰当动作的建议
- 函数拟合/时间序列建模：学习输入向量和期望输出向量之间的函数关系
- 数据挖掘：从数据中发现隐藏的模式

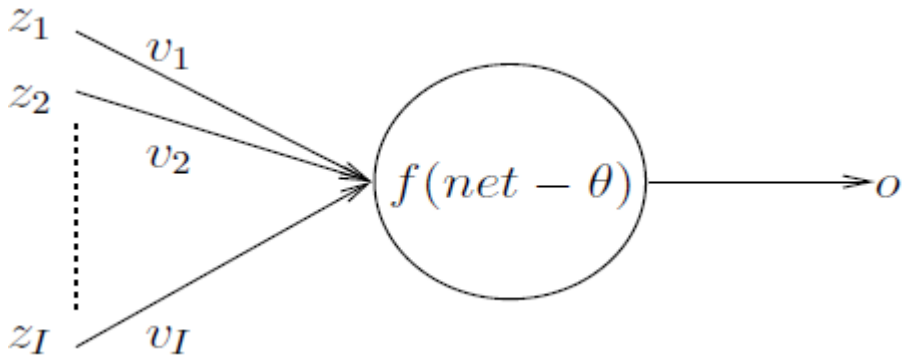
本章内容

- 人工神经网络简介
- 人工神经元结构
  - 输入信号
  - 激活函数
  - 几何构型
  - 学习规则

人工神经元的结构

- 一个人工神经元（即神经元），通常实现了从R^I到[0, 1]或[1, −1]的一个非线性映射，映射区间取决于使用的激活函数。

$$f_{AN}: \mathrm{R}^{\mathrm{I}} \to [0,1] \qquad 或 \qquad f_{AN}: \mathrm{R}^{\mathrm{I}} \to [-1,1]$$



$$z = (z_1, z_2, \cdots, z_I)$$

- 输入向量z来自环境或者其他神经元。每一个输入信号z_i都关联一个增强或者衰减该输入信号的权重v_i。
- 输出信号的强度也受阈值θ的影响， θ也称为偏置。

输入信号

- 一个神经元的网络输入信号通常为所有输入信号的加权和，这类神经元称为求和单元。

$$net = \sum_{i=1}^{I} z_i v_i$$

- 计算网络输入信号的另一种方法是采用所有输入信号的乘积，这类神经元称为乘积单元。

$$net = \prod_{i=1}^{I} z_i^{v_i}$$

激活函数

- 函数f_{AN}接受网络输入信号和偏置，并且决定神经元的输出（或者激活强度），这个函数称作激活函数。

- 一般而言，激活函数是单调递增映射，其中：

$$f_{AN}(-\infty)=0 \text{或者} f_{AN}(-\infty)=-1$$

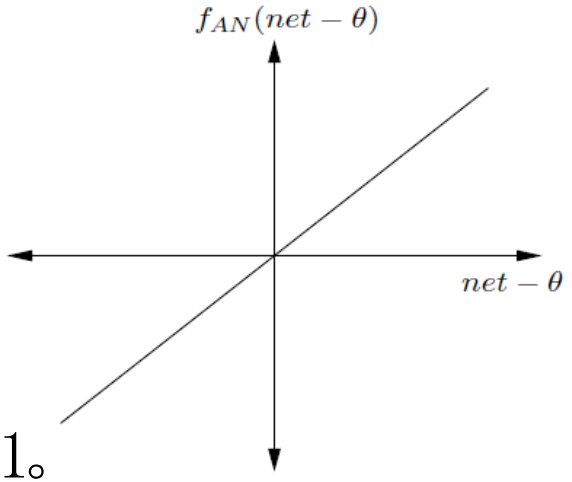
且

$$f_{AN}(\infty)=1$$

常用的激活函数

- **线性函数**

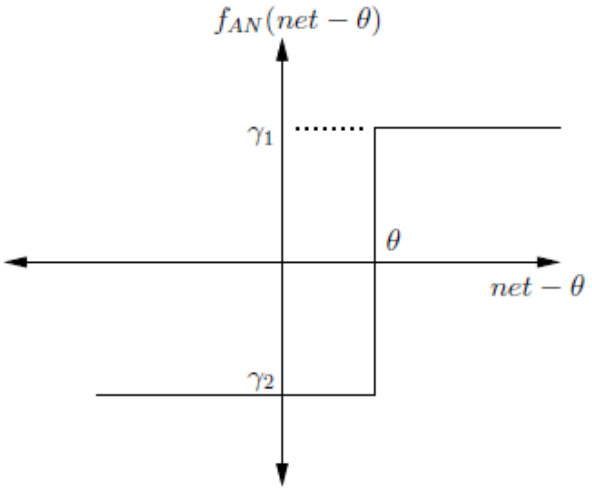
$$f_{AN}(net-\theta)=\lambda(net-\theta)$$



注意：$f_{AN}(-\infty)\neq 0$或-1且$f_{AN}(\infty)\neq 1$。

- **阶跃函数**

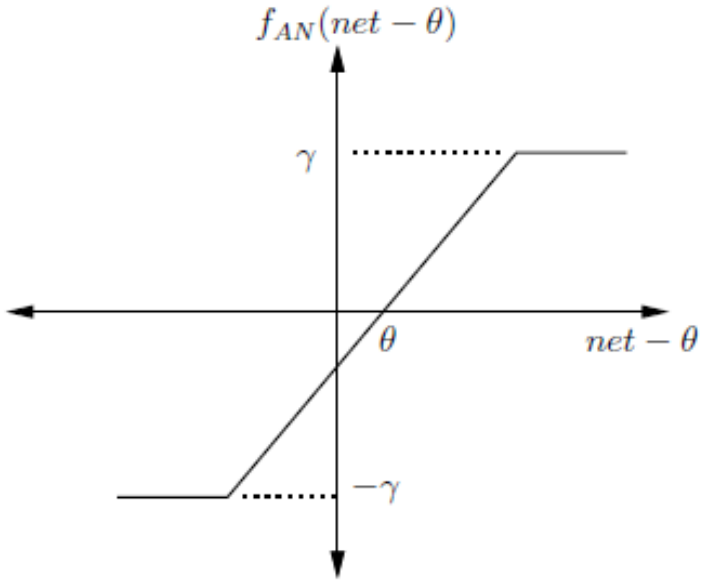
$$f_{AN}(net-\theta)=\begin{cases}\gamma_1 & \text{如果} net\geq\theta\\ \gamma_2 & \text{如果} net\leq\theta\end{cases}$$



常用的激活函数

- 斜坡函数

$$f_{AN}(net-\theta)=\begin{cases}\gamma & \text{如果} net-\theta \geq \varepsilon \\ net-\theta & \text{如果} -\varepsilon < net-\theta < \varepsilon \\ -\gamma & \text{如果} net-\theta \leq -\varepsilon\end{cases}$$

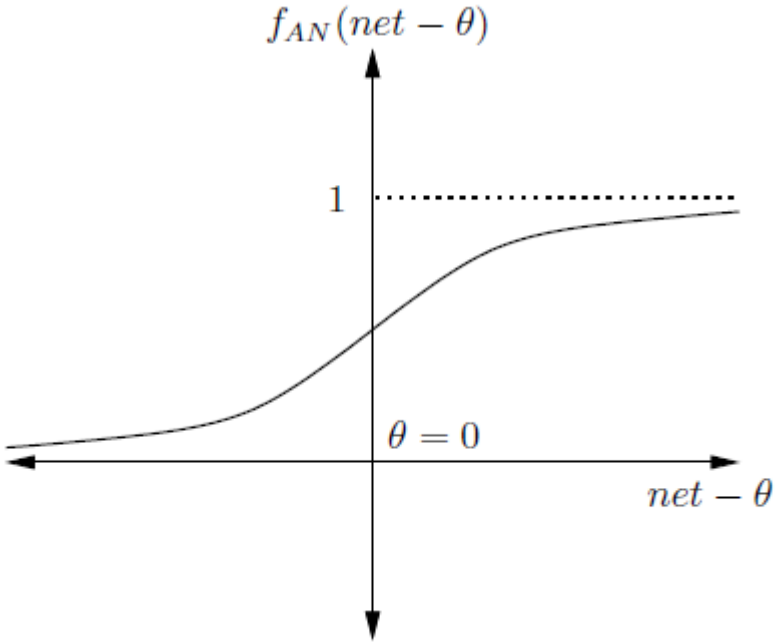


常用的激活函数

- **Sigmoid函数**

$$f_{AN}(net-\theta)=\frac{1}{1+e^{-\lambda(net-\theta)}}$$

其中，参数λ控制函数的陡度，通常λ=1。



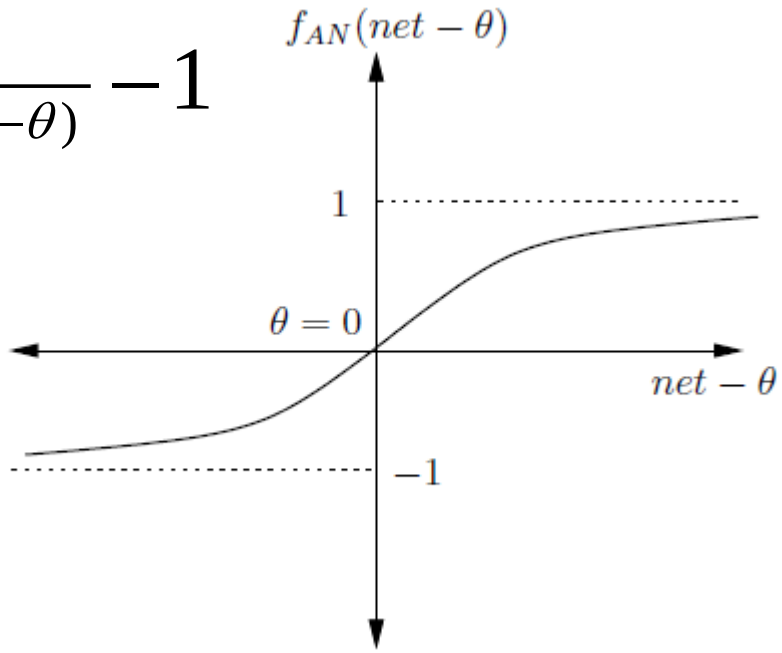
常用的激活函数

- 双正切函数

$$f_{AN}(net-\theta)=\frac{e^{\lambda(net-\theta)}-e^{-\lambda(net-\theta)}}{e^{\lambda(net-\theta)}+e^{-\lambda(net-\theta)}}$$

或近似为

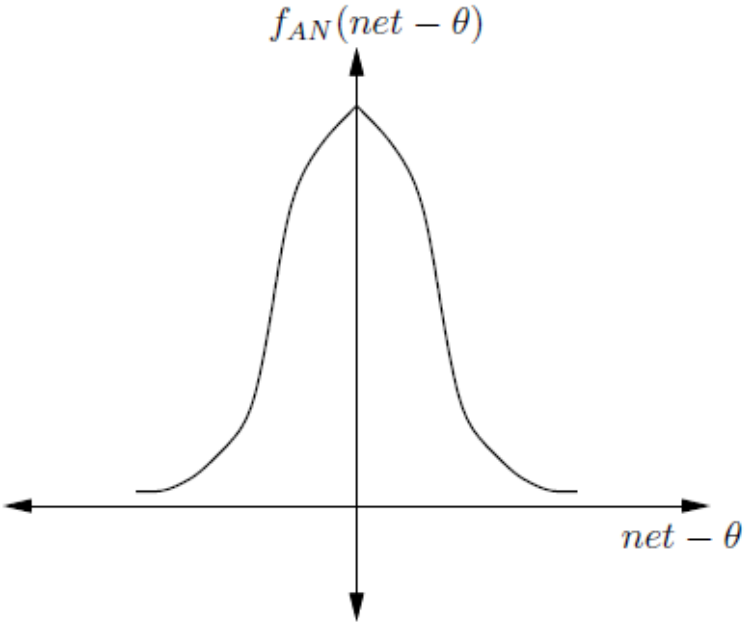
$$f_{AN}(net-\theta)=\frac{2}{1+e^{-\lambda(net-\theta)}}-1$$



常用的激活函数

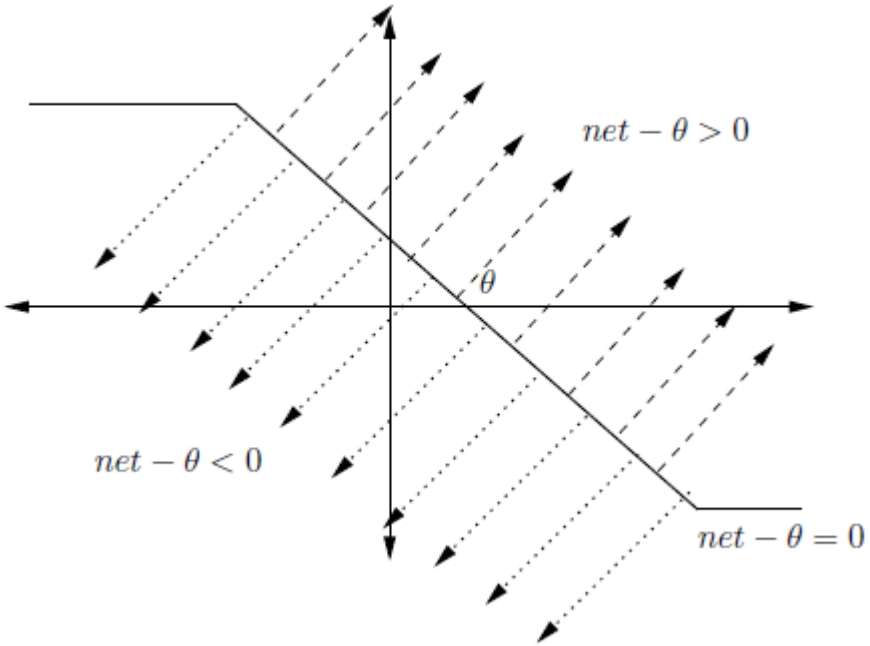
- 高斯函数
  - 其中，***net*－θ**是均值，σ是高斯分布的标准差。

$$f_{AN}(net-\theta)=e^{\frac{-(net-\theta)^2}{\sigma^2}}$$



几何构型

- 单个神经元可实现没有任何错误的线性可分函数。
  - 线性可分是指神经元可以通过一个*I*维超平面分割*I*维的输入向量空间，将超过阈值的响应和那些低于阈值的响应分割开。
- 例如，斜坡激活函数神经元的决策边界：



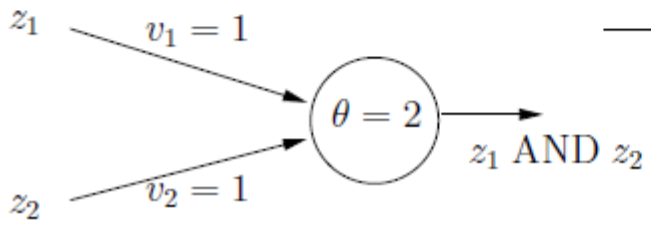
例1

- 用单个感知器实现布尔函数**AND**
  - 感知器是由美国计算机科学家罗森布拉特（**F. Roseblatt**）于**1957**年提出的，是一种最简单形式的前馈式人工神经网络，是一种线性分类器。

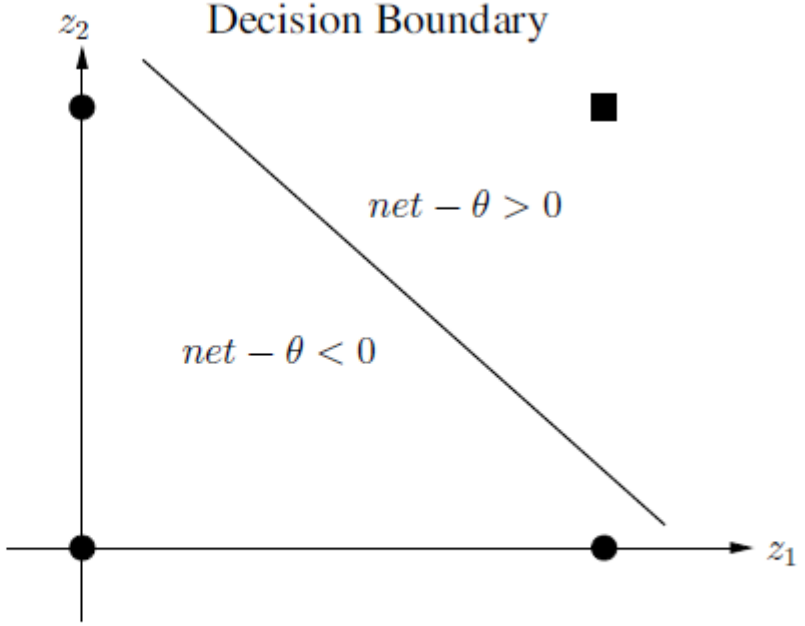
Truth Table

z_1	z_2	z_1 AND z_2
0	0	0
0	1	0
1	0	0
1	1	1

The Artificial Neuron



Decision Boundary



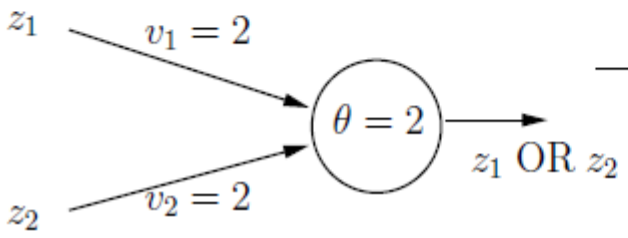
例2

- 用单个感知器实现布尔函数OR

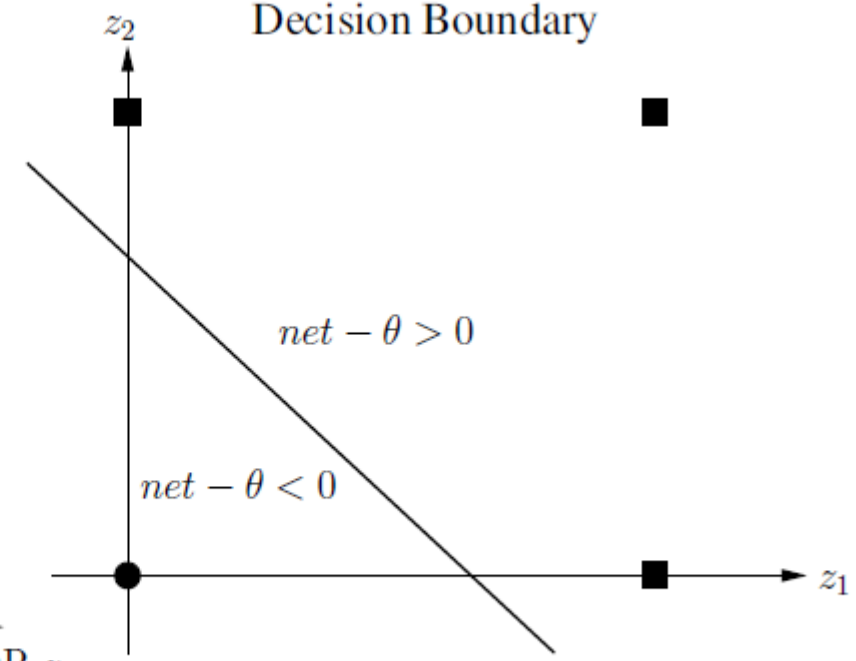
Truth Table

z_1	z_2	z_1 OR z_2
0	0	0
0	1	1
1	0	1
1	1	1

The Artificial Neuron



Decision Boundary



布尔函数AND和OR

- 布尔函数AND和OR都是线性可分函数的例子。
  - 对于这种简单函数，可以很容易手工决定偏置和权重的取值。

- 在给定输入信号以及θ的前提下，权值v_i的取值可以通过求解下式得到。

对于每个输入模式$z=(z_1,z_2,\cdots,z_I)$，应有

$$\sum_i z_i v_i-\theta>0 \text{或} \sum_i z_i v_i-\theta<0$$

思考：

- 如何实现异或函数？

Truth Table

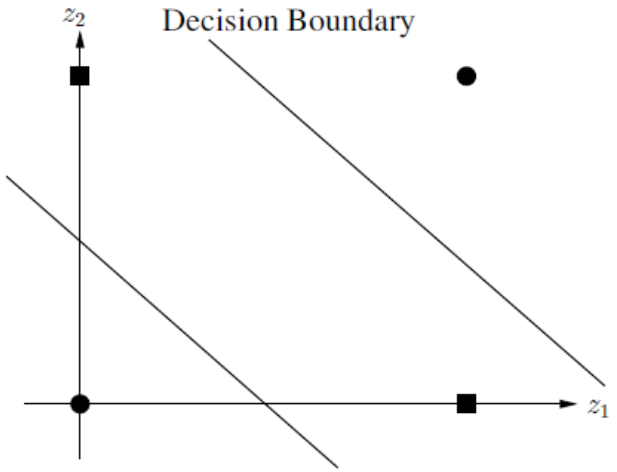
z_1	z_2	z_1 XOR z_2
0	0	0
0	1	1
1	0	1
1	1	0

布尔函数XOR

- 布尔函数XOR是非线性可分布尔函数。
  - 单个感知器不能实现该函数。
  - 若使用单个感知器，则所能获得的最佳正确率是75%。

Truth Table

z_1	z_2	z_1 XOR z_2
0	0	0
0	1	1
1	0	1
1	1	0



- 为了能够学习非线性可分的函数，需要一个由若干神经元组成的分层的 神经网络。
  - 例如，异或函数需要两个输入单元、两个隐层单元和一个输出单元。

学习

- 是否存在一个自动的方法来确定权值v_i和阈值θ的值？
  - 对于简单问题，可以很容易计算。
  - 但是，假定出来数据以外，并没有关于函数的任何先验知识，那么将如何计算v_i和θ的值呢？
    - 通过学习！

- 学习包括调整权值和阈值，直到满足要求。有三种主要的学习方法：
  - 监督学习
  - 无监督学习
  - 增强学习

学习方法

- 监督学习（Supervised Learning）
  - 提供训练集，目标是调整权值以使得神经元的真实输出和目标输出之间的误差最小化。
- 无监督学习（Unsupervised Learning）
  - 在没有外部辅助的情况下，从输入数据中发掘模式和特征。许多无监督的学习算法是对训练模式进行聚类。
- 增强学习（Reinforcement Learning）
  - 目标是对性能良好的神经元（或神经网络的一部分）给予奖励，而对性能不佳的神经元给予惩罚。

增广向量

- 人工神经元的网络输入信号（假定求和单元）：

$$net = \sum_{i=1}^{I} z_i v_i - \theta$$

$$= \sum_{i=1}^{I} z_i v_i + z_{i+1} v_{i+1}$$

$$= \sum_{i=1}^{I+1} z_i v_i$$

其中，$z_{i+1} = -1$，$v_{i+1} = \theta$。

- 输入向量被增广，从而使其包含一个成为偏置单元的额外输入单元z_{I+1}，而且权值v_{I+1}被用作阈值，目的是简化学习方程。

梯度下降学习规则

- 梯度下降法是最常用的方法。梯度下降法需要定义一个误差函数（目标函数）来衡量逼近目标时神经元的误差。

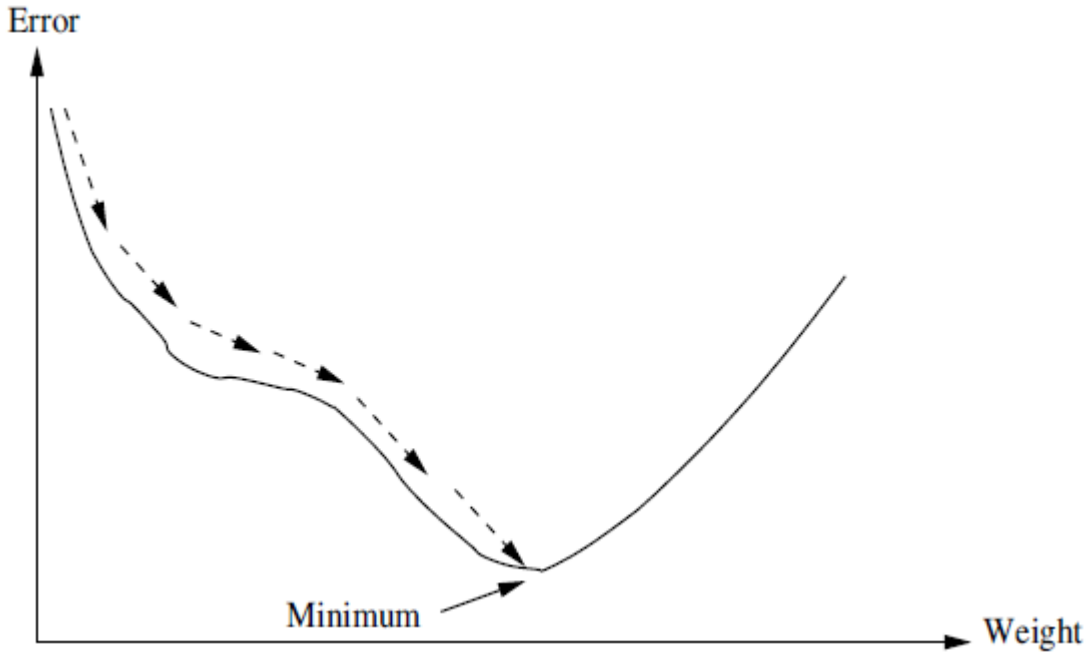
- 通常使用平方误差。

$$\varepsilon = \sum_{p=1}^{P_T} (t_p - o_p)^2$$

其中，t_p和o_p分别是第p个输入模式的目标输出和真实输出，P_T是训练集中输入—目标向量对（模式）的总数。

梯度下降学习规则

- 梯度下降的目标是找到使得ε最小的权值。
  - 这可以通过计算权值空间中ε的梯度，并沿负梯度方向移动权值向量来实现。



梯度下降学习规则

- 给定单个训练模式，一个神经元的权值更新公式为：

$$v_i(t) = v_i(t-1) + \Delta v_i(t)$$

$$\Delta v_i(t) = \eta\left(-\frac{\partial \varepsilon}{\partial v_i}\right)$$

$$\frac{\partial \varepsilon}{\partial v_i} = -2\left(t_p - o_p\right)\frac{\partial f}{\partial net_p} z_{i,p}$$

- η是学习率（即负梯度方向上所取的步长大小）。
- $z_{i,p}$是对应于模式p的第i个输入向量。
- 对于所有非连续的激活函数（例如阶跃和斜坡函数等），f相对于net_p（模式p的网络输入）的偏导数的计算存在问题。

Widrow-Hoff学习规则

假定$f = net_p$，则$\frac{\partial f}{\partial net_p} = 1$，有

$$\frac{\partial \varepsilon}{\partial v_i} = -2(t_p - o_p)z_{i,p}$$

那么权值更新公式为：

$$v_i(t) = v_i(t-1) + 2\eta(t_p - o_p)z_{i,p}$$

- **Widrow-Hoff学习规则，即最小均方（least-means-square）算法，是最早用于训练具有多个自适应线性神经元的分层神经网络算法之一。**
  - **这种网络通常称为Madaline。**

广义delta学习规则

- 广义delta学习规则是Widrow-Hoff学习规则的一个推广，该规则假定激活函数是可微的。

- 假定激活函数是Sigmoid函数，则

$$\frac{\partial f}{\partial net_p} = o_p(1-o_p)$$

$$\frac{\partial \varepsilon}{\partial v_i} = -2(t_p - o_p)o_p(1-o_p)z_{i,p}$$

注：$f_{AN}(net) = \dfrac{1}{1+e^{-net}}$

误差修正学习规则

- 误差修正学习规则使用二值激活函数，例如阶跃函数。权值仅当神经元引发错误时才被调整。

即，仅当$(t_p - o_p) = 1$或$(t_p - o_p) = -1$时，才使用下式调整。

$$v_i(t) = v_i(t-1) + 2\eta(t_p - o_p)z_{i,p}$$

小结

- 神经元的基本结构
- 人工神经元结构
  - 输入信号
  - 激活函数
  - 几何构型
  - 学习规则

习题

1. 下列哪一个布尔函数可以通过一个使用求和单元的单个神经元来实现？给出权值和阈值的取值来证明你的答案。

$(a)\ z_1 z_2 \bar{z}_3$

$(b)\ z_1 \bar{z}_2 + \bar{z}_1 z_2$

$(c)\ z_1 + z_2$

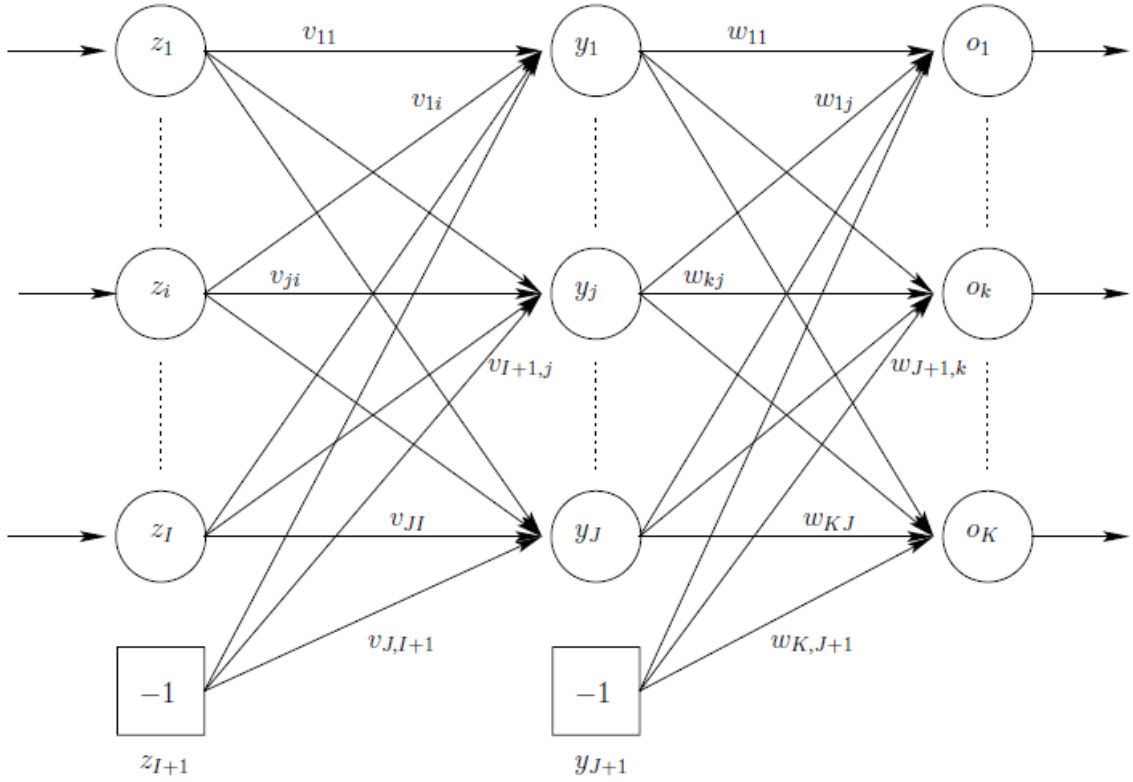
其中，$z_1 z_2$表示z_1和z_2取与操作，$z_1 + z_2$表示z_1和z_2取或操作，$\bar{z}_1$表示z_1取非操作。

本章内容

- 概述
- 神经网络的类型
- 监督学习规则
- 隐层单元的功能
- 集成神经网络

监督学习的例子

- 手写数字识别



神经元→神经网络

- 单个神经元所能够学习的函数类型有很多限制。
  - 一个实现求和单元的单个神经元仅仅能被用于线性可分函数。
- 对于非线性可分函数，需要一个分层的神经元网络。
- 训练分层的神经网络比训练单个神经元更为复杂。
  - 监督学习（本章）
  - 非监督学习
  - 增强学习

本章内容

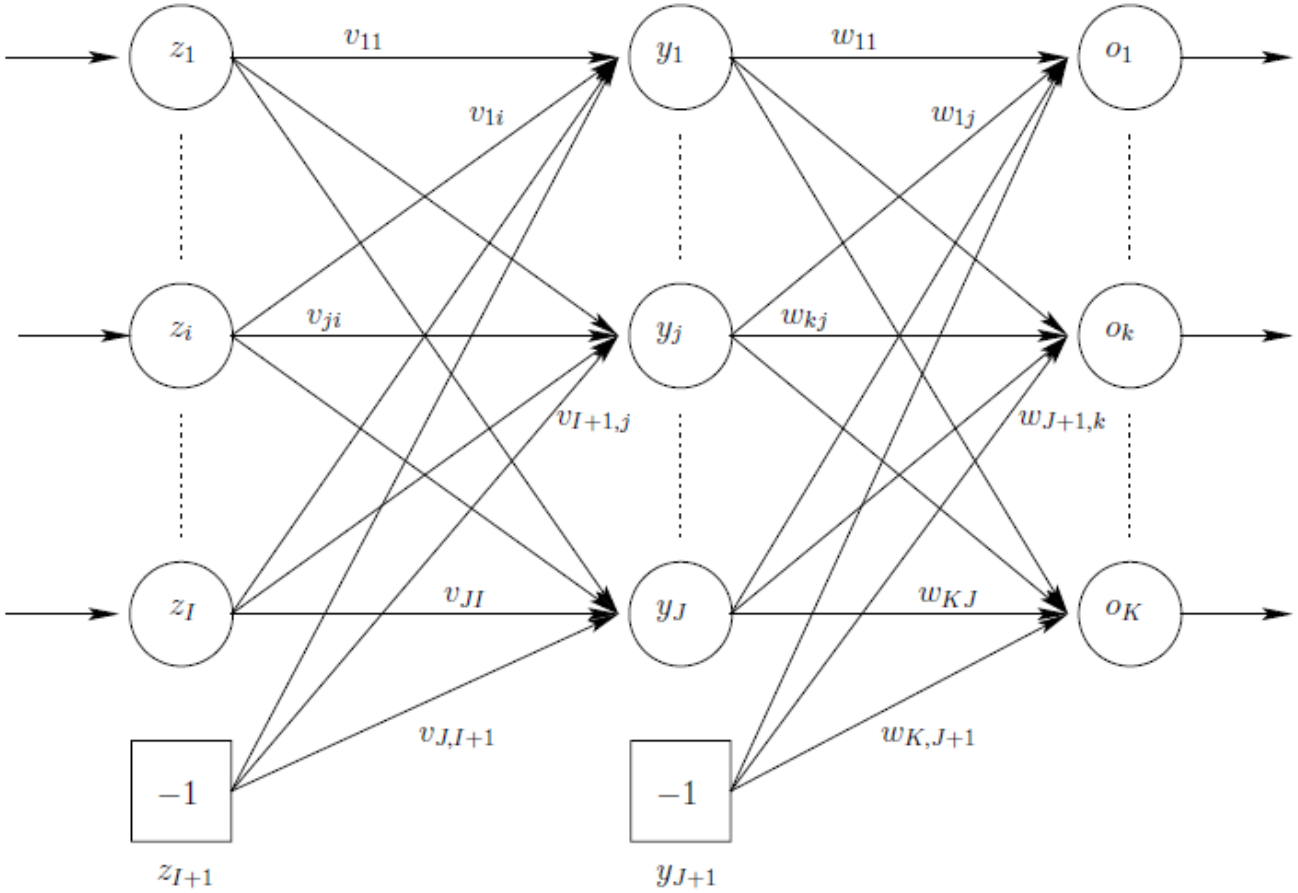
- 概述
- 神经网络的类型
- 监督学习规则
- 隐层单元的功能
- 集成神经网络

神经网络的类型

1. 前馈神经网络
2. 乘积单元神经网络
3. 简单反馈神经网络
4. 时延神经网络
5. 级联神经网络

前馈神经网络

- 标准的前馈神经网络
  - 一个输入层、一个隐层、一个输出层



前馈神经网络

- 对于任意给定的一个输入模式z_p，前馈神经网络的输出单元o_k的输出为

$$o_{k,p} = f_{o_k}(net_{o_{k,p}})$$

$$= f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}(net_{y_{j,p}})\right)$$

$$= f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}(\sum_{i=1}^{I+1} v_{ji} z_{i,p})\right)$$

f_{o_k}和f_{y_j}分别是输出单元o_k和隐层单元y_j的激活函数。

$z_{i,p}$是输入模式z_p的输入单元z_i的值。第$(I+1)$个输入单元和第$(J+1)$个隐层单元代表下一层中的神经元的阈值的偏置单元。

前馈神经网络

- 有些有关神经网络的文献，不将输入层记为一层。
- 一个前馈神经网络可以有多个隐层。
  - 已经证明了使用单调递增可微函数的单隐层前馈神经网络能够逼近任意的连续函数，只要隐层具有足够多的隐层神经元。
- 并不需要所有的神经元使用相同的激活函数。每个输入单元也可以实现一个激活函数。
- 一个前馈神经网络也可以在输入层和输出层之间建立直接（线性）连接）。

神经网络的类型

1. 前馈神经网络
2. 乘积单元神经网络
3. 简单反馈神经网络
4. 时延神经网络
5. 级联神经网络

乘积单元神经网络

- 乘积单元神经网络的神经元的网络输入是输入信号的加权乘积而不是加权和。

- 已有多种不同类型的乘积单元神经网络：
  ① 每个输入单元都连接到求和单元和一组乘积单元
  ② 乘积单元层和求和单元层相交替
  ③ ……

乘积单元神经网络

- 本节：隐层仅包括乘积单元，不含求和单元；输出层仅含有求和单元，并且假定网络中的所有神经元均使用线性激活函数。
  - 因此，对于每一个隐层单元y_j，其网络输入为：

不含偏置：

$$net_{y_{j,p}} = \prod_{i=1}^{I} z_{i,p}^{v_{ji}}$$

$$= \prod_{i=1}^{I} e^{v_{ji}\ln(z_{i,p})}$$

$$= e^{\sum_i v_{ji}\ln(z_{i,p})}$$

含失真度（distortion），对所有模式均有$z_{I+1,p}=-1$，$v_{j,I+1}$代表失真度：

$$net_{y_{j,p}} = \prod_{i=1}^{I+1} z_{i,p}^{v_{ji}}$$

引入失真度的目的是在训练中动态调整激活函数以使其更好地逼近训练数据所代表的真实函数。

神经网络的类型

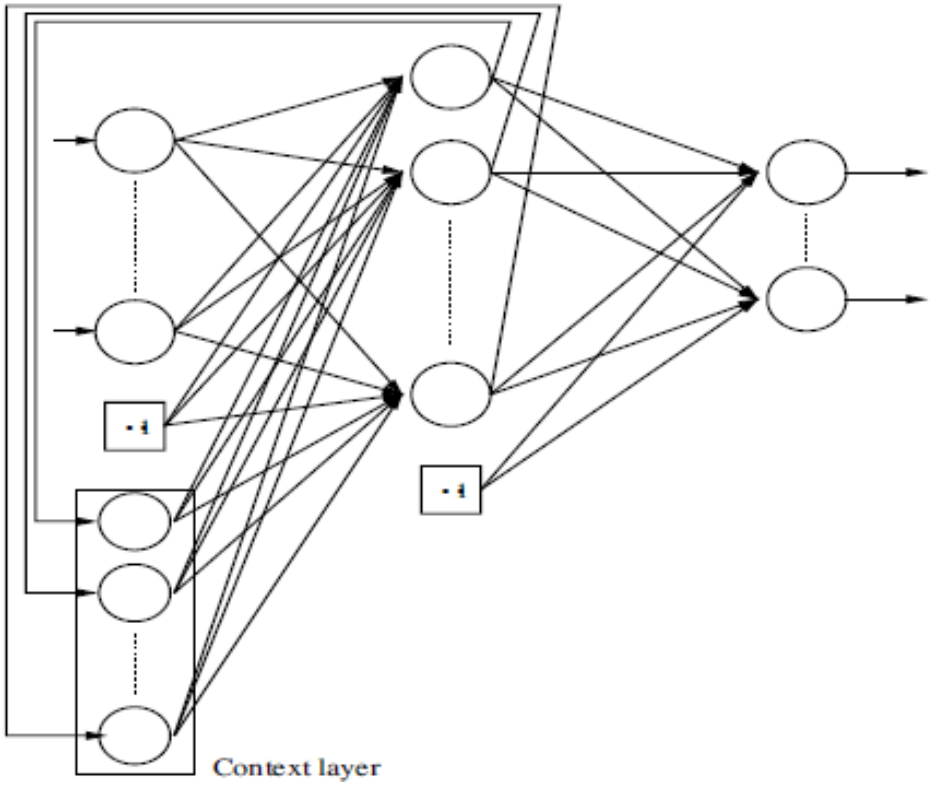
1. 前馈神经网络
2. 乘积单元神经网络
3. 简单反馈神经网络
4. 时延神经网络
5. 级联神经网络

简单反馈神经网络

- 简单反馈神经网络具有反馈连接。

- 已经建立了几个不同类型的简单反馈神经网络。其中，**Elman**和**Jordan**简单反馈神经网络是前馈神经网络的简单扩展。

Elman简单反馈神经网络



- 通过复制隐层得到环境层，而环境层作为输入层的一个扩展，给隐层提供表示先前网络状态的信号。
  - 环境层的目的是存储隐层的先前状态。

Elman简单反馈神经网络

- 环境单元z_{I+2}，…，z_{I+1+J}和所有隐层单元均为全连接。每一个隐层单元y_j（j=1,…,J)到其对应的环境单元z_{I+1+j}的连接权值为1。
  - 每一个输出单元的输出值为：

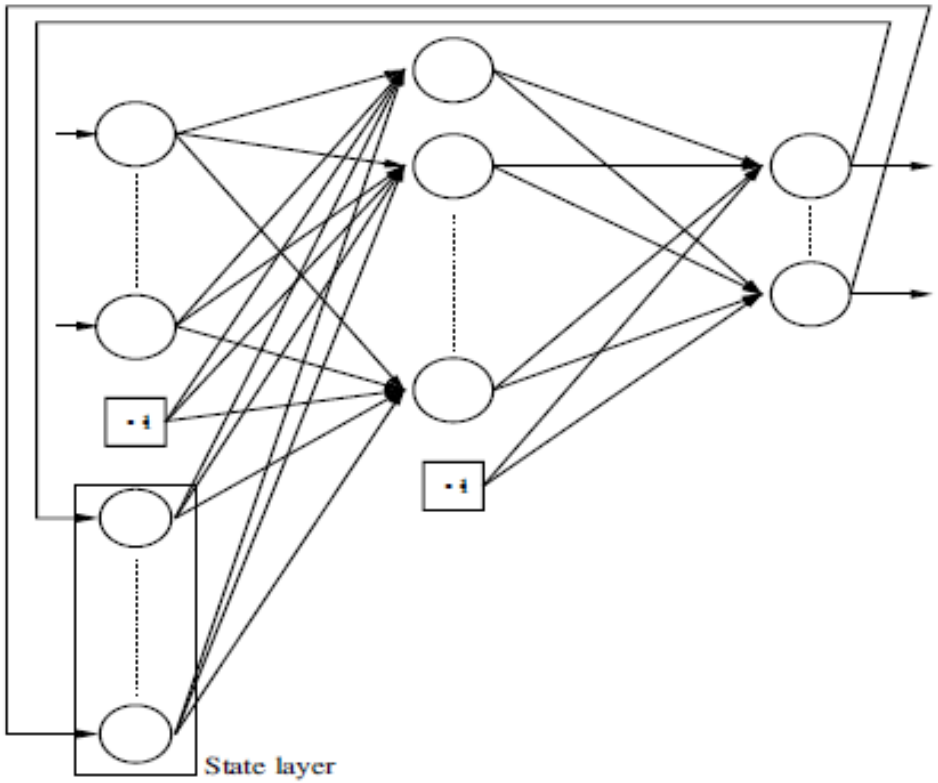
$$o_{k,p} = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}\left(\sum_{i=1}^{I+1+J} v_{ji} z_{i,p}\right)\right)$$

其中$(z_{I+2,p},\cdots,z_{I+1+J,p}) = (y_{1,p}(t-1),\cdots,y_{J,p}(t-1))$

- 每一个隐层单元y_j（j=1,…,J)到其对应的环境单元z_{I+1+j}的连接权值也可以不为1。
  - 先前状态的影响被加权。确定这些权值为训练步骤增加了额外的复杂度。

Jordan简单反馈神经网络

- 对输出层进行复制。
  - 复制的输出层称作状态层。



Jordan简单反馈神经网络

- 输入层扩展为：

$$(\underbrace{z_{1,p},\cdots,z_{I+1,p}}_{\text{实际输入}},\underbrace{z_{I+2,p},\cdots,z_{I+1+K,p}}_{\text{状态单元}})$$

- 对于每一个输出单元，其输出为：

$$o_{k,p}=f_{o_k}\left(\sum_{j=1}^{J+1}w_{kj}f_{y_j}\left(\sum_{i=1}^{I+1+K}v_{ji}z_{i,p}\right)\right)$$

其中$(z_{I+2,p},\cdots,z_{I+1+K,p})=(o_{1,p}(t-1),\cdots,o_{K,p}(t-1))$

神经网络的类型

1. 前馈神经网络
2. 乘积单元神经网络
3. 简单反馈神经网络
4. 时延神经网络
5. 级联神经网络

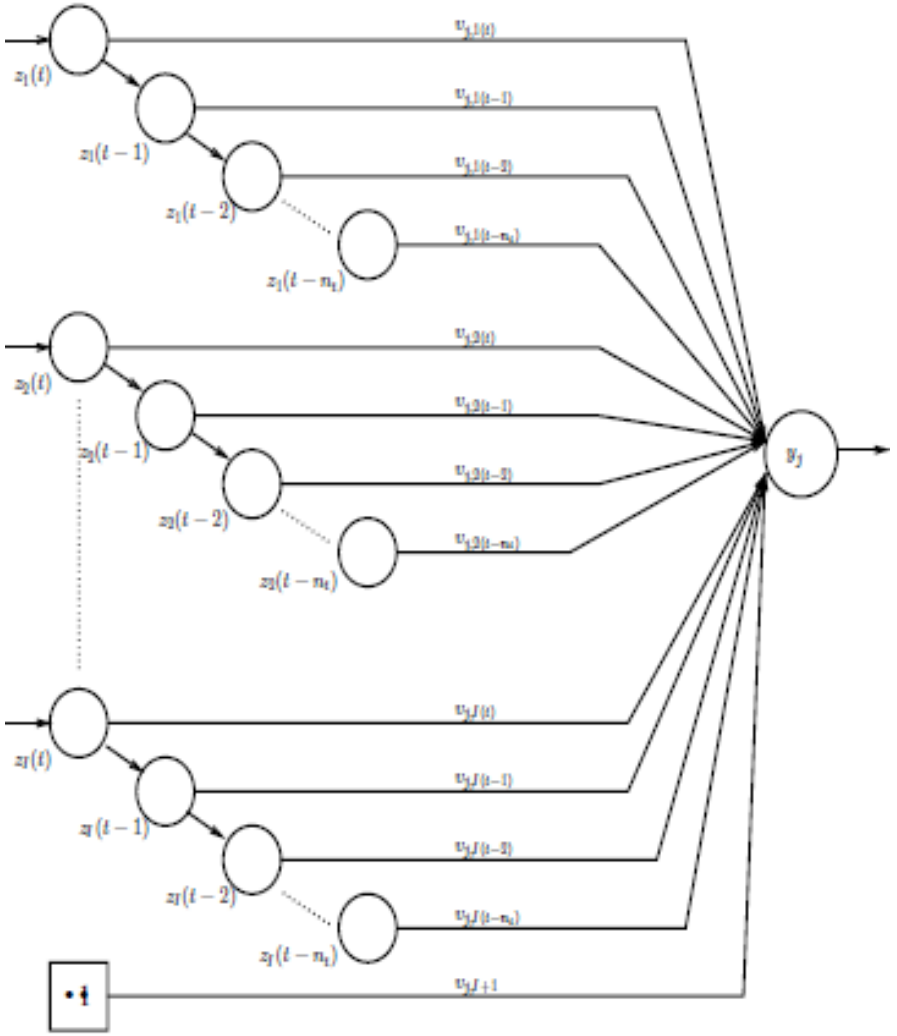
时延神经网络

- 时延神经网络是一个将其输入模式连续地延时的时域神经网络。

单个时延神经元

（每个输入都被n_t时延）

用于构造一个完整的前馈时延神经网络。



时延神经网络

初始时，仅$z_{i,p}(0)$有值，其他所有的$z_{i,p}(t-t')$（$i=1,\cdots,\mathrm{I};t'=1,\cdots,\mathrm{n_t}$）均为0。$\mathrm{n_t}$为时延模式的个数。

在第一个模式出现后并在给出第二个模式前，有$z_{i,p}(t-1)=z_{i,p}(t)$。

在t'个模式出现后并在给出$t'+1$个模式前，对于所有的$t=1,\cdots,t'$，有$z_{i,p}(t-t')=z_{i,p}(t-t'+1)$。

- **如此类推，共有n_t个模式影响权值的更新，从而使得时域特征可以 引导所学习到的函数曲线。**

时延神经网络

- 时延神经网络的输出为：

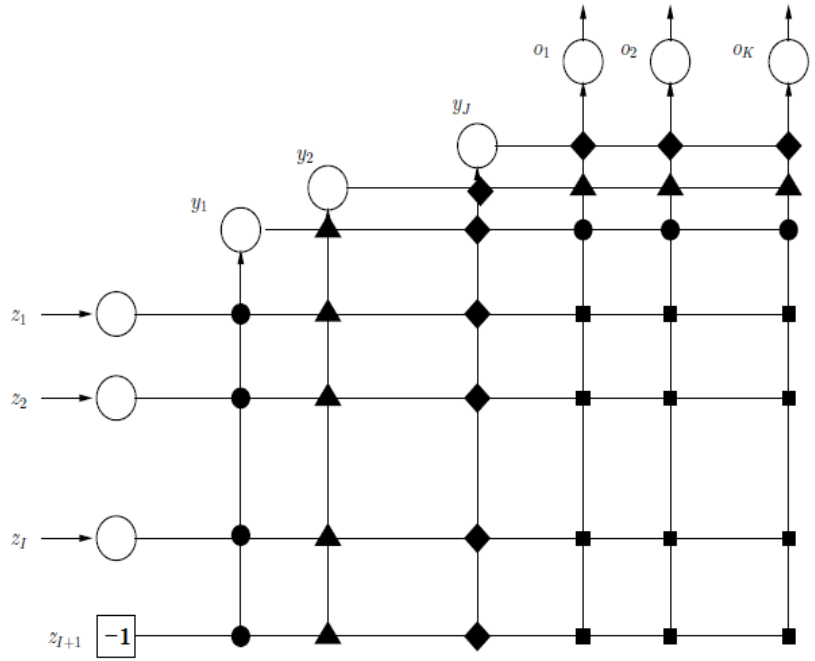
$$o_{k,p} = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{y_j}\left(\sum_{i=1}^{I}\sum_{t=0}^{n_t} v_{ji}(t) z_{i,p}(t) + z_{I+1} v_{j,I+1}\right)\right)$$

神经网络的类型

1. 前馈神经网络
2. 乘积单元神经网络
3. 简单反馈神经网络
4. 时延神经网络
5. 级联神经网络

级联神经网络

- 级联神经网络是一个多层的前馈神经网络，并且：
  - 所有的输入单元都对所有的隐层单元和输出单元建立了直接连接。
  - 隐层单元也是级联的，即每一个隐层单元的输出均作为后续所有隐层单元和输出单元的一个输入。



级联神经网络

- 级联神经网络的输出为：

$$o_{k,p} = f_{o_k}\left(\sum_{i=1}^{I+1} u_{ki} z_{i,p} + \sum_{j=1}^{J} w_{kj} f_{y_j}\left(\sum_{i=1}^{I+1} v_{ji} z_{i,p} + \sum_{l=1}^{j-1} s_{jl} y_l\right)\right)$$

其中，u_{ki}表示输出单元k和输入单元i之间的权重，s_{jl}是隐层单元j和l之间的权重，y_l是隐层单元l的输出。

本章内容

- 概述
- 神经网络的类型
- 监督学习规则
- 隐层单元的功能
- 集成神经网络

监督学习规则

- 监督学习问题
- 梯度下降优化
- 量化共轭梯度
- **LeapFrog优化**
- 粒子群优化

监督学习问题

考虑从平稳密度$\Omega(D)$采样的数据集

$$D=\{d_p=(z_p,t_p)\,|\,p=1,\cdots,P\}$$

其中，z_p是输入，t_p是对应的输出。

对于所有$i=1,\cdots,I$，$k=1,\cdots,K$，$z_{i,p}$是模式p对应输入单元z_i的值，$t_{k,p}$是输出单元o_k的目标值，有$z_{i,p},t_{k,p}\in R$。

根据信号加噪声模型，有：

$t_p=\mu(z_p)+\varsigma_p$，其中$\mu(z)$是未知函数；$\varsigma_{k,p}$是以密度$\phi(\varsigma)$采用的独立的、同分布的噪声样本，其均值为0。

- **学习的目标是使用有限集合D中包含的信息来逼近未知函数$\mu(z)$。**

监督学习问题

- 对于神经网络学习而言，通常将数据集D随机划分为训练集合D_T、验证集合D_V以及测试集合D_G，集合之间相互独立。
  - 对$\mu(z)$的逼近由训练集D_T建立（用于调节权值）。
  - 记忆（memorization）由D_V确定（用于最小化过拟合）。
  - 泛化精度则由测试集D_G进行估计（测试最终解证实网络的真实效果）。
- 对于多层神经网络，给定网络结构时，其搜索空间是由权值向量W所描述的搜索空间。
- 通常，关于Ω(D)的先验知识一无所知，因此神经网络学习器会使用非参数回归方法在其假设空间$\mathcal{H}$中搜索能对未知函数$\mu(z)$给出良好估计的函数$f_{NN}(D_T,W)$。

监督学习问题

- 最小化经验误差：

$$\varepsilon_T(D_T;\mathbf{W})=\frac{1}{P_T}\sum_{p=1}^{P_T}(f_{NN}(z_p,\mathrm{W})-t_p)^2$$

其中，P_T是训练模式的总数，函数F_{NN}：$\mathrm{R}^{\mathrm{I}}\rightarrow\mathrm{R}^{K}$。

- 期望：低的经验（训练）误差同样会给出低的真实误差，或泛化误差。

- 真实误差或泛化误差被定义为：

$$\varepsilon_T(\Omega;\mathbf{W})=\int(f_{NN}(z,\mathrm{W})-t)^2\,\mathrm{d}\Omega(z,t)$$

监督学习问题

- 训练神经网络的优化算法分3类：
  - 局部优化，例如梯度下降法、量化共轭梯度法等。
  - 全局优化，例如跳蛙（LeapFrog）、模拟退火、进化算法、粒子群优化等。
  - 局部优化和全局优化组合起来的混合训练算法。

- 根据何时更新权值，有两类监督学习算法。
  - 随机/在线学习：每出现一个模式，立即更新权值。为防止模式在训练集中的出现次序可能引起的误差，通常随机选取下一个输入模式。
  - 批/离线学习：仅当使用完所有的训练模式后，才使用累积变化来对权值进行调整。

梯度下降优化

- 从梯度下降优化方法导出了一个最流行的学习方法，即反向传播（**backpropagation**）。

- 一次学习迭代包括两个阶段：
  - 前馈传递：对于每一个训练模式简单地计算输出值。
  - 反向传播：从输出层向输入层反向传播误差信号。权值作为反向传播误差信号的函数被调整。

- 前馈神经网络
- 乘积单元神经网络

梯度下降优化—前馈神经网络

- 假定以误差平方和作为目标函数，则对于每一个模式z_p，有

$$\varepsilon_p = \frac{1}{2}\left(\frac{\sum_{k=1}^{K}(t_{k,p}-o_{k,p})^2}{K}\right)$$

其中，K是输出单元的个数，$t_{k,p}$和$o_{k,p}$分别是第k个输出单元的目标输出值和实际输出值。

以下推导均针对单个模式。因此，为了表示上的方便，将模式的下标p省略。同时，假定隐层和输出层均使用Sigmoid激活函数，使用增广向量，所有隐层和输出层单元均使用求和单元。

梯度下降优化—前馈神经网络

- 则：

$$o_k = f_{o_k}(net_{o_k}) = \frac{1}{1+e^{-net_{o_k}}},\ y_j = f_{y_j}(net_{y_j}) = \frac{1}{1+e^{-net_{y_j}}}$$

- 在随机学习中，权值根据下列等式进行更新：

$$w_{kj}(t)+= \Delta w_{kj}(t) + \alpha \Delta w_{kj}(t-1)$$

$$v_{ji}(t)+= \Delta v_{ji}(t) + \alpha \Delta v_{ji}(t-1)$$

其中，α是冲量。

- 下面将推导计算$\triangle w_{kj}(t)$和$\triangle v_{ji}(t)$等式。为了表示上的便捷，推导中把表示时间的符号t省略。

梯度下降优化—前馈神经网络

- 因为

$$o_k = f_{o_k}(net_{o_k}) = \frac{1}{1+e^{-net_{o_k}}}$$

- 有

$$\frac{\partial o_k}{\partial net_{o_k}} = \frac{\partial f_{o_k}}{\partial net_{o_k}} = (1-o_k)o_k = f'_{o_k},$$

（注：f'_{o_k}是对应激活函数的导数）

$$\frac{\partial net_{o_k}}{\partial w_{kj}} = \frac{\partial}{\partial w_{kj}}\left(\sum_{j=1}^{J+1} w_{kj} y_j\right) = y_j,$$

$$\frac{\partial o_k}{\partial w_{kj}} = \frac{\partial o_k}{\partial net_{o_k}} \cdot \frac{\partial net_{o_k}}{\partial w_{kj}} = (1-o_k)o_k y_j = f'_{o_k} y_j$$

梯度下降优化—前馈神经网络

- 又因为

$$\varepsilon_p = \frac{1}{2}\left(\frac{\sum_{k=1}^{K}(t_{k,p}-o_{k,p})^2}{K}\right), \quad 记E = \frac{1}{2}\sum_{k=1}^{K}(t_k-o_k)^2$$

- 因此有

$$\frac{\partial E}{\partial o_k} = \frac{\partial}{\partial o_k}\left(\frac{1}{2}\sum_{k=1}^{K}(t_k-o_k)^2\right) = -(t_k-o_k)$$

梯度下降优化—前馈神经网络

定义需要被反向传播的输出误差为$\delta_{o_k} = \frac{\partial E}{\partial net_{o_k}}$，则

$$\delta_{o_k} = \frac{\partial E}{\partial net_{o_k}} = \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial net_{o_k}}$$

$$= -(t_k - o_k)(1 - o_k)o_k = -(t_k - o_k) f'_{o_k}$$

因此，从隐层到输出层权值的改变可计算为：

$$\Delta w_{kj} = \eta\left(-\frac{\partial E}{\partial w_{kj}}\right) = -\eta \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial w_{kj}} = -\eta \times \left(-(t_k - o_k)\right) \times f'_{o_k} y_j$$

$$= -\eta \delta_{o_k} y_j$$

梯度下降优化—前馈神经网络

- 进一步，分析输入层到隐层的权值变化量：

$$\frac{\partial y_j}{\partial net_{y_j}} = \frac{\partial f_{y_j}}{\partial net_{y_j}} = (1-y_j)y_j = f'_{y_j}$$

$$\frac{\partial net_{y_j}}{\partial v_{ji}} = \frac{\partial}{\partial v_{ji}}\left(\sum_{i=1}^{I+1} v_{ji} z_i\right) = z_i$$

$$\frac{\partial y_j}{\partial v_{ji}} = \frac{\partial y_j}{\partial net_{y_j}} \cdot \frac{\partial net_{y_j}}{\partial v_{ji}} = (1-y_j)y_j z_i = f'_{y_j} z_i$$

$$\frac{\partial net_{o_k}}{\partial y_j} = \frac{\partial}{\partial y_j}\left(\sum_{j=1}^{J+1} w_{kj} y_j\right) = w_{kj},$$

梯度下降优化—前馈神经网络

$$\frac{\partial E}{\partial y_j}=\frac{\partial}{\partial y_j}\left(\frac{1}{2}\sum_{k=1}^{K}(t_k-o_k)^2\right)=\sum_{k=1}^{K}\frac{\partial E}{\partial o_k}\cdot\frac{\partial o_k}{\partial net_{o_k}}\cdot\frac{\partial net_{o_k}}{\partial y_j}$$

$$=\sum_{k=1}^{K}\frac{\partial E}{\partial net_{o_k}}\cdot\frac{\partial net_{o_k}}{\partial y_j}=\sum_{k=1}^{K}\delta_{o_k}w_{kj}$$

- 将需要反向传播的隐层误差定义为：

$$\delta_{y_j}=\frac{\partial E}{\partial net_{y_j}}=\frac{\partial E}{\partial y_j}\cdot\frac{\partial y_j}{\partial net_{y_j}}=\sum_{k=1}^{K}\delta_{o_k}w_{kj}f'_{y_j}$$

- 最终，输入层到隐层权值的变化量为：

$$\Delta v_{ji}=\eta\left(-\frac{\partial E}{\partial v_{ji}}\right)=-\eta\frac{\partial E}{\partial y_j}\cdot\frac{\partial y_j}{\partial v_{ji}}=-\eta\delta_{y_j}z_i$$

梯度下降优化—前馈神经网络

- 如果包含输入层到输出层的直接权值，则需要下列额外的权值更新：

$$\Delta u_{ki} = \eta\left(-\frac{\partial E}{\partial u_{ki}}\right) = -\eta\frac{\partial E}{\partial o_k}\cdot\frac{\partial o_k}{\partial u_{ki}} = -\eta\delta_{o_k} z_i$$

其中，u_{ki}是第i个输入单元到第k个输出单元的权值。

- 如果是批学习，权值按照下式更新的同时

$$w_{kj}(t)+= \Delta w_{kj}(t) + \alpha\Delta w_{kj}(t-1)$$

$$v_{ji}(t)+= \Delta v_{ji}(t) + \alpha\Delta v_{ji}(t-1)$$

且：$\Delta w_{kj}(t) = \sum_{p=1}^{P_T}\Delta w_{kj,p}(t),\quad \Delta v_{ji}(t) = \sum_{p=1}^{P_T}\Delta v_{ji,p}(t)$

其中$\Delta w_{kj,p}(t)$和$\Delta v_{ji,p}(t)$是对于单个模式p的权值改变。

梯度下降优化—前馈神经网络

//随机梯度下降学习算法

初始化权值、η、α以及学习迭代次数，令t=0；

While 终止条件不满足 **do**

设ε_T=0；

for 每一个训练模式p **do**

在前馈阶段计算$y_{j,p}$和$o_{k,p}$（其中，j=1,…,J, k=1,…,K）

计算输出误差信号$\delta_{o_{k,p}}$和隐层误差信号$\delta_{y_{j,p}}$；

调整权值w_{kj}和v_{ji}（误差反向传播）；

$$\varepsilon_T += [\varepsilon_p = \sum_{k=1}^{K}(t_{k,p} - o_{k,p})^2];$$

end

t=t+1；

end

梯度下降优化—前馈神经网络

- 随机梯度下降学习算法的终止准则包括：
  ① 当超过学习迭代的最大次数时终止。
  ② 当训练集的均方误差MSE足够小时停止（或方根均方误差足够小）。

$$\varepsilon_T = \frac{\sum_{p=1}^{P_T}\sum_{k=1}^{K}(t_{k,p}-o_{k,p})^2}{P_T K}$$

  ③ 当观察到过拟合时终止，即当训练数据被记忆时终止。

过拟合的一个现象是：$\varepsilon_V > \bar{\varepsilon}_V + \delta_{\varepsilon_V}$，其中是$\bar{\varepsilon}_V$在先前学习迭代中的平均验证误差，$\delta_{\varepsilon_V}$是验证误差的标准差。

梯度下降优化—前馈神经网络

- 可以直接将梯度下降优化应用于前馈神经网络、简单反馈神经网络和时延神经网络。
  - 具体的权值更新公式的推导？

- 乘积单元神经网络的梯度下降学习？

梯度下降优化—乘积单元神经网络

- 推导示例：仅在隐层中使用乘积单元的学习等式，并假定使用梯度下降优化和线性激活函数。
  - 因为只有输入层到隐层的等式改变，因此仅给出这部分权值更新等式的推导。
- 权值v_{ji}的变化量 Δv_{ji}为：

$$\Delta v_{ji} = \eta\left(-\frac{\partial E}{\partial v_{ji}}\right) = -\eta\frac{\partial E}{\partial net_{y_{j,p}}}\frac{\partial net_{y_{j,p}}}{\partial v_{ji}} = -\eta\delta_{y_{j,p}}\frac{\partial net_{y_{j,p}}}{\partial v_{ji}}$$

其中，$\delta_{y_{j,p}}$是误差信号，其计算方式与求和单元相同。

且
$$\frac{\partial net_{y_{j,p}}}{\partial v_{ji}} = \frac{\partial}{\partial v_{ji}}\left(\prod_{i=1}^{I} z_{i,p}^{v_{ji}}\right) = \frac{\partial}{\partial v_{ji}}\left(e^{\rho_{j,p}}\cos(\pi\phi_{j,p})\right)$$

$$= e^{\rho_{j,p}}\left[\ln\left|z_{i,p}\right|\cos(\pi\phi_{j,p}) - \Gamma_i\pi\sin(\pi\phi_{j,p})\right]$$

梯度下降优化—乘积单元神经网络

- **Durbin和Rumelhart的工作表明：单个乘积单元的信息容量（ information capacity）近似为3I，而单个求和单元的信息容量为2I，其中I为该单元输入的个数。**
  - **信息容量：能够学习的随机布尔模式数。**

- **更大的容量意味着，与求和单元相比，用乘积单元逼近相同的函数需要更少的处理单元。**

Function	SUs	PUs
$f(z) = z^2$	2	1
$f(z) = z^6$	3	1
$f(z) = z^2 + z^5$	3	2
$f(z_1, z_2) = z_1^3 z_2^7 - 0.5 z_1^6$	8	2

梯度下降优化—乘积单元神经网络

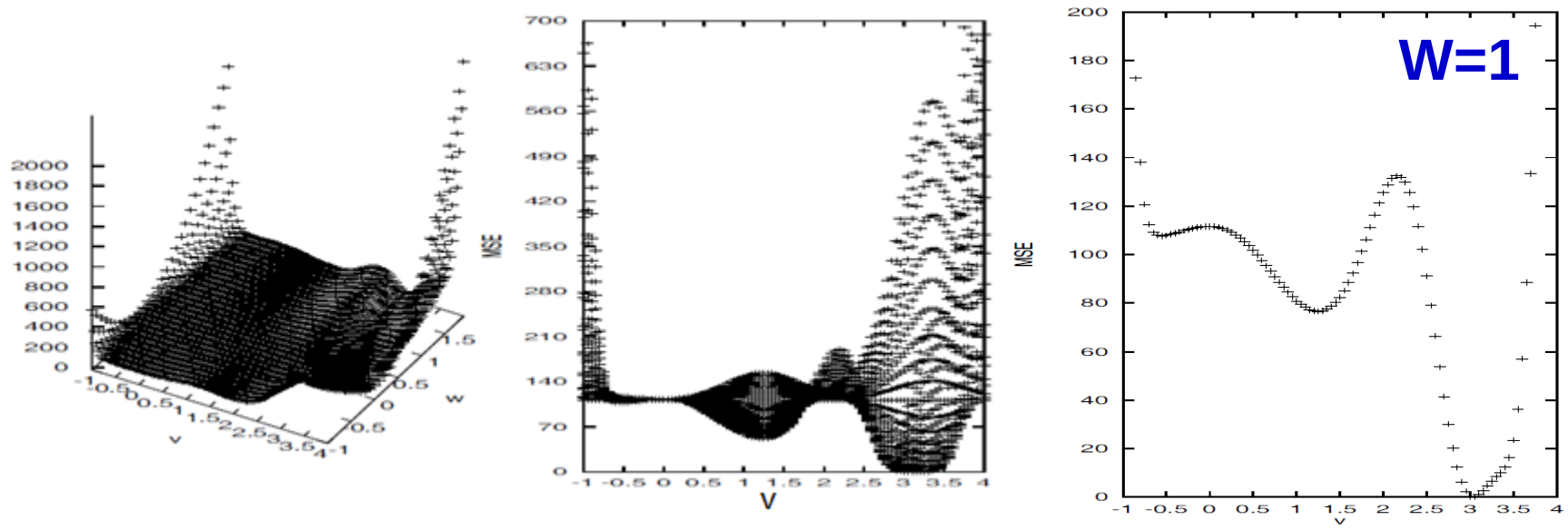
- 乘积单元神经网络：能够提供更小的网络结构。
  - 缺点：增加了局部最小值的数量以及深的峡谷。
  - 因此，乘积单元的搜索空间极其复杂。

- 但是，梯度下降在搜索空间相对平滑时效果较好。

- **Leerink**等人的工作表明了**6**位奇偶问题不能够使用梯度下降乘积单元来训练。两个的原因导致失败：
  - 权值初始化
  - 局部最小值的存在

梯度下降优化—乘积单元神经网络

- 以函数$f(z)=z^3$为例。
  - 仅需一个乘积单元，即1-1-1结构的神经网络，输入层到隐层的权值为3，隐层到输出层的权值为1。



- 需用全局随机优化算法以便能在搜索空间中更广阔的区域进行搜索，而不是过分依赖梯度信息。例如，模拟退火，遗传算法，粒子群优化，LeapFrog等。

量化共轭梯度

- 量化共轭梯度：**Scaled Conjugate Gradient**

- 共轭梯度优化是在梯度下降的简单性和牛顿法的快速二次收敛性之间的一种平衡。

- 共轭梯度神经网络训练算法
- **Fletcher-Reeves**共轭梯度算法
- 量化共轭梯度算法

LeapFrog优化和粒子群优化

- **LeapFrog优化**
  - 基于n维守恒力场中单位质量例子运动物理问题的一个优化方法。

- 粒子群优化
  - 一种概率的、基于群体的搜索方法。

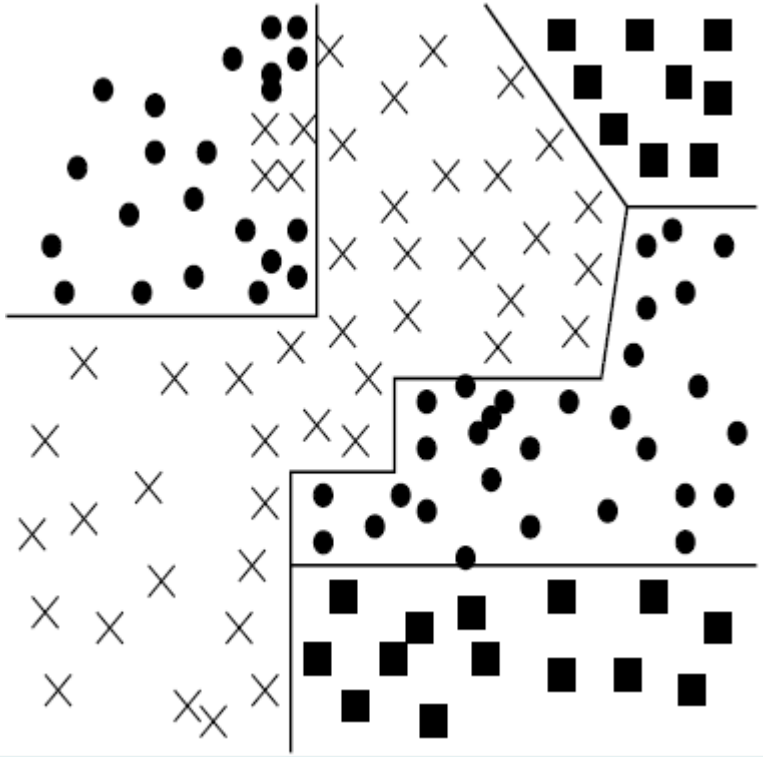
本章内容

- 概述
- 神经网络的类型
- 监督学习规则
- 隐层单元的功能
- 集成神经网络

隐层单元的功能

- 分类问题：假定是二维问题，使用单隐层标准前馈神经网络，隐层单元的任务是形成决策边界来分隔不同的类。

- 例如，右图所示的三类问题
  - 10个边界
  - 10个隐层单元
  - 左上角有分类误差，需增加3个隐层单元
  - 隐层单元减少，误差增加

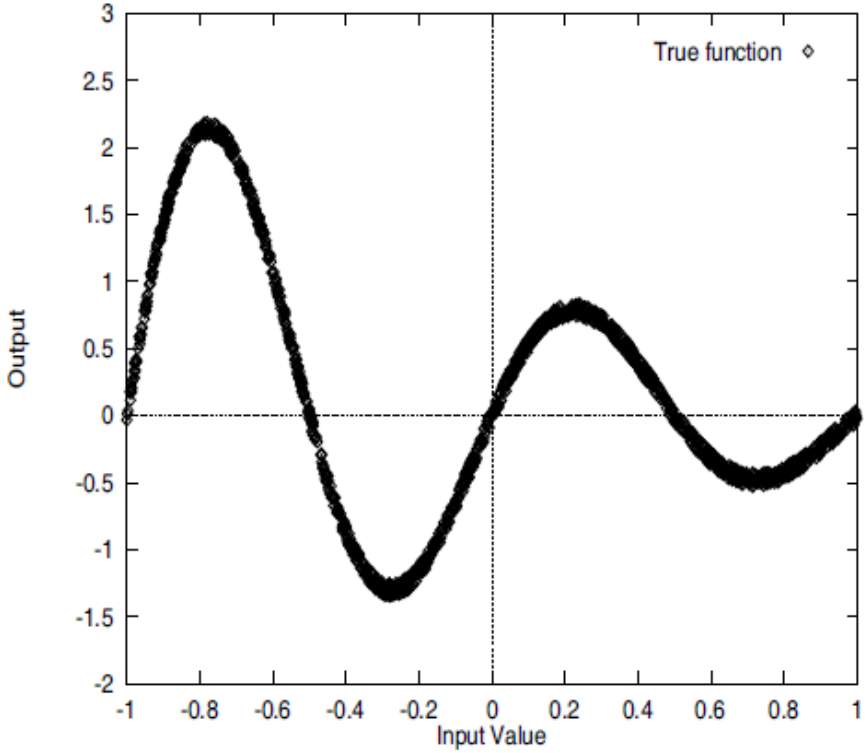


如何能在不使用关于输入空间的任何先验知识的前提下确定隐层单元的个数？

隐层单元的功能

- 函数逼近：假定是一维函数，使用单隐层标准前馈神经网络。

- 右图所示的一维函数，需要5个使用Sigmoid激活函数的隐层单元来学习该函数。对于目标函数的每一个拐点，需要一个Sigmoid函数进行拟合。隐层隐层单元的个数是拐点的个数加1.
- 采用线性激活函数的隐层单元执行同样的任务。为了达到与使用Sigmoid函数一样的正确率，需要更多的线性激活函数来学习该函数。



本章内容

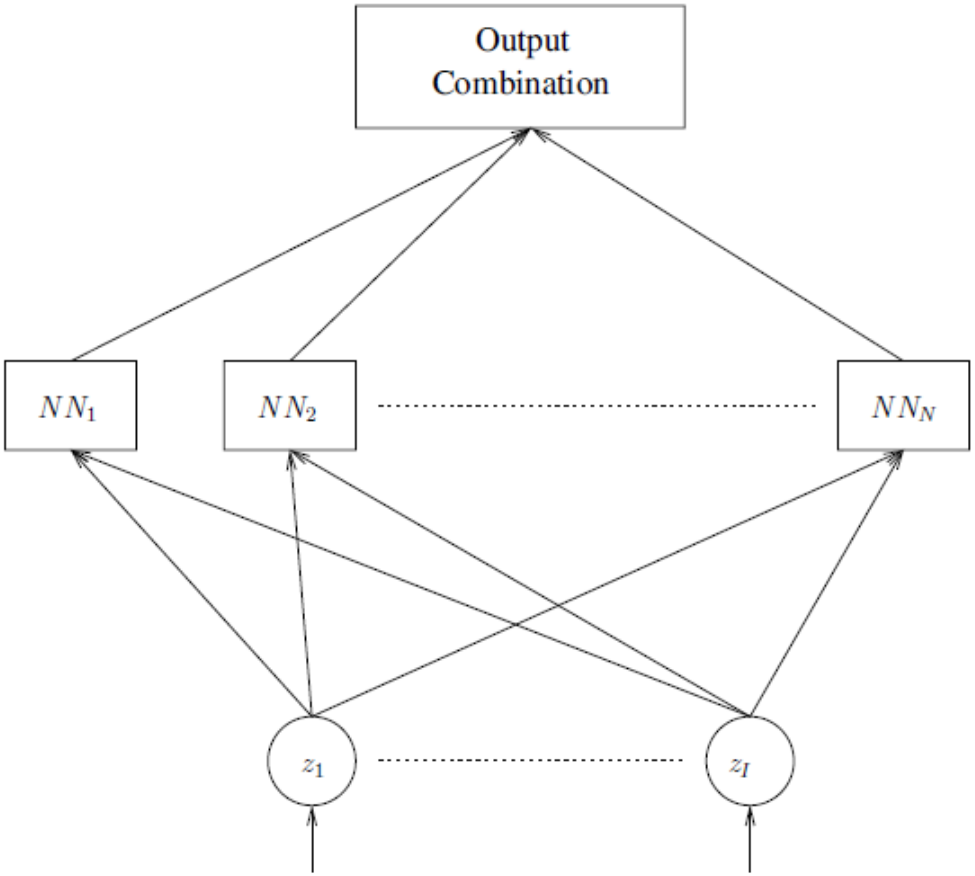
- 概述
- 神经网络的类型
- 监督学习规则
- 隐层单元的功能
- 集成神经网络

集成神经网络

- 神经网络的训练通过随机选择的初始权值开始。每一次在在同一数据集上重新训练一个神经网络时，都可能得到一个不同的结果。

- 神经网络的这一问题推动了集成神经网络的发展。
  - 集成神经网络的目标是通过一些不同类型网络的组合来优化结果，这些神经网络将在同样的任务上进行训练。

集成神经网络



- 仅当集成成员之间存在不一致，或者成员在搜索空间的不同部分产生误差时，集成神经网络才有意义。

集成神经网络

- 集成网络的最终结果可通过不同的方式计算。

- 例如：
  - 在集成网络中选择给出最佳泛化性能的神经网络。
  - 对集成网络所有成员的输出求平均值。
  - 对集成网络中所有神经网络的输出进行线性组合。在这种情况下，可每一个神经网络分配一个权值作为其可靠性的指示。

集成神经网络

- 以彼此独立的方式训练个体（基本的 神经网络）
  - 例如，每个神经网络使用不同的训练数据集。
  - 典型的方法是**Bagging**（一种**Bootstrap**集成方法）。

- **Bootstrap**集成方法是通过在原始训练集的一个随机分布上训练每一个网络成员，创建起集成网络的个体。如果原始训练集包含P_T个模式，则对于每一个集成神经网络的成员，其由P_T个模式构成的数据集是从原始训练集随机采样的（有可能重复）。

集成神经网络

- 协作式集成策略
  - 不同类型的神经网络在训练阶段交换它们的经验和知识。
  - 例如，**Boosting**策略。

- **Boosting策略：**按次序训练参与集成的神经网络个体。已经训练好的成员将模式分为简单模式和困难模式，未训练的新成员更多地关注于先前训练好的成员所确认的困难样本。

小结

- 神经网络的类型
  - 熟悉典型的神经网络。
- 监督学习规则
  - 了解简单学习问题，熟悉其基本学习方法。
- 隐层单元的功能
  - 对隐层单元的功能有所了解。
- 集成神经网络
  - 了解基本思想。

习题

1. 对于一个在输入层和输出层之间使用直接连接的前馈神经网络，给出$o_{k,p}$的表达式。

2. 假定使用梯度下降作为优化算法，推导Elman简单反馈神经网络的学习等式。