M4 - 神经网络：巨量并行，智慧无限

Neural Networks

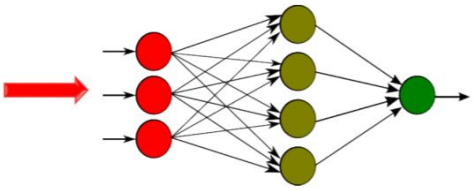
Lecturer: Dr. Bo Yuan

E-mail: yuanb@sz.tsinghua.edu.cn

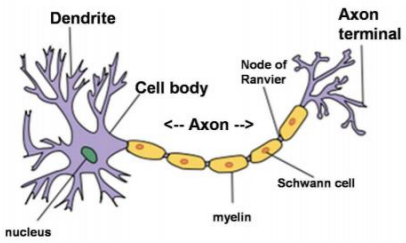
同学们好那这个单元我们给同学们简单的介绍 一下神经网络 就是大名鼎鼎的artificial neural networks神经网络 是博大精深我记得 我上本科的 时候就看到过这种神经网络的 著作大开本 这么厚所以 今天我们只是蜻蜓点水 介绍一些神经网络最一般的 一些概念给同学们介绍一下这是 一个什么样的一个分类器它的 基本的训练学习这些方法有 什么特点它有的优点缺点等等 这样同学们以后可以根据自己 的需要去做一些延伸的学习 特别是现在这个深度学习非常 火爆那它们基础它就是传统的 这个多层感知机

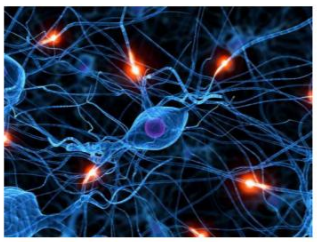
4.1 智慧之源神经元

Overview



Biological Motivation





- 10^{11}: The number of neurons in the human brain
- 10^{4}: The average number of connections of each neuron
- 10^{-3}: The fastest switching time of neurons
- 10^{-10}: The switching speed of computers
- 10^{-1}: The time required to visually recognize your mother

神经网络是一种和自然界联系相对比较 紧密的这样的一种分类器其实 它的生物界中的一个一个体现 就是我们的大脑我们叫一个 大脑当中有很多的一个神经元 而神经网络就是对大脑做了 一个高度的一个抽象和简化我们 的神经网络和大脑当中的神经 网络相比那是就说简单的非常多 所以我们的神经网络一般来讲是 这种有输入有输出神经元之间 还有一些联系是这样的一个 模型当然神经网络有各种各样的 我们展示的是一个feedforward的是一种 前馈性网络.

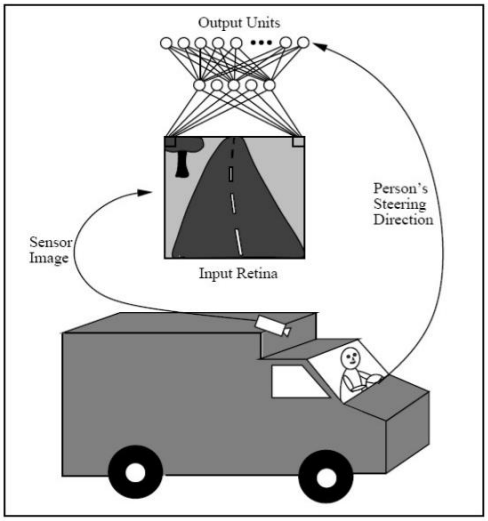
大家还记得以前这个生物学课本上学过的关于神经元的 讲了一个图这有什么轴突 有细胞核等等的它一边是 可以去接生这种从其他神经元 传递过来的一些刺激信号然后 可以传导然后再连到别的神经元 这是它基本的一个工作构架就为 数众多的massive的这样一个 神经元互相去传递信号神经元 可以是处于激活状态比如它收集 到的信号足够的强大它就会被 激活他也可能会处于一种抑制 状态这里给大家列了一些数字 第一个数字就是人脑当中 神经元的个数这个它没有 一个特别统一的一个这样的一个 计数吧反正大体上是这样的一个 数量级反正是非常多还几百亿 这样的神经元都是有可能存在 也存在大脑当中 每一个神经元 可能和比如说上万的这样其他 的神经元去进行互联所以大家 可以想象非常多的神经元神经元 和神经元之间又是这样密切的 这样一个联系就是这是大脑当中 神经元基本上是这样一个构架 但是神经元它比较慢 10的-3 次方秒大概就是这样的一个量级 就是它的一个开关速度就是说它 从零变一一变零你这样可以去 理解所以它的计算速度是比较 慢的而计算机的速度是非常快的 等10的-10次方将比如我们 现在计算机的速度都是Gfloats 都是以每秒10亿次运算以 这个为单位可能是几十个Gfloats 所以说元器件的计算速度 要远远高于神经元的计算速度 神经元因为它这个切换速度很慢 它就是10的-3次方秒这样的 一个量级但是你只需要可能 零点一秒就可以认出你妈来 就说人脑的神经元虽然说它每 一个神经元比较弱它不如 计算机那么强悍但是它联系起来 以后它其实可以实现非常强悍的 一个功能

总结:

Biological Motivation

- The power of parallelism
- The information processing abilities of biological neural systems follow from highly parallel processes operating on representations that are distributed over many neurons.
- The motivation of ANN is to capture this kind of highly parallel computation based on distributed representations.
- Group A
  - Using ANN to study and model biological learning processes.
- Group B
  - Obtaining highly effective machine learning algorithms, regardless of how closely these algorithms mimic biological processes.

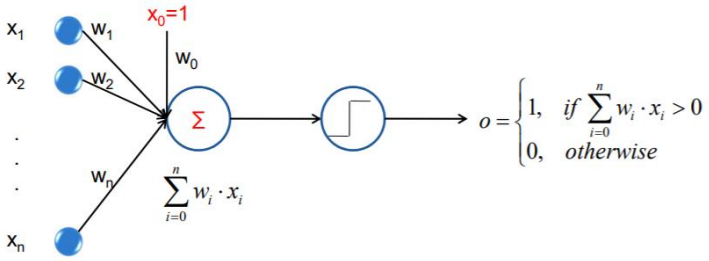
Case Study



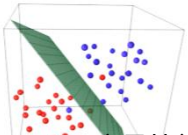
这个power来自于什么就来自于这种并行处理其实我们 可以想象人脑和计算机有多少 区别是吧计算机当中它怎么存储 数据的比如说给你拍张照片然后 存在计算机当中它的硬盘上它是 线性存储的 如果它有足够的 空间它就是一个一个挨着存的 就是这些比特位 所以我是可以说 我刚才拍的那张照片我存在了 计算机当中这个硬盘上面的从哪 一个扇区哪一个词道开始的第几 个上去到第几个扇区我都可以说 得一清二楚的然后我不需要这张 照片了OK把它delete 我会把它删除掉非常简单但人脑 可以吗 你们看到我以后看到我 这样的一个照片在你的大脑当中 你会留下某种印象 但是你能说得 清你是留在哪里了吗 而且你能 轻易的说我要忘掉一个人吗 这个 都是很难做到的所以说大脑的 这种存储记忆处理都是一种 分布式的 你不能说是哪一个神经 元进入了某一张图片的哪一个 像素 它是没有这样的一个对应 关系的 所以它整体全部都是 分布式的 它计算的时候也是 分布式 所以说我们用神经网络 就是要去模拟人脑的这种巨量 的一种并行的知识的表达存储 处理加工这样的一个能力 所以 神经网络他可以有两种用途 一种用途就是从cognitive science 这种角度来讲我们可以用神经 网络去理解人们是怎么去 思考的这是一种思路但是对于 我们来讲我们只是希望去从 自然界中 从人的大脑的构造 当中去寻找一些灵感 这样我 可以去构建一些比较强大的 一个分类器 其实说这个分类器跟 人脑的神经元之间有多么密切的 关系 这个不是特别考虑的一个 范畴但是我只需要希望 这个分类器非常强大 在一定 程度上模拟的这种并行处理能力 就OK了 所以这是有两种研究的 一种流派.

这个是我是一个比较喜欢举一个例子 就神经网络到底 怎么学习 比如说我想让一个 神经网络学习开车 现在这个无人 车越来越多 特斯拉的无人车 后来出了事故 但你怎么去训练 一个计算机 或者说说白了 就是他的大脑就是个神经网络去 开车 很早以前人们都做过这样的实验就一个人开着车有 一个司机去开车他上面架了个 摄像头 就是一个是也好然后 这个拍下来的这些视频也好图像 也好 就作为神经网络的一个输入 随着神经网络假设说我能 看到这些东西那我怎么开车他 会进入这个司机打方向盘的动作 我们先不考虑什么换挡加油门 这些东西方向盘这个神经网络他 的输入是图片输出也是一排 神经元 我们叫neures从左到右 比如说最左边如果被激活的话就代表说你的方向盘要往左边 打死打到底 如果说右边的被激活 就说明你要往右边打到底 所以 说他就去记录你这个驾驶员你的 操作习惯 你是轻轻地往左边转了 一下 还是就是非常大幅度的往 右边去打方向盘 他把这些都 作为一个老师的一个训练的信号 这个图像本身就是它视野中 看到这个东西作为一个 输入 它根据这样的一个 输入输出去 进行训练最后神经网络 它自己开车上街它看到这个东西 一个输入它就会知道说我现在 应该把稳方向盘 还是应该是往左 还是往右

Perceptrons

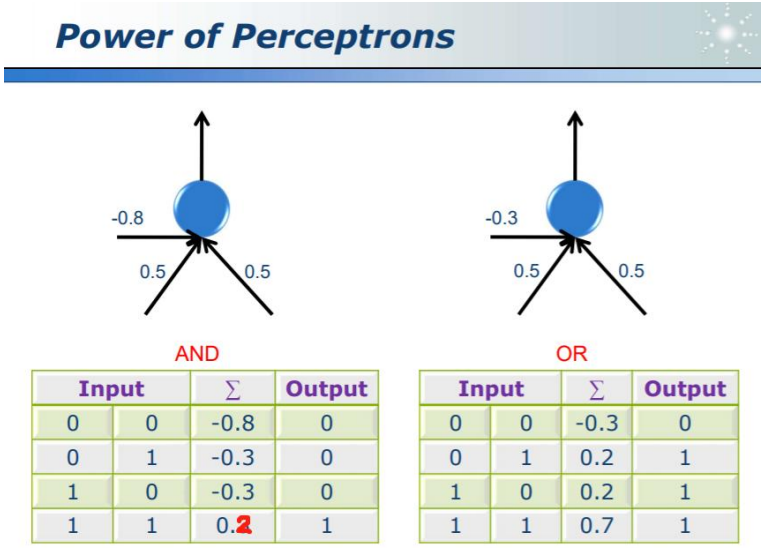


$$o(x_1,\ldots,x_n) = \begin{cases} 1, & if\ w_0 + w_1 \cdot x_1 + \cdots + w_n \cdot x_n > 0 \\ 0, & otherwise \end{cases}$$



神经元有时候称为感知机 就是一个就很简单perceptrons 它最简单因为它就是 一个神经元 他这个神经元有 若干个输入 从X一到XN 其实 就是属性吗 你有N个属性把它输 进去然后是西格玛起来但是做 一个这个向量的一个内积因为每 一条输入它是有一个权重的 大家注意W1到WN跟X1到XN 它们要做一个dot product 做 一个内积 特别注意的是他还有 一个叫W0对应的是X0 X0 始终是等于1的 W0它也是额外 的一个参数 其实它就是相当于 一个偏执 我们可以看到这样的 当我把这个全部西格玛起来以后 我加一个一般叫门限的 这样的一个function就把它变成零 或者一什么时候是一就是说 如果西格玛起来它大于零它 就认为是一然后小于等于0 就是零当然这只是一一种可能你 也可以说他大于0.5的时候是 一小于等于0.5的时候零都是 可以的 它实际上是什么东西 他就是形成了这样的一个判决 平面 大家看W1 X1加WN乘XN 为什么要W0要注意如果 没有W0的话这个判决平面它 永远都是要经过原点的所以说 这个W0跟X0是非常重要的 很多同学学的时候它就忘掉这点 了它以为有N个输入的话就是N 个权重 其实不是的是N加一个 权重这点要特别注意你其实形成 的就是这样一个判决平面W0 加上后面这个wx这个东西如果 它大于零小于零然后你认为它一 或者零其实在空间当中就是个超 平面 这是区分 所以大家要能够从 这种能够想象出来这样的一个 感知机它实际上对应的是个什么 东西 它就是一个超平面

总结:



AND

Input		∑	Output
0	0	-0.8	0
0	1	-0.3	0
1	0	-0.3	0
1	1	0.2	1

OR

Input		∑	Output
0	0	-0.3	0
0	1	0.2	1
1	0	0.2	1
1	1	0.7	1

感知机能做什么事情我们看左边和右边 这两个感知机你可以猜一猜它 实现了什么逻辑功能我们讲这个 逻辑这个元器件当它输入都是 00的时候我们看它的输出 是多少然后01 10 11这样的 时候我们可以看到这样一个 表格就会发现原来前面全都是 0只有当输入是11的时候它的 输出才是1这不就是and吗不 就是个与门吗 同样我们再看这边 只有当是00的时候才是0其他 的时候都是1那不就是个或门 是不是必须是两个都是0它 是0只要有一个是1输出就是1 所以我们可以看到输入是11 的时候是0.5 0.5这样dot product 出来它是1 1-0.3就是0.7 那么它就是大于零的我们就 认为它是一所以这就是刚才所说 的有一个这个门限的一个这样 一个function把这个输入转化成为 0或者1这样其实也就是一个 逻辑上的一个功能 所以我们看到 小小的一个感知机 它就可以去 实现不同的逻辑功能 只要我给它 那个权重设置的好0.5 0.5 关键是上面还有一个这个这 就是W0一定不要忘记因为有 太多同学忘了还有一个W0所以 后面再训练神经网络的时候经常 会告诉我说什么不收敛等等那 是因为你那个分前面你总是经过 0的 你没有W0它没法上下这样 去做平移的所以这一点要特别 注意所以我们到这里就大概 知道神经网络中间一个最基本的 单元就一个神经元它叫感知机 那么感觉机它有输入然后有 一个activation function一个激活函数 要有个输出通常是01而且它 可以实现这样的一个逻辑上的 功能

我们很难刻意忘掉一个人的原因是：

- ○ 记性好，没办法
- ○ 刻骨铭心，矢志不渝
- ○ 天长地久有时尽，此情绵绵无绝期
- ◉ 神经元的大规模分布式信息存储机制

✔

在感知机的判决函数中，w0的作用是：

$$o(x_1,\ldots,x_n)=\begin{cases}1, & if\ w_0+w_1\cdot x_1+\cdots+w_n\cdot x_n>0\\ 0, & otherwise\end{cases}$$

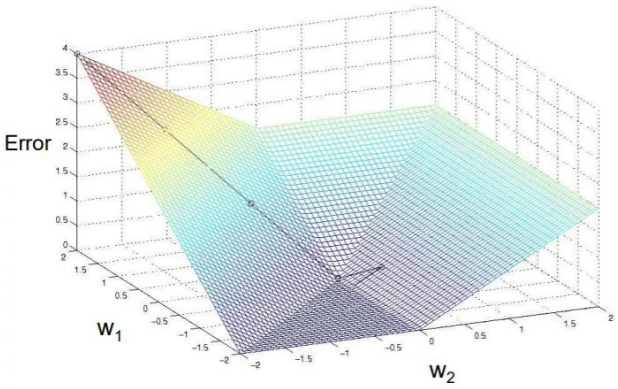
- ○ 为了后续学习算法推导的方便
- ○ 其实在实际中可以略去
- ◉ 控制判决平面到原点的距离
- ○ 控制判决平面的方向

✔

总结：

4.2 会学习的神经元

Error Surface



Gradient Descent

$$E(\vec{w}) \equiv \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2$$ ⟸ Batch Learning

$$\nabla E(\vec{w}) \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right]$$

$$w_i \leftarrow w_i + \Delta w_i \quad where \quad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Learning Rate

刚才给大家介绍了一个小小的感知机通过调整它的 权重可以实现不同的逻辑功能 或门或者与门等等那么到底 怎么样去设置这些权重呢 我刚才 只是告诉你们这样0.5 0.5 -0.8 这个东西怎么得到的呢 其实呢对于一个感知机如果它 有两个权重W1W2我去调整 这个权重的时候它其实呢是会有 一个误差的就是不同的权重取值 取得好的时候它的误差呢就会 比较低 取得不好的时候误差就 比较高 当然就是相对于特定的 任务比如说我想训练出来一个 或门是吧那它该是一的时候它就 应该是一 该是零的时候呢就应该 是零 可是有的时候呢它物它可能 很大有的时候呢误差会比较小 所以说我们就需要有这么一种 学习的方法让它能够沿着这个 误差的这个这个梯度让他逐渐 的往下走 最终呢能够收敛到这个 误差最小的一个地方 这是我们 希望要能够达到这样的一个效果。

我们呢就讲一下怎么去做这个事情 这就是刚才所说的梯度 gradient descent梯度下降的方法 首先呢我们先定义误差 什么叫 误差呢就是你T减去O T在这 里面就是你的target 就是我 期望比如说它输出应该是O是 它的实际的一个输出 那么T减O 不就是它的误差嘛 然后呢当然我 要用这个误差的平方 而且前面有 二分之一 在二分之一呢其实这个大家 了解一点都不知道一会要求导 我会把这个二分期呢给约掉 我们的训练集呢是用大D来表示 我有可能很多训练样本每个样本 上呢都有误差所以呢要在西格玛 起来这个呢叫batch learning就叫 批处理的这种学习就是很多个 样本每个样本上的误差都累加 起来 然后用这个误差信息呢去 调整我的权重 后面我们可以看到 其实我也可以知错就改 我来个 样本一算有误差马上就改也是 可以的 我们先看批处理怎么去做 当你知道误差了以后呢其实呢你 就是要去求一个偏导 比如说我 想去调整这个W0 那我就要知道 误差对W0求偏导这个数值 如果 我要知道这个数值了 我就可以 知道我该增加还是减小这个权重 所以说求偏导呢是非常重要的 一个事情 当你真正求这个ΔWi 求出来了以后那么你对 wi就是把它对这就是一个updata 学习我现在比如说权重是0.5 然后你告诉我说ΔWi应该 等于0.05就增加一点所以呢我 要把它加上去 就这样这个ΔWi怎么算呢 你看后面就是这样的 一个误差的这样一个求偏导 这样一个信息 我们就认为它是 一个误差的一个梯度信息这个呢 η呢是它的一个学习率前面 还要加一个负号 问题就在于说 为什么要加一个负号 那我们学 这些公式这个神经网络我们 后面会有好多公式 这些公式一定 要明白它的物理含义 否则你看到 一个东西你不理解你只是记下来 这个意思不大 而且你要明白它 背后的含义以后你再看各个公式 之间你会发现它们其实是遵循着 相同或者类似的一个原理 这样 可以帮助你们去很好地去学习 误差对于W求偏导大家想一下 如果他是正的代表什么意思 随着 W的增加这个误差应该增加还是 减小 当然需要增加是因为它的 偏导是正数 但是我的目的呢是要 减小这个误差 所以呢我要减小 这个权重 所以它前面是一个负号 叫明白这个道理 为什么是负的η 乘以这样的一个梯度信息这个学习率呢是控制它每次修改的 幅度 有的时候会矫枉过正就是 你每次调整就是说你让你增加 行我增加但你别增加太大也 0.5也增加个增加到0.55或者 0.6都还可以你要一下增加到5 你可能一下就跳过去了就增加得 太狠了这个梯度这个信息他只是 个方向而且是告诉你说我应该 增加还是你在减小但是通常呢 我们希望有一个小的学习率它 比较稳健 就慢慢的去去这个调整 和改变 不要一夜之间突然间什么 都变 需要慢慢的去改这是它的 一个基本的思路 所以我们后面 就要去看主要是这一块怎么去求 误差对于这个W求偏导怎么去求

总结:

Delta Rule

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{d\in D}(t_d - o_d)^2$$

$$= \frac{1}{2}\sum_{d\in D}\frac{\partial}{\partial w_i}(t_d - o_d)^2$$

$$= \frac{1}{2}\sum_{d\in D}2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d)$$

$$= \sum_{d\in D}(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - w\cdot x_d)$$

$$= \sum_{d\in D}(t_d - o_d)(-x_{id})$$

$$\Delta w_i = \eta\sum_{d\in D}(t_d - o_d)x_{id}$$

$$o(x) = w\cdot x$$

Batch Learning

GRADIENT_DESCENT (*training_examples*, η)

- Initialize each w_i to some small random value.
- Until the termination condition is met, Do
  - Initialize each Δw_i to zero.
  - For each <*x*, *t*> in *training_examples*, Do
    - Input the instance *x* to the unit and compute the output *o*
    - For each linear unit weight w_i, Do
      - $\Delta w_i \leftarrow \Delta w_i + \eta(t\text{-}o)x_i$
  - For each linear unit weight w_i, Do
    - $w_i \leftarrow w_i + \Delta w_i$

这个推导公式呢其实呢也并不复杂主要在这个地方t-o t呢是刚才说了它是它的 期望 O呢是实际 这个东西怎么去 求导呢 o跟W乘以X之间 什么关系呢W乘X它是它那个 输入o是它的输出你怎么求导 为了好求导其实呢我们做了一个 假设就是认为这个这个节点是 线性的就是输入等于输出这样的 话呢就用W典型X代替了这个O 这样的话呢就比较好去求导如果 你要用那个门限的话你这个在零 的那个地方它是导数不存在在 别的地方呢到处都是零 其实也是 不好去算的 所以我们呢是假设它 是一个线性的 这样最后呢我们就 可以得出这样一个表达式ΔWi 就等于还是这个有一个 学习率放在这 乘以 什么呢西格玛因为是很多的 样本 每个样本来了以后呢就是td-od然后再乘以Xid这又 是什么意思 那大家还是要去理解 一下 这是怎么一个物理上的含义 td-od比如说现在的期望 的输出是1 但是你实际的输出呢 没有那么大 你比它小所以呢 t-o 这个数呢它是个正数如果他是 正的话我应该增加权重还是减小 权重 你不知道 你取决于你当前 的输入的那个X X如果是正的话 X本身就是正的 然后呢我的我又 想让它变得更大 那我就要增加 这个W才能让W乘以X的变得更大是吧 所以说你看这个ΔWi 应该是个正数就像刚才 一样就是你要去理解一下每一个 公式它背后体现的一个含义 当我那个输出不够大的时候会 怎么样 当我的输出就是超过期望 输出时候 会怎么样当我的X本身 的那个输入那个分量是正数会 怎么样是负数会怎么样 那把各种 情况大家都要去考虑一下能够 充分地去理解这个公式这个 公式呢看上去并不复杂 但是他是 大名鼎鼎的叫delta rule这是 训练感知机还有一些线性分类器 的时候 都是可以用这样的一个方法 去训练的。

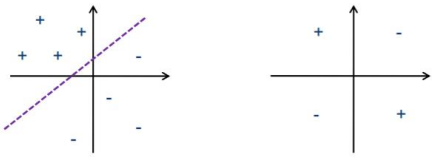
如果写成一个算法的话也没有什么复杂的但是大家要 注意一点它是一种批处理的一个 学习 所以说呢在这个for循环当中 在这一步要特别看清楚它只是 调整了ΔWi它并没有去 修改W就是来一个样本它要计算 T减O然后再乘以当时那个X 然后算出来它那个某一个调整的 分量 因为这条样本来了你可能 告诉我说你得增加0.05那个 样本来了他说你要减小什么 0.03所以有的时候大有的缩小 我把这些东西全部都累积起来 最后才用红颜色标出的这一块 这才是真正的做了一次更新所以 这是叫batch learning就是我把我里面 的比如说50条样本 每一条 样本都先过一遍 计算出来相应的 ΔWi我这个地方只是更新 ΔWi大家要记住最后才是 更新W这就是它作batch learning 一个基本的一个思路。

Stochastic Learning

$$w_i \leftarrow w_i + \Delta w_i \quad where \quad \Delta w_i = \eta(t-o)x_i$$

For example, if x_i=0.8, η=0.1, t=1 and o=0

$\Delta w_i = \eta(t\text{-}o)x_i = 0.1\times(1\text{-}0)\times 0.8 = 0.08$



Stochastic Learning: NAND

Input			Target	Initial Weights			Output: Individual			Output: Sum	Output: Final Output	Error	Correction	Final Weights		
x_0	x_1	x_2	t	w_0	w_1	w_2	C_0	C_1	C_2	S	o	E	R	w_0	w_1	w_2
							$x_0 \cdot w_0$	$x_1 \cdot w_1$	$x_2 \cdot w_2$	$C_0+C_1+C_2$		t-o	LR x E			
1	0	0	1	0	0	0	0	0	0	0	0	1	+0.1	0.1	0	0
1	0	1	1	0.1	0	0	0.1	0	0	0.1	0	1	+0.1	0.2	0	0.1
1	1	0	1	0.2	0	0.1	0.2	0	0	0.2	0	1	+0.1	0.3	0.1	0.1
1	1	1	0	0.3	0.1	0.1	0.3	0.1	0.1	0.5	0	0	0	0.3	0.1	0.1
1	0	0	1	0.3	0.1	0.1	0.3	0	0	0.3	0	1	+0.1	0.4	0.1	0.1
1	0	1	1	0.4	0.1	0.1	0.4	0	0.1	0.5	0	1	+0.1	0.5	0.1	0.2
1	1	0	1	0.5	0.1	0.2	0.5	0.1	0	0.6	1	0	0	0.5	0.1	0.2
1	1	1	0	0.5	0.1	0.2	0.5	0.1	0.2	0.8	1	-1	-0.1	0.4	0	0.1
1	0	0	1	0.4	0	0.1	0.4	0	0	0.4	0	1	+0.1	0.5	0	0.1
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	1	0	1	0.8	-.2	-.1	0.8	-.2	0	0.6	1	0	0	0.8	-.2	-.1

threshold=0.5 learning rate=0.1

其实呢也可以有另外一种思路叫stochastic learning 就是知错就改就是你拿一条 样本我一算那么T减O然后 再乘以学习率再怎么样也算出来 马上我就知道我要怎么修改 既然知道我现在就改这是另外 一种思路 比如说呢希望输出T 是等于1实际呢我又输出这个 0那么1-0就是我的误差学习率 呢是0.1当前的输入呢是0.8 这样一算呢ΔW呢不就是 等于0.08吗这是这样的一种 更新方法 delta rule虽然说很简单 包括感知机就非常简单但是呢 它有这样一套比较完善的学习的 方法所以说呢对于这种问题来 讲比如两两分类问题比如加号 代表1减号代表0我总是可以 通过得它住去训练出来这样一个 分界面所以这是一个非常 有价值的这样的一个学习算法 所以当时呢人们会觉得 哇非常 有趣 给你两堆数据然后我 可以去训练出来学习出来这样 一个decision boundary这是很好的 事情可是呢人们后来很快就 发现好像不太妙比如说对于 这样的情况我两个加号分别在 二四象限 两个减号在一三象限你 无论如何都训练不出来一个分 界面能够把他们分开的其实 这就是所谓的线性不可分问题 我们后面会看到怎么样用其他的 分类器去来解决这种问题 所以到 这里呢大家先要记住感知机它 其实就是一个线性分类器 毕竟 线性可分问题没有问题它可以用 delta rule这些方法都可以做得 很好但是对于这种问题呢它就 无能为力了。

最后呢我们看一个例子怎么样去用这个感知机 去解决一个叫与非门我们 之前展示过这个或门 与非门 就001 011 101 110 就是这样的一个训练集就是四 个 但是大家注意第一列全都是 1 这个是要特别注意的所以 这个呢叫X0相对应的呢它有三 个W W0 W1 W2所以在每一次呢 我就把这个输入三个X跟那三 个W做一个内积这就是它的 一个sum 我假设这个Threshold是 0.5就大于0.5或者小于等于 05大于05呢就是一 小于等于 0.5呢就是零 这样的话我可以算 出实际的一个输出 然后呢物业是 T减O 然后根据刚才这样的话 可以一代一代地去迭代去计算 稍微注意一点的是我们推导的 delta rule的时候呢我假设这个神经元 或者说感知机它都是线性的 但是这个时候呢 我又把它假设有 一个Threshold看上去好像不太 一样跟我们当时推导的时候 这个假设不一致其实呢实际当中 呢它一样work就是还是用 刚才那种方法就η乘以T减O 再乘以当时那个X是一样 可以做那么一代一代迭代到最后 你会发现0.8 -0.2 -0.1这个时候 我们的网络呢就可以完美的 解决我们的这个与非门的问题也 就是说它的error的就是零了 零的 话呢就没有更新了就这个权重的 就等于收敛 所以一开始慢慢 慢慢收敛到0.8 -0.2 -0.1 这个例子呢大家可以慢慢看一下 这一行一行的就可以去看一看它 是怎么样一步步去收敛到他最终 的一个这样的一个比较好的一个 这个solution

总结:

若神经元的误差对某输入的权重的偏导大于零说明：

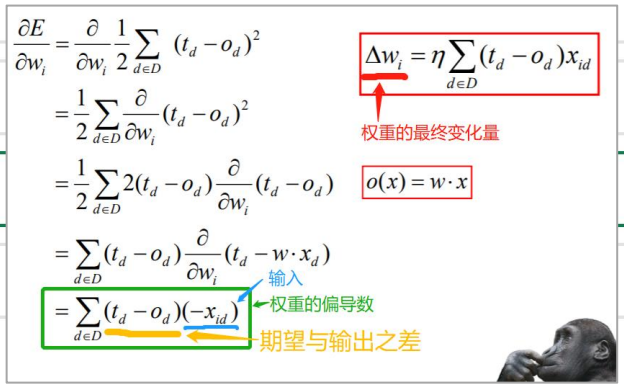
- ○ 权重应增加
- ◉ 权重应减小　　乘以-η之后，Δw变负，即权重变小。
- ○ 不能确定

✔

根据Delta规则，在stochastic learning模式下，若神经元的实际输出大于期望输出，权重应：

下图Δw_i的表达式中，η为正，若t_d-o_d为负，若输入xid为正，则Δw_i为负，即权重变小。

- ○ 顺势而为：增大
- ○ 反其道而行之：减小
- ◉ 若相应输入大于零：减小
- ○ 若相应输入小于零：减小



✔

以下关于感知机说法正确的是：

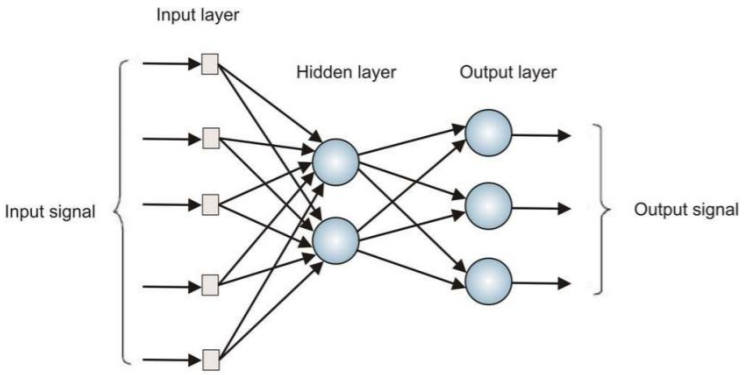
- ○ 在batch learning模式下，权重调整出现在学习每个样本之后
- ○ 只要参数设置得当，感知机理论上可以解决各种分类问题
- ◉ 感知机的训练过程可以看成是在误差空间进行梯度下降
- ○ 感知机的激励函数必须采用门限函数

✔

总结：

4.3 从一个到一群

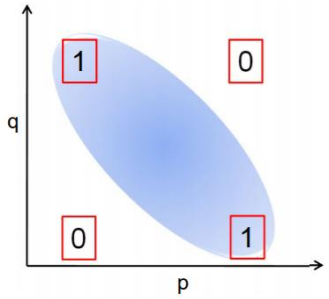
Multilayer Perceptron



刚才给大家介绍了感知机这是神经网络最基本的element 你 可以理解为也是一种神经网络 但是呢它有非常明显的缺陷就是 它不能够解决这种所谓的线性 不可分问题 因此呢咱们后面呢就 提出了叫multilayer perceptron就是 多层感知机 这其实就是所谓的 artificial neural networks 多层感知机它通常呢是这样的一个 架构 我们有输入有隐含和输出 当然隐含层可以是两层三层 但是我们现在呢去给大家介绍 就是有一个隐含层 所以说它 就是原来的感知机中间呢又加了 一层 因为原来本身其实就是有 输入的然后呢也是有输出的那它 中间呢多了一个隐含层 但是多 这个隐含层就把它的这个power就 一下子就提升了一个档次

XOR

$$p \oplus q = p\bar{q} + \bar{p}q = (p+q)(\overline{pq})$$

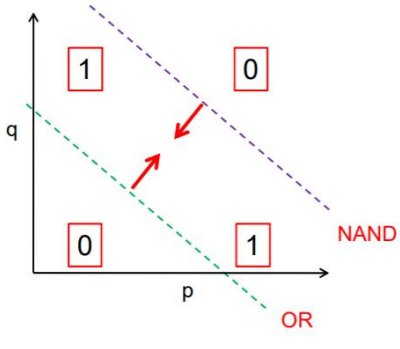


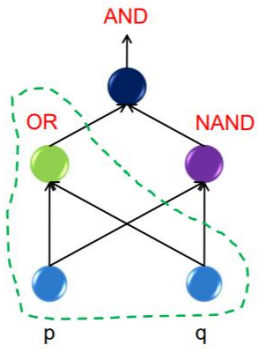
Input		Output
0	0	0
0	1	1
1	0	1
1	1	0

Cannot be separated by a single line.

XOR

$$p \oplus q = \neg(p \wedge q) \wedge (p \vee q)$$



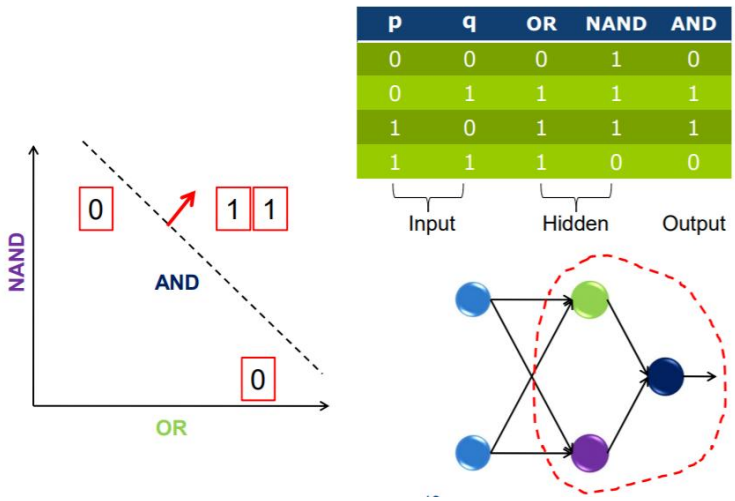


我们下面呢通过一个简单的例子来 详细的学习一下神经网络是如何 解决线性不可分问题 异或问题 XOR是线性不可分问题当中的 一个典型代表是非常简单我们 可以看就是四个 000 011 101 110 就是这四个二维的输入一个输出 但是这个问题本身它是线性不可 分的 如果同学们忘了什么是异或 的话你们还可以看一下上面这个 表达形式所谓异或就是说它 的两位要不同才是1 若同它 就是0 对于这个问题你本身你是 无论如何用一根线都分不开的你 可以去比划一下你怎么切分 呢其实呢都是分不开 到底怎么去 分这种问题呢一根线不行 你可能 就得用两根线或者用其他的 形式比如说你会觉得可能是 一个像椭圆形这样能分开或者 说是两条平行线能分开所以说这 就是一个线性不可分问题 一根 线无论如何都分不开但是呢我们 又想去解决它应该怎么办呢 对于 异或问题的话我们先用一个解析的方法 手工的来把先把它去 解决。

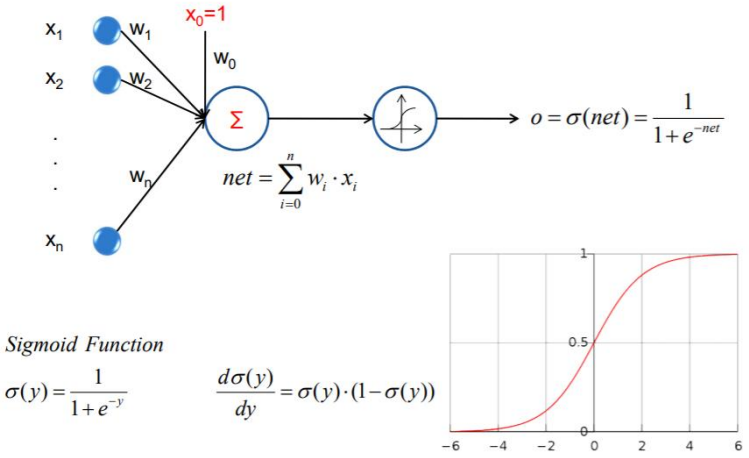
我们可以看到这个异或问题其实呢它可以把它拆分它可以 变成P和Q本身的一个或然后呢 再和P和Q的一个与非这两个 输出结果呢在做一个与我们根据 这样一个逻辑表达式就可以所以 如果我知道这样一个逻辑表达式 那我是不是就可以手工的去设计 出来这样一个神经网络呢我们 可以看到在这一部分我用绿 颜色的框把它标识出来的时候这 一部分其实呢我就认为他是一个 或门 同样呢另外一边呢是一个 与非门 我们看一下现在这个 情况变成什么样子 还是P和Q 直接分呢是肯定是分不开的 但是 呢我一步一步来 先看这个或门或 是什么意思 就是00的时候是0 其他都是1所以或门它相对应 的那条线呢应该是长成这样的 一个形状就是他网上面就红色 箭头的上面全都是一下面全都是 0那么与非门呢又反过来了 与非门的时候呢只有十五十一的 时候才是0下面呢全都是1 所以看着红色箭头代表了说对于 两个决策面来讲哪一面是一哪 一面是0好最后怎么办呢中间这 一部分这两个都是1也就是说 两个决策面都要说它是一的时候 它才是一那不就是个与 所以你 可以看到这上面呢他是一个与门 所以非常有意思一个或一个雨霏 然后再加上一个与门就可以去 解决了一个叫异或这样的一个 问题而且呢我之前其实已经给 大家展示过了这个感知机其实 绿颜色这一块它就是个感知机 或门是可以实现的 我给大家画 过那个图 然后与非门也可以 实现那个大表格 与门也是可以的 所以你可以看到原来一个比较 复杂的问题现在被我拆成了三 部分拆成了三个逻辑电路 然后 合起来就可以去解决这个问题。

总结:

Hidden Layer Representations



The Sigmoid Threshold Unit



我们更进一步去看看神经网络到底是怎么解决这个事情原始 的输入是在这里 前两列就是 原始的输入 那么这个网络从左往 右看 这两个输入进来以后它会 正向传播到绿颜色和紫颜色的 这个节点 一旦当它传播到了紫 颜色和绿颜色的节点以后我们 就只看红颜色这个圈里面它的 内容了这个时候呢其实网络的 输出就跟这两块就没有关系的就 像马尔可夫过程一样 当你 知道了这两个的输出他的输出就 已经唯一确定了而这一部分本身 呢就又是一个感知机所以我们 可以看一看隐含层是什么 隐含层 的输入是01 11 11 10就 跟这个本身的这个原始数层呢就 不一样了 所以我们看看在隐含层 上发生了什么有趣的事情 这个 时候我们再画一个图 要注意 这个横坐标和纵坐标已经不在是 P和Q了这个时候呢已经变成 了这个绿颜色的这个或和这个紫 颜色的这个与非了所以我们再 看010是哪个东西 010 010是这个111 111 100好像跟刚才有点类似又 有点不一样了就像我们所说的 这个移形换位 乾坤大挪移一样 就原来这些样本的空间样本中 的这些这些点他现在换位置了 标签是1的这两个样本呢被 挪到一块去了被挪到这个右上角 了零和零呢是那两个样本的放在这两个角所以说这个时候呢 你们可以看一下这个问题是复杂 还是简单了 原来呢是一根线是 无论如何是分不开的 现在我把它 重新在空间当中的位置调整了 以后 其实你一眼就会发现这个 问题简单多了 为什么简单多了你 直接画一条线这条线只要是与门 不就可以了吗只有当是11的 时候才是1 其它全都是零不就 分开了吗 实际上我今天不是讲了 这个深颜色的这个节点它就是 个与门的功能吗 所以这就是从 逻辑上给大家去展示了神经网络 是怎么去解决线性不可分问题的 所以我们看这个标题叫隐含层 表示 也就是说在输入层来讲P 和Q它和这个输出的关系它们 之间是线性不可分的 我没法直接 去解决 但是我真正是怎么解决的 我先把它做了一个映射 映射 到隐含层了 映射到隐含层以后 这个问题呢就变成了一个线性可 分的问题 然后呢再用一个感知机 就可以把它分开了所以说隐含层上的这个模式跟它本身的输入层 的模式呢是不同 这就是神经网络 解决线性不可分问题的一个基本 思想 我要把原始问题做一个 转化转化成为一个容易的问题 然后再去解决这个思想一定要 学会因为不同的分类器它的 思想是不同的我们后面要选SVM 支持向量机这些向量机怎么 解决呢它也是在做一个映射只 不过说它可能是像一个很高维度 去做映射 但是呢基本思想还是有 一定的结果这种相似性的

在神经网络当中呢每一个神经元呢其实 跟之前学的感知机呢是非常接近 的就还是有W X要做内积 当然它还是有W0这是一个你 认为它是一个虽然事后的只不过 说呢他的这个激活函数不一样了 之前那个激活函数通常会是一个 门限的这样一个函数现在 通常会用一个长成这个样子的 函数这个函数呢其实叫sigmod 的函数有一加上E的负X方分 之一这sigmod函数 为什么 要用sigmod函数呢因为它有 一些比较好的特性我说不管你的 输入多小它的输出呢不会低于零 不管你的收入多大它的输出不会 超过一 换句话说呢就是它可以很好地把这个输入输出这个把 这个映射关系把它做得很漂亮 就是它可以保证这个输出始终是 在零一区间内 不会说要搞得特别 大或者特别小这是零一区间 而且它这个平滑曲线平滑曲线的 好处呢可以求导 而且呢这个sigmod 函数一个特别好的一个属性 就是它的导数可以直接算出来 也对Y求导其实就是它的输出乘 以一减它的输出 而且呢你们可以 看得出来这个sigmod函数当 输入是几的时候它的倒数最大你 可以看到当他是在这个位置的 时候他的这个地方梯度是最大 所以我们可以先记住这一点当它 的输入越小的时候就越接近于零 的时候就绝对值越接近于零的 时候它这个地方的这个梯度呢是 最大的因为它的输出值是0.5 0.5×（1-0.5）它这个 时候呢它的这个导数的是最大 如果你的输入都特别小或者 特别大 那么它的导数呢就会接近 于零那么在训练的时候就不好 因为导数接近于零那它这个训练 就非常非常的缓慢 所以我们一般 来讲呢希望从这个地方开始 所以你要明白为什么它一开始 这个权重都会比较小 是因为它要 让这个sigmod的函数呢处于这个 位置它的导数呢会比较大 所以这 是神经网络它的这个在神经元 这个层面和感觉期呢就是略有 不同 它这样设计呢是为了后面去 做一些推导的时候也方便我们 一会就能看到

总结:

以下关于感知机说法正确的是：

- ○ 多层感知机比感知机只多了一个隐含层
- ◉ 感知机只能形成线性判决平面，无法解决异或问题
- ○ 多层感知机可以有多个隐含层，但是只能有一个输出单元
- ○ 隐含层神经元的个数应当小于输入层神经元的个数

✔

多层感知机解决线性不可分问题的原理是：

- ○ 分而治之，对原始问题空间进行划分
- ○ 将原始问题向更高维空间映射
- ○ 在输出层和隐含层之间形成非线性的分界面
- ◉ 将原始问题在隐含层映射成线性可分问题

✔

采用Sigmod函数作为激励函数的主要原因是：

- ☑ 有固定的输出上下界
- ☐ 计算复杂度较低
- ☑ 导数存在解析解
- ☑ 处处可导

✔

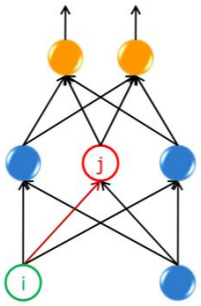
总结：

4.4 层次分明，责任到人

Backpropagation Rule

$$E_d(\vec{w}) \equiv \frac{1}{2}\sum_{k\in outputs}(t_k - o_k)^2 \qquad \Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j}\cdot\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} x_{ji}$$



- x_{ji} = the i^{th} input to unit j
- w_{ji} = the weight associated with the i^{th} input to unit j
- $net_j = \sum w_{ji}x_{ji}$ (the weighted sum of inputs for unit j)
- o_j = the output of unit j
- t_j = the target output of unit j
- σ = the sigmoid function
- *outputs* = the set of units in the final layer
- *Downstream* (j) = the set of units directly taking the output of unit j as inputs

Training Rule for Output Units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j}\cdot\frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k\in outputs}(t_k - o_k)^2$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}(t_j - o_j)^2 = \frac{1}{2}2(t_j - o_j)\frac{\partial(t_j - o_j)}{\partial o_j} = -(t_j - o_j)$$

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = o_j(1-o_j)$$

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)o_j(1-o_j)$$

$$\Delta w_{ji} = -\eta\frac{\partial E_d}{\partial w_{ji}} = \eta(t_j - o_j)o_j(1-o_j)x_{ji}$$

刚才呢我们已经展示了神经网络呢它可以去处理 这种线性不可分问题也就是说它 的功能是非常强大的但是在强大 的分类器 你也得有好的学习算法 对于神经网络来讲呢 它的学习 算法呢是不唯一的 不是说只有 我们现在要讲的一种 但是这是 一个非常经典的算法 叫BP算法 叫一传播这个误差逆传播算法 它不是数据挖掘的十大算法之一 但是呢它是在这个神经网络的 发展当中起到非常重要的一个 作用它的基本思想其实和这个 感知器的训练是非常像的 我都是 要求这个东西都是要求误差对于 这个某一个权重求偏导 稍微复杂一点点我大概同学们介绍一下Xji 就是第j个神经元的第i 个输入 wji呢就是以这个 神经元呢第i个这个权重它有 输入有权重稍微定义的稍微 复杂一点点 还有一个叫当时有 这个就是downstream这个神经 元的它的这个下游的这样的 神经元就是都用j的输出作为 输入这些神经元叫downstream 但是 不管怎么样你最后都是要求这个 误差对于某一个权重求偏导那么 利用这个**链式规则**把它展开完 以后就可以变成误差它对于D 这个神经元它的输入求偏导 所以我们最后要求的是误差对于 这个net net j呢 就是第j个神经元它的输入对它去求偏导把这个求出来了所有 的问题呢都可以解决了我们下面 呢就来一步一步的去看一看 对于这些神经元怎么去计算这个 误差对权重求偏导 我们要分两 步走 第一步呢是去计算对于这些 黄颜色的神经元 它们是输出神经 元 这个呢是很简单的 对于这个J 就是这种隐含层的呢 相对比较 复杂一点 我们先看简单的输出 神经元怎么去计算他们的这个 权重。

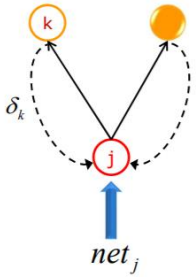
首先你看我们看到了这是training rule for output unit 就是输出的其实这个东西跟感知 机就是一模一样 就是一回事 我们看上面这个公式 误差呢对 这个net j追求偏导应该把它 展开 误差对于这个第j个神经元的 输出求偏导 然后它的输出在对于 它的这个输入求偏导是吧花开两 朵各表一枝 我们先看往左边走 就前面这一项是什么样 定义 这个误差 这个一看非常熟悉 就是我们之前感知机然后呢对他 去求偏导 这个导完以后一看 就是T减O非常简单就是期望 输出跟实际输出之间有这么 一个差 然后再再看这边这边呢 就是这个神经元它的输出对它的 输入求偏导 能看得出来为什么是 等于O乘以1减O吗 这不就是 我们之前讲的sigmod函数的 特性吗 就是因为我们用了sigmod 函数 所以在这个地方给我们带来 了极大的便利 我直接就可以写出 o 1减o你要换一个不太 不太 好用的这个函数你求偏导什么的 你可能都没法直接去写 现在直接 一步解析解出来了 然后呢把 这两部分呢再把它合并起来不就 可以了吗 你看我们就得到什么 东西了 误差对于你这个神经元的 输入求偏导就等于负的t-o o 1-o 然后呢最后我们要求到的是 Δwji学习率放在这 就跟 以前一模一样 然后求这个偏导把 这个呢带进去 不就可以了吗最后 呢Xji好这个式子看上去好像 稍微复杂了一点 但是你仔细 看一看是不是有一种似曾相识的 感觉 我之前已经说过学习这些 公式的时候一定要能够把这些 公式连在一起 不要认为每个公式 都是天上掉下来的就一个公式 又一个新公式 那样的话你会很快 就这个脑子就会炸掉你根本记 不住那么多东西但你要会学习的 人一眼就能看得出来这和我们 前面的感觉记得公式是不是 如出一辙呀 学习率在这 t-o在这 xji在这 是不是一模一样 都在这但是有一点点不同他多了 一点东西多了一个O乘以1减O 这是什么东西这不就是sigmod 函数求导为什么perceptron里面 感知机里面没有这一项呢那 是因为我假设的是线性 输入等于 输出一求导不就是一吗 无非就是 这样区别就在于这个地方所以说 output这些神经元去求 这些编导去算这些权重跟感知机是一模一样就那么一点点 区别还是因为它的激励函数或者 激活函数不同所以要注意就 就是这一点点不同其他它们的 原理是一模一样的 所以我一开始 就说对于输出层神经元 他的这个 计算呢是比较简单的 麻烦是麻烦 在后面 对于隐含层就是他不是 输出层怎么办。

总结:

Training Rule for Hidden Units

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k}\frac{\partial net_k}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j}\frac{\partial o_j}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k w_{kj}\frac{\partial o_j}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k w_{kj} o_j(1-o_j)$$



$$\delta_k = -\frac{\partial E_d}{\partial net_k}$$

$$\delta_j = o_j(1-o_j)\sum_{k \in Downstream(j)} \delta_k w_{kj}$$

$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

BP Framework

- BACKPROPAGATION (*training_examples*, η, n_{in}, n_{out}, n_{hidden})
- Create a network with n_{in} inputs, n_{hidden} hidden units and n_{out} output units.
- Initialize all network weights to small random numbers.
- Until the termination condition is met, Do
  - For each <x, t> in *training_examples*, Do
    - Input the instance *x* to the network and computer the output *o* of every unit.
    - For each output unit *k*, calculate its error term δ_k

      $$\delta_k \leftarrow o_k(1-o_k)(t_k - o_k)$$

    - For each hidden unit *h*, calculate its error term δ_h

      $$\delta_h \leftarrow o_h(1-o_h)\sum_{k \in outputs} w_{kh}\delta_k$$

    - Update each network weight w_{ji}

      $$w_{ji} \leftarrow w_{ji} + \Delta w_{ji} = w_{ji} + \eta \delta_j x_{ji}$$

这一点同学们可能一开始想我不就这样算嘛算 这个误差什么对这个权重求偏导 就行了吧 其实你再想想你知道 误差吗 你能把那个误差期间 这样算出来吗 对于输出层神经元 比如说他应该输出是1 你输出的 是0.8 那么T减O是0.2这就是 他的误差也知道这个误差 所以你 可以去说你这个W应该怎么去改 但是现在就不一样了你根本不 知道隐含层神经元它的真正的 希望的输出应该是多少 你不知道的话你都不知道它应该输出多少 你怎么说它的误差这是没有办法 去算的一件事情 所以人们在这个 问题上其实get stuck了很长时间 很多年你们搞不定这个事情我举 一个实际的一个例子 你可以想象 如果我是领导我派了两个人 就像判了这两个人出去干活结果 呢把事情搞砸了 出了差错了那么 实际办事的这个人他的责任是 非常好这个认定的 你应该 做成什么样你没做到什么样这不 就是你的错误吗 但是我的责任有 多大呢现在不是讲什么领导问责 那到底领导要担多大的责任 怎么说 这个就麻烦了你想想 我又不是直接去做这件事情的人 我怎么能够说 我应该输出多少 我实际出多少 去算这个误差的是 是很难将直接去算的 所以呢人们 后来想那就把这个误差呀就你 传播回来 就张三李四反正都是我 派出去的那你们做错了那你们 这些错误呢都是由我来承担 都要反馈给我 所以看这两条线就是反馈回来 ΔK呢代表就是 一个误差的这样一个信息就把 它反馈回来 但是呢并没有这么 简单 你可以想象张三李四这两 个人张三比如说跟我关系特别好 它对我的话言听计从那么它 犯了错误你觉得我承担的责任 应该大还是小 如果李四这个人跟 我关系一般或者我说的话它也 不是那么听说十句它听一句那么 这种人犯了错误我的责任该大 还是太小 所以你可以想象你 不仅要把这个误差信息传回来 而且呢还要乘以他的一个权重 权重呢就是这条边 就是我们两 个人之间的这个联系的紧密度 或者说我对你的影响力有多大 这个呢在乘起来 这是一个基本的 一个思路所谓输出层的这个 ΔK就是它的这个误差对于 这个第k个神经元呢输入求偏导 就这个我们认为它是一个误差的 一个信息 这是对于输出神经元 我们上一页这个PPT呢已经把 它算出来了 我们现在呢要求的 是对于底下这个神经元这个 Δj应该是多少 但是我已经给 你们描述一下 他的基本思想就是 它下面的这些神经元 算好的这些 Δ呢把它反馈回来 然后再跟 这些权重进行就把加权 整体的 公式呢是这个样子 我就不一一条 一条去介绍 但是最后算出来以后 是等于这个Δk乘以W再 乘以o乘以1减o是这样 一个公式 所以呢Δj就 等于这样的一个表达式 所以 我们也看到K属于什么downstream j 就是所有用我的这个j这个 神经元的输出作为输入的神经元 都是我的downstream 你们这些都跟 我有关所以你们犯了错误都要西 格玛起来 反馈给我而且呢还要 乘上我们两个之间的一个权重Wkj 这就是我们两个之间的一个 权重所以你的错误和咱俩的权重 盛起来所有的这些都汇总起来就 到我这儿了 那么o 1-o 一看 大家都知道这还是那个sigmod 函数求导 所以说最终最终ΔWji 第j这个神经元的第 i条输入这条边的权重他的这个 修改量就等于学习率乘以δj 乘以这个xji所以说虽然 上面两个公式算的不一样 但是 最后修改的时候是一致的 这就是 神经网络做bp算法它的一个 基本思想 就先把外面的outputs的 这些算好 算完以后把这些误差把 它一层一层的逆传播所以它用的 时候feedforward是这个正向 传播前馈网络 但是训练的时候它 要反馈逆传播回来 所以你们会 听到什么误差你传播算法指的 就是这样因为外面的误差你得先 算 算完你才能够算里面的这个 顺序呢是不能够这个颠倒的。

刚才讲了一堆公式同学们看了可能 有点头大其实不用怕bp算法 虽然里面的公式推导比较多 但是 呢他真正写成算法很简单 公式就三个 每次这个让同学们去编程 实现bp算法的时候 我就跟他们讲 其实就三个公式 是吧先算这个 输出层的ΔK就是刚才算的 o 1-o t-o就完了 这个就 跟感知机是一样的 算完输出层 以后反馈回来 对于隐含层的每 一个有我用小H代表隐含层 ΔH呢就是把这个西格玛 起来就行了 其实就是这样的一个 把它把误差它都反馈回来 当你把 两个两种这个Δ都能算好 以后那后面这个更新就很简单了 更新的时候这个公式是 一模一样的 只不过说这个Δ 有的是这条公式算的 有的呢是下 面的公式算 所以说bp算法呢 看上去推导过程是有点复杂的 因为它有很多定义就是Wji 这些东西 但是呢推导完了以后它 的表达式是非常的简洁的而且 我们可以看到就是学习率是乘 以这个误差的这个它其实是个 其实是误差对于那它的一个偏导 的一个信息 根据这个信息我来 判断我到底增加还是减少当然你 还要考虑你现在的这个输入X是 正是负 这个综合起来就可以知道 我这个权重到底是增加还是减少 这就是大名鼎鼎的bp算法的 一个基本的一个框架结构。

总结:

More about BP Networks ...

- Convergence and Local Minima
  - The search space is likely to be highly multimodal.
  - May easily get stuck at a local solution.
  - Need multiple trials with different initial weights.

- Evolving Neural Networks
  - Black-box optimization techniques (e.g., Genetic Algorithms)
  - Usually better accuracy
  - Can do some advanced training (e.g., structure + parameter).
  - Xin Yao (1999) "Evolving Artificial Neural Networks", *Proceedings of the IEEE*, pp. 1423-1447.
- Representational Power
- Deep Learning

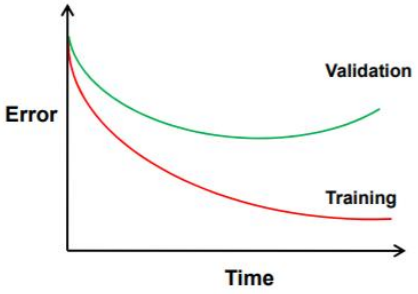
这些是他的一个算法 但其实呢你要 在实际当中去用好这个算法的还 需要描写很多很多的内容bp 算法你一看就叫他求导 求导最怕什么呀就是local 如果你有很多局部的这样的一些 最优点是最怕的但是很不幸这个 神经网络呢它这个误差空间 就是有很多这样的局部的最优点 所以你用bp算法做很容易就掉 到这些局部最优点 你会发现那个 网络那个误差呀它就平了 它不再 降低了 就是经常会遇到的一些 问题 所以may get stuck at a local solution这是很容易出现的一个 事情如果出现了该怎么办呢通常 人们就会就是再试一次是吧 你从 不同的这个点出发从不同的这个 initial weights出发就是你不会走 那么点背 有的时候运气不好 你刚收敛一会就平了 就像地上有 坑你刚走两步你就掉坑里了 有的 时候呢 你运气好就可以继续 往下去收敛 这是特别需要 注意的 就一般来讲训练神经网络 都是需要就是就try好多回的 当然呢我们也可以就是用其他的 方法比如说去净化这个神经网络 bp算法是其中之一而不是 只能用bp算法去训练 也可以用 这个进化计算等等像我以前 硕士期间都会做我这样的一些 工作就是用genetic algorithms这些方法 来去训练神经网络去避免他们跌 入到这个局部最优点 这有一篇 很著名的文章 有一篇综述在 proceedings of the IEEE上发表的 这个有一个非常详细的一个描述 另外神经网络呢它的表达能力 是非常强的为什么人们那么喜欢 用它 就是你可以去理论上去证明 比如说当我隐含层有两个隐含 层的时候 当我这个节点数足够多 的时候我的神经网络它是可以 去逼近任意复杂的函数 所以说它 的表达能力是很强的而且近几年 这个深度学习又特别火爆什么叫 深度学习 其实深度学习并没有 什么高深莫测的地方 他名字起得 好其实深度学就是多层嘛 我们之前讲就multiply perceptron 就是多层感知机只不过他往上再 多吗可能十几层几十层这就叫 深度学习了 它可以一层一层的 去吸取这个特征它的能力呢可能 更加强大 只不过说多层嘛以前在 训练的时候会有一些困难像bp 的时候它误差逆传播它的这个 这个导数信息很快就会衰减到 接近于零了所以它不好那样去 训练深度学习呢它有哪些其他的 技巧绕过这些困难所以它 本质上呢也还是一种神经网络 就是多层更多层神经网络

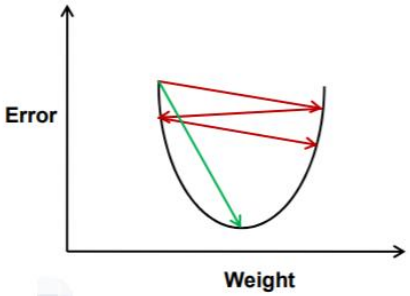
总结:

More about BP Networks ...

- Overfitting
  - Tend to occur during later iterations.
  - Use validation dataset to terminate the training when necessary.
- Practical Considerations
  - Momentum
  - Adaptive learning rate
    - Small: slow convergence, easy to get stuck
    - Large: fast convergence, unstable

$$\Delta w_{ji}(n) = \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(n-1)$$





神经网络呢也有这种所谓的这种overfiting 问题就是跟这个决策树样都 有这个过学习问题那么实际上 应该怎么办呢我怎么知道什么时候不能再学习了他会过学习呢 我们在训练的时候往往是这样 训练就你有一个这个一个训练集 它的误差呢会慢慢慢慢下降 同时你还要再有一个validation set 就叫校验集你同时去测试 你的网络在校验集上的性能一般 来讲在这个外地的satvalidation set的误差 是开始会降低降低但是到一定 程度呢他就会增加开始的降低是 对的 但是为什么又会增加呢就 说明过学习了 你学过头了你训练 的太狠了 以至于说他再没有见过 的样本上面的这个这样的误差呢 就开始上升了 所以说呢你马上 就要停下来 就要在这个拐点的 地方把它停下来 就是在这个地方 神经网络我们认为是比较好的它 的训练无差小 它的测试误差呢也 小所以它其实是用这样的方式 要来实时的去关注我网络学习的 一个情况。

另外神经网络让bp方法进行训练的时候最怕就是 陷入到这种局部最优点就是说 或者说是遇到这种很平的这种 地方 你可以想象这个空间当中 如果有一个地方是很平的那它 导数就是零导数是零的话它就 没法往前走了它的ΔW就 等于0了就像你是做个小车 往下滑结果划到平地上了你没有 方向了也走不动了所以人们提出 一个叫家冲量就你从山上往下滑 即便你现在暂时是个平地了 但是呢你还有原来的那股劲还 可以继续往前把你往前再推一推 说不定前面呢就可以继续往下滑 所以这就是momentum可以加 这么一个公式表达就是说 后面就是在这个第N时刻你 的这个修改量呢还取决于说你N 减一时刻如果N减一是个就你 之前的时候你往前冲那么到 那N时刻的时候呢你还可以借着 这股就继续往前走所以即便说你 在N时刻的时候这个这那个 表达式呢是零 因为你的这个很多 梯度是零的话它还能往前再走 它不会马上就停下来。

第二个需要注意的呢是关于这种叫学习率 学习率η我一直没有讲应该多 大通常我们就设计成0.05 0.1 0.2 不能太大也不能太 小太小会怎么样呢 他容易get stuck比如说这个地上有有 很多坑我要走过去 你觉得我是迈 着小碎步走比较安全 还是说迈着 大步走比较安全如果迈小碎步走 就相当于说学习率特别小这叫 小碎步那么你一定会掉到第一个 坑里去 你迈大步的话呢所以要坑 比较小你运气好 你可能唉直接就 迈过去了你就直接就过去 但是呢大的学习率呢也不一定 就是好事 我们可以看这样的一个 例子当你的这个训练已经接近于 最优值的时候比如说它在这个 位置它一算这个梯度 梯度是负 前面再加一个负号就 变成正的了就说明说我全中应该 增加 我权重增加可能学习力太 大 一口气迈过就像我其实要 到这个位置结果我一下走过了 走过一看梯度呢现在是正的 正当的我要减小我要再往回走 结果呢一步呢又迈大了又走到这 了就是来回再障蹦来蹦去 这就是 这就是所谓有的时候 网络的这个 训练 他会产生一种叫用震荡 就 你看它的误差 会这样来回波动这 就说明他可能是其实已经接近于 这个这真正的solution但是它 来回跳来回跳 它没有办法很好的 很稳定的收敛 其实呢如果在这个 位置 你要学习并选择好叫绿颜色 这条线 你一步就找到最优点了这 不是很好的事情吗 所以说这个 learning rate的选择其实是很有讲究的 开始的时候是怎么样到慢慢的 时候会怎么样 都可以去动态地去 进行去调整

在权重更新公式中引入冲量的主要目的是:

- ○ 提高算法的收敛精度
- ○ 提高算法的稳健性
- ○ 提高算法的全局优化能力
- ◉ 有助于摆脱误差平缓区域

✔

总结:

在误差逆传播算法中，输出层神经元权重的调整机制和感知机的学习规则相比：

- ○ 考虑到线性不可分问题，学习规则更为复杂
- ○ 一模一样，等价于多个感知机
- ◉ 遵循相同的原理，激励函数可能有所不同
- ○ 所有输出层神经元的权重需要同步调整

✔

在误差逆传播算法中，隐含层节点的误差信息应当：

- ○ 根据自身的期望输出和实际输出的差值计算
- ○ 根据所有输出层神经元的误差的均值计算
- ◉ 根据自身下游神经元的误差进行加权计算
- ○ 根据自身下游神经元的误差的均值计算

✔

为了克服学习空间中存在的局部最优点应当：

- ◉ 尝试从不同的初始点开始训练
- ○ 将权重初始化为接近于0的值
- ○ 采用较小的学习率
- ○ 增加隐含层神经元个数

✔

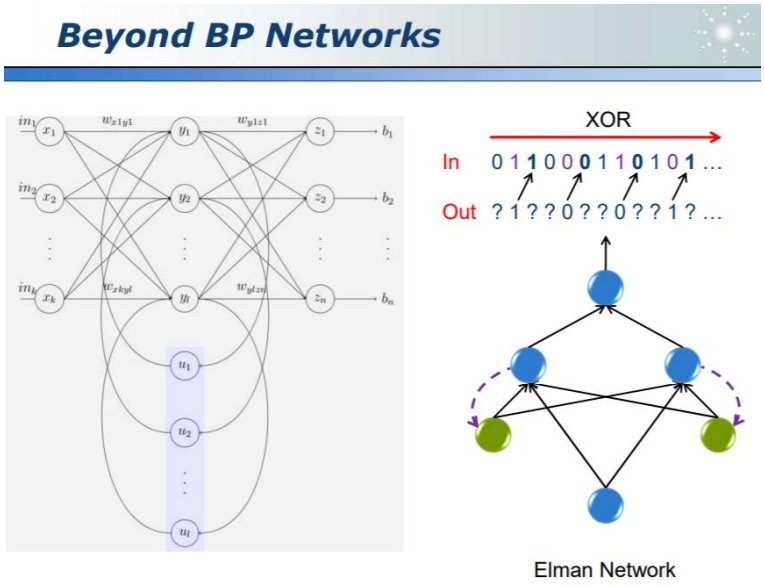
关于学习率参数的设置，正确的描述是：

- ○ 较大的值有助于提高算法的收敛稳定性
- ○ 较小的值有助于提高算法的收敛速度
- ◉ 在开始阶段应该较大，然后逐渐减小
- ○ 在开始阶段应该较小，然后逐渐增大

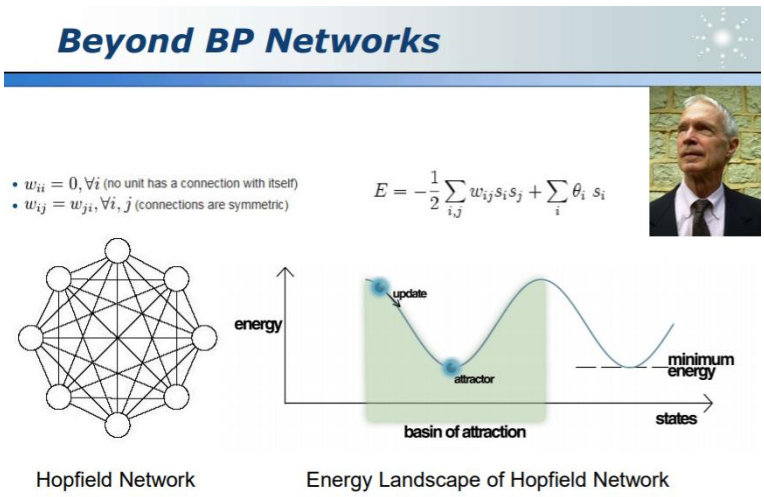
✔

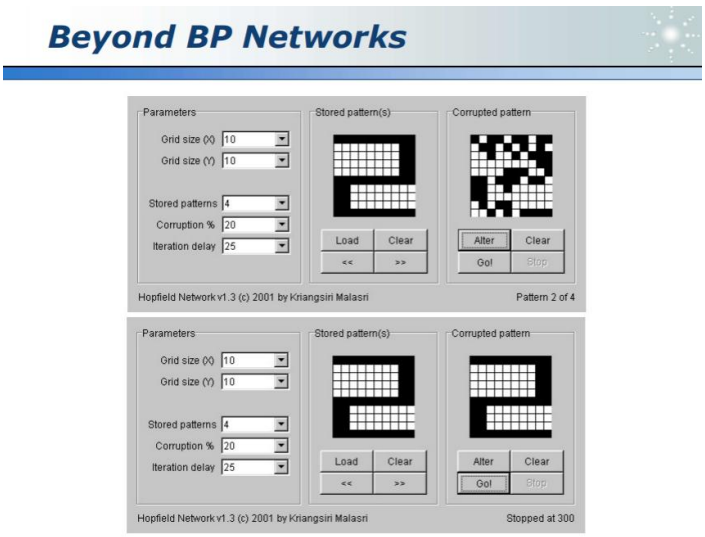
总结：

4.5 管中窥豹，抛砖引玉



刚才介绍的呢叫前馈形的网络 有的时候就干脆说它是叫 bp网络 但其实神经网络 博大精深 它这个各种各样的网络 非常多 我们最后呢再利用一点 时间给大家介绍几种比较有 特点的 还可能大家一般没有见过 那个神经网络 这种网络呢叫做elman 网络 它呢这个网络其实也是做 一个类似于异或的问题但是它的 这个异或呢和我们的bp网络的 异或呢非常不一样 我们看一下它 的输入 我上面那一行输入呢是 他是一个串 而且每次只输入一 位 因为这个神经网络呢它只有 一个输入单元而这个就非常 有意思了 它是有个时间顺序的我出入01000还是按照 这个顺序输入的而且我用这个粗 体字表示了你可以看到他是三个 三个一组的 每三个里面第一位 和第二位它做异或其实正好呢就是 第三位 大家可以看到01 变成1 00呢是0 11是0它是 这样的一个关系 这个网络的 输出应该是什么呢 也很有意思 当第一位0进去以后他的输出呢 我打了一个问号 问号就代表说I don't care我不管他是什么东西因为我 不知道你要干嘛 但是当你在输入 1的时候 它的输出应该是1这个 1呢就对应了上面那个1 就是第 三位的那个1 我用这个斜的 箭头呢给大家展示了 换句话说你 数是0的时候我不知道你要 干什么 但是你数1的时候我就 知道0 1做异或是1我就猜出来 你第三位应该输入的是个1就是 这样所以它是有一定的这种 上下文关系的所以可以想象这种 网络它是要有一定的这种时间 或者记忆就是这样的一些元素要 能够放在里面传统的网络第T 时刻的输出就是等于第T时刻的 这样一个输入然后做一些这些 激活函数等等这样操作他和T-1 时刻是没有任何关系的 但是呢 在网络里面就不一样了 你得 记得它以前输入的是什么就是它 要有一定的这种记忆功能 要不然 你怎么去做我们看一下他的记忆 是怎么实现的 在隐含层之外他 其实还有两个这样呢我们用绿 颜色表示的一个神经元这两个 神经元其实就是他的一个记忆 功能模块但这两个神经元看 有点奇怪他好像只有输出没有 输入它其实是个copy就 T-1时刻网络的状态就是T-1 时刻这两个栏时应元它们的这个值会被copy到这个绿 颜色的神经元 所以在第T时刻的 时候呢这个网络的输出就不仅仅 取决于当前的输入 还取决于T-1 时刻网络的内部状态通过这种 模式呢它就可以有一定的记忆 功能 它我知道你上次输入的是0 这次又输的是1所以你说下 一次你我我猜应该还是个1是 这样一个有关系所以它是通过 这样一种copy然后呢再反馈回来 通过这样的方式去实现了一定 程度上的这种memory的这样一个 功能。





第二个网络呢也是有这种记忆功能 包括它的名气呢可能会 比较大一些叫hopefield的网络 有些同学也都听说过这个网络呢 它实现的是一种也是一种memory 的功能是非常接近于人 的大脑的一种功能这个网络很有 特点它是一种全互联网络你 说不上来哪个神经元是什么输出 输入它全部都连在一起了它 没有这种层次型的概念 它的 好处是干嘛呢他可以去给你一个 pattern进来它可以去记忆它就我 比如说我给他八个pattern让他 记住所以拍的你就比如说 0110就是每个神经元1个值 就是一个pattern他 通过训练它可以怎么样呢让每 一个这个模式正好对应于这个 网络能量的一个极小值网络 能量它有一个定义它通过调整 这些权重预支使得说当你在 某一个pattern进来以后它 就正好处于其中的一个极小值它 通过这种方式就等于是说我记住 了这个模式 当然他是有这个就 说能够记多少东西它是capacity 就像人的大脑一样你不能 无限制的让我记东西它是有这个 有一定的限制 然后呢当你再来 一个新的pattern的时候 他可能就 对应于这个位置 网络然后通过 不断的这个收敛 他会收敛到其中 的某一个局部最小值 然后把原来 存在那个地方的那个pattern取出来 这就是你真正的一个联想记忆的 一个结果 这就像人比如说你给 我看一张照片这个人我可能很多 年前我看到过他但是现在在看 这个照片好像都不太一样了那 怎么办我们经常会有这种这种 这个这个体验 你看到一张 照片 哇我一下子好像想不起来他是谁你想不起来怎么办呢你就 努力的想 在你的大脑当中苦苦 的思索 我们就会有这样 的一个经历其实你的思索的过程 就是网络做了一个收敛的一个 过程 然后突然间说这不就是 高中的那个同学吗 就这个道理就 像一首歌中唱的吗 本来模糊 的脸却又渐渐清晰就是这个道理 就你给我来的这个这个pattern跟 我原来记得了它不完全一样 不 完全一样的没有关系我可以慢慢 的去收敛 收敛呢找到跟它最 接近的那个就行了所以它是一定 程度上去模拟了人的大脑的这样 的一些memory就或者联想就 这些功能 所以我觉得是非常 有趣的这样的一种神经网络的 一个架构。

有一个小例子就 这个网络呢它存了一些数字作为 pattern比如说1234 这样一些我们可以写一个数字 进去让它学习 然后把它存下来 我们呢再输入一个什么东西呢 右边这个右边这个其实是那个2 他加了20%的一些噪声 所以这个 都已经看上去就很不好看了对 所以猛一你猛一看可能都看不 出来这是个什么东西但是呢这个 网络呢通过不断的迭代不断地 收敛呢它最终还是会收敛到 就是这个它认为这个和它原来 呢是最接近它就收敛到了这样的 一个位置 这就是一个 demo就是说你输入的一个很 模糊的或者有噪声的不完整的一个信息 但是呢它可以逐渐的去 联想到原来存的一个标准的 一个pattern 这就是Hopefield 的网络一个非常重要的一个 特点。

总结:

When does ANN work?

- Instances are represented by attribute-value pairs.
  - Input values can be any real values.
- The target output may be discrete-valued, real-valued, or a vector of several real- or discrete-valued attributes.
- The training samples may contain errors.
- Long training times are acceptable.
  - Can range from a few seconds to several hours.
- Fast evaluation of the learned target function may be required.
- The ability to understand the learned function is not important.
  - Weights are difficult for humans to interpret.

Reading Materials

- Text Book
  - R. O. Duda et al., *Pattern Classification*, Chapter 6, John Wiley & Sons Inc.
  - Tom Mitchell, *Machine Learning*, Chapter 4, McGraw-Hill.
  - http://page.mi.fu-berlin.de/rojas/neural/index.html.html
- Online Demo
  - http://neuron.eng.wayne.edu/software.html
  - http://facstaff.cbu.edu/~pong/ai/hopfield/hopfieldapplet.html
- Online Tutorial
  - http://www.autonlab.org/tutorials/neural13.pdf
  - http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/faces.html

神经网络呢它的功能呢是非常的强大但是呢我们要注意 什么时候使用它什么时候呢 可能用起来会有一些麻烦这个要 特别注意如果说一个分类器非常 强大包治百病那我干嘛还要再去 学别的分类器 就每种分类它有 自己适用的一个特点比如对于 神经网络来讲它的训练时间会 比较长 我们以前做作业由训练 一些比如跟图像压缩相关的 你 随便跑了几个小时过去了 就是它 的训练是很慢的这点要特别 注意你的这个应用场景必须能够 接受相对来讲比较长的训练过程 更不要说深度学习了 现在深度 学习经常训练都是好多天所以它那个训练过程是很慢的 但是它用起来呢是比较快 这是 它的一个特点所以如果你所以我 对使用这个反应速度要很快那 可以用神经网络但是他训练是很 慢 另外呢跟决策树相比呢 神经网络呢他可能准确度特别 高 但是呢他可解释性呢就比较差 因为神经网络作为是什么东西 全是权重 神经元连上去然后 权重然后这个输出再连到别的 神经元这个东西你很难去解释的 比如说他说这个东西是苹果 那我 怎么去给别人解释说这个东西 为什么是苹果 你没办法去去解释 的对于很简单的网络向异或网络 我们刚才展示了我可以把它分析这是或门这是与非门那是与门 等等我可以去把它拆开去解释 但是稍微复杂一点的问题你都 没有办法这样去做所以神经网络 的可解释性是它的在实际应用 当中可能是一个比较严重的 一个问题。

最后呢我们可以看一些参考文献pattern recognition machine learning因为神经网络是非常 核心的一个方法所以不管是机器 学习模式识别数据挖掘这些书 里面呢都可以重点的去介绍 这些神经网络相关的知识 另外呢 在网上给大家找了一些demo 你可以去动态去看一下 包括我们 刚才讲的那个Hopefield那个网络 的例子识别这些数字等等 的都是有很多非常有趣的一些 一些资源 一些残破可以帮助 同学们呢去更好的去理解所以 我们今天呢只是讲了一些非常 浅显的一些神经网络的基本的 思路和原理 那么它理论里面还有 很多可以去研究的这样的一些 工作包括它的应用也是非常的 广泛所以呢我们呢同学们下来呢 还有很多的工作需要自己去学习 和研究。

请同学们认真学习课件所附学习资源。在这里给大家补充一些有用的信息：

深度学习入门：

http://ufldl.stanford.edu/wiki/index.php/Deep_Networks:_Overview

神经网络与自主车：从1995到2015

http://www.cs.uml.edu/~holly/91.549/readings/pomerleau-alvinn-1995.pdf

http://deepdriving.cs.princeton.edu/

Matlab资源：

https://cn.mathworks.com/help/nnet/getting-started-with-neural-network-toolbox.html （工具包）

https://cn.mathworks.com/help/nnet/examples/crab-classification.html （分类应用示例）

凤凰卫视《世纪大讲堂》栏目关于智能时代的讲座：

http://v.ifeng.com/gongkaike/sjdjiangtang/201610/0588f09f-609b-4d49-97e6-a9e0f9e85f5d.shtml

总结：

以下关于Hopfield网络特性的描述正确的是：

- [x] 基于内容的检索
- [x] 联想记忆功能
- [ ] 误差逆传播
- [x] 含噪声的模式识别

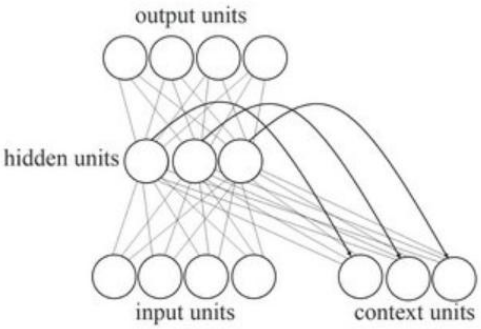
✔

前馈神经网络适用的场景为：

- [ ] 训练时间有限
- [x] 训练样本含有噪声
- [x] 需要较快的测试响应速度
- [ ] 较好的可解释性
- [x] 多分类问题

✔

在Elman网络中，第T时刻网络的输出取决于：



- ( ) 当前的网络输入
- (•) 当前的网络输入和第T-1时刻网络的内部状态
- ( ) 第T-1时刻网络的内部状态
- ( ) 当前的网络输入和第1到T-1时刻网络的内部状态

✔

总结：

总结: