



中国科学技术大学  
University of Science and Technology of China

# 人工智能讲义

## 神经网络

April 25, 2018



- ① 特征提取函数
- ② 神经网络的例子
- ③ 神经网络
- ④ 深度神经网络
- ⑤ 卷积神经网络 CNN
- ⑥ 循环神经网络 RNN

## 作用

- 消去线性  $h(x)$  的常数项:  $h(x) = \mathbf{W} \cdot \mathbf{x} + b = \mathbf{W} \cdot \phi(\mathbf{x})$
- 从输入变量空间（如图片，一段语音）映射到一个更有预测意义的“特征空间”
- 实现某些非线性的处理

寻找通用的特征提取函数的构造方法！



## 讨论线性判决的例子

- 线性  $h(x) = \mathbf{W} \cdot \phi(x)$ , 分数  $score$  由  $\mathbf{W}$  和特征提取函数  $\phi(\cdot)$  来共同决定。
- 我们有一套方法/机制来评价  $h(x)$ , 而  $h(x)$  的部分影响因素  $\phi(\cdot)$  如何评价?
- $\phi(\cdot)$  被线性分类器  $h(x)$  “使用”, 不同线性分类器使用方式不同的 ( $\mathbf{W}$  会产生相互影响), 发生的作用大小不一样, 不好评价。

因此, 摒弃或屏蔽  $\mathbf{W}$  带来的影响!

考虑相同  $\phi(\cdot)$  的所有  $\mathbf{W}$  可以屏蔽  $\mathbf{W}$  带来的影响

- 假设类:  $\mathcal{H}_{\Phi} = \{h_{\mathbf{W}} : \mathbf{W} \in R^d\}$ , 所有满足下列条件的假设/模型  $h$  的集合:
  - $\phi(\cdot)$  相同
  - $\mathbf{W}$  不同, 即参数不同
- 用假设类中使得  $h(x)$  性能最好的  $\mathbf{W}$  来代表  $\phi(\cdot)$  的性能

我们比较不同“假设类”的性能, 来评价特征提取函数  $\phi(\cdot)$  的性能

- 由  $\phi(\cdot)$  确定的“假设类”是所有可能假设的一个子集
- 学习算法从“假设类”中搜索最优的参数  $\mathbf{W}$ , 确定最优假设
- 如果特征提取函数不好, 即特征空间的表示能力不行, 再怎么学习也得不到一个好的假设  $h$
- 我们不关心  $\phi(\cdot)$  是否会导致很多“糟糕的”  $h$  存在于“假设类”中, 只关心它产生的那个最优假设性能如何
- (正则化: 控制假设类中  $\mathbf{W}$  的取值范围 (复杂性), 减小  $\mathcal{H}_{\Phi}$ , 如  $\min \|\mathbf{W}\|$ )

## 假设类的例子

$$x \in \mathbb{R}, y \in \mathbb{R}$$

Linear function:  $\Phi(x) = x$

$$\mathcal{H}_1 = \{x \rightarrow w_1 x : w_1 \in \mathbb{R}\}$$

Quadratic functions:  $\Phi(x) = [x, x^2]^t$

$$\mathcal{H}_2 = \{x \rightarrow w_1 x + w_2 x^2 : w_1 \in \mathbb{R}, w_2 \in \mathbb{R}\}$$

## 解释说明

- 我们增加了特征  $x^2$ ，就能够描述二次函数/用二次函数  $h$  来进行逼近真实  $f$ ，增加了表达能力；
- 增加特征，往往意味着增加“假设类”的大小/size，即在更大的范围内搜索最优假设  $h$ ，但是付出的代价是，通常会增加寻找最优参数  $W$  的难度



$x$  表示病人,  $y = +1$  表示身体健康,  $y = -1$  表示身体不好;

- 特征是什么?
- 基本思路: 提取和  $y$  相关的所有特征!
- 病人  $x$  的特征: 体温/血压/是否咳嗽/体重/.....
- 存在的问题: (用于线性预测  $\mathbf{W} \cdot \phi(x)$ )
  - 特征  $\phi(x)$  和输出  $y$  之间的关系可能是非单调的, 例如由体温判断健康状况;
  - 特征  $\phi(x)$  和输出  $y$  之间的关系不一定是线性的, 例如由产品购买人数预测用户是否购买;
  - 特征  $\phi(x)$  内部不同维度之间有相互作用/影响/相关, 例如由身高和体重来预测健康状况;

## $\phi(x)$ 和输出 $y$ 之间是非单调的关系

- 由体温判断健康状况：（体温低于或高于正常体温 37，意味着可能会有问题）
- 特征  $\phi(x) = [1, T(x)]^t$ ，描述偏差（常数项）和体温
- 特征  $\phi(x) = [1, (T(x) - 37)^2]^t$ ，将“正常人体温 37”硬编码进入特征提取函数，需要领域知识；
- 特征  $\phi(x) = [1, T(x), T(x)^2]^t$ ，通用方法，将构建正确特征提取函数的“基本构造因子”列入特征向量，领域知识  $(T(x) - 37)^2$  可以通过足够的样本用基本构造因子表示出来。

问题：包括哪些“基本构造因子”？





## $\phi(x)$ 和输出 $y$ 之间不一定是线性关系

- 由产品的购买人数预测用户是否会购买产品
- 特征  $\phi(x) = N(x)$ ，即购买人数。但是 100 人购买 100 件产品和 100 人购买 1000 件产品的差异被“忽视”了
- 特征  $\phi(x) = \log(N(x))$ ，当值域较大时，缩小值域
- 特征  $\phi(x) = [1[0 < N(x) < 11], 1[10 < N(x) < 101], \dots,]^t$ ，有些类似上一个方法，离散化技术

## $\phi(x)$ 内部不同维度之间相互影响

- 由身高和体重判断病人  $x$  的健康状况
- 特征  $\phi(x) = [height(x), weight(x)]^t$ , 身高和体重之间有合适的“健康”关系
- 特征  $\phi(x) = (52 + 1.9(height(x) - 60) - weight(x))^2$ , from J.D.Robinson, 需要领域/专家知识才能给出这个式子
- 特征  $\phi(x) = [1, height(x), weight(x), height(x)^2, weight(x)^2, height(x)weight(x)]^t$ , 通用方法, 给出“基本构造因子”构成的特征抽取函数
- 存在的一个问题, 基本构造因子最高次数如何确定?



## 特征提取函数设计的通用方法

- 定义输入  $x$  的各个单项式构成的向量为特征  $\phi(x)$
- 存在问题：单项式的最高次数难以确定，太高会过拟合，增加计算代价；太低可能不能很好地处理复杂的非线性决策边界

## 线性 $h(x)$ + 特征提取函数: 处理非线性 $f$

- $W$  和特征  $\phi(x)$  都是值向量，它们做内积运算得到分数  $score$ ，是线性运算，整个过程非常简单高效；
- 非线性的部分，可以用非线性的特征提取函数来获得。



### 问题描述

- 假设输入是两车的位置  $x = [x_1, x_2]$ ，一维位置
- 当车距大于等于 1 时，是安全的 ( $y = +1$ )，当车距小于 1 时，认为是危险的，二者会相撞  $y = -1$ ，真实  $f$  的表达式如 nn-1 所示
- 这是一个非线性的分类问题，可以用特征提取的方法来处理非线性，使之转化为用线性  $h$  来近似真实的  $f$ ，关键在合适地选择特征提取函数
- 这里用神经网络来实现一个  $h$ ，去近似  $f$

$$y = f(x) = \text{sign}(|x_1 - x_2| - 1) \quad (\text{nn-1})$$

## 分情况讨论

- 如果车 1 在车 2 的右边很远处, 即  $h_1 = 1[x_1 - x_2 \geq 1]$
- 如果车 2 在车 1 的右边很远处, 即  $h_2 = 1[x_2 - x_1 \geq 1]$
- 上述两种情况不可能同时出现, 只要出现一种, 就会  $h_1 + h_2 \geq 1$ , 就可以判定两车处于安全车距  $y = +1$ 。上述两种情况包括了所有安全车距的情形, 故我们可以定义  $h$  为

$$h = \text{sign}(h_1 + h_2)$$

## 数据例子

$[x_1, x_2]$	$h_1$	$h_2$	$y = h(x)$
$[3, 1]$	1	0	1
$[1, 3]$	0	1	1
$[1, 0.5]$	0	0	0

Table: 具体数据例子

## 求解方法总结与提高

- 定义  $\Phi(x) = [1, x_1, x_2]$
- 分别求两个子问题:  
 $h_1 = 1[v_1 \cdot \Phi(x) \geq 0]$   
 $h_2 = 1[v_2 \cdot \Phi(x) \geq 0]$
- 然后综合两个子问题的解,  
 $h = \text{sign}(w_1 h_1 + w_2 h_2)$ , 得到原问题的解。
- 反思: 非线性问题, 用多个线性问题来近似 (本例子中要解三个线性子问题)。**分而治之**

## 求解子问题 $h_1$

- $h_1 = 1[v_1 \cdot \Phi(x) \geq 0]$ ，分段线性函数（两条射线的并集）
- 任意二值的两段线性函数，总可以用一个解析式表示出来： $h_1 = \text{sign}(v_1 \cdot \Phi(x))$
- 因为已知  $h_1 = 1[x_1 - x_2 \geq 1]$ ，故能看出来该子问题  $v_1 = [-1, 1, -1]$ ，通常我们对  $h_1$  一无所知，只有数据集，该如何求解  $v_1$ ？
- 如左下图所示的符号函数，无法利用梯度信息来求解  $v_1$ ，因此我们引入右下图所示的可导的 logistic 函数来替换符号函数
- 数据集 + 线性模型  $\rightarrow$  优化问题  $\rightarrow$  梯度下降

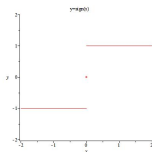


Figure: 符号函数:  $\text{sign}$

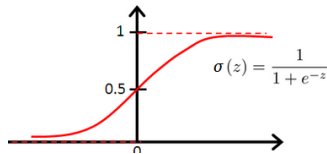


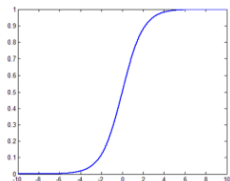
Figure: logistic 函数

## 激活函数

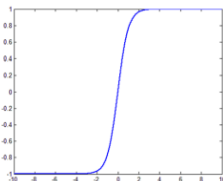
- 对分数/score 做一种非线性映射/变换，增加非线性处理能力，是对生物机制的模拟。

## 常见的激活函数<sup>1</sup>

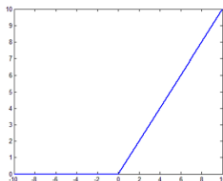
Sigmoid



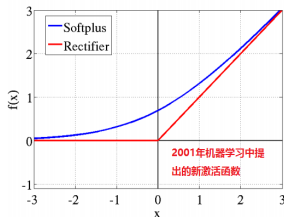
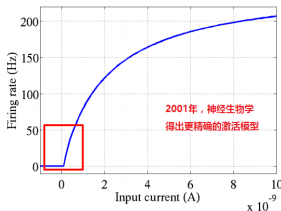
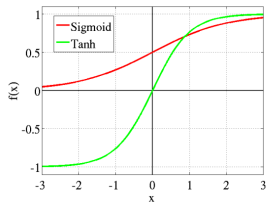
Tanh



Rectified Linear



<sup>1</sup>sigmoid 就是 logistic



## ReLU/Rectified Linear Units/Rectifier 的优点

- 稀疏激活性：激活少量的神经元，2003 年 Lennie 等人估测大脑同时被激活的神经元只有 14%；快速提取（稀疏）特征，噪声的剔除；线性特征，非线性处理变成选择性激活
- 单侧抑制（有界）和相对较宽阔的兴奋边界，降低梯度“消失”的影响（梯度的计算要乘两个小于 1 的缩减因子）



## 图形化线性函数

- 我们用一种图形来直观描述一个线性函数，如图所示

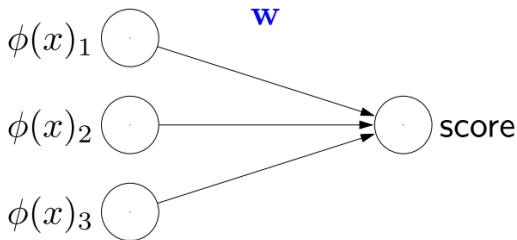


Figure: 线性函数  $score = [\phi(x_1), \phi(x_2), \phi(x_3)] \cdot W$

### 图形化“激活的”线性函数

- 若将激活函数加入，得到下图

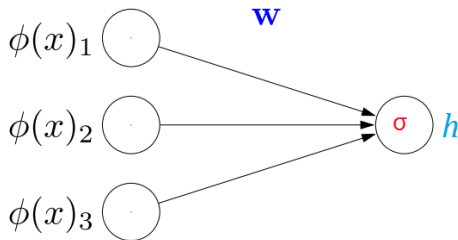


Figure: “激活的” 线性函数  $h = \sigma(\text{score}) = \sigma([\phi(x_1), \phi(x_2), \phi(x_3)] \cdot W)$

## 图形化线性函数

- 将例子中三个线性子问题的求解过程描述依据求解的次序描述出来 ( $h_1, h_2$  可以并行求解), 如图所示, 这就是“神经网络”

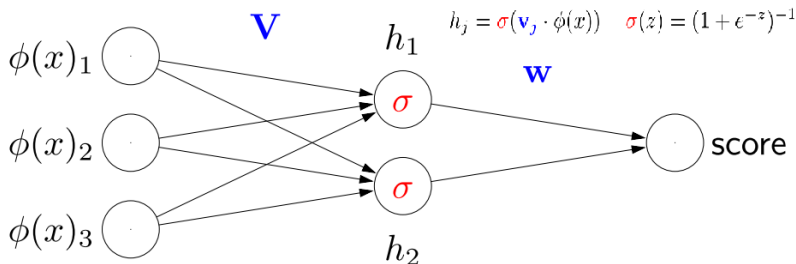
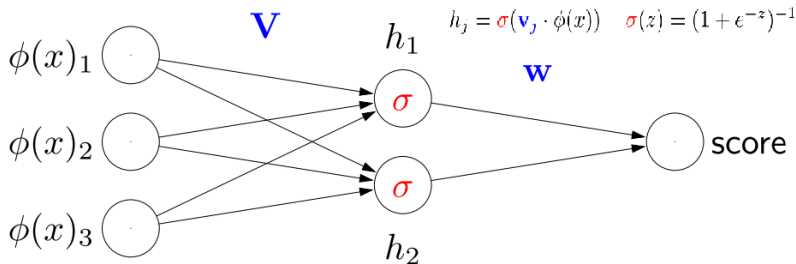


Figure: 例子：神经网络

## 基本概念: NN

- 网络结构: 输入层、隐层、输出层, 权值, 隐层单元/节点
- 激活函数
- 训练信号/样本数据
- 隐层输出与数据特征



## 例子中存在的基本问题

- 我们不知道  $h_1$  的意义，即车 1 在车 2 很远的地方
- 求解子问题  $h_1$  需要有训练数据，实践中，没有用于训练  $h_1$  的数据
- 具体来说，如下表， $h_1, h_2$  列是手工分解问题后的中间结果，一般情况下，我们没有手工分解问题及其中间结果，即中间两列不存在。如何获得  $v_1, v_2$ ?  $w$  的训练是有“监督信号”的。
- 思路：能不能利用  $w$  的监督信号，把它无法解决的“误差”反向传递给  $v_1, v_2$ ，让它们去解决？

$[x_1, x_2]$	$h_1$	$h_2$	$y = h(x)$
$[3, 1]$	1	0	1
$[1, 3]$	0	1	1
$[1, 0.5]$	0	0	0

Table: 具体数据例子

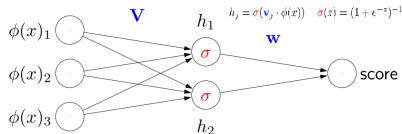


Figure: 例子：神经网络



## NN 中权值学习问题描述

- 给定数据集  $\mathcal{D}$
- 给定网络结构：隐层数目及各层节点数，连边方式等；
- 求： “各边的权值”

## 导数/偏导数的意义

- 输入  $in$  的微小改变是如何影响输出  $out$  的? 一种“敏感性分析”
- 当  $in_1$  发生改变  $\epsilon$  时, 输入  $out$  发生的改变为  $\frac{\partial out}{\partial in_1} \epsilon$
- 用图形来描述导数计算, 如下图 ( $out = function(in_1, in_2, in_3)$ )。边上的绿色字体标记导数, 孩子节点是输入变量。

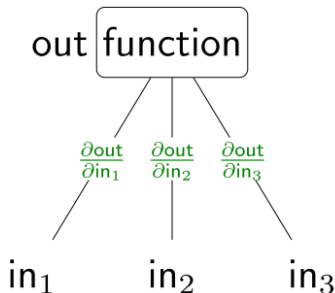


Figure: 导数计算的图形化描述

# 常见函数的偏导数图形表示

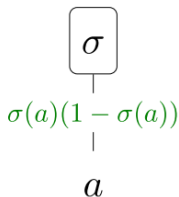
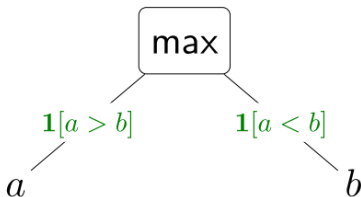
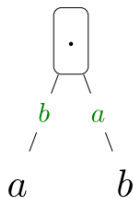
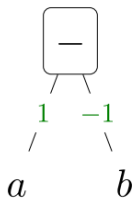
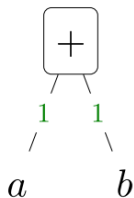
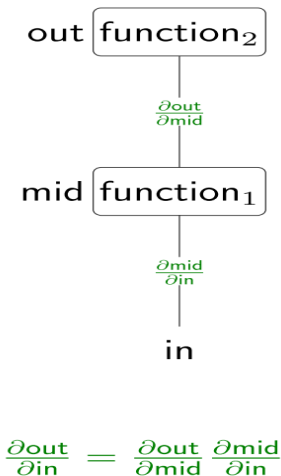


Figure: 常见函数的偏导数图形表示

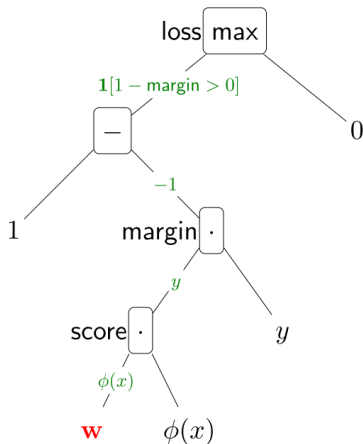




## 进一步解释

- 复合函数求导的“链式法则”
- 神经网络权值训练需要用梯度下降法，计算一个复杂的复合函数的梯度/偏导数
- 如右图， $in$  的微小改动  $\epsilon$  最后在  $out$  出的变动为  $\frac{\partial out}{\partial in} \epsilon = \frac{\partial out}{\partial mid} \frac{\partial mid}{\partial in} \epsilon$
- 计算复合函数的导数时，计算其孩子节点所在单路径子树的分支乘积。

Figure: 复合函数的偏导数图示



## 进一步解释说明

- hinge loss:  $loss(x, y, W) = \max\{1 - W \cdot \phi(x)y, 0\}$
- 右图由“常见偏导数的图形表示”，以及“复合函数的偏导数描述”二者一起绘制出来
- 最终偏导数/梯度信息，由边上的绿色“权值”相乘获得。 $\frac{\partial loss(x, y, W)}{\partial W} = -1[margin < 1]\Phi(x)y$

Figure: Hinge loss 函数的偏导数

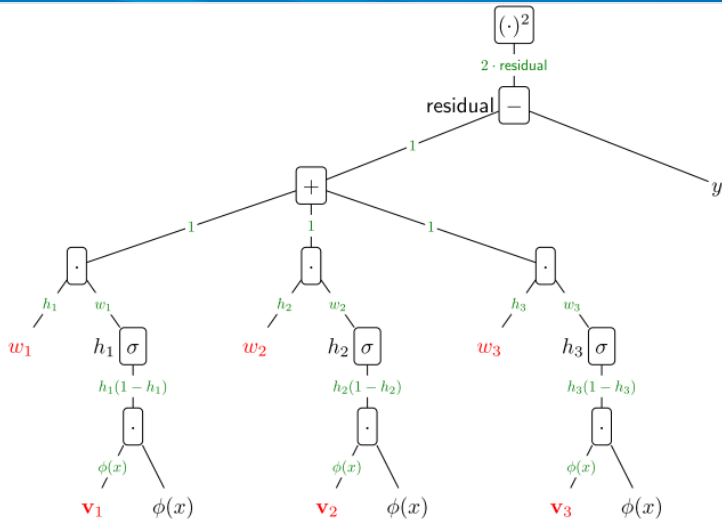


Figure: 平方损失函数的梯度/偏导数计算

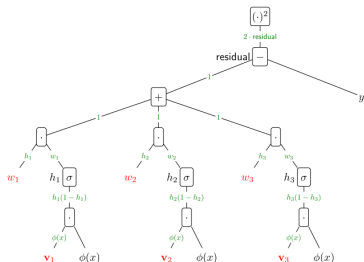


Figure: 平方损失函数的梯度计算

- 核心亮点：快速梯度计算
- 权值更新：知道梯度信息，也就知道了“最佳”修改权值的方法，权值沿负梯度方向行走一个步长/学习率。

## BP 算法核心思想

- 给图中红色叶子节点/变量赋随机初值；
- 将数据  $(x, y)$  代入叶子输入变量，从叶到根，计算每个中间节点的输出（前向值  $f_i$ ），直到根节点/输出节点，得到  $f_0$ ；此为 **前向过程**；
- 我们希望根  $f_0^* = 0$ ，故，从根开始，把“误差”向下（向输入层）反馈。回答问题：根节点有误差  $g_0 = f_0 - f_0^*$ ，是如何被任意一个中间节点  $i$  的误差所影响？求每个中间节点对根节点影响，即后向值：  

$$g_i = \frac{\partial f_0}{\partial f_i} \frac{\partial f_i}{\partial w_i} = \frac{\partial f_0}{\partial f_i} g_j$$
 此为 **后向传播**



- 步长/学习率如何设置达到最优？
- 网络有几个隐层，每个隐层各有几个节点？节点间如何连边？（网络结构设置）
- 如何对时间序列问题进行预测？
- 如何使用样本？先把一个样本数据进行多次循环（前向/后向过程），训练完毕，再训练下一个样本（在线学习），还是一批样本依次轮换循环？
- .....

## 深度学习

- 在图像/语音识别领域获得了突破性进展<sup>2</sup>，成为目前机器学习领域最热门的研究课题。
- 深度学习的出发点: 类标号也是一种输入数据的“抽象”特征，分类器  $h$  被视为一种“特征提取函数”。

## 几点解释说明

- 以前，我们总是用神经网络直接来实现图像或语音的识别/分类；现在，认为神经网络可用来自动提取图像/语音的特征；
- 模式识别中，一般来说，如果特征提取的非常好，分类器就会很容易获得，如用简单的线性判别即可；
- 如何设计好的“特征提取函数”？具体问题具体分析，领域知识等。
- 深度学习，近乎于任何图像/语音数据都可以用神经网络来进行自动特征提取，而且效果非常好！神奇！

<sup>2</sup>参考《Natruue》2015 年综述性论文 “deep learning”

## 神经网络被诟病的缺点之一

- 在做预测/分类之前，神经网络的训练代价大，收敛速度慢（获得收敛的权值）；当隐层的层数和节点的数据增加时，该问题尤为突出；以前，通常应用中都只用一个隐层。

## 深度神经网络

- 很多个隐层，但是不用 BP 算法去训练权值。

如何做？新的权值训练方法用于深度神经网络！

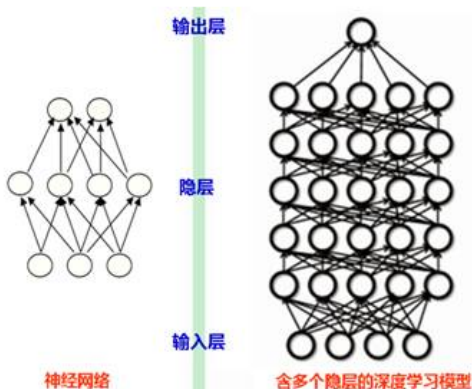


Figure: 例子：深度神经网络



## 特征提取的核心思想：自编码器

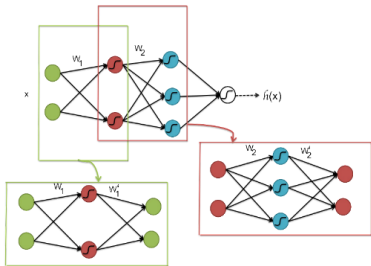


Figure: 例子：深度神经网络学习方法的改进

- 每个隐层独立训练，训练一个三层 BP 网络；
- 对任意一个三层 BP 网络：输入层是前一个隐层的输出（或原始输入数据），输出层与输入层完全一样；训练该三层 BP 网络；
- 丢弃输出层及其关联边；
- 重复上述过程，进行多次，得到多个连续的隐层；这个循环过程构建了一系列的自编码器。
- 最后，利用“监督信号”训练最后一个输出层连边的权值。

## 特征提取不需要“监督信号”

- 现实中，训练数据/类标签通常难以获得；有大量的数据没有标签（即有  $x$ ，没有  $y$ ），该方法的特征提取过程可以充分利用大量没添加标签的数据。（分类就是为了给数据自动添加标签）

## 训练代价相对于整体的梯度下降要小很多

- 虽然每个三层 BP 网络的训练还是比较费时间，相对于多层的 BP，时间代价已经降低了。

## 隐藏的约束条件

- 假设数据来自某个具有“特点”数据生成器
- 或者说给定的数据集具有“可分类”的特点

从左下角，沿蓝色箭头方向查看。

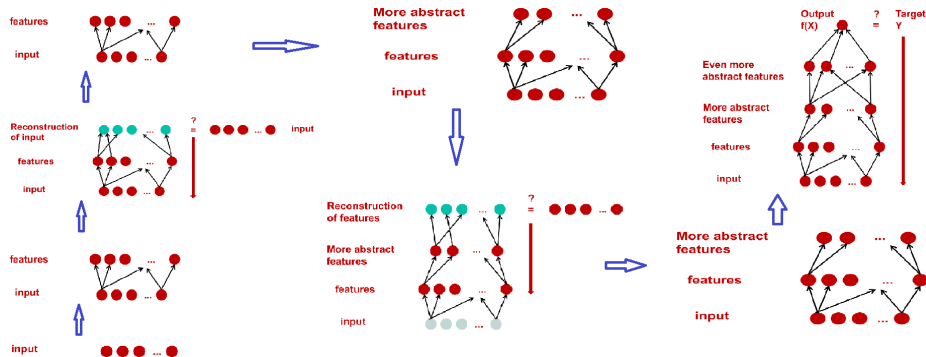


Figure: 深度神经网络构建示意图



## 深度神经网络

- 任何一个隐层，实现了对原始输入数据的一种特征提取，找到了数据的“规律”，实现了一种“数据压缩”。

## 问题

- 超参数的设计：层数，每层的节点数；
- 如何避免隐层节点及其参数是“平凡的”；
- 一般认为：随着层次的增加，后续的特征越来越抽象；越抽象就能更好地去实现分类吗？
- 不同类型/超高维原始输入数据的应用（大数据的特点）
- 理论分析困难：随机性
- 深度神经网络更接近生物大脑？

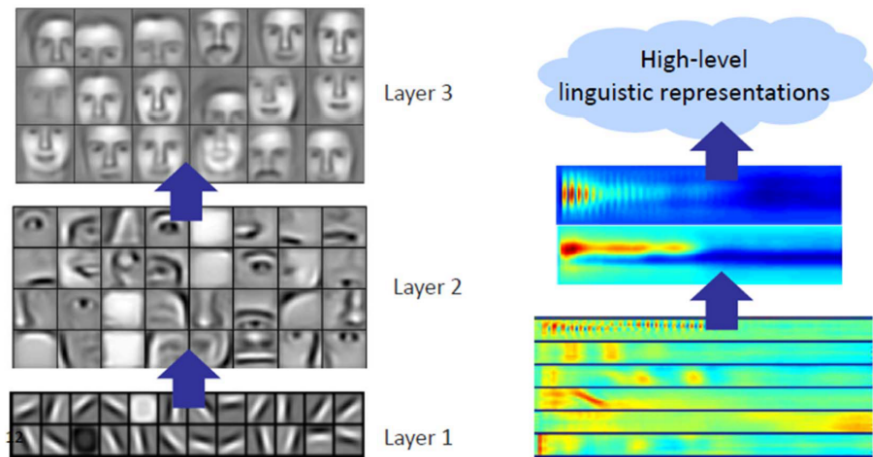


Figure: 例子：分层提取的特征



## 两个关键点

- 观点的改变：神经网络用于特征提取，隐层是一个特征提取函数；
- 权值学习方法的改变：自编码器，实现隐层代表的特征提取函数。

## 两大类深度神经网络<sup>3</sup>

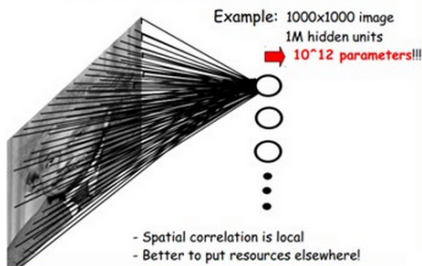
- 卷积神经网络 / Convolutional Neural Network, CNN: 对连接方式做了约简处理
- 回归神经网络 / Recurrent Neural Networks, RNN: 应用于序列数据

<sup>3</sup>进一步学习请网上搜索相关文献和材料，slides 中仅仅阐述了基本原理。

## 输入

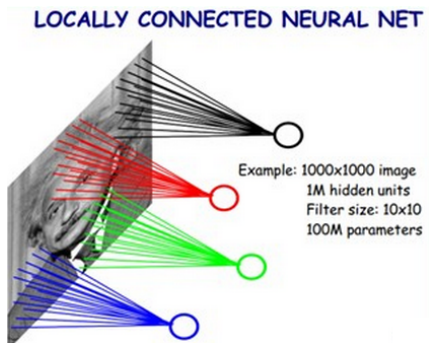
- 假设输入是二维图像，一个数值矩阵描述，矩阵每个元素代表对应位置像素点的灰度值；如图所示
- 构建全连接神经网络，隐层节点数为 1 百万，输入层和隐层之间连边有  $10^{12}$  条，也就是需要确定的参数个数
- 训练难度高，需要样本量大

### FULLY CONNECTED NEURAL NET



## 利用：隐层节点的局部性

- 将隐层的每个节点视为一个神经元，能感知外界输入信息（和输入层相连），处理输入层的信息；
- 一个（合理的）假设：每个隐层节点能感受到的信息都是输入信息的某个局部（图像的某个小块）；如图所示；每个隐层节点连接  $10 \times 10$  的局部区域，共  $10^8$  个参数要确定
- 隐层节点感知的信息处理后被后续的隐层节点处理，综合成更大范围、直至全局的信息。

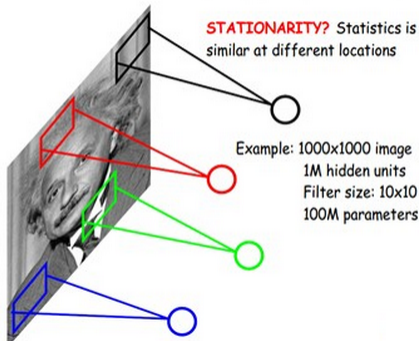




## 利用：隐层节点的同伦性

- 将隐层的每个节点都是一样的，假设其对输入的加权方式和处理方式都是同样的，即功能完全一样的神经元；（合理性：统计显示不同位置，神经元处理方式相似）
- $10^8$  个参数变成了 100 个参数需要确定；这种技术被称为“权值共享”。其中这 100 个参数构成的神经元处理方式称为一个滤波器/filter 或卷积核

### LOCALLY CONNECTED NEURAL NET





## 卷积核：例子 1

## 卷积核：例子 2



0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

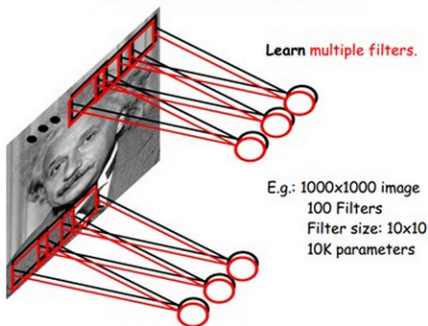
\*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

## 不同的卷积核，不同的特征

- 不同的卷积核，对应一类不同的图像特征 (或者叫特征提取方法); 每个卷积核作用在输入图像上, 就得到一个特征映射 (feature map); 特征映射可视为图像的一种变换
- 取多个不同的卷积核, 提取多种特征, 多角度描述图像; 每个特征形成一个“平面图像”
- 参数的个数和卷积核的结构大小, 卷积核的个数相关, 与输入层节点数、隐层节点数无直接关系!
- 卷积层的层数越高, 提取到的特征就越全局化。

### CONVOLUTIONAL NET





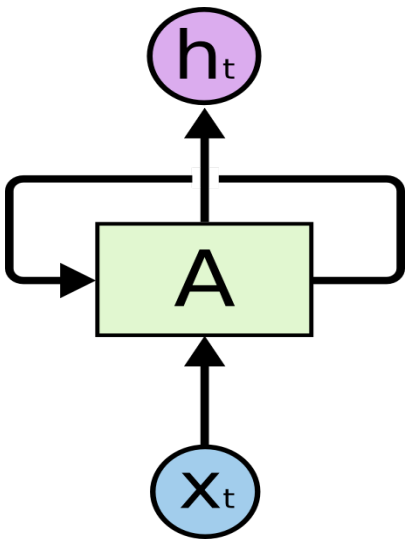
### 池化/Down-pooling/下采样:

- 聚合特征、降维，达到减少运算量的目的；
- 对一块数据进行抽样或聚合，例如选择该区域的最大值（或平均值）取代该区域；
- 下图的例子将  $10 \times 10$  的区域聚合成  $1 \times 1$  的小方块。思考一下，所有的卷积结果都这样处理一遍。



### 输出层:

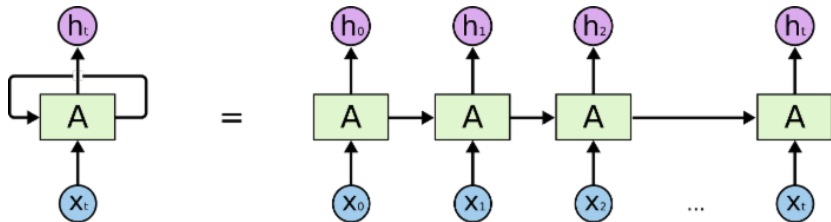
- 全连接层
- 前面的层输出作为输入
- 以此构建一个经典的用于分类的神经网络



## RNN 图示与符号

- A: 神经网络模块,  $x_i$  输入,  $h_i$  输出
- 循环使得前一时刻的信息能当成当前时刻的输入

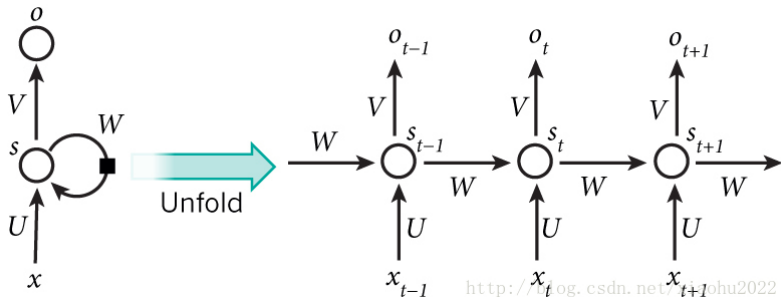
理解：将 RNN 的循环展开



长期依赖问题

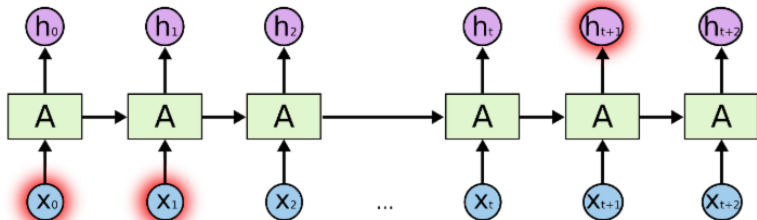
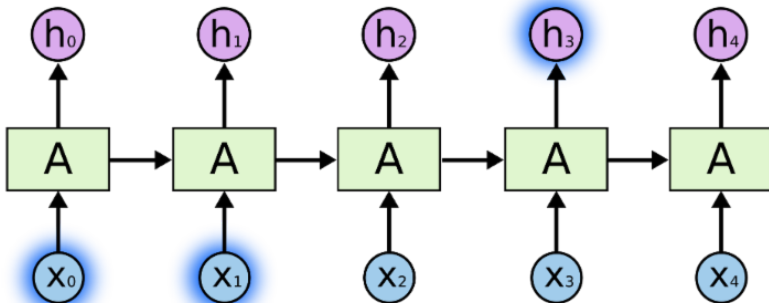
- RNN 的特点就是能把历史信息应用到当前
- 多长的历史信息会对当前产生影响？
- RNN 很难处理时间上很长的历史信息





## RNN 背后

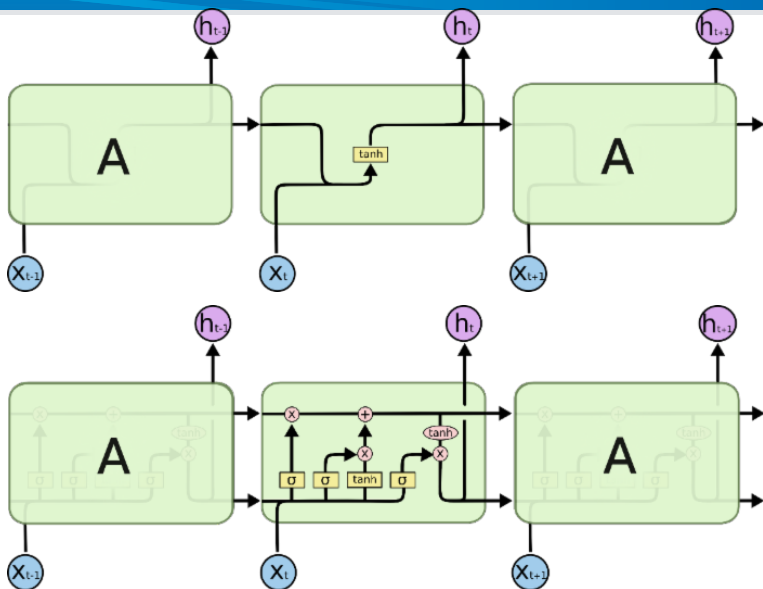
- 序列数据，先后出现的数据之间存在关联
- 对序列数据的每个数据，RNN 执行相同操作，但是输入和前面的状态或输出相关
- 隐藏层  $s$  的状态计算  $s_t = f(Ux_t + Ws_{t-1})$ ，其中  $f$  可以是  $\tanh$  或  $\text{ReLU}$ ,  $s_t$  含有历史信息
- RNN 用隐含层状态来捕获数据的特征

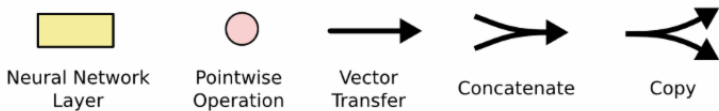




## LSTM: Long Short Term Memeory, 长短时记忆网络

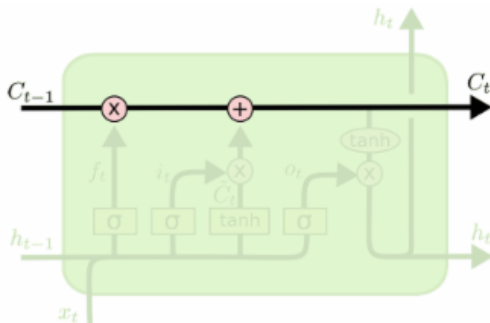
- 特殊类型的 RNN，可用于学习长期依赖信息；
- 刻意的设计，LSTM 的默认行为是记住长期信息；
- 改进之处：RNN 中的 A(重复的循环体) 只有一个单一的层结构，例如 tanh





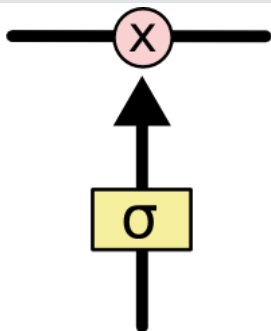
## 图例说明

- 带箭头的实线：表示在两个节点间传输一个向量
- 粉色圆：表示 pointwise 的操作，例如求两个向量的和
- 黄色矩形框表示学习的神经网络层



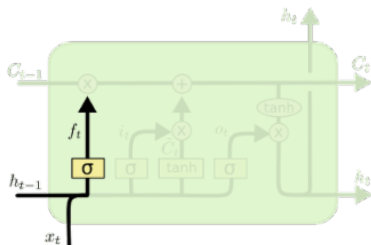
## 类比解释:

- 最重要的信息流: 如图, 从左到右水平方向  $C_{t-1} \rightarrow C_t$
- 信息在黑实线上传输, 途中经过其它信号的叠加或干涉



## “门”的设计:

- 信息流中增加或删除信息的控制开关
- 包含一个 sigmoid 神经元/网络, 和一个 pointwise 的乘法操作
- sigmoid 输出  $[0, 1]$  之间的实数, 描述信息流每个分量能通过多少, 例如: 0 表示不允许任何量通过, 1 表示允许任意量通过。

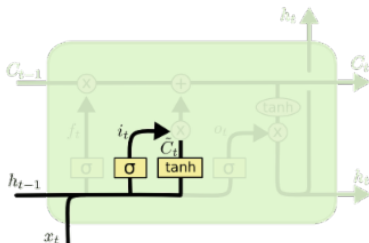


$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

## LSTM 的遗忘门设计:

- 第一步：信息流中删除什么信息？
- 从上一时刻的输出  $h_{t-1}$  和当前的输入  $x_t$  通过一个 sigmoid 神经元，输出一个  $[0, 1]$  之间的实数，0 表示完全舍弃，1 表示完全保留；也就是这个输出和  $C_{t-1}$  的每个分量做乘法运算。

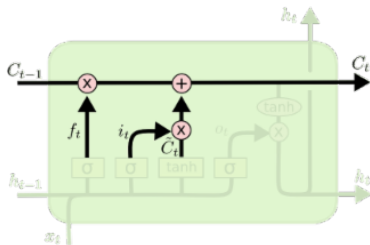




$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \sigma(W_C \cdot [h_{t-1}, x_t] + b_C)$$

## LSTM 的输入门设计:

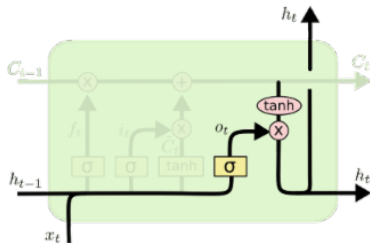
- 第二步：信息流中增加什么信息？
- 第一个 sigmoid 层，被称为输入门层，用来选择/确定更新值；一个 tanh 层来创建各个更新值的候选值  $\tilde{C}_t$



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

## LSTM 的遗忘门和输入门的实现：

- 前两步确定了信息如何删减和增加
- 如图，开始实现： $C_{t-1} * f_t$  实现历史信息的遗忘，然后加上  $i_t * \tilde{C}_t$  得到新的状态值/信息流；

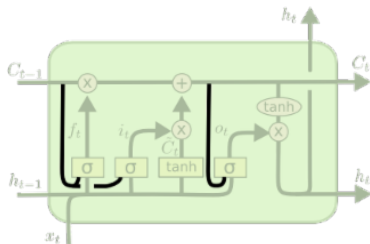


$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

## LSTM 的输出门的设计与实现:

- 第三步，信息流中的什么信息需要被输出？
- 一个 sigmoid 层来确定那一部分信息被输出；信息流的 tanh 值被计算，形成输出的候选



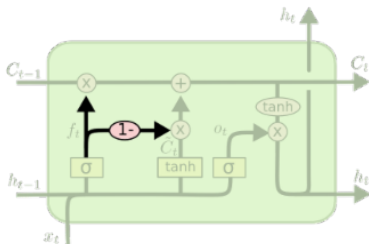
$$f_t = \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma(W_o \cdot [C_t, h_{t-1}, x_t] + b_o)$$

## 流形 LSTM:

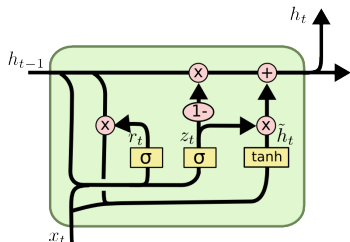
- 增加 peephole 连接, 如图
- sigmoid 层接受信息流状态的影响
- 并不是每个 sigmoid 门都添加 peephole 连接



$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t$$

## 耦合型 LSTM:

- 如图，删除信息和增添信息由一个 sigmoid 门确定
- 输入的新信息仅仅是用来替代删除的旧信息



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

## GRU-LSTM:

- GRU: Gated Recurrent Unit, 遗忘门 + 输入门  $\rightarrow$  更新门
- 如图, 还混合了信息流状态, 隐藏状态等; 比标准 LSTM 简单?