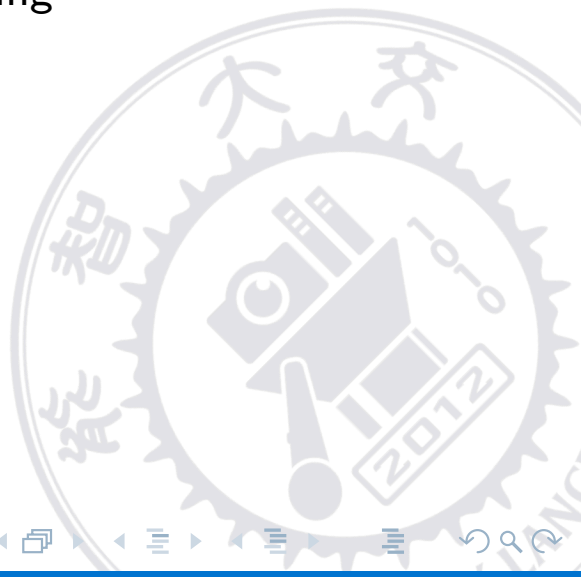
Lecture 10: 语音识别的深度神经网络模型

Deep Neural Network for Speech Recognition

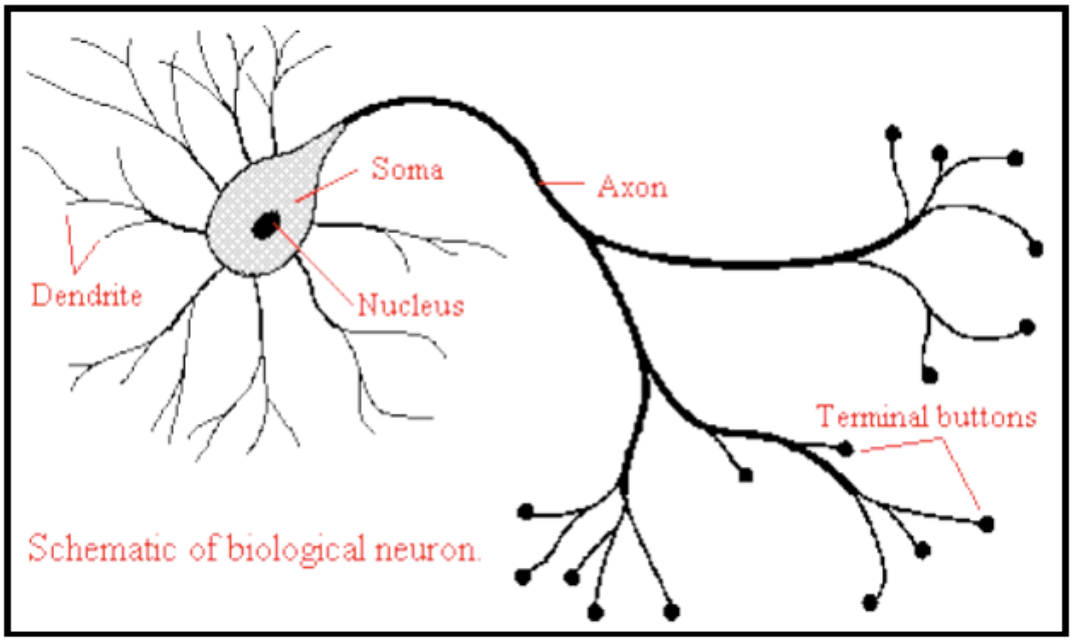
Kai Yu and Yanmin Qian

Cross Media Language Intelligence Lab (X-LANCE)
Department of Computer Science & Engineering
Shanghai Jiao Tong University

Spring 2021



生物神经元



- 人类大脑中有约 100 亿个神经元
- 输入刺激的总和
  - 空间 (信号)
  - 时间 (脉冲)

生物神经网络

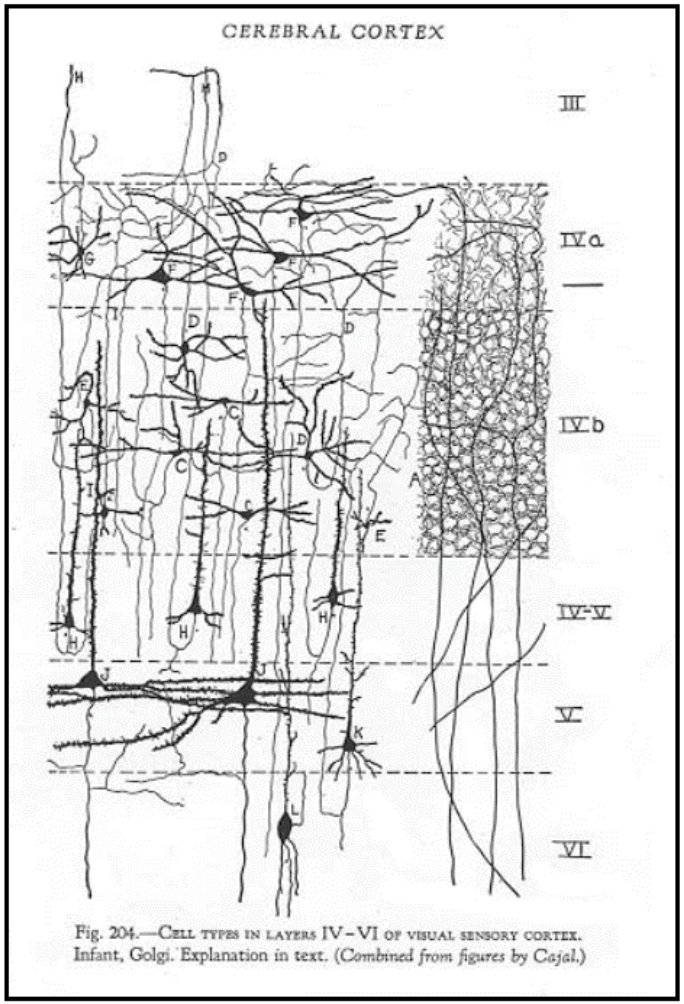


Fig. 204.—Cell types in layers IV–VI of visual sensory cortex. Infant, Golgi. Explanation in text. (*Combined from figures by Cajal.*)

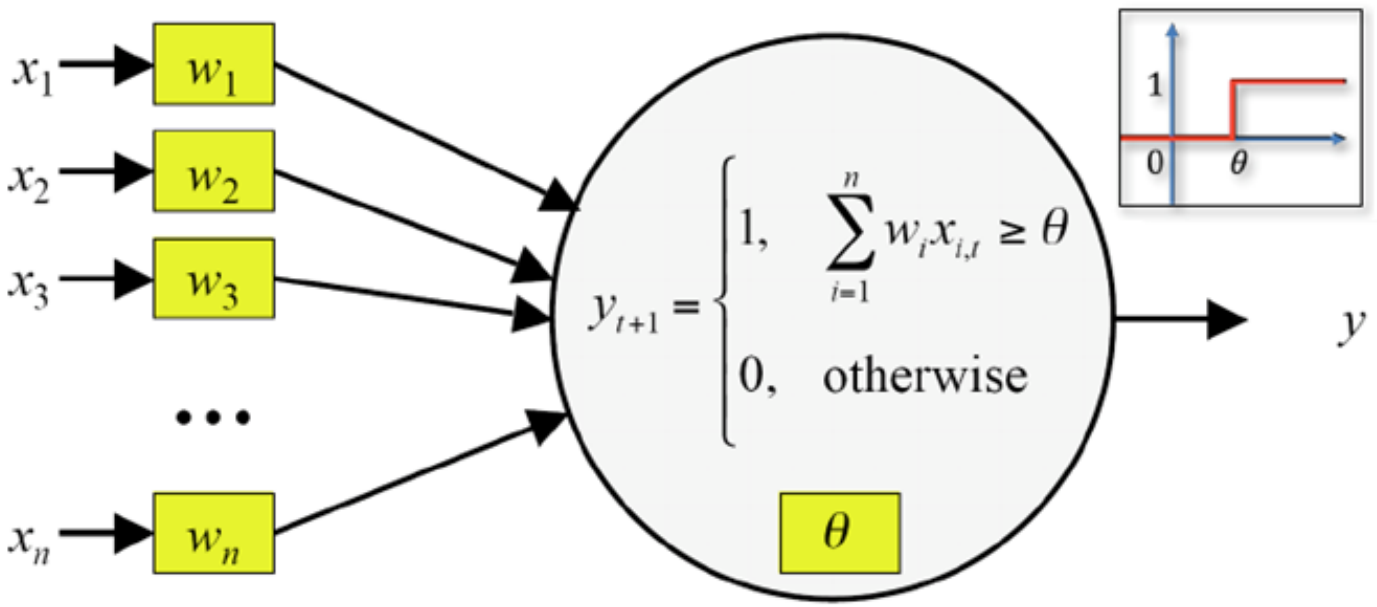
- 更多神经元
- 计算能力 = 连接度 (connectivity)
- 可塑性
  - 新的连接
  - 修改后的连接强度

什么是人工神经网络

- 人工神经网络是由许多简单的处理器（神经元、单元）组成的网络。
  - 单元之间由连接联系起来
  - 每个连接都有一个权重
  - 每个单元仅通过其权重和从连接接收的输入来局部运行

第一代神经网络

- 1943 年，McCulloch 和 Pitts 开发了神经元的基本模型
- 没有隐层的感知机



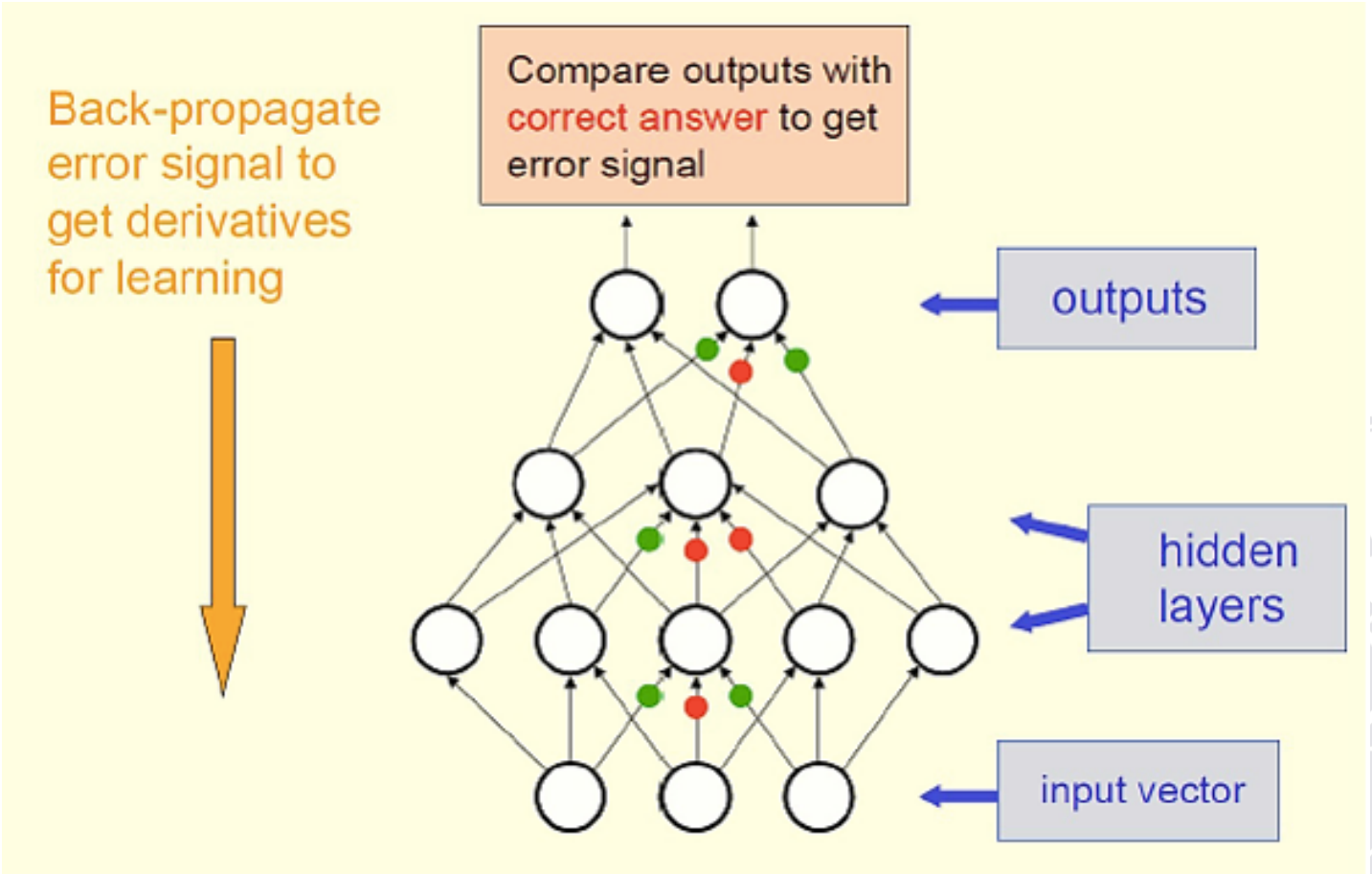
第一代神经网络

- 1948, Wiener: 控制论
- 1949, Hebb: 学习理论 (Hebb's rule)
- 1958, Rosenblatt: 感知器模型和感知器收敛算法
- 1960, Widrow-Hoff: 最小均方算法

- 1969, Minsky-Papert: 感知机的局限性: (不能解决非线性可分的问题)

第二代神经网络

- 多层感知机 (Multi-Layer Perceptron, MLP) (80’-90’)
- 反向传播 (Back-Propagation)



第二代神经网络

- 1980s, Stephen Grossberg: 自适应共振理论
- 1982, Hopfield: 能量函数, 循环网络模型
- 1982, Kohonen: 自组织映射
- 1986, Rumelhart, Hinton et. al.: 反向传播

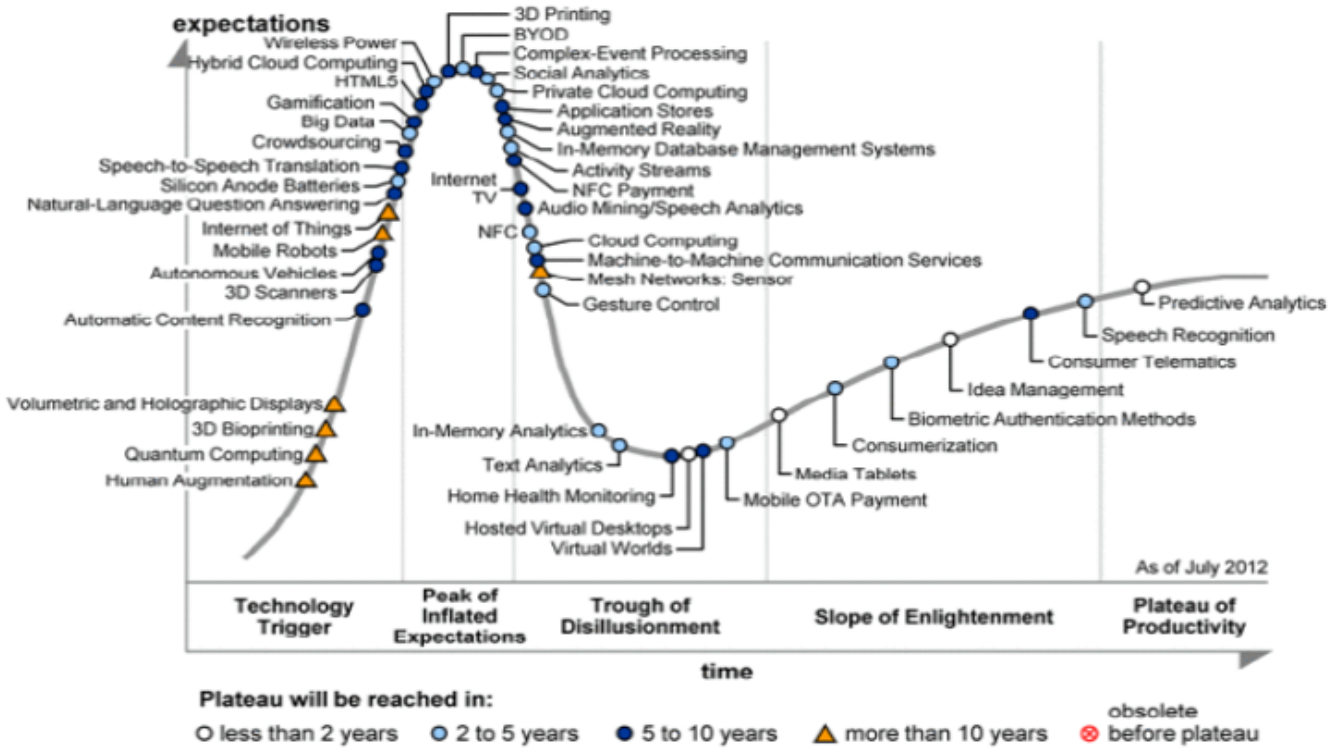
- 1990s: 衰退
  - 需要经验和技能
  - 容易过训练 (over-train) 或陷入局部最优点
  - 很难做得更深

神经网络复兴

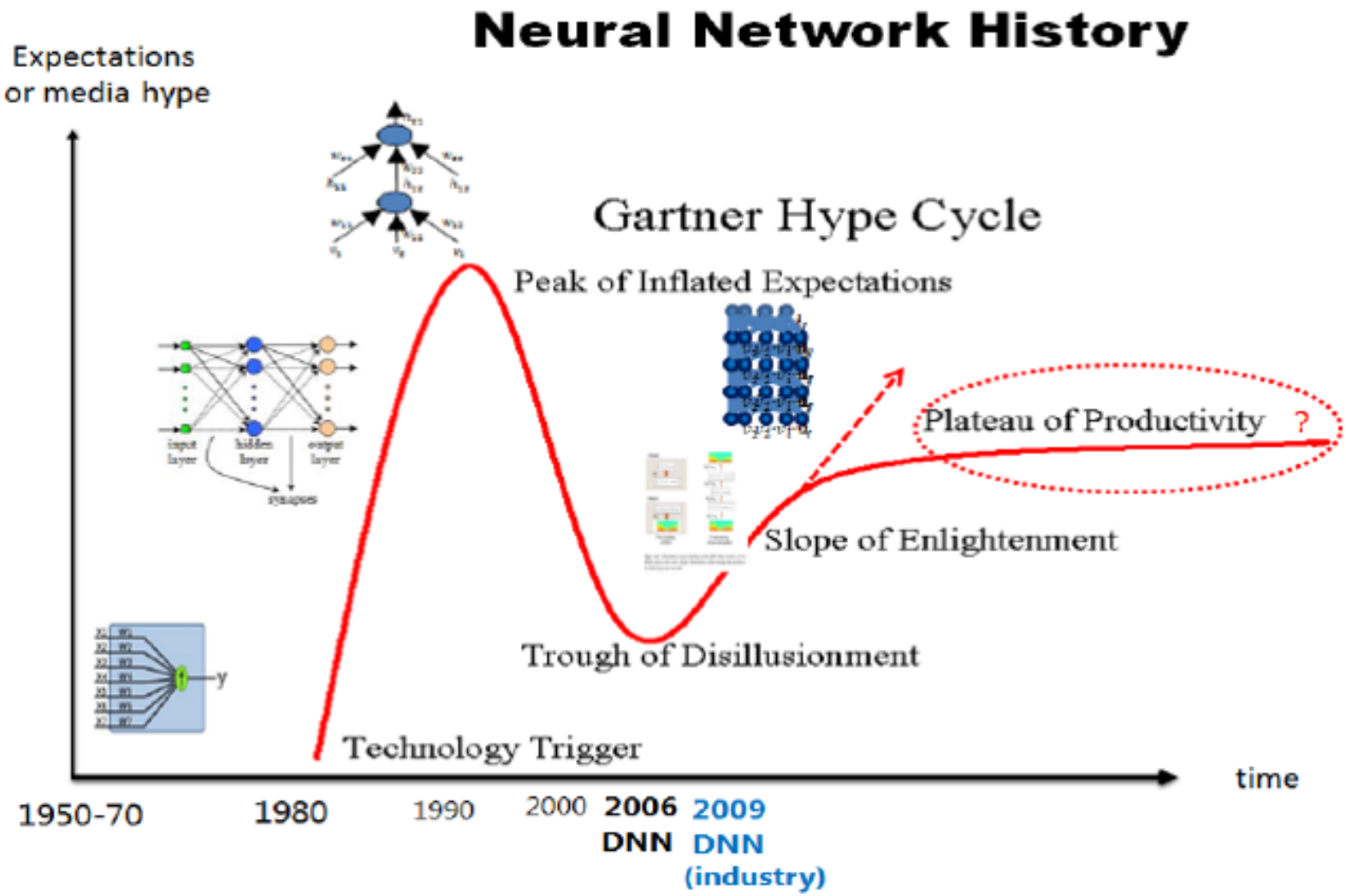
- 2006 年，Geoffrey Hinton 发明了深度信念网络 (Deep Belief Networks, DBN)，可以快速有效地进行深度神经网络学习
  - 自底向上地逐层预训练
  - 每一对网络层都是一个受限波尔兹曼机 (Restricted Boltzmann Machine, RBM)
  - 使用反向传播对所有层进行联合训练调整 (Fine-tune)

Gartner IT 技术成熟度曲线

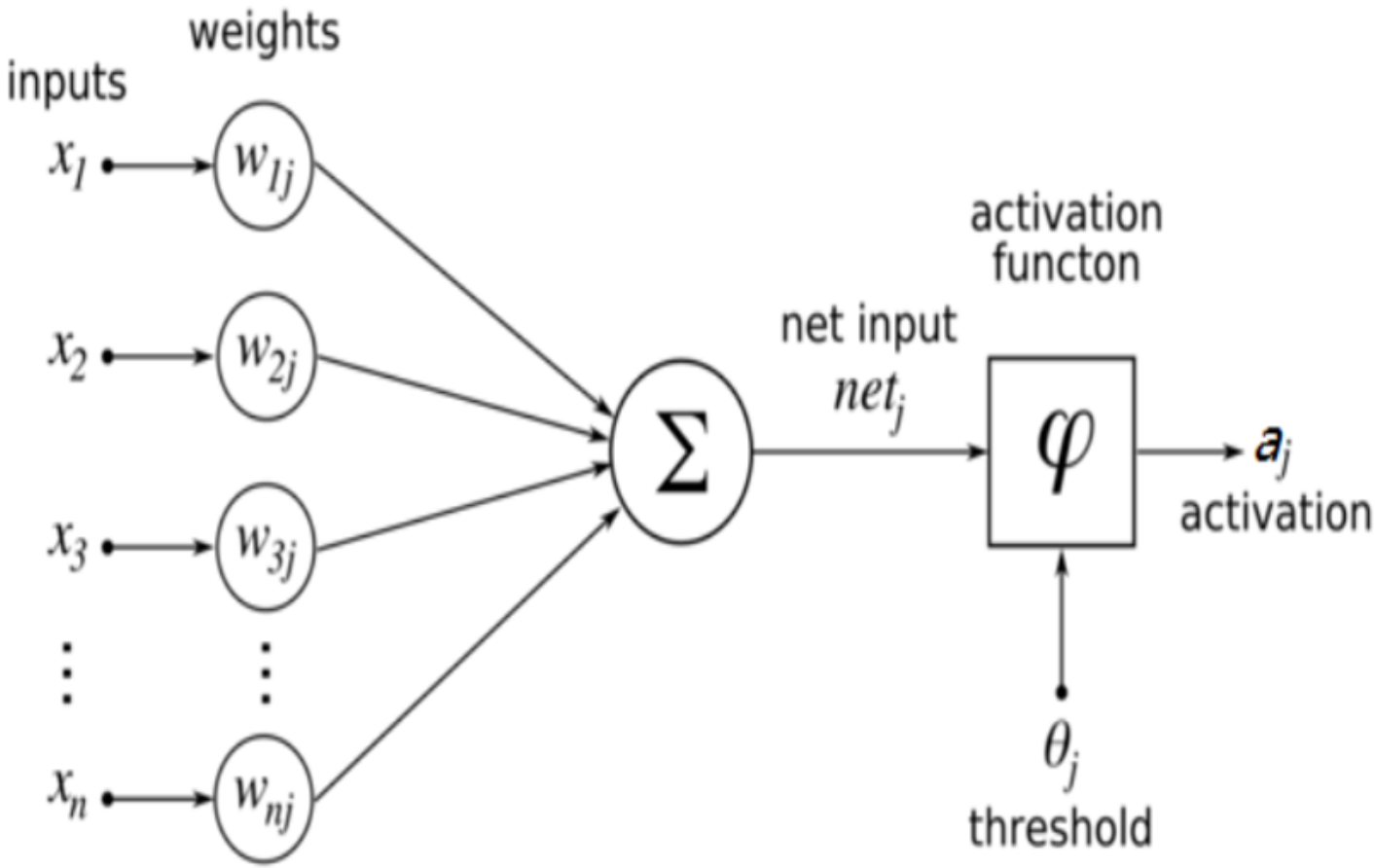
- 代表着具体技术的成熟度、接受和社会应用情况
- 五个阶段：技术触发；期望值膨胀的高峰；幻灭的低谷；再次启蒙的斜坡；生产力阶段的高原



Gartner 人工神经网络成熟度曲线



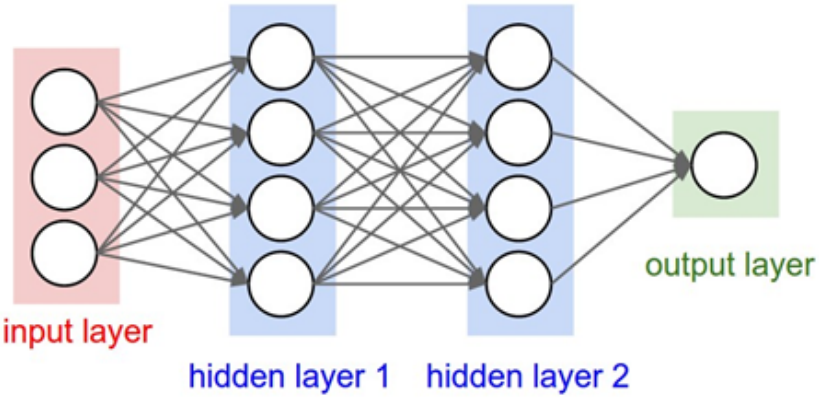
感知机：人工神经网络 (ANN) 的基石

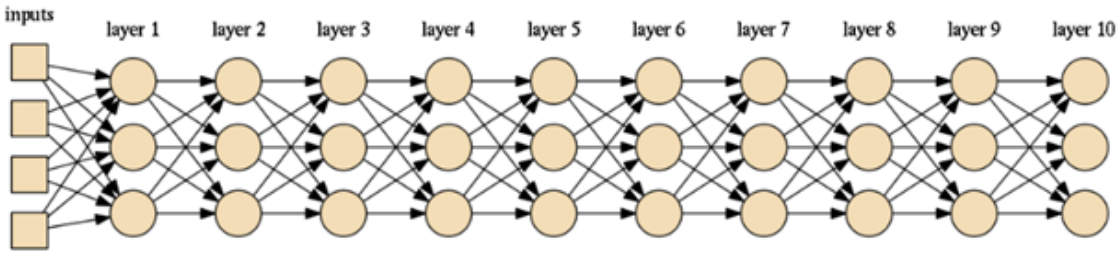


$$a_j = \varphi(\theta_j + \sum_i (w_{ij} x_i))$$

全连接 (FC) 神经网络: 从浅到深

- 全连接层:





深度神经网络 (DNN) 架构

- DNN 即具有多个（>2）隐层的传统多层感知机 (MLP)
- 前 L 层

$$\mathbf{v}^{\ell} = f(\mathbf{z}^{\ell}) = f(\mathbf{W}^{\ell}\mathbf{v}^{\ell-1} + \mathbf{b}^{\ell}),$$

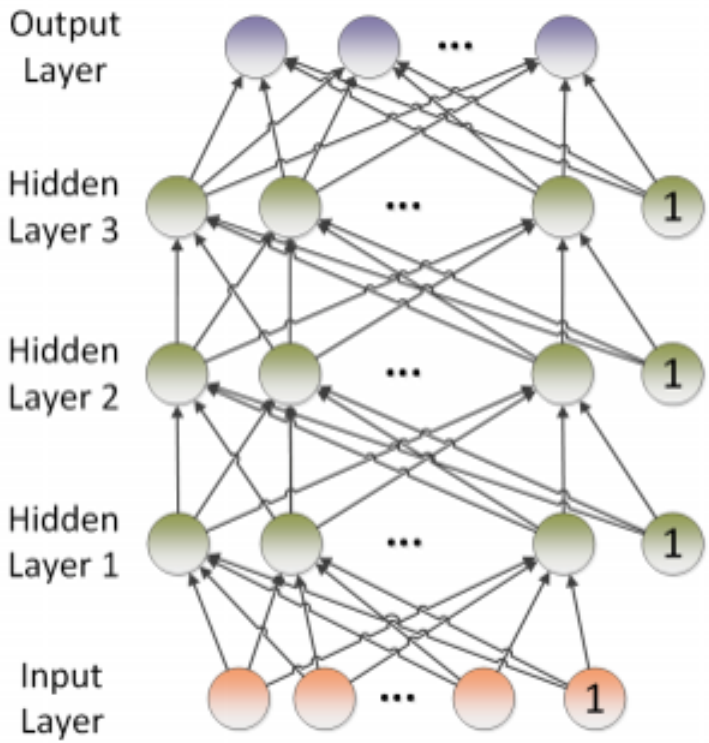
for $0 < \ell < L$

- 回归 (regression) 问题中的输出层

$$\mathbf{v}^{L} = \mathbf{z}^{L} = \mathbf{W}^{L}\mathbf{v}^{L-1} + \mathbf{b}^{L}$$

- 进一步应用在分类问题中

$$v_i^L = P_{dnn}(i|\mathbf{o}) \quad = softmax_i(\mathbf{z}^L)$$
$$= \frac{e^{z_i L}}{\sum_{j=1}^{C} e^{z_i L}}$$



DNN 中的激活函数 (Activation)

- 非线性激活函数被逐元素地应用于激励向量 (excitation vector)
- Sigmoid: 最常用

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- Tangent: 缩放版的 Sigmoid，具有同样的建模能力

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

- Rectified linear unit (ReLU): 施加稀疏的激活输出

$$ReLU(z) = \mathtt{max}(0, z)$$

DNN 前向计算

- Forward
  - 给定一个观测向量，DNN 的输出由模型指定，并逐层计算

Algorithm 1.1 DNN Forward Computation

procedure FORWARDCOMPUTATION($\mathbf{O}$)
  ▷ Each column of $\mathbf{O}$ is an observation vector
  $\mathbf{V}^0 \leftarrow \mathbf{O}$
  for $\ell \leftarrow 1; \ell < L; \ell \leftarrow \ell + 1$ **do** ▷ L is the total number of layers
    $\mathbf{Z}^\ell \leftarrow \mathbf{W}^\ell \mathbf{V}^{\ell-1} + \mathbf{B}^\ell$ ▷ Each column of $\mathbf{B}^\ell$ is $\mathbf{b}^\ell$
    $\mathbf{V}^\ell \leftarrow f\left(\mathbf{Z}^\ell\right)$ ▷ $f(.)$ can be sigmoid, tanh, ReLU, or other functions
  end for
  $\mathbf{Z}^L \leftarrow \mathbf{W}^L \mathbf{V}^{L-1} + \mathbf{B}^L$
  if regression **then** ▷ regression task
    $\mathbf{V}^L \leftarrow \mathbf{Z}^L$
  else ▷ classification task
    $\mathbf{V}^L \leftarrow \text{softmax}\left(\mathbf{Z}^L\right)$ ▷ Apply softmax column-wise
  end if
  Return $\mathbf{V}^L$
end procedure

DNN 前向计算

- 前向计算过程

$$h_1^{(1)} = f(z_1^{(1)}) = f(z_1^{(1)}) = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3)$$
$$h_2^{(1)} = f(z_2^{(1)}) = f(z_2^{(1)}) = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3)$$
$$h_3^{(1)} = f(z_2^{(1)}) = f(z_3^{(1)}) = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3)$$
$$h_4^{(1)} = f(z_2^{(1)}) = f(z_4^{(1)}) = f(W_{41}^{(1)}x_1 + W_{42}^{(1)}x_2 + W_{43}^{(1)}x_3)$$
$$h_1^{(2)} = f(z_1^{(2)}) = f(W_{11}^{(2)}h_1^{(1)} + W_{12}^{(2)}h_2^{(1)} + W_{13}^{(2)}h_3^{(1)} + W_{14}^{(2)}h_4^{(1)})$$
$$h_2^{(2)} = f(z_2^{(2)}) = f(W_{21}^{(2)}h_1^{(1)} + W_{22}^{(2)}h_2^{(1)} + W_{23}^{(2)}h_3^{(1)} + W_{24}^{(2)}h_4^{(1)})$$
$$h_3^{(2)} = f(z_3^{(2)}) = f(W_{31}^{(2)}h_1^{(1)} + W_{32}^{(2)}h_2^{(1)} + W_{33}^{(2)}h_3^{(1)} + W_{34}^{(2)}h_4^{(1)})$$
$$\hat{y} = f(z^{(3)}) = f(W_{11}^{(3)}h_1^{(2)} + W_{12}^{(3)}h_2^{(2)} + W_{13}^{(3)}h_3^{(2)})$$

参数估计

- 模型训练
  - DNN 中的参数需要从训练样本中进行估计
- 一些定义
  - 模型参数：$\{\mathbf{W}, \mathbf{b}\}$
  - 训练样本：$\mathbb{S} = \{(\mathbf{o}^m, \mathbf{y}^m) | 0 \leq m < M\}$
  - 观测向量：$\mathbf{o}^m$
  - 想要的输出向量：$\mathbf{y}^m$

训练准则

- 模型训练应该最小化预期损失
  - 损失函数：$J(\mathbf{W}, \mathbf{b}; \mathbf{o}, \mathbf{y})$
  - 观测的 PDF：$p(\mathbf{o})$

$$J_{EL} = \mathbb{E}(J(\mathbf{W}, \mathbf{b}; \mathbf{o}, \mathbf{y})) = \int_{\mathbf{o}} J(\mathbf{W}, \mathbf{b}; \mathbf{o}, \mathbf{y}) p(\mathbf{o}) d(\mathbf{o})$$

- 回归问题的均方误差 (Mean Square Error, MSE)

$$J_{MSE}(\mathbf{W}, \mathbf{b}; \mathbb{S}) = \frac{1}{M} \sum_{m=1}^{M} J_{MSE}(\mathbf{W}, \mathbf{b}; \mathbf{o}^m, \mathbf{y}^m)$$

$$J_{MSE}(\mathbf{W}, \mathbf{b}; \mathbf{o}, \mathbf{y}) = \frac{1}{2} \parallel \mathbf{v}^L - \mathbf{y} \parallel^2 = \frac{1}{2} (\mathbf{v}^L - \mathbf{y})^\top (\mathbf{v}^L - \mathbf{y})$$

训练准则

- 分类问题的交叉熵 (Cross Entropy, CE)
  - 经验概率：$y_i = P_{emp}(i|\mathbf{o})$
  - DNN 估计的概率：$v_i^L = P_{dnn}(i|\mathbf{o})$
  - i.e. 最小化下面两个分布的 Kullback–Leibler 散度

$$J_{CE}(\mathbf{W}, \mathbf{b}; \mathbb{S}) = \frac{1}{M} \sum_{m=1}^{M} J_{CE}(\mathbf{W}, \mathbf{b}; \mathbf{o}^{\mathbf{m}}, \mathbf{y}^{\mathbf{m}})$$

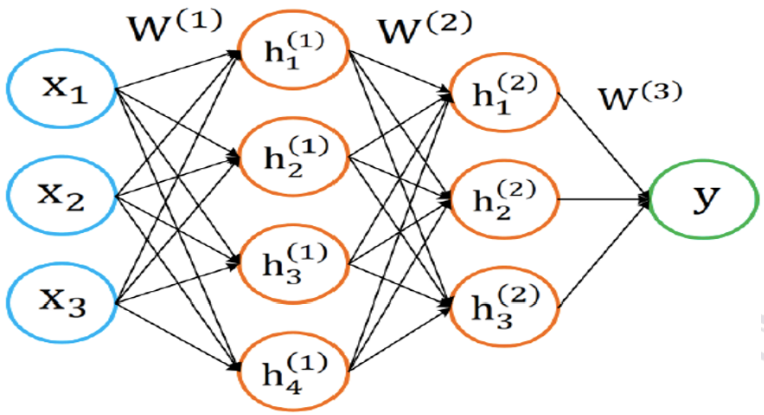
$$J_{CE}(\mathbf{W}, \mathbf{b}; \mathbf{o}, \mathbf{y}) = -\sum_{i=1}^{C} y_i \log v_i^L$$

  - 通常是采用 hard 的类标签（离散值）：

$$y_i = \mathbb{I}(c = i) \qquad \mathbb{I}(x) = \begin{cases} 1, & \text{if } x \text{ is true} \\ 0, & \text{otherwise} \end{cases}$$

训练算法

- 一个简单神经网络架构
  - 三个输入
    x_1, x_2, x_3
  - 加权和
    $z_i^{(l)} = \sum_j W_{ij}^{(l)} h_j^{(l-1)}$
  - 激活输出
    $h_i^{(l)} = f(z_i^{(l)})$
  - 参数
    $\boldsymbol{W} = \{\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{W}^{(3)}\}$
  - 网络输出
    $\hat{y} = f(z^{(3)}) = f(\sum_j W_{ij}^{(3)} h_j^{(2)})$



神经网络的误差反向传播

- 反向传播（Back-propagation, BP）是神经网络中应用最广泛的参数更新方法
- BP 算法可以分解为以下四个步骤
  - 前向 (Feed-forward) 计算
  - 反向传播到输出层
  - 反向传播到隐层
  - 参数更新

批量梯度下降
Batch Gradient Descent

- 参数更新
  - η: 学习率
  - 简单起见，以 $\boldsymbol{W}^{(3)}$ 为例

$$\begin{aligned}\boldsymbol{W}^{(3)} &:= \boldsymbol{W}^{(3)} - \eta \nabla_{\boldsymbol{W}^{(3)}} J(\boldsymbol{W}) \\ &:= \boldsymbol{W}^{(3)} - \frac{\eta}{m} \sum_{i}^{m} ((\hat{y}^i - y^i) f'(\boldsymbol{z}^{(3)}) (\boldsymbol{h}^{(2)})^\top)\end{aligned}$$

  - **批量梯度下降**

    训练数据很大时，更新会变得非常慢。

随机梯度下降

Stochastic Gradient Descent

如果我们从训练数据 $\{\boldsymbol{X}, Y\}$ 中随机挑选一个样本 $\{\boldsymbol{x}, y\}$，而不是使用所有数据

$$J(\boldsymbol{W}; X, Y) = \frac{1}{2m}\sum_{i}^{m}(\hat{y}^i - y^i)^2 \Rightarrow J(\boldsymbol{W}; X, Y) = \frac{1}{2}(\hat{y}^i - y^i)^2$$

$$\begin{aligned}\boldsymbol{W}^{(3)} &:= \boldsymbol{W}^{(3)} - \eta\nabla_{\boldsymbol{W}^{(3)}} J(\boldsymbol{W}) \\ &:= \boldsymbol{W}^{(3)} - \eta(\hat{y} - y)f'(\boldsymbol{z}^{(3)})(\boldsymbol{h}^{(2)})^\top\end{aligned}$$

参数更新很快，但准确率下降

小批次梯度下降

Mini-batch Gradient Descent

- 权衡 (Tradeoff):
  从训练数据 $\{\boldsymbol{X}, Y\}$ 随机挑选 b 个样本 $\{\boldsymbol{X}_b, Y_b\}$，而不是使用所有数据
  when $b = 1$: SGD
  when $b = m$: BGD

$$\begin{aligned}\boldsymbol{W}^{(3)} &:= \boldsymbol{W}^{(3)} - \eta \nabla_{\boldsymbol{W}^{(3)}} J(\boldsymbol{W}) \\ &:= \boldsymbol{W}^{(3)} - \frac{\eta}{b} \sum_{y \in Y_b} ((\hat{y} - y) f'(\boldsymbol{z}^{(3)})(\boldsymbol{h}^{(2)})^\top)\end{aligned}$$

实际考虑因素

- 数据预处理（Data Preprocessing）
- 模型初始化（Model Initialization）
- 权重衰减（Weight Decay）
- 随机失活（Dropout）
- 批量大小选择（Batch Size Selection）
- 样本随机化（Sample Randomization）
- 动量（Momentum）
- 学习率（Learning Rate）
- 停止准则（Stopping Criterion）
- 网络架构（Network Architecture）

数据预处理
Data Preprocessing

- 数据预处理发挥重要作用
  - 逐样本的特征归一化
  - 全局特征标准化
- 每个样本的范数（norm）
  - 应从特征中减去平均数，以减少最终特征的可变性
  - E.g. 图像处理: 由亮度引入的可变性
  - E.g. 手写: 偏移的字符位置
  - E.g. 语音识别：声学信道失真

- 倒谱均值归一化 (Cepstral mean normalization, CMN)
  - 减去每句话特征平均值
  - 首先估计逐句均值

$$\bar{\mu}_i = \frac{1}{T} \sum_{t=1}^{T} o_i^t$$

  - 然后对于语句中的每一维 i，减去所有帧 $(t = 1, \cdots, T)$ 中的平均值

$$\bar{o}_i^t = o_i^t - \bar{\mu}_i$$

- 全局特征标准化
  - 使用全局变换沿着每一维来缩放数据
  - 最终的数据向量在相似的范围内
    - 图像处理：$[0, 255] \rightarrow [0, 1]$
    - 语音处理：将 MFCC 和 FBANK 归一化为零均值和单位方差
  - 全局矩阵是从训练数据中估计的

$$\mu_i = \frac{1}{M}\sum_{m=1}^{M} o_i^m \quad \sigma_i = \sqrt{\frac{1}{M}\sum_{m=1}^{M}(o_i^m - \mu_i)^2} \quad \tilde{o}_i^m = \frac{o_i^m - \mu_i}{\sigma_i}$$

模型初始化
Model Initialization

- 模型初始化很重要
  - DNN 是高度非线性的模型
  - 训练准则是非凸的
- 很多启发式的技巧，但都基于两点考虑
  - 首先，权重需要初始化
  - 如果权重都很大，许多神经元将会饱和（接近 0 或 1），且梯度会非常小

模型初始化
Model Initialization

- 语音识别中 DNN 的初始化
  - 受限波尔兹曼机 (RBM) 预训练
    - DBN 的基础模块
  - 鉴别性预训练
    - 仅用一轮逐层式训练 DNN
  - 用精心设计的分布来初始化
    - 从高斯分布中采样：$\mathcal{N}(\mathbf{w}; 0, 0.05)$
    - 或从均匀分布 [-0.05,0.05] 中采样

权重衰减
Weight Decay

- 权重衰减是控制过拟合的一种方法
- 过拟合
  - 可能是一个问题，特别是 DNN 中的参数数量与许多其他机器学习模型相比是巨大的
  - 我们感兴趣的是最小化预期损失，但实际上我们是最小化定义在训练集上的经验损失

权重衰减
Weight Decay

- 通过正则化训练准则来避免过拟合

$$\hat{L}(\mathbf{W}, \mathbf{x}) = L(\mathbf{W}, \mathbf{x}) + \lambda R(\mathbf{W})$$

- 最广泛使用的正则化项
  - L1 范数

$$R_1(\mathbf{W}) = \|\mathrm{vec}(\mathbf{W})\|_1 = \sum_{\ell=1}^{L} \|\mathrm{vec}(\mathbf{W}^\ell)\|_1 = \sum_{\ell=1}^{L} \sum_{i=1}^{N_\ell} \sum_{j=1}^{N_{\ell-1}} |W_{ij}^\ell|$$

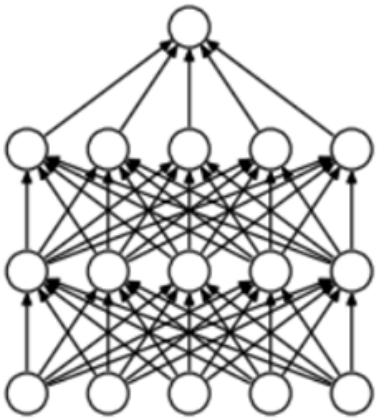
  - L2 范数

$$R_2(\mathbf{W}) = \|\mathrm{vec}(\mathbf{W})\|_2^2 = \sum_{\ell=1}^{L} \|\mathrm{vec}(\mathbf{W}^\ell)\|_2^2 = \sum_{\ell=1}^{L} \sum_{i=1}^{N_\ell} \sum_{j=1}^{N_{\ell-1}} (W_{ij}^\ell)^2$$

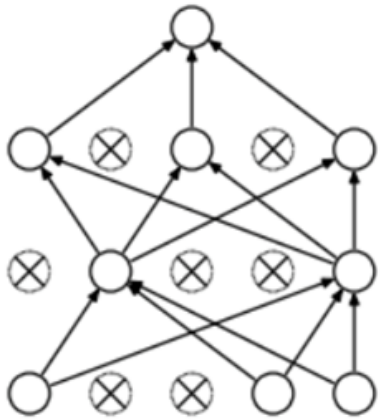
随机失活

Dropout

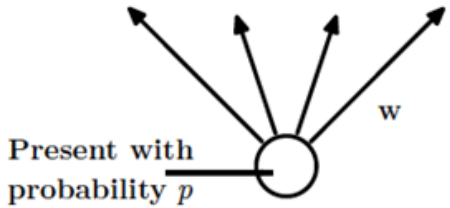
- 更新参数时忽略某些节点
- 释放一些连接节点的依赖关系
- 增加泛化性来避免过拟合
- 测试时将输出乘以 p



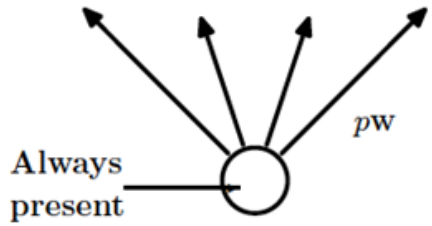
(a) Standard Neural Net



(b) After applying dropout.



(a) At training time



(b) At test time

批量大小选择
Batch Size Selection

- 批量大小的选择将影响收敛速度和最终的模型
  - 批量模式训练
  - 随机梯度下降 (SGD)
  - 小批量 (minibatch) SGD

- 小批量 SGD
  - 基于小批量随机抽取的训练样本来估计梯度
    - minibatch 训练使得在 minibatch 内很容易并行化，因此可以比 SGD 更快地收敛
  - 我们可以在初期选择较小的 minibatch 大小，后期选择较大的
  - E.g. 在初期使用 64 到 256 个样本，后期使用 1024 到 8096 个样本

样本随机化
Sample Randomization

- 样本随机化与批量训练无关，因为所有样本都被用来估计梯度

- 这对 SGD 和小批量训练非常重要
  - 这是因为要想获得无偏的梯度估计，样本必须是 IID
  - 连续的样本不是从训练集中随机抽取的（例如所有的样本都属于同一个说话人），模型参数很可能会沿着相似的方向移动太长时间

样本随机化

Sample Randomization

- 实现
  - 小训练集
    - 可以加载到内存中，而样本的随机化可以很容易地通过索引数组的变换来实现
  - 大训练集
    - 每次将一大块 (chunk) 数据（通常是 24–48 小时的语音，或 8.6M 到 17.2M 的样本）加载到内存中
    - 如果训练数据来自不同的来源（例如不同的语言），在将其输入 DNN 训练工具之前，随机化语句列表文件也将有帮助

不使用动量
Without momentum

- 负梯度 $-\nabla_{\boldsymbol{W}^{(3)}} J(\boldsymbol{W})$ 被认为是一个在参数空间中移动粒子的力
- 假设在加速过程中粒子为单位质量，且平均的力衰减系数为 η，则速度变为

$$\boldsymbol{v}_t = -\eta \nabla_{\boldsymbol{W}^{(3)}} J(\boldsymbol{W})$$

- 一个时间步长后，粒子移动到一个新的参数点, i.e.

$$\boldsymbol{W}^{(3)} := \boldsymbol{W}^{(3)} + \boldsymbol{v}_t$$

动量
Momentum

- 众所周知，如果模型更新是基于以前所有的梯度（更多的全局视野），而不是只基于当前的梯度（局部视野），那么收敛速度可以提高
- 在 DNN 训练中，通常用一种名为动量的简单技术来实现

动量
Momentum

- 物质具有惯性或动量!
- 考虑到前一次更新的速度，并假设平均衰减因子为 γ，速度更新为

$$\boldsymbol{v}_t = \gamma \boldsymbol{v}_{t-1} - \eta \nabla_{\boldsymbol{W}^{(3)}} J(\boldsymbol{W})$$

- 一个时间步长后，粒子移动到一个新的参数点

$$\boldsymbol{W}^{(3)} := \boldsymbol{W}^{(3)} + \boldsymbol{v}_t$$

动量算法

Momentum Algorithm

- 定义在第 $\mathtt{t}^{\mathtt{th}}$ 次迭代的动量: $\boldsymbol{v}_t$
- 定义动量衰减值 $\gamma(< 1$, 通常选 0.9)
- 回顾更新规则 $\boldsymbol{W}^{(3)} := \boldsymbol{W}^{(3)} - \eta \nabla_{\boldsymbol{W}^{(3)}} J(\boldsymbol{W})$
- **使用动量**

$$\boldsymbol{v}_t = \gamma \boldsymbol{v}_{t-1} - \eta \nabla_{\boldsymbol{W}^{(3)}} J(\boldsymbol{W})$$

$$\boldsymbol{W}^{(3)} := \boldsymbol{W}^{(3)} + \boldsymbol{v}_t$$

- SGD 有很多嘈杂的梯度 (noisy gradient)，动量有效解决了这个问题

动量
Momentum

- 动量平滑了参数更新，降低了梯度估计的方差
- 实际中它可以减少常规反向传播中常见的振荡问题
- 加速训练

学习率
Learning Rate

- 减半学习率
  - 设置初始学习率
  - 训练并关注 CV 误差
  - 当 CV 误差增加时
    - 减半学习率
    - 继续训练

- 指数级衰减
  - 设置初始学习率
  - 设置衰减率（通常为 0.95 或 0.99）
  - 设置衰减步长（通常为一轮）
  - 训练时
    - 在每一个衰减步长时衰减学习率

停止准则
Stopping Criterion

- 停止策略
  - 每当在开发集上测得的训练准则出现波动时，batch size 增加一倍，学习率降低四倍；当学习率小于阈值或达到预设的数据传递次数时，停止训练
  - 在训练准则波动后，将学习率降低到一个很小的数字；当训练集或开发集再次发生波动时，停止训练

网络架构
Network Architecture

- 宽而浅的模型更容易过拟合 (overfit)，深而窄的模型更容易欠拟合 (underfit)

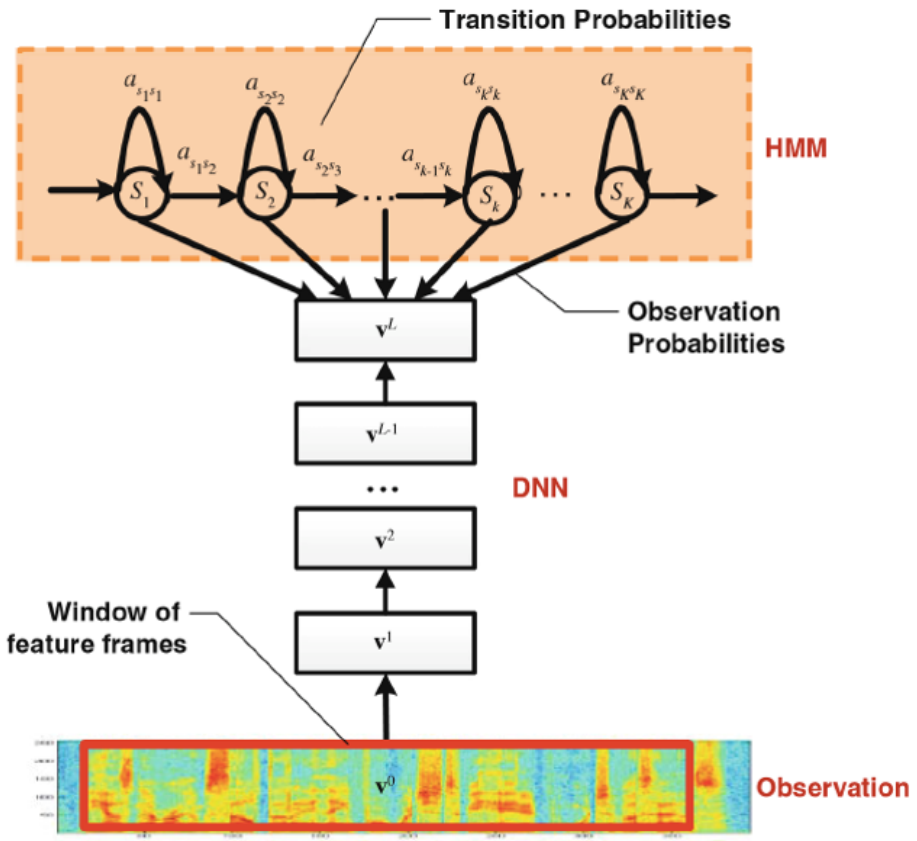
- 在语音识别任务中
  - 我们发现 5-7 层的 DNN（每层有 1000-3000 个神经元）效果非常好
  - 与窄而浅的模型相比，宽而深的模型往往更容易找到好的配置
  - 这是因为在宽而深的模型上，有很多好的局部最优点类似的表现

语音识别的 DNN

- DNN 不能直接用于建模语音信号
  - 语音信号是变长的
  - DNN 需要固定尺寸的输入
- DNN-HMM 混合系统
  - HMM：建模语音信号的动态变化
  - DNN：建模观测概率

CD-DNN-HMM 混合系统

- 三个关键组成部分
  - 输入
    - 上下文窗长大小的特征向量
  - 输出
    - senones (tied tri-phone state) 的后验概率
  - 深度神经网络
    - 很多层的非线性特征变换



与 GMM-HMM 的对比

- GMM-HMM
  - 每个状态都有自己的 GMM

$$p(\boldsymbol{o}_t|q_t,\theta) \sim \sum_{m=1}^{M} c_{q_t,m}\mathcal{N}(\boldsymbol{o}_t|\boldsymbol{\mu}_{q_t},\ _{q_t})$$

- DNN-HMM
  - 所有状态共享一个模型

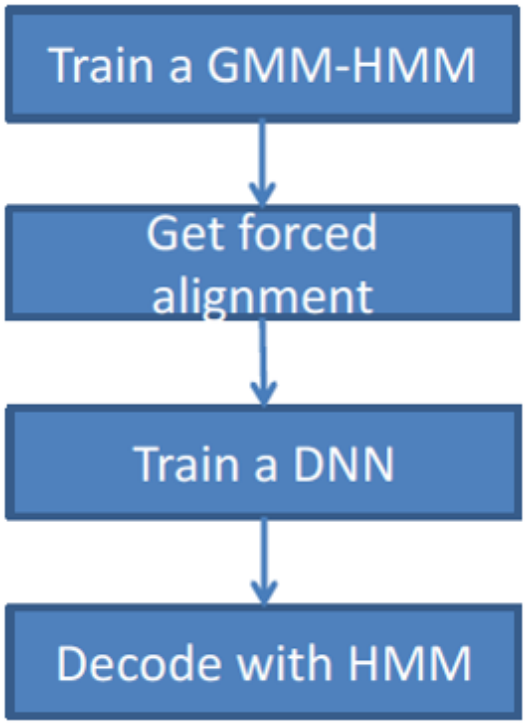
$$p(\boldsymbol{o}_t|q_t,\theta) \propto \frac{p(q_t=s|\boldsymbol{o}_t)}{p(s)}, \text{ where } p(q_t=s|\boldsymbol{o}_t) \text{ is DNN output}$$

与 GMM-HMM 的对比

	GMM-HMM	CD-DNN-HMM
State output distribution	$\sum_{m=1}^{M_j} c_{jm}\mathcal{N}(\boldsymbol{o}_t;\boldsymbol{\mu}^{(jm)},\boldsymbol{\Sigma}^{(jm)})$	$\frac{p(\boldsymbol{o})p(q=j\|\boldsymbol{o})}{p(q=j)} \propto \frac{p(q=j\|\boldsymbol{o})}{p(q=j)}$
Feature	No span, one level, spectrum-related	Long span, multiple-level features
Model	Shallow, one layer parameter	Deep, multiple layer parameters
Hierarchy	No hierarchy	Hierarchical
Criterion	$\sum_t\sum_{q_t} P(q_t\|\boldsymbol{O},\widehat{\mathcal{M}})\log p(\boldsymbol{o}_t\|q_t,\mathcal{M})$	$\sum_t\sum_{q_t} P_{ref}(q_t\|\boldsymbol{o}_t)\log P(q_t\|\boldsymbol{o}_t,\mathcal{M})$

训练过程

- 对齐 (alignment) 是通过 GMM-HMM 用 Viterbi 算法生成的
- 训练一个好的 GMM-HMM 系统作为种子模型很重要
- RBM 预训练有帮助，尤其是只有很小的训练集时



用 CD-DNN-HMM 解码

- 由于 HMM 需要条件似然 $p(\boldsymbol{o}_t|q_t)$
- 通过下式将后验转换成条件似然
  - $p(\boldsymbol{o}_t|q_t = s) = \frac{p(q_t=s|\boldsymbol{o}_t)p(\boldsymbol{o}_t)}{p(s)}$
  - $p(s)$ 是从训练集中估算出的每个状态的先验概率

使用 DNN 的 LVCSR 结果总结

- Voice Search

AM (Training Set)	Setup	Test SER
GMM-HMM (24 hrs)	ML	39.6%
GMM-HMM (24 hrs)	MPE	36.2%
DNN-HMM (24 hrs)	5 layers x 2048	30.1% (-17%)
DNN-HMM (48 hrs)	5 layers x 2048	28.5%
GMM-HMM (2100 hrs)	ML	37.1%

- Switch Board

AM (Training Set)	Setup	Hub5'00-SWB WER	RT03S-FSH WER
GMM-HMM (309 hrs)	ML Single-Pass SI	26.5%	30.2%
GMM-HMM (309 hrs)	BMMI Single-Pass SI	23.6%	27.4%
DNN-HMM (309 hrs)	7 x 2048 Single-Pass SI	16.1% (-32%)	18.5% (-33%)
GMM-HMM (1700 hrs)	DT Multi-Pass, Adaptive	17.1%	

更深的模型更强大

Deeper Models More Powerful

L×N	DBN-Pretrain	BP	LBP	Discri-Pretrain	1×N	DBN-Pretrain
1×2k	24.2	24.3	24.3	24.1	1×2k	24.2
2×2k	20.4	22.2	20.7	20.4	-	-
3×2k	18.4	20.0	18.9	18.6	-	-
4 ×2k	17.8	18.7	17.8	17.8	-	-
5×2k	17.2	18.2	17.4	17.1	1×3772	22.5
7 ×2k	17.1	17.4	17.4	16.8	1×4634	22.6
9×2k	17.0	16.9	16.9	-	-	-
9× 1k	17.9	-	-	-	-	-
5×3k	17.0	-	-	-	-	-
					1× 16k	22.1

缩减尺寸与稀疏性

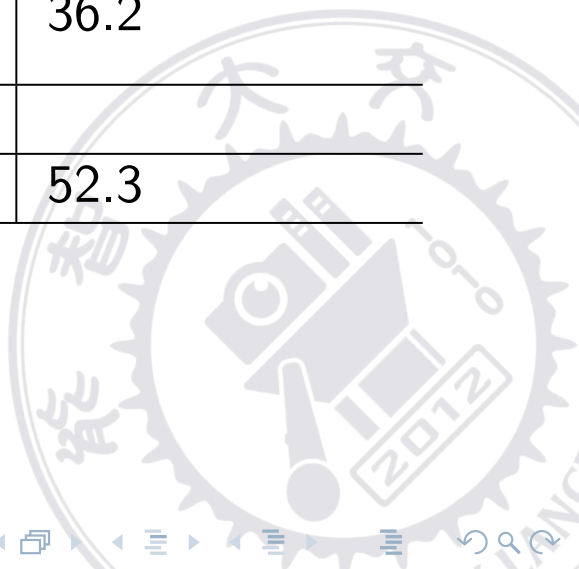
Size Reduction with Sparseness

- SWB 309hrs $(429 \times 2048^7 \times 9304)$ with ML alignment

Acoustic model	# nonzero params	Model size	Calc time	Hub5'00 SWB	RT03S FSH
CD-GMM-HMM, BMMI	29.4M	-	-	23.6	27.4
CD-DNN-HMM	45.1M	100%	100%	16.4	18.6
Sparse: 69% nz	31.1M	104%	83%	16.2	18.5
Sparse: 52% nz	23.6M	78%	62%	16.1	18.5
Sparse: 34% nz	15.2M	51%	41%	16.1	18.4
Sparse: 24% nz	11.0M	36%	29%	16.2	18.5
Sparse: 19% nz	8.6M	29%	23%	16.4	18.7
Sparse: 15% nz	6.6M	22%	18%	16.5	18.7

语音识别的变革

task	hours of training data	DNN-HMM	GMM-HMM with same data
Switchboard(test set 1)	309	18.5	27.4
Switchboard(test set 2)	309	16.1	23.6
English Broadcast News	50	17.5	18.8
Bing Voice Search (Sentence error rates)	24	30.4	36.2
Google Voice Input	5870	12.3	
Youtube	1400	47.6	52.3



ASR 仍然很难

- 多变性，多变性，多变性

Speaker	Environment	Device
• Accents • Dialect • Style • Emotion • Coarticulation • Reduction • Pronunciation • Hesitation • …	• Noise • Side talk • Reverberation • …	• Head phone • Land phone • Speaker phone • Cell phone • …

- 这些因素之间的相互作用是复杂且非线性的

更多高级的神经网络

- 卷积神经网络（Convolutional neural network, CNN）
- 循环神经网络（Recurrent neural network, RNN）
- 长短期记忆（Long-short term-memory, LSTM）
- 组合: C+L+D NN