五、神经网络

主讲教师：周志华

什么是神经网络？

“神经网络是由具有适应性的简单单元组成的广泛并行互连的网络，它的组织能够模拟生物神经系统对真实世界物体所作出的交互反应”

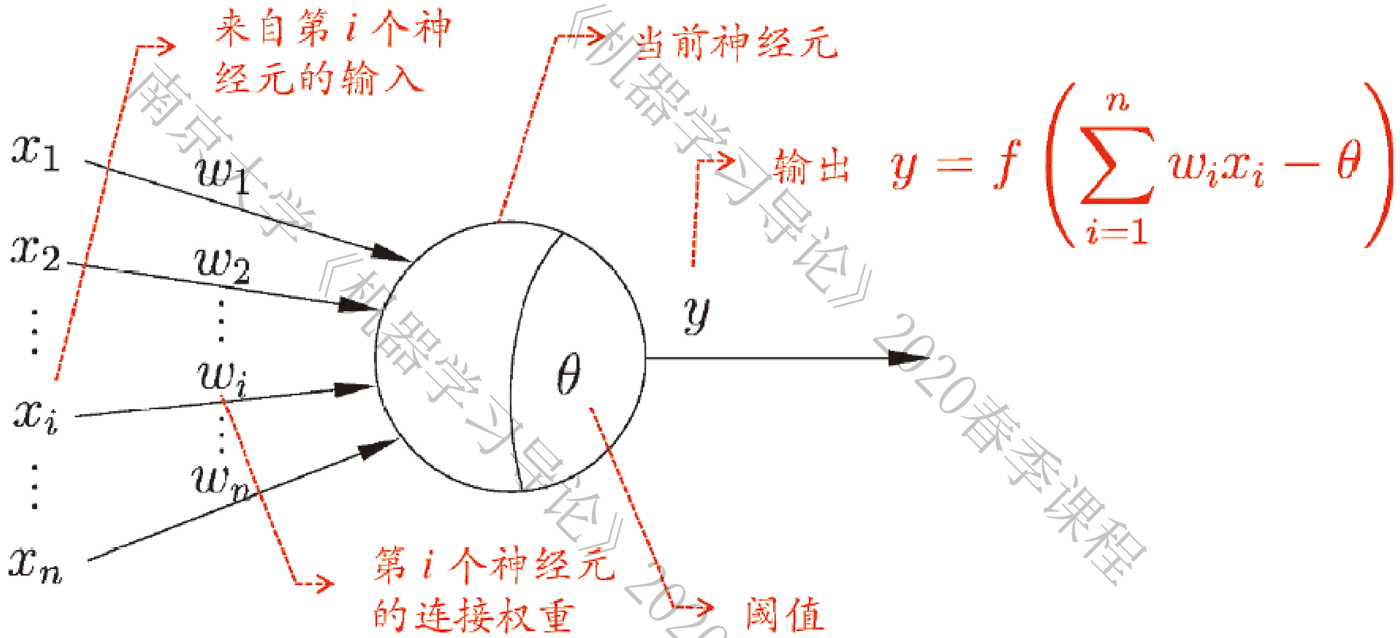
[T. Kohonen, 1988, *Neural Networks* 创刊号]

神经网络是一个很大的学科领域，本课程仅讨论神经网络与机器学习的交集，即“神经网络学习”

亦称“连接主义(connectionism)” 学习

“简单单元”：神经元模型

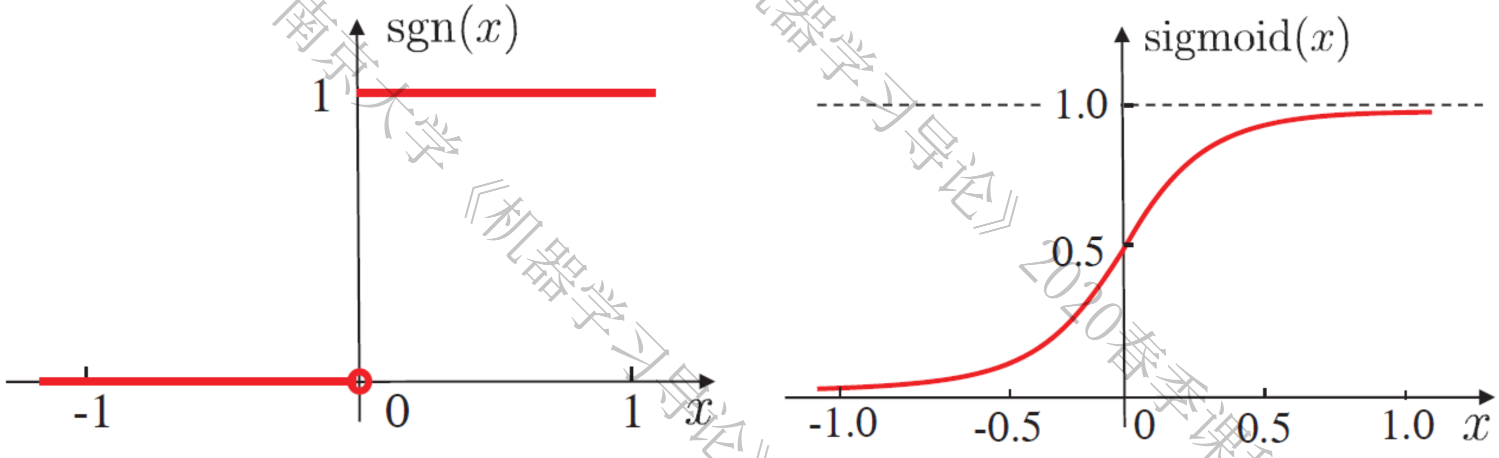
M-P 神经元模型 [McCulloch and Pitts，1943]



神经网络学得的知识蕴含在连接权与阈值中

神经元的“激活函数”

- 理想激活函数是阶跃函数，0表示抑制神经元而1表示激活神经元
- 阶跃函数具有不连续、不光滑等不好的性质，常用的是 Sigmoid 函数



$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{if } x < 0. \end{cases}$$

$$\mathrm{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

(a) 阶跃函数　　(b) Sigmoid 函数

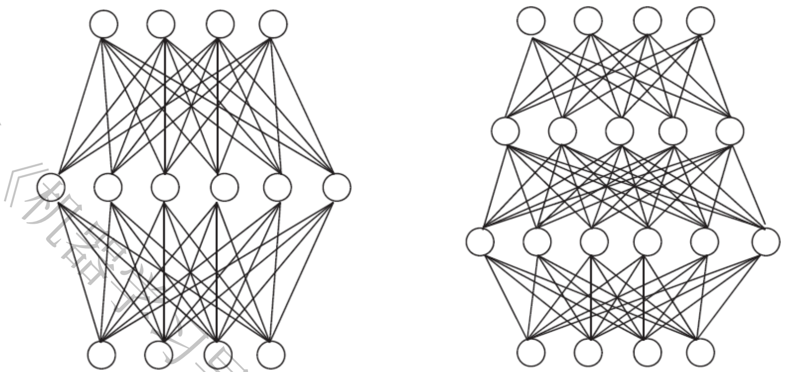
图 5.2　典型的神经元激活函数

多层前馈网络结构

多层网络：包含隐层的网络

前馈网络：神经元之间不存在同层连接也不存在跨层连接

隐层和输出层神经元亦称“功能单元” (functional unit)



(a) 单隐层前馈网络　　(b) 双隐层前馈网络

多层前馈网络有强大的表示能力（”万有逼近性”）

> 仅需一个包含足够多神经元的隐层，多层前馈神经网络就能以任意精度逼近任意复杂度的连续函数　[Hornik et al., 1989]

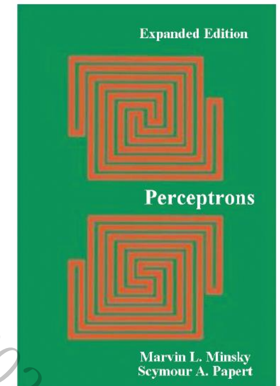
但是，如何设置隐层神经元数是未决问题. 实际常用“试错法”

神经网络发展回顾

1940年代-萌芽期：M-P模型 (1943), Hebb 学习规则 (1945)

1956左右-1969左右~繁荣期：感知机 (1958), Adaline (1960), …

1969年：Minsky & Papert “Perceptrons”



马文·闵斯基
(1927-2016)
1969年图灵奖

冰河期

1984左右 -1997左右~繁荣期：Hopfield (1983), BP (1986), …

1997年左右：SVM文本分类成功 及 统计学习 兴起

沉寂期

2012-至今~繁荣期：深度学习

交替模式:
热十三？年
冷十五？年

2019年3月27日，ACM宣布：

Geoffrey Hinton, Yann LeCun, Yoshua Bengio

因对深度学习的卓越贡献获得图灵奖

科学的发展总是
“螺旋式上升”

三十年河东
三十年河西

坚持才能有结果

BP (BackPropagation:误差逆传播算法)

迄今最成功、最常用的神经网络算法，可用于多种任务（不仅限于分类）

P. Werbos在博士学位论文中正式完整描述:

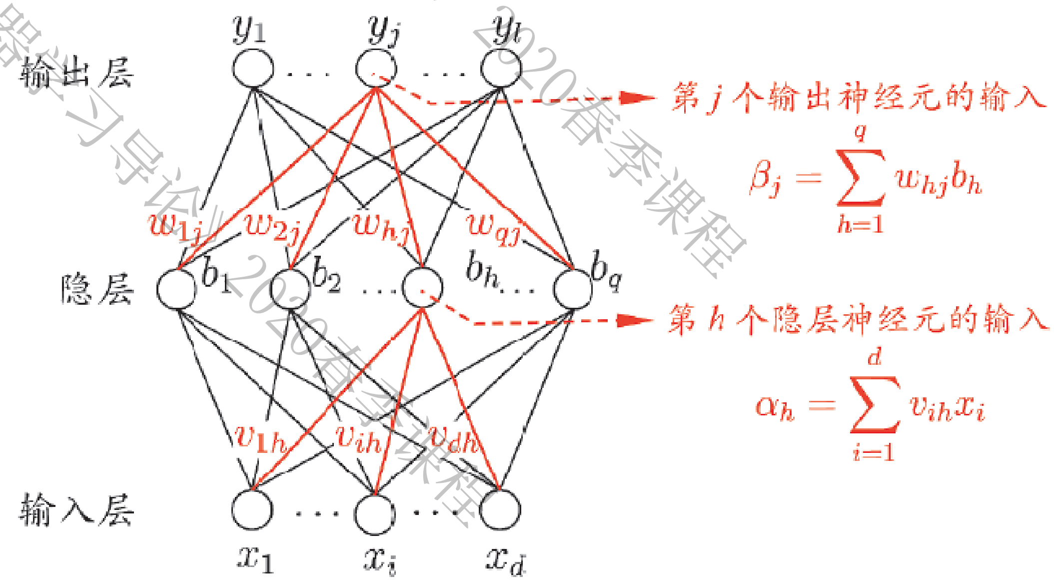
P. Werbos. Beyond regression: New tools for prediction and analysis in the behavioral science. Ph.D dissertation, Harvard University, 1974

给定训练集 $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_m, \boldsymbol{y}_m)\}, \boldsymbol{x}_i \in \mathbb{R}^d, \boldsymbol{y}_i \in \mathbb{R}^l$

输入：d 维特征向量
输出：l 个输出值

隐层：假定使用 q 个隐层神经元

假定功能单元均使用 Sigmoid 函数



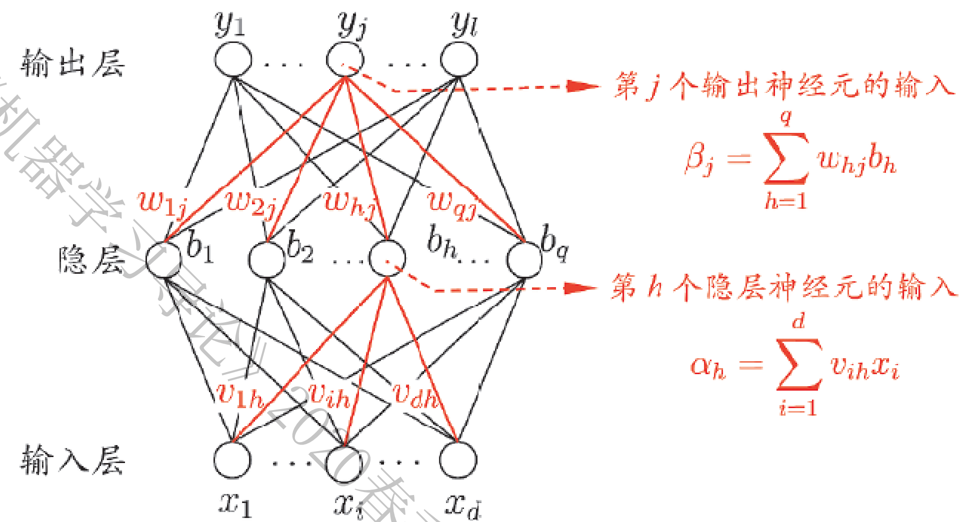
BP 算法推导

对于训练例 $(\boldsymbol{x}_k, \boldsymbol{y}_k)$, 假定网络的实际输出为 $\hat{\boldsymbol{y}}_k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_l^k)$

$$\hat{y}_j^k = f(\beta_j - \theta_j)$$

则网络在 $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ 上的均方误差为：

$$E_k = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_j^k - y_j^k)^2$$



需通过学习确定的参数数目：$(d+l+1)q+l$

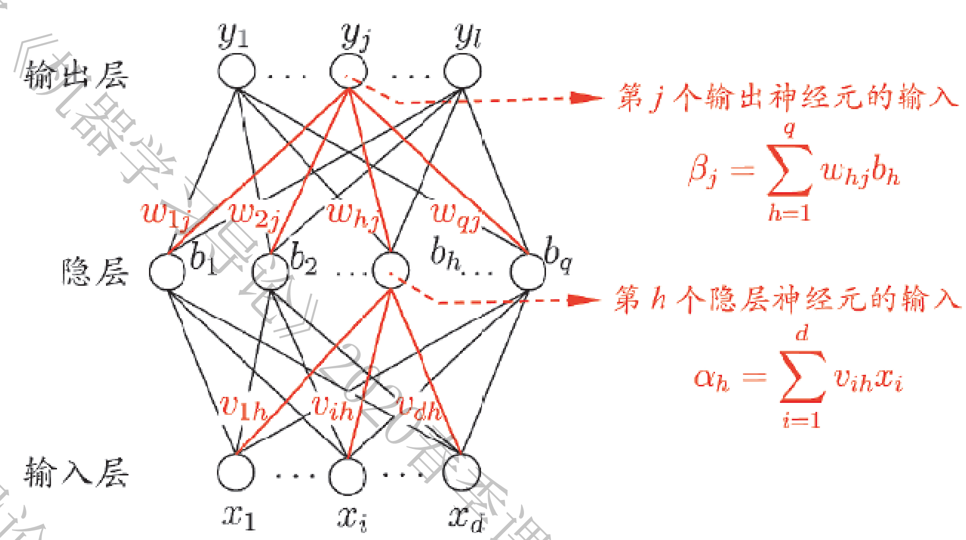
BP 是一个迭代学习算法, 在迭代的每一轮中采用广义感知机学习规则

$$v \leftarrow v + \boxed{\triangle v.}$$

BP 算法推导 (续)

BP 算法基于**梯度下降**策略，以目标的负梯度方向对参数进行调整

$\boxed{\text{以 } w_{hj} \text{ 为例}}$



对误差 E_k，给定学习率 η，有：

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$$

注意到 w_{hj} 先影响到 β_j，

再影响到 $\hat{y}_j^k$，然后才影响到 E_k，有：

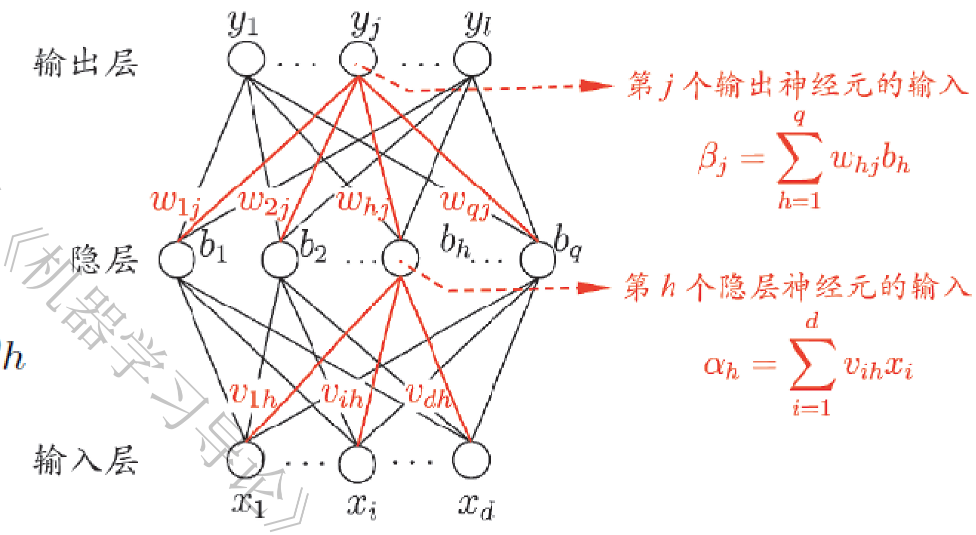
$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$

“链式法则”

BP 算法推导 (续)

$$\frac{\partial E_k}{\partial w_{hj}} = \boxed{\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j}} \cdot \boxed{\frac{\partial \beta_j}{\partial w_{hj}}}$$

$$= b_h$$



$$g_j = -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j}$$

$$= -(\hat{y}_j^k - y_j^k) f'(\beta_j - \theta_j)$$

$$= \underline{\hat{y}_j^k (1 - \hat{y}_j^k)} (y_j^k - \hat{y}_j^k) \ .$$

对 $\mathrm{sigmoid}(x) = \frac{1}{1+e^{-x}}$，有

$$f'(x) = f(x)(1 - f(x))$$

再注意到 $\hat{y}_j^k = f(\beta_j - \theta_j)$

于是，

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}} = \eta g_j b_h$$

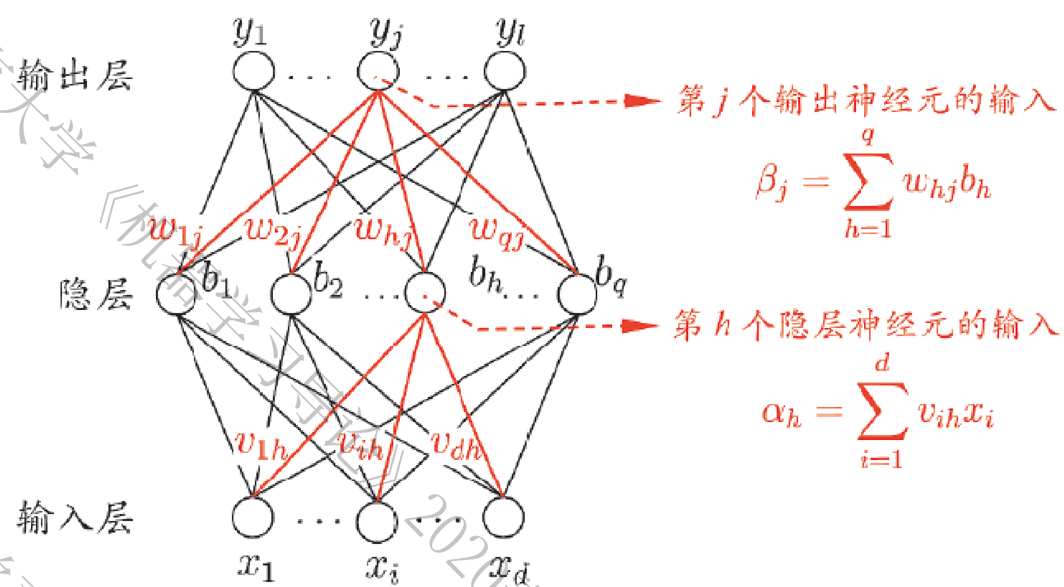
BP 算法推导 (续)

类似地，有：

$$\Delta\theta_j = -\eta g_j$$

$$\Delta v_{ih} = \eta e_h x_i$$

$$\Delta\gamma_h = -\eta e_h$$



其中：

$$\begin{aligned} e_h &= -\frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \\ &= -\sum_{j=1}^{l} \frac{\partial E_k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} f'(\alpha_h - \gamma_h) = \sum_{j=1}^{l} w_{hj} g_j f'(\alpha_h - \gamma_h) \\ &= b_h(1-b_h) \sum_{j=1}^{l} w_{hj} g_j \end{aligned}$$

学习率 $\eta \in (0,1)$ 不能太大、不能太小

BP 算法

输入: 训练集 $D = \{(\boldsymbol{x}_k, \boldsymbol{y}_k)\}_{k=1}^m$;
学习率 η.

过程:

1: 在(0, 1)范围内随机初始化网络中所有连接权和阈值
2: **repeat**
3: **for all** $(\boldsymbol{x}_k, \boldsymbol{y}_k) \in D$ **do**
4: 根据当前参数和式(5.3) 计算当前样本的输出 $\hat{\boldsymbol{y}}_k$;
5: 根据式(5.10) 计算输出层神经元的梯度项 g_j;
6: 根据式(5.15) 计算隐层神经元的梯度项 e_h;
7: 根据式(5.11)-(5.14) 更新连接权 w_{hj}, v_{ih} 与阈值 θ_j, γ_h
8: **end for**
9: **until** 达到停止条件

输出: 连接权与阈值确定的多层前馈神经网络

图 5.8 误差逆传播算法

标准 BP 算法 vs. 累积 BP 算法

标准 BP 算法

- 每次针对单个训练样例更新权值与阈值
- 参数更新频繁, 不同样例可能抵消, 需要多次迭代

累积 BP 算法

- 其优化目标是最小化整个训练集上的累计误差
- 读取整个训练集一遍才对参数进行更新, 参数更新频率较低

在很多任务中, 累计误差下降到一定程度后, 进一步下降会非常缓慢, 这时标准BP算法往往会获得较好的解, 尤其当训练集非常大时效果更明显.

南京大学《机器学习导论》2020春季课程

缓解过拟合

主要策略：

- 早停(early stopping)
  - 若训练误差连续 a 轮的变化小于 b，则停止训练
  - 使用验证集：若训练误差降低、验证误差升高，则停止训练

- 正则化 (regularization)
  - 在误差目标函数中增加一项描述网络复杂度

例如 $$E = \lambda \frac{1}{m} \sum_{k=1}^{m} E_k + (1-\lambda) \underline{\sum_i w_i^2}$$

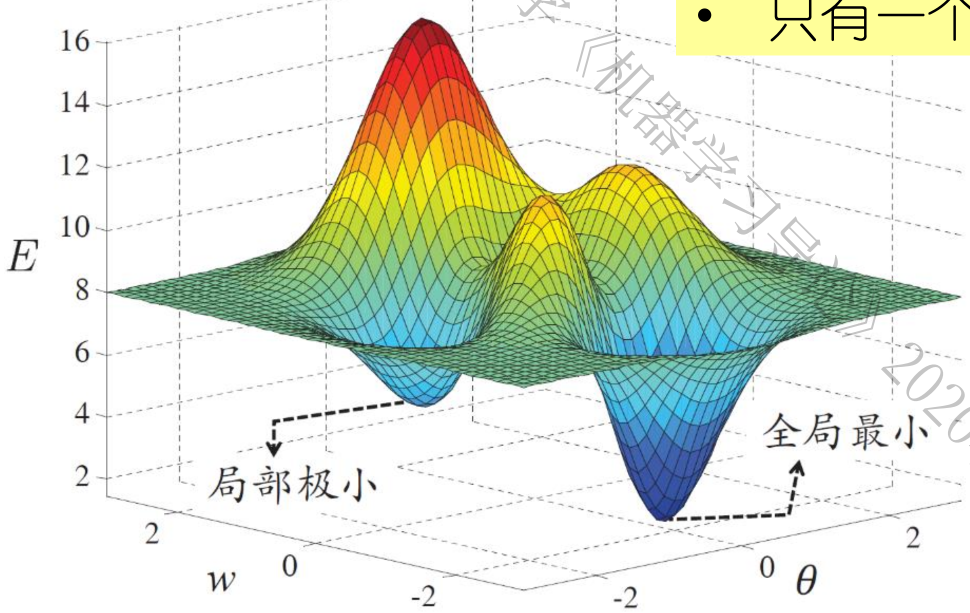
偏好比较小的连接权和阈值，
使网络输出更“光滑”

全局最小 vs. 局部极小

神经网络的训练过程可看作一个参数寻优过程：

在参数空间中，寻找一组最优参数使得误差最小

- 存在多个“局部极小”
- 只有一个“全局最小”



“跳出”局部极小的常见策略：

✓ 不同的初始参数
✓ 模拟退火
✓ 随机扰动
✓ 演化算法
✓ ……

其他常见神经网络模型

- RBF：分类任务中除BP之外最常用
- ART："竞争学习"的代表
- SOM：最常用的聚类方法之一
- 级联相关网络："构造性"神经网络的代表
- Elman网络：递归神经网络的代表
- Boltzmann机："基于能量的模型"的代表
- ……

RBF 神经网络

RBF: Radial Basis Function (径向基函数)

- 单隐层前馈神经网络
- 使用径向基函数作为隐层神经元激活函数

  例如高斯径向基函数 $\rho(\boldsymbol{x}, \boldsymbol{c}_i) = e^{-\beta_i \|\boldsymbol{x} - \boldsymbol{c}_i\|^2}$

- 输出层是隐层神经元输出的线性组合

$$\varphi(\boldsymbol{x}) = \sum_{i=1}^{q} w_i \rho(\boldsymbol{x}, \boldsymbol{c}_i)$$

训练：

Step1：确定神经元中心，常用的方式包括随机采样、聚类等

Step2：利用BP算法等确定参数

SOM 神经网络

SOM: Self-Organizing feature Map (自组织特征映射)

- 竞争型的无监督神经网络
- 将高维数据映射到低维空间（通常为2维），高维空间中相似的样本点映射到网络输出层中邻近神经元
- 每个神经元拥有一个权向量
- 目标：为每个输出层神经元找到合适的权向量以保持拓扑结构

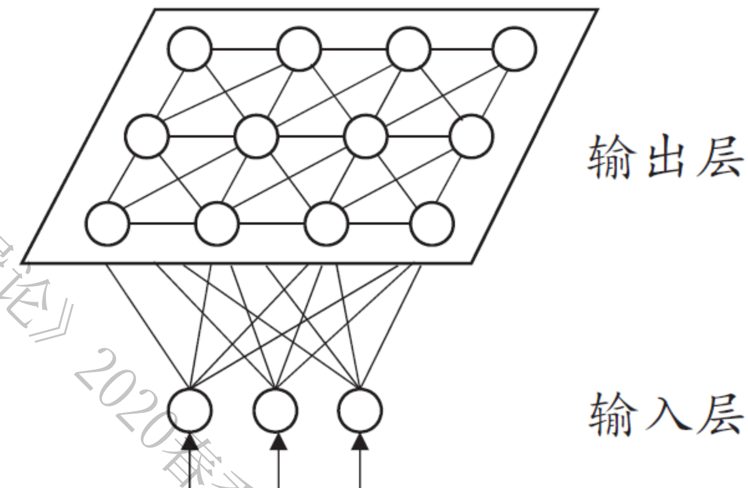


图 5.11 SOM 网络结构

训练：

- 网络接收输入样本后，将会确定输出层的“获胜”神经元（“胜者通吃”）
- 获胜神经元的权向量将向当前输入样本移动

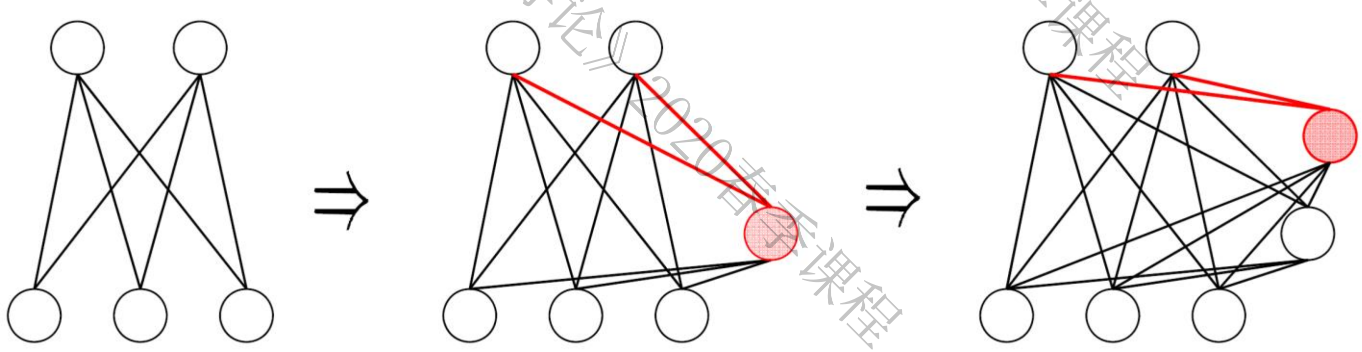
级联相关网络

CC: Cascade-Correlation (级联相关)

构造性神经网络：将网络的结构也当做学习的目标之一，希望在训练过程中找到适合数据的网络结构

训练：

- 开始时只有输入层和输出层
- 级联 - 新的隐层结点逐渐加入，从而创建起层级结构
- 相关 - 最大化新结点的输出与网络误差之间的相关性



(a) 初始状态　(b) 增加一个隐层结点　(c) 增加第二个隐层结点

Elman 网络

递归神经网络： Recurrent NN, 亦称 Recursive NN

- 网络中可以有环形结构, 可让使一些神经元的输出反馈回来作为输入
- t 时刻网络的输出状态：由 t 时刻的输入状态和 t-1 时刻的网络状态共同决定

Elman 网络是最常用的递归神经网络之一

- 结构与前馈神经网络很相似, 但隐层神经元的输出被反馈回来
- 使用推广的BP算法训练

目前在自然语言处理等领域常用的 LSTM 网络, 是一种复杂得多的递归神经网络

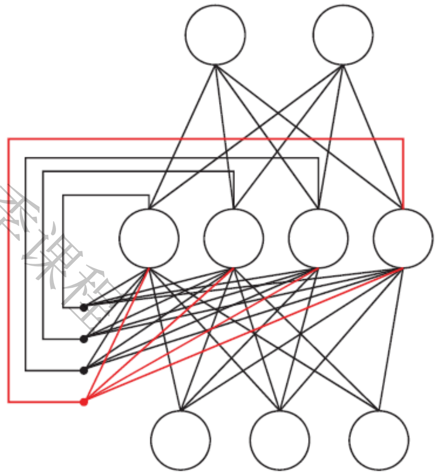


图 5.13　Elman 网络结构

南京大学《机器学习导论》2020春季课程

深度学习的兴起

- 2006年, Hinton 组发表深度学习的 Science 文章
- 2012年, Hinton 组参加ImageNet 竞赛, 使用 CNN 模型以超过第二名10个百分点的成绩夺得当年竞赛的冠军
- 在计算机视觉、语音识别、机器翻译等领域取得巨大成功

最著名的深度学习模型：卷积神经网络（**CNN**: Convolutional NN）
[LeCun and Bengio，1995; LeCun et al.，1998]

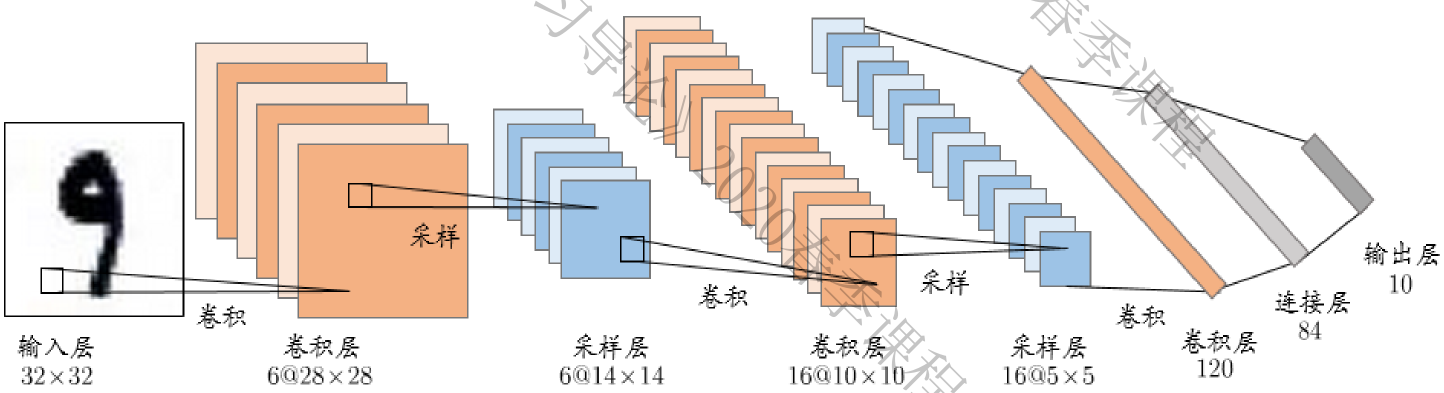


图 5.15 卷积神经网络用于手写数字识别 [LeCun et al., 1998]

深度学习是“模拟人脑”吗？

《IEEE 深度对话 Facebook 人工智能负责人 Yann LeCun》

Yann LeCun

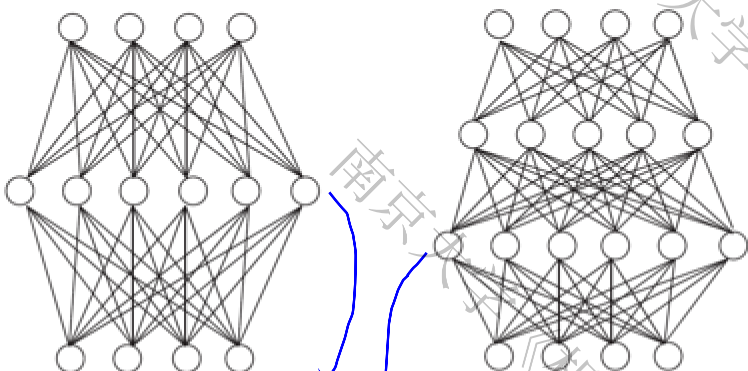
CNN的主要发明人
深度学习“三架马车”之一
2019年图灵奖得主

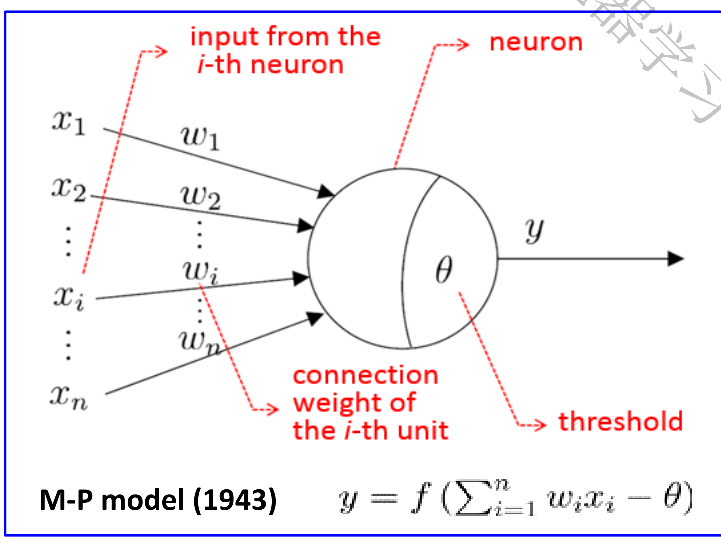
IEEE Spectrum：这些天我们看到了许多关于深度学习的新闻……

Yann LeCun：我最不喜欢的描述是「它像大脑一样工作」，我不喜欢人们这样说的原因是，虽然深度学习从生命的生物机理中获得灵感，但它与大脑的实际工作原理差别非常非常巨大。将它与大脑进行类比给它赋予了一些神奇的光环，这种描述是危险的。

深度神经网络

以往神经网络采用单或双隐层结构



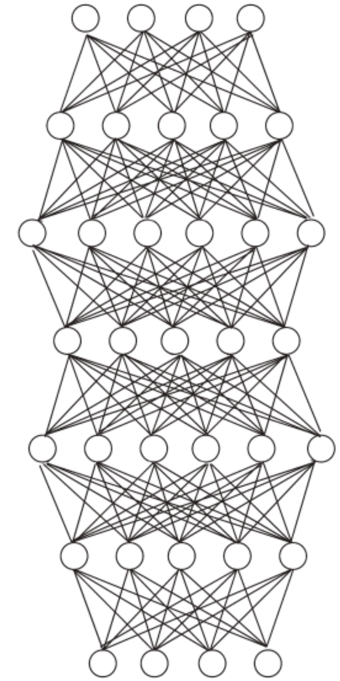


M-P model (1943) $y = f\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$

例如，ImageNet 胜者:
2012: 8 层
2015: 152 层
2016: 1207 层

deep

深度神经网络：
很多层



神经网络实质上是多层函数嵌套形成的数学模型

可以说受到了一点生物神经机制的“启发”，但远没有“受指导”

至今最常用的算法：BP [Rumelhart et al., 1986]，是完全从数学上推导出来的

重要诀窍 (tricks)

- **预训练+微调**
  - 预训练：监督逐层训练，每次训练一层隐结点
  - 微调：预训练全部完成后，对全网络进行微调训练

可视为将大量参数分组，对每组先找到较好的局部配置，再全局寻优

- **权共享 (weight-sharing)**
  - 一组神经元使用相同的连接权值

减少需优化的参数

- **Dropout**
  - 在每轮训练时随机选择一些参数令其不被更新(下一轮可能被更新)

降低 Rademacher 复杂度

- **ReLU (Rectified Linear Units)**
  - 将 Sigmoid 激活函数修改为修正线性函数 $f(x) = \max(0, x)$

求导容易；缓解梯度消失现象

- **交叉熵 (Cross-entropy)**
  - BP算法中以交叉熵 $-\frac{1}{m}\sum_{i=1}^{m} y_i \log \hat{y}_i$ 代替均方误差 $\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$

更能体现分类任务的特性

尚有许多trick缺乏关于奏效原因的合理猜测

深度学习并非“突然出现”的
“颠覆性技术”，

而是经过了长期发展、
很多研究者做出贡献，

“冷板凳”坐“热”的结果

南京大学《机器学习导论》2020春季课程

例如：CNN (卷积神经网络)

信号处理中的卷积 [最晚1903年已在文献中出现]

D. Hubel & T. Wiesel 关于猫视皮层的研究 [1962]

福岛邦彦(Fukushima)在神经网络中引入卷积 [1982]

Y. LeCun 引入BP算法训练卷积网络, CNN成型 [1989]

Y. LeCun and Y. Bengio, 完整描述CNN [1995]

Y. LeCun et al., CNN 用于支票手写字符识别[1998]

20年

G. Hinton通过无监督逐层训练, 构建深层模型 [2006]

H. Lee et al. 引入无监督逐层训练CNN [2009]

30年

G. Hinton研究组将8层CNN用于ImageNet竞赛获胜 [2012]

引发深度学习热潮，被广泛应用

新型深度学习的探索

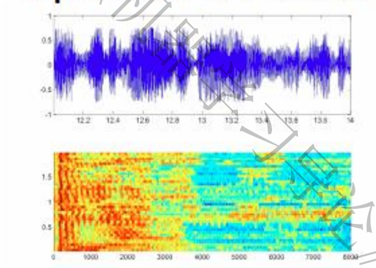
Deep learning

Nowadays, deep learning achieves great success

Images & Video

flickr
Google
YouTube

Speech & Audio



Text & Language

WIKIPEDIA The Free Encyclopedia
REUTERS
AP Associated Press

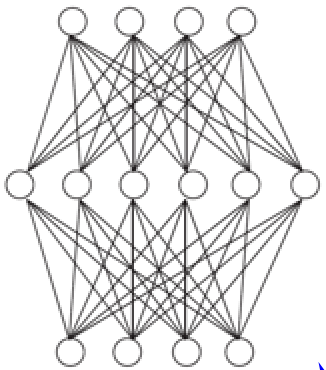
What's "Deep Learning"?

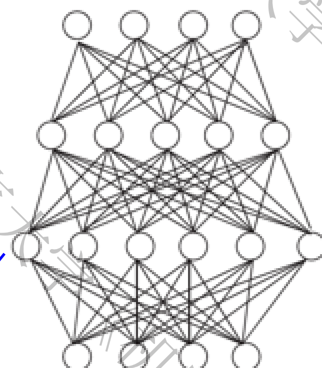
nowadays,

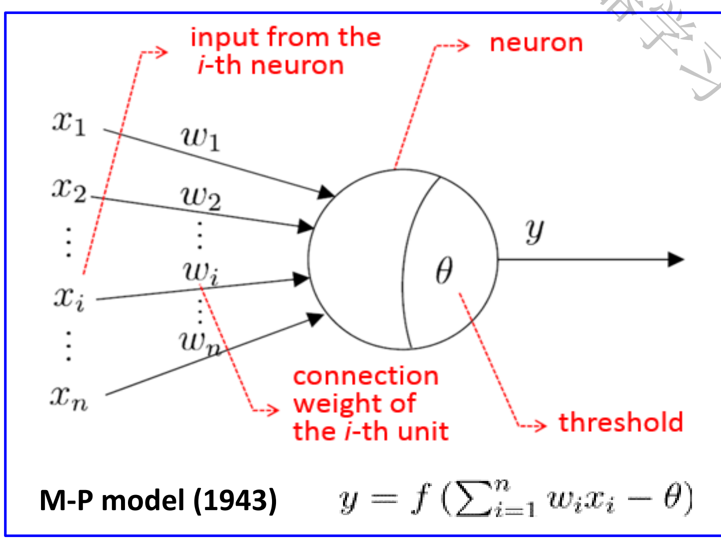
= Deep neural networks (DNNs)

深度神经网络

以往神经网络采用单或双隐层结构





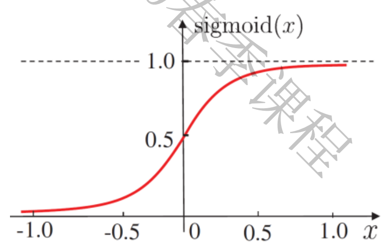


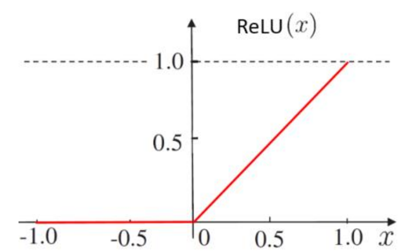
例如，ImageNet 胜者:
2012: 8 层
2015: 152 层
2016: 1207 层

deep

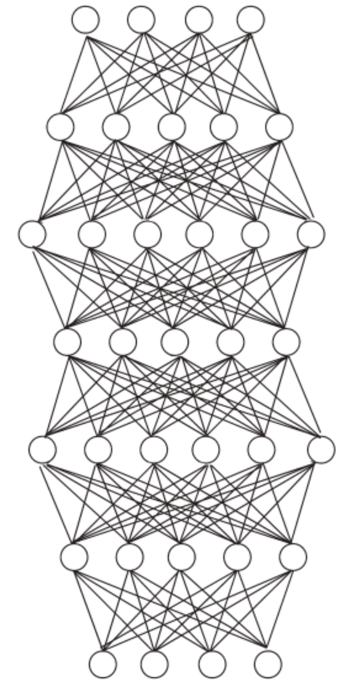
训练：
Backpropagation (BP) 或其变体

f: continuous, differentiable
e.g.,





深度神经网络：
很多层



Why deep? … One explanation

Increase model complexity → increase learning ability

- Add hidden units (model width)
- Add hidden layers (model depth)

Adding layers is more effective than adding units

increasing not only the number of units with activation functions, but also the embedding depths of the functions

Increase model complexity → increase risk of overfitting; difficulty in training

- For overfitting: Big training data
- For training: Powerful comp facilities

Error gradient will diverge when propagated in many layers, difficult to converge to stable state, and thus difficult to use classical BP algorithm

Lots of tricks

One explanation: High complexity matters?

- **BIG training data**

  The most simple yet effective way to reduce the risk of overfitting

- **Powerful computational facilities**

  Big models: Without GPU acceleration, DNNs could not be so successful

- **Training tricks**

  Heuristics, even mysteries

Error gradient will diverge when propagated in many layers, difficult to converge to stable state, thus difficult to use classical BP algo

⇒ **Enable to use high-complexity models**

⇒ **DNNs**

CANNOT explain: Why "flat" not good?

- *one-hidden-layer proved to be universal approximater*
- *complexity of one-hidden-layer can be arbitrarily high*

DNN 最重要的作用 → 表示学习（Representation learning）

以往

特征工程

Manual Feature Design

分类器学习

深度学习

关键

表示学习

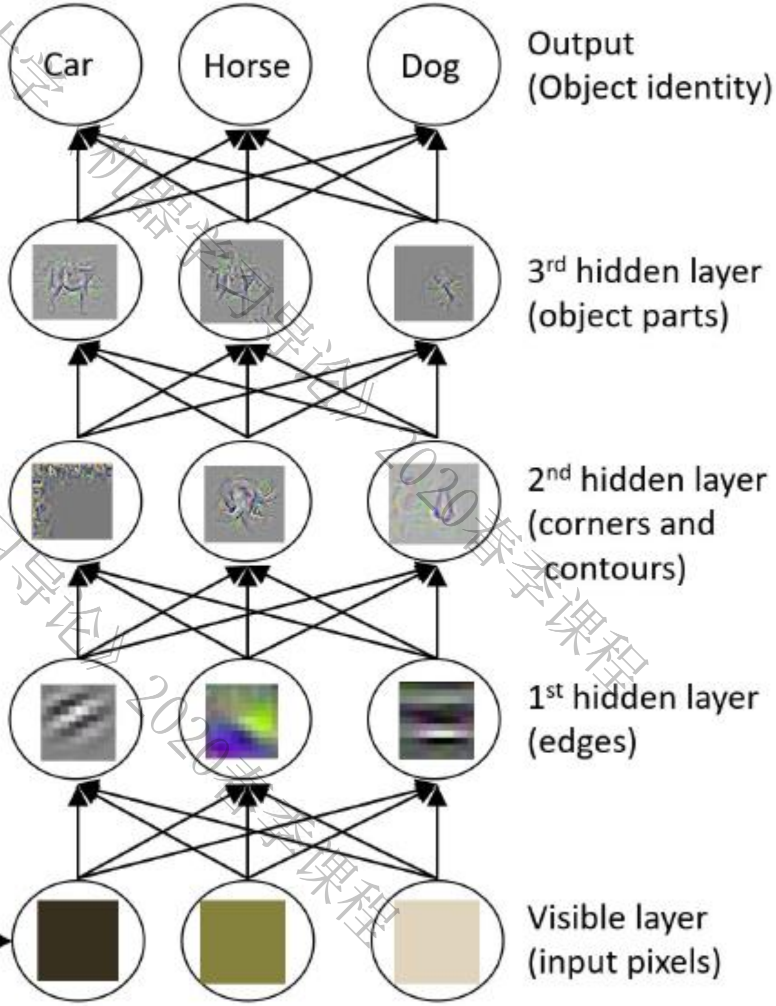
自动学习特征

端到端学习
(not that important)

分类器学习

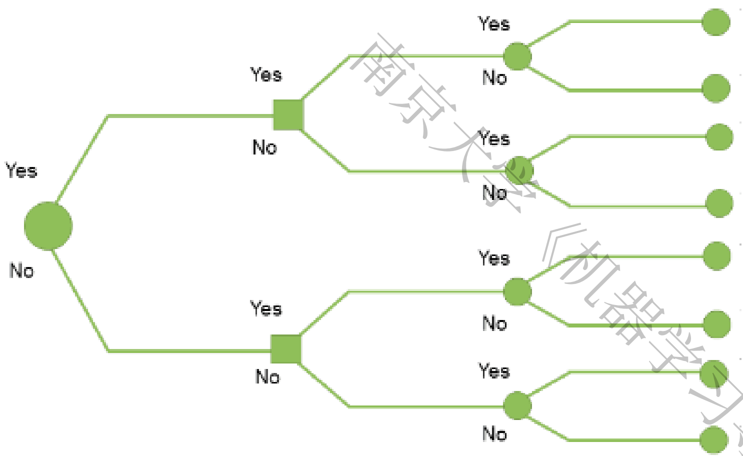
表示学习的关键

逐层处理
（Layer-by-layer processing）

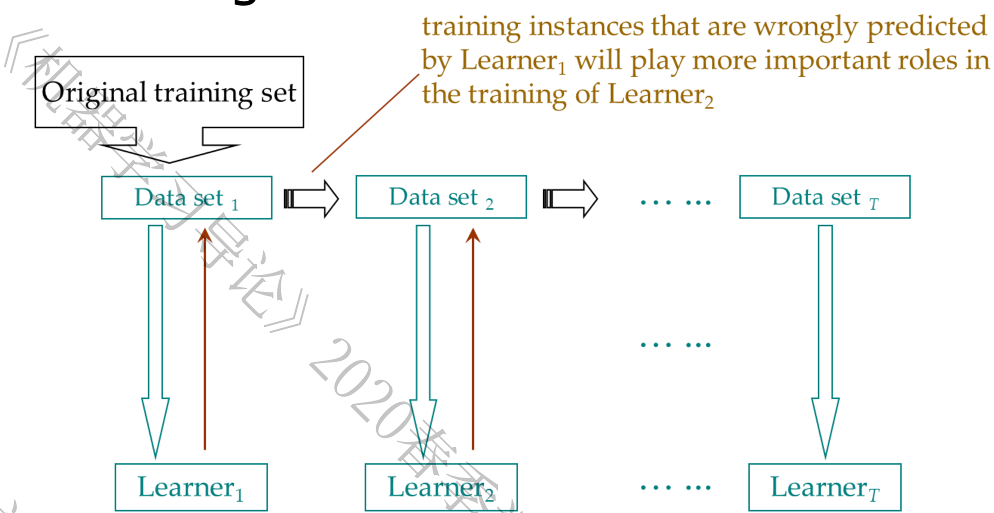


仅有逐层处理够不够？

决策树?



Boosting?

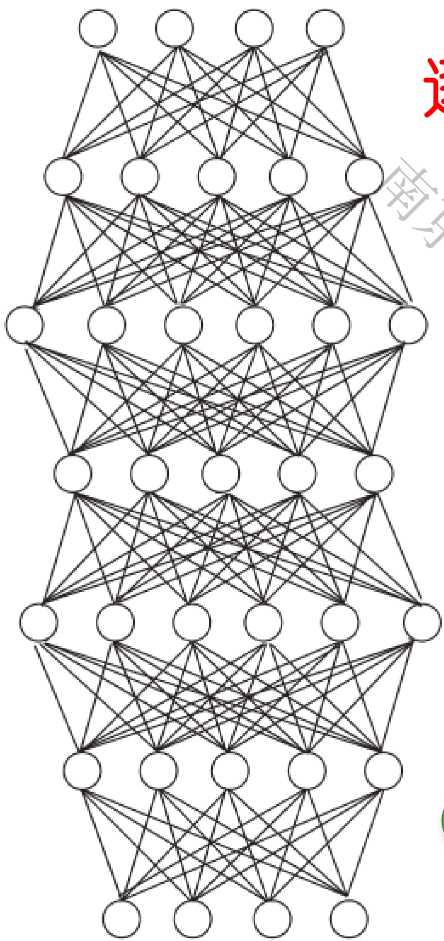


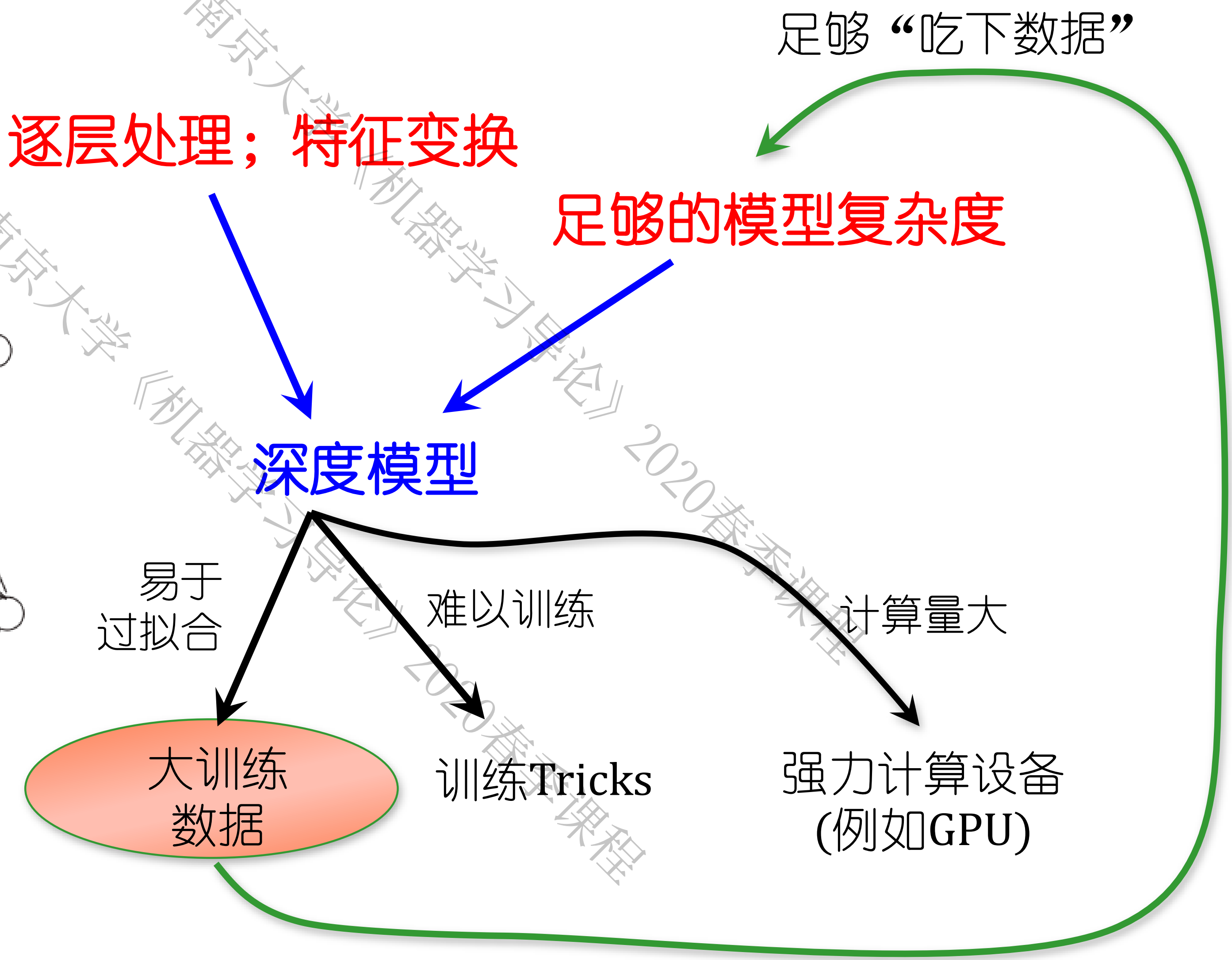
逐层处理，但是……

- 复杂度不足
- 始终基于初始特征，无特征变换

- 同样，复杂度不足
- 始终基于初始特征，无特征变换

为何要“深度”？





足够“吃下数据”
逐层处理；特征变换
足够的模型复杂度
深度模型
易于
过拟合
难以训练
计算量大
大训练
数据
训练Tricks
强力计算设备
(例如GPU)

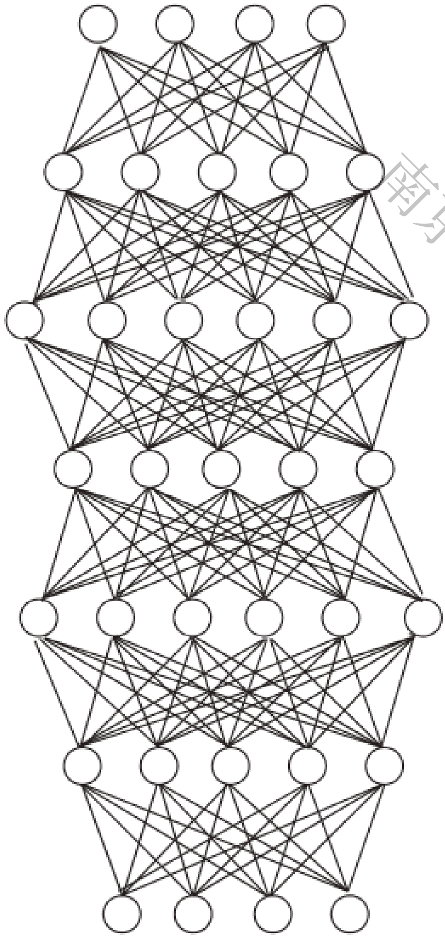
一个猜想

Deep models 的关键:

- 逐层加工处理
- 内置特征变换
- 模型复杂度够

"非神经网络不可"? NO!

深度神经网络的缺陷



- 太多超参数
  - 调参难，“跨任务”经验难分享
  - 重复结果难，即便使用相同数据、相同模型，不知道超参数设置就无法重现结果
- 模型一旦选定，模型复杂度即确定；通常远大于“所需”复杂度
- 大训练数据
- 理论分析难
- 黑箱模型
- …

从应用角度

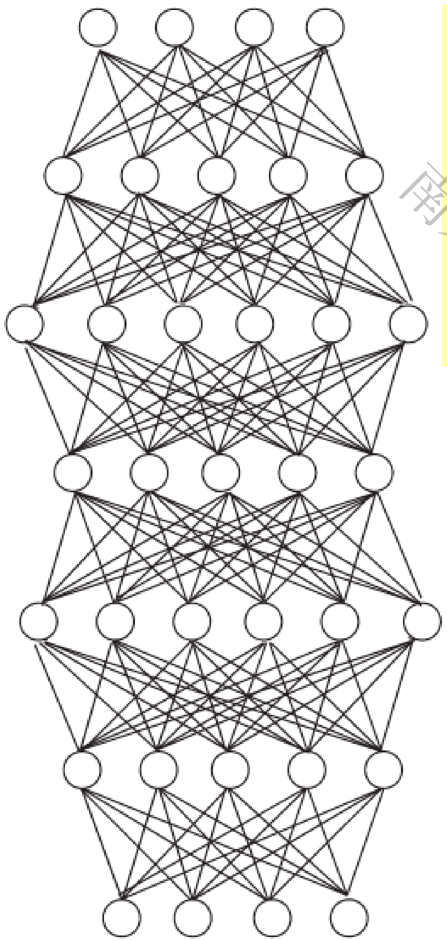
- 在图像、视频、语音之外的很多任务上，深度神经网络并非最佳选择，不少时候甚至表现不佳

  例如，在很多 Kaggle competition 任务上， 随机森林或者 XGBoost 表现更好

No Free Lunch！

没有任何一个模型能“包打天下”

重新审视深度模型



目前，深度模型就是深度神经网络，更确切地说:

multiple layers of parameterized differentiable nonlinear modules that can be trained by backpropagation

- 现实世界中并非所有规律性质都是“可微”(differentiable)，或者通过可微构件建模最优
- 机器学习中有很多“不可微”构件 (它们无法通过backpropagation训练)

A Grand Challenge

能否基于不可微构件进行深度学习 ？

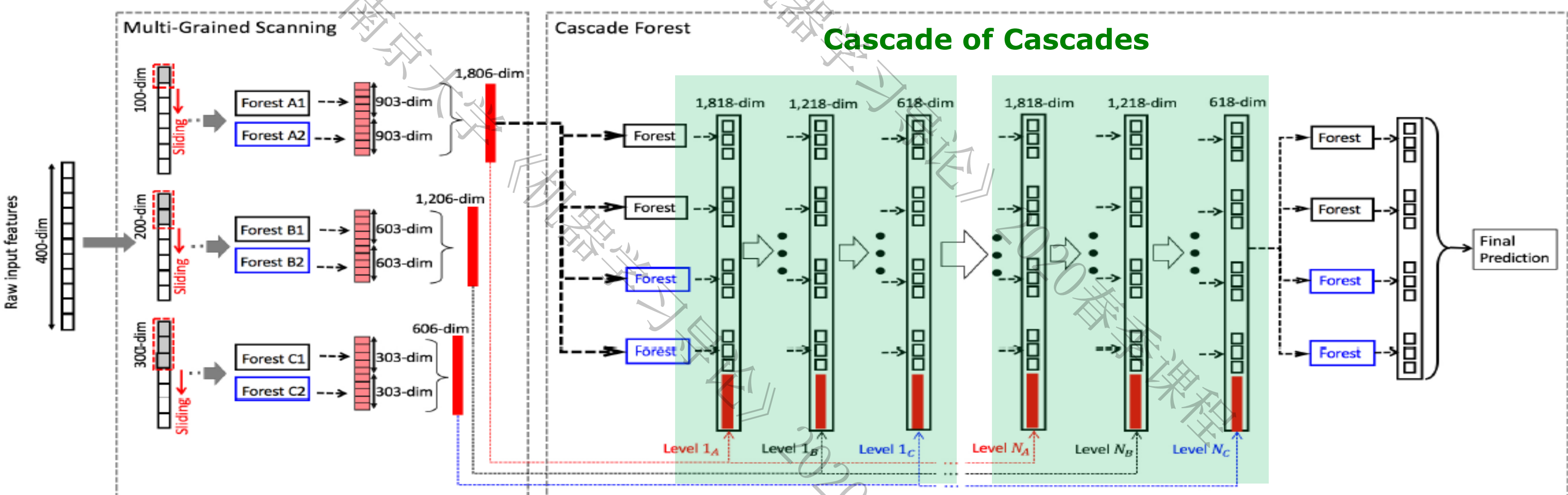
Can we realize deep learning with non-differentiable modules?

这个问题相当本质，对它的研究将可能使我们理解:

- Deep models ?= DNNs
- 如何能基于不可微构件"做深"？(不使用BP)
- 能否使得图像、语音、视频之外的更多任务受益于深度模型?
- ... ...

非神经网络的深度学习模型与方法

我们提出的“深度森林”(Deep Forest)，这是第一个“非神经网络”、不使用BP算法训练的深度学习模型

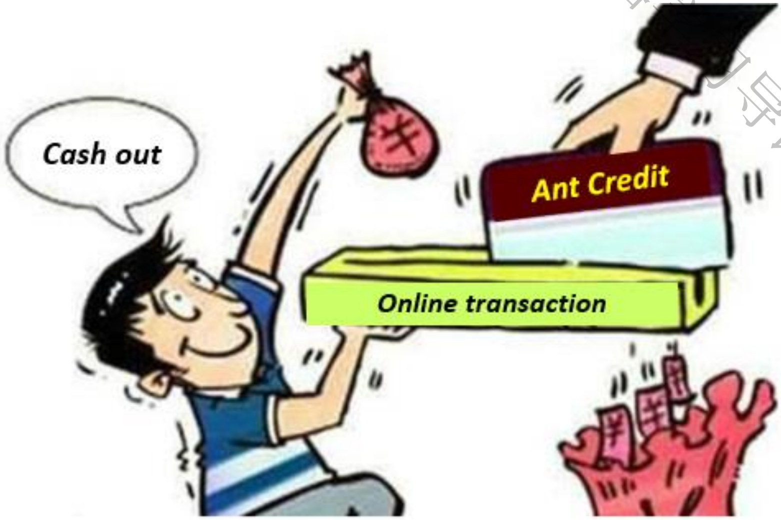


在基于非可微构件的深度学习模型方面，这是目前仅知的探索

- Z.-H. Zhou and J. Feng. Deep forest. **National Science Review**, 2019, 6(1): 74-86. (early version in IJCAI 2017) Code: http://lamda.nju.edu.cn/code_gcForest.ashx (for small- or medium-scale data)

A real application: Detection of Illegal cash-out (非法套现)

Very serious, particularly when considering the big amount of online transactions per day



For example, in 11.11 2016, more than 100 millions of transactions are paid by *Ant Credit Pay*

Big loss even if only a very small portions were fraud

Results

Table 1: The number of the training and test samples.

	# Pos. Ins.	# Neg. Ins.	# All Ins.
Train	171,784	131,235,963	131,407,704
Test	66,221	52,423,308	52,489,529

More than 5,000 features per transaction, categorical/numeric (details are business confidential)

Evaluation with common metrics

	AUC	F1	KS
LR	0.9887	0.4334	0.8956
DNN	0.9722	0.3861	0.8551
MART	0.9957	0.5201	0.9424
gcForest	**0.9970**	**0.5440**	**0.9480**

Evaluation with specified metrics

	1/10000	1/1000	1/100
LR	0.3708	0.5603	0.8762
DNN	0.3165	0.4991	0.8471
MART	0.4661	0.6716	0.9358
gcForest	**0.4880**	**0.6950**	**0.9470**

1/100 means that 1/100 of all transactions are interrupted

Deep forest performs much better than others

However,
not to expect too much immediately

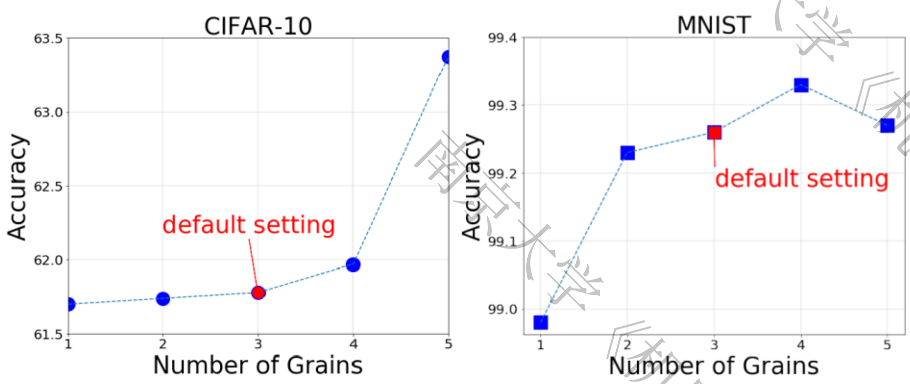
New tech usually has a long way to go

Challenges/Open Problems

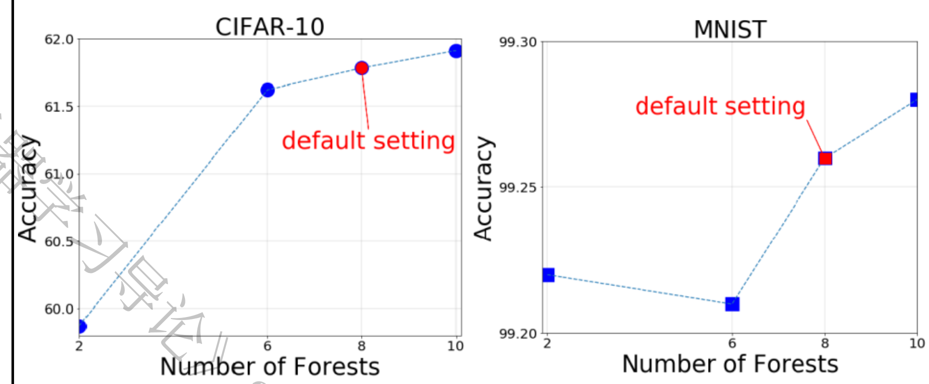
-- Hardware Speedup

Larger models tend to be better

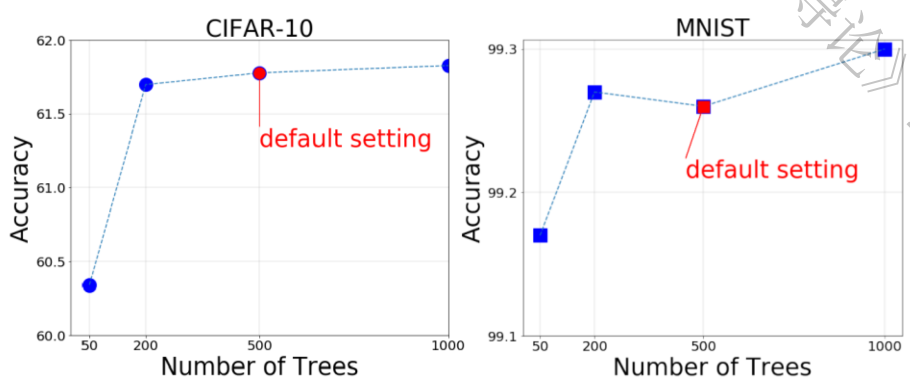
with increasing number of grains



with increasing number of forests per grade



with increasing number of trees per forest



Larger model might tend to offer better performances

But, currently we could not do larger

Computational facilities crucial for training larger models

e.g., GPUs for DNNs.

Hardware

Computational cost: DF training < DNN training

However,

- DNN gets great speedup by GPU
- DF naturally unsuited to GPU

If GPU speedup counted, DNN even more efficient

Can DF get speedup from suitable hardware?

*Can **KNL** (or some other architecture) do for DF as GPU for DNN ?*

南京大学《机器学习导论》2020春季课程

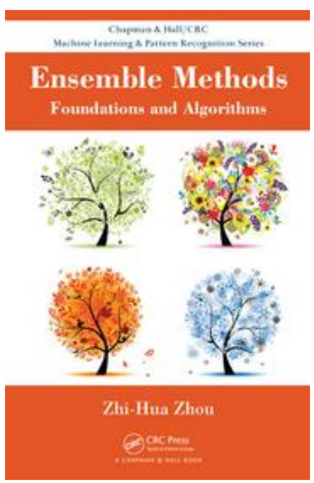
Challenges/Open Problems

-- Diversity

南京大学《机器学习导论》2020春季课程

gcForest is a success of ensemble methods

- “**Diversity**” is crucial for ensembles
- gcForest utilizes almost all kinds of strategies for diversity enhancement



Z.-H. Zhou.
[Ensemble Methods: Foundations and Algorithms](), Boca Raton, FL: Chapman & Hall/CRC, Jun. 2012.
(ISBN 978-1-439-830031)

Diversity

During training process,

- **Deep NN**: to avoid **Gradient vanishing**
- **Deep Forest**: to avoid **Diversity vanishing**
  - **It is a fundamental challenge to maintain sufficient *diversity* to enable DF to go deeper**
  - Tricks currently inspired by *ensemble methods; more fresh ones?*

Effort worthwhile

No Free Lunch !

No learning model "always" superior

Our conjecture:

- **Numerical modeling → DNNs**
  e.g., image/vision data
- **Non-numerical modeling → DF ?**
  e.g., symbolic/discrete/tabular data

The most important

DF offers a verification to our conjecture:

The essentials of deep learning:

- Layer-by-layer processing, and
- Feature transformation, and
- Sufficient model complexity

Forest model improved significantly through the DF way

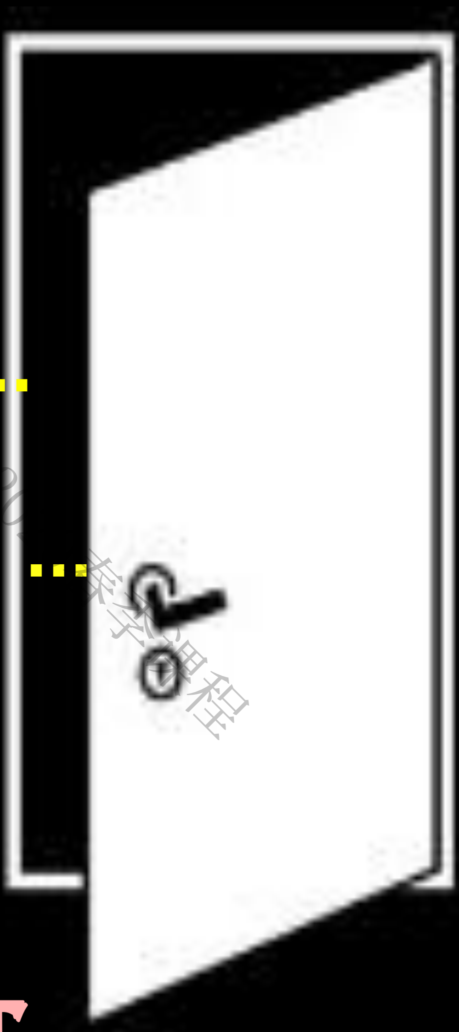
Can we improve other machine learning models by considering the above?

DEEP LEARNING ROOM

Deep neural networks

Deep Forest … …

… …



THE IS JUST A START

前往第七站……