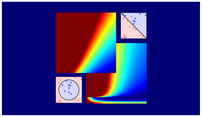
Machine Learning Techniques
(機器學習技巧)



Lecture 11: Neural Network

Hsuan-Tien Lin (林軒田)
htlin@csie.ntu.edu.tw

Department of Computer Science
& Information Engineering

National Taiwan University
(國立台灣大學資訊工程系)

Agenda

Lecture 11: Neural Network

- Random Forest: Theory and Practice
- Neural Network Motivation
- Neural Network Hypothesis
- Neural Network Training
- Deep Neural Networks

Theory: Does Diversity Help?

strength-correlation decomposition (classification):

$$\lim_{T\to\infty} E_{\text{out}}(G) \leq \rho \cdot \left(\frac{1-s^2}{s^2}\right)$$

- strength: average voting margin within G
- correlation: similarity between g_t
- similar for regression (bias-variance decomposition)

RF good if diverse and strong

Practice: How Many Trees Needed?

theory: the more, the 'better'

- NTU KDDCup 2013 Track 1: predicting author-paper relation
- $1 - E_{\text{val}}$ of thousands of trees: $[0.981, 0.985]$ depending on seed; $1 - E_{\text{out}}$ of top 20 teams: $[0.98130, 0.98554]$
- decision: take 12000 trees with seed 1

cons of RF: may need lots of trees **if random process too unstable**

Fun Time

Disclaimer

Many parts of this lecture borrows
Prof. Yaser S. Abu-Mostafa's slides with permission.

Learning From Data

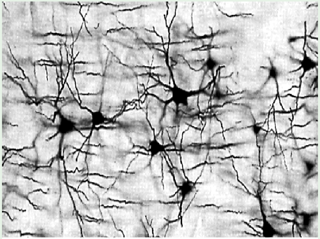
Yaser S. Abu-Mostafa
California Institute of Technology

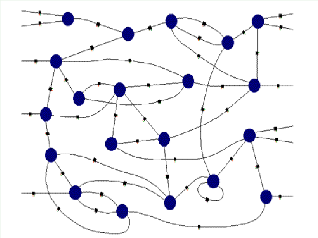
Lecture 10: **Neural Networks**

Sponsored by Caltech's Provost Office, E&AS Division, and IST • Thursday, May 3, 2012

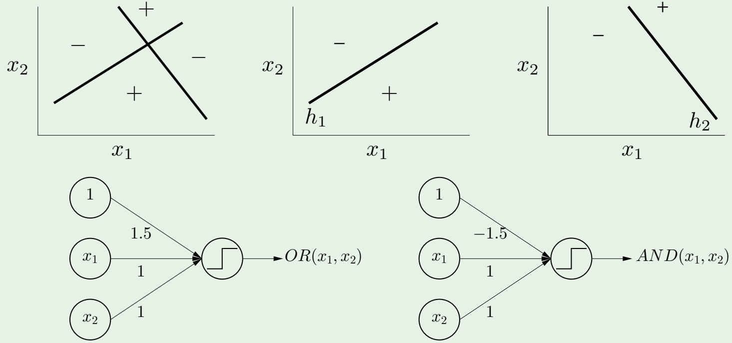
Biological inspiration

biological function $\longrightarrow$ biological structure



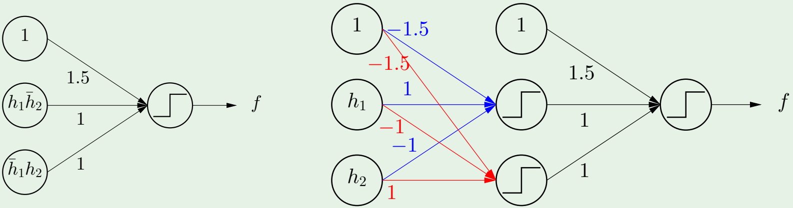


Combining perceptrons



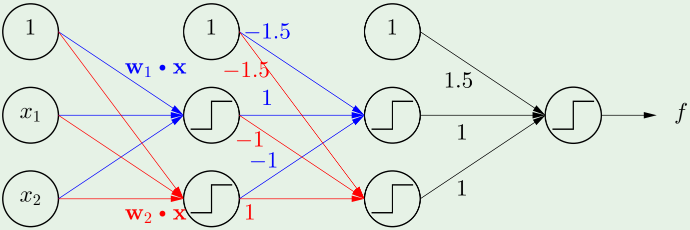
Learning From Data - Lecture 10   9/21

Creating layers



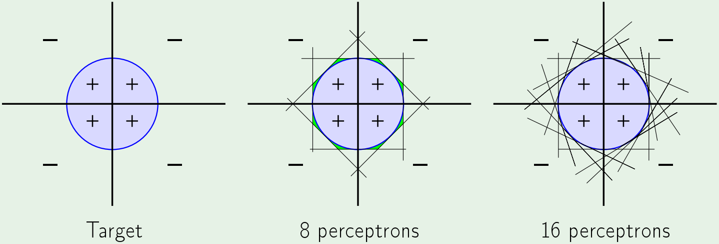
Learning From Data - Lecture 10 10/21

The multilayer perceptron



3 layers "feedforward"

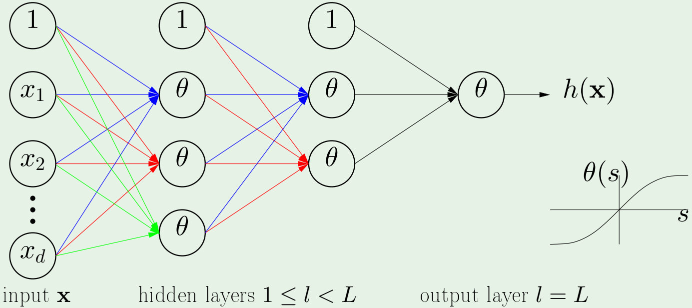
A powerful model

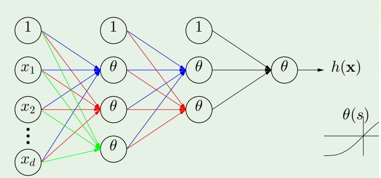


2 red flags for **generalization** and **optimization**

Fun Time

The neural network



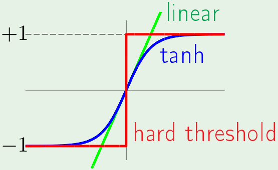


How the network operates

$$w_{ij}^{(l)} \quad \begin{cases} 1 \le l \le L & \text{layers} \\ 0 \le i \le d^{(l-1)} & \text{inputs} \\ 1 \le j \le d^{(l)} & \text{outputs} \end{cases}$$

$$x_j^{(l)} = \theta(s_j^{(l)}) = \theta\left(\sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)}\, x_i^{(l-1)}\right)$$

Apply $\mathbf{x}$ to $x_1^{(0)} \cdots x_{d^{(0)}}^{(0)} \rightarrow\rightarrow x_1^{(L)} = h(\mathbf{x})$



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

Fun Time

Applying SGD

All the weights $\mathbf{w} = \{w_{ij}^{(l)}\}$ determine $h(\mathbf{x})$

Error on example $(\mathbf{x}_n, y_n)$ is

$$\mathbf{e}\left(h(\mathbf{x}_n), y_n\right) = \mathbf{e}(\mathbf{w})$$

To implement SGD, we need the gradient

$$\nabla\mathbf{e}(\mathbf{w}): \quad \frac{\partial\ \mathbf{e}(\mathbf{w})}{\partial\ w_{ij}^{(l)}} \quad \text{for all } i, j, l$$

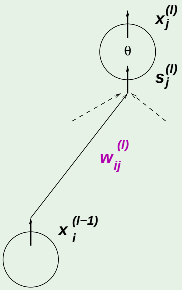
Computing $\frac{\partial \ \mathrm{e}(\mathbf{w})}{\partial \ w_{ij}^{(l)}}$

We can evaluate $\frac{\partial \ \mathrm{e}(\mathbf{w})}{\partial \ w_{ij}^{(l)}}$ one by one: analytically or numerically

A trick for efficient computation:

$$\frac{\partial \ \mathrm{e}(\mathbf{w})}{\partial \ w_{ij}^{(l)}} = \frac{\partial \ \mathrm{e}(\mathbf{w})}{\partial \ s_j^{(l)}} \times \frac{\partial \ s_j^{(l)}}{\partial \ w_{ij}^{(l)}}$$

We have $\frac{\partial \ s_j^{(l)}}{\partial \ w_{ij}^{(l)}} = x_i^{(l-1)}$ We only need: $\frac{\partial \ \mathrm{e}(\mathbf{w})}{\partial \ s_j^{(l)}} = \delta_j^{(l)}$



δ for the final layer

$$\delta_j^{(l)} = \frac{\partial \, \mathrm{e}(\mathbf{w})}{\partial \, s_j^{(l)}}$$

For the final layer $l = L$ and $j = 1$:

$$\delta_1^{(L)} = \frac{\partial \, \mathrm{e}(\mathbf{w})}{\partial \, s_1^{(L)}}$$

$$\mathrm{e}(\mathbf{w}) = (x_1^{(L)} - y_n)^2$$

$$x_1^{(L)} = \theta(s_1^{(L)})$$

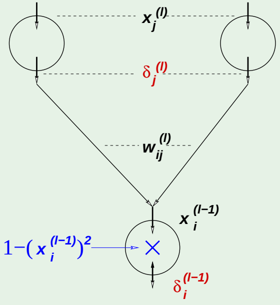
Back propagation of δ

$$\delta_i^{(l-1)} = \frac{\partial\ \mathrm{e}(\mathbf{w})}{\partial\ s_i^{(l-1)}}$$

$$= \sum_{j=1}^{d^{(l)}} \frac{\partial\ \mathrm{e}(\mathbf{w})}{\partial\ s_j^{(l)}} \times \frac{\partial\ s_j^{(l)}}{\partial\ x_i^{(l-1)}} \times \frac{\partial\ x_i^{(l-1)}}{\partial\ s_i^{(l-1)}}$$

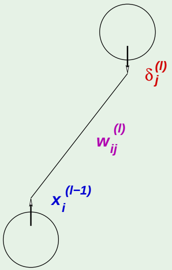
$$= \sum_{j=1}^{d^{(l)}} \delta_j^{(l)} \times w_{ij}^{(l)} \times \theta'(s_i^{(l-1)})$$

$$\delta_i^{(l-1)} = (1 - (x_i^{(l-1)})^2) \sum_{j=1}^{d^{(l)}} w_{ij}^{(l)}\ \delta_j^{(l)}$$



Backpropagation algorithm

Initialize all weights $w_{ij}^{(l)}$ at random
for $t = 0, 1, 2, \ldots$ **do**
    Pick $n \in \{1, 2, \cdots, N\}$
    Forward: Compute all $x_j^{(l)}$
    Backward: Compute all $\delta_j^{(l)}$
    Update the weights: $w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta \, x_i^{(l-1)} \delta_j^{(l)}$
    Iterate to the next step until it is time to stop
Return the final weights $w_{ij}^{(l)}$

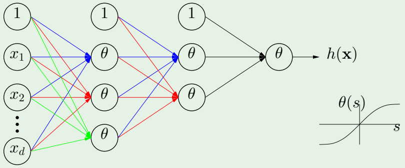


Fun Time

Final remark: hidden layers

learned nonlinear transform

interpretation?



Learning From Data - Lecture 10 21/21

Shallow versus Deep Structures

shallow: few hidden layers; deep: many hidden layers

Shallow

- efficient
- powerful if enough neurons

Deep

- challenging to train
- needing more structural (model) decisions
- 'meaningful'?

deep structure (deep learning) **re-gain attention recently**

Key Techniques behind Deep Learning

- (usually) **unsupervised pre-training** between hidden layers,
  —viewing hidden layers as **'condensing'** low-level representation to high-level one
- **fine-tune with backprop** after initializing with those 'good' weights
  —because **direct backprop may get stuck more easily**
- speed-up: better optimization algorithms, and faster **GPU**

currently very useful for **vision and speech recognition**

Fun Time

Summary

Lecture 11: Neural Network

- Random Forest: Theory and Practice
- Neural Network Motivation
- Neural Network Hypothesis
- Neural Network Training
- Deep Neural Networks