Course ID 80250993 Dates: 9/16-12/23/2021 @ I-205
Tencent/VooV Meeting 521 4678 6257 (passwd 1205)

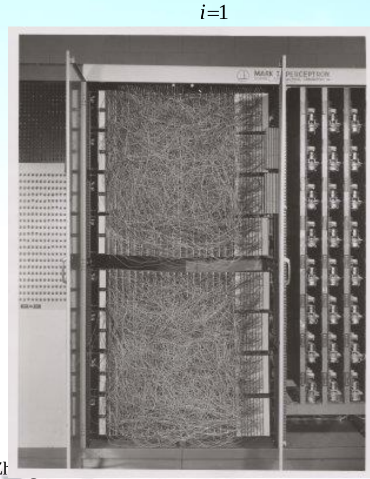
Chapter 5
Classical Artificial Neural Networks

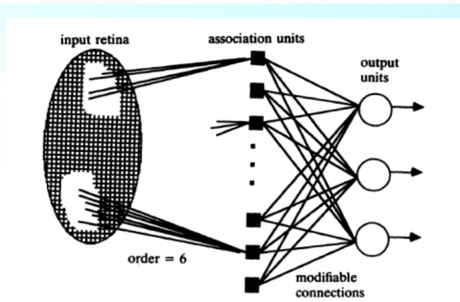
Xuegong Zhang
September 30, 2021

Perceptron

Frank Rosenblatt, *The Perceptron – a perceiving and recognizing automaton*, Report 85-460-1, Cornell Aeronautical Laboratory, Jan. 1957

$$y = \mathrm{sgn}\left(\sum_{i=1}^{d} w_i x_i + w_0\right)$$



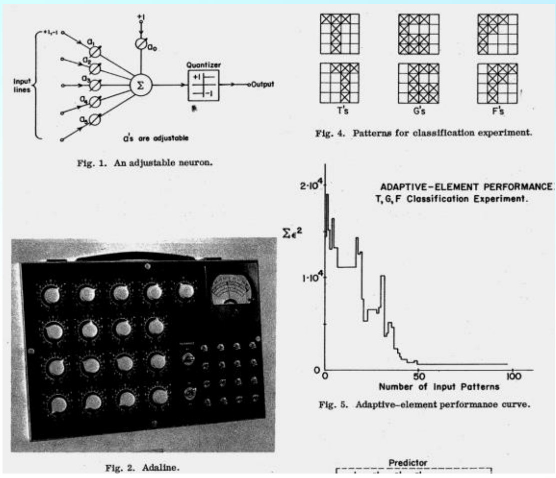


https://en.wikipedia.org/wiki/Perceptron

M. Olazaran, A sociological study of the official history of the perceptrons controversy, *Social Studies of Science*, 1996

ADALINE

Widrow & Hoff, Adaptive switching circuits, *1960 IRE Western Electric Show and Convention Record, Part 4*, pp.96-104, Aug, 1960



Widrow & Hoff, Adaptive switching circuits, *1960 IRE Western Electric Show and Convention Record, Part 4*, pp.96-104, Aug, 1960

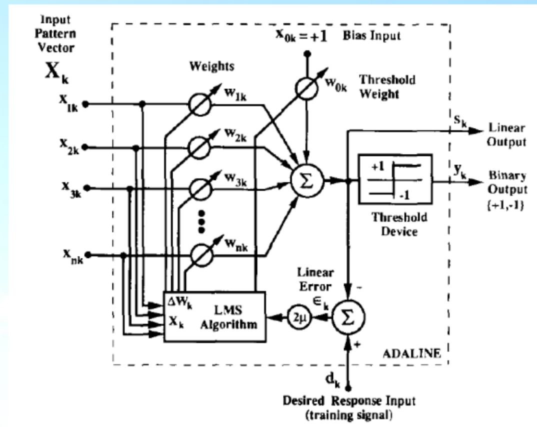


Fig. 2. Adaptive linear element (Adaline).

Widrow & Lehr, 30 years of adaptive neural networks: Perceptron, Madaline, and Backpropagation, *Proceedings of the IEEE*, 78(9): 1415-1442, 1990

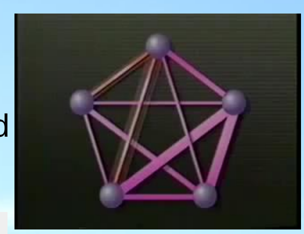
Xuegong Zhang 3

An early prediction on perceptrons

- In a 1958 press conference organized by the US Navy, Rosenblatt made statements about the perceptron that caused a heated controversy among the fledgling AI community; based on Rosenblatt's statements, *The New York Times* reported:

The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence. Later perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech and writing in another language, it was predicted.[17]

M. Olazaran, A sociological study of the official history of the perceptrons controversy, *Social Studies of Science*, 1996

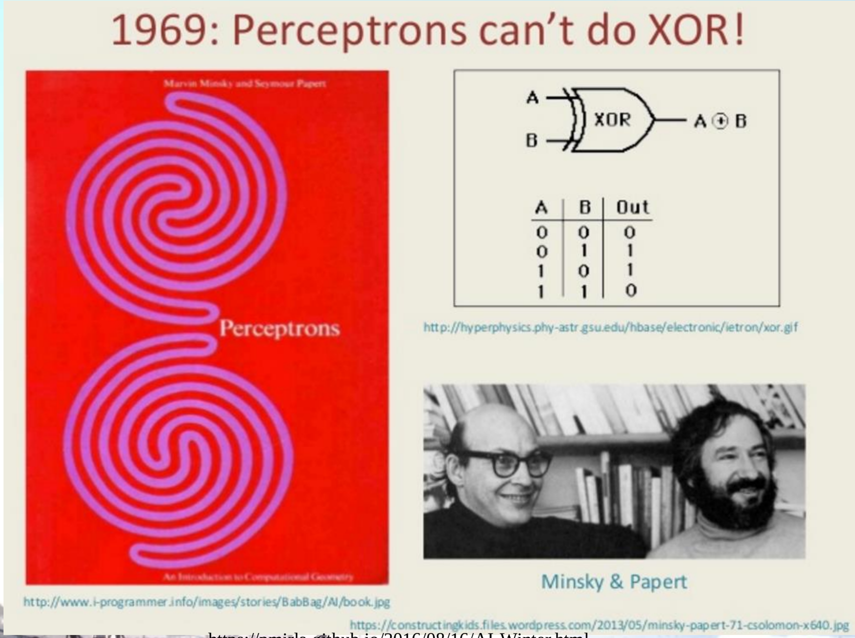


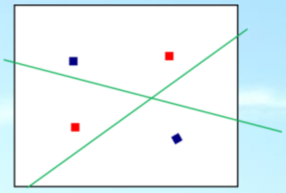
https://pmirla.github.io/2016/08/16/AI-Winter.html

D

Xuegong Zhang 4

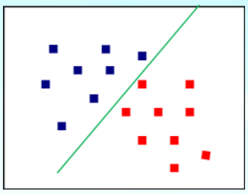
Criticisms on Perceptrons

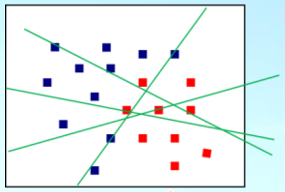


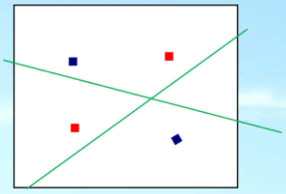


https://pmirla.github.io/2016/08/16/AI-Winter.html

Linearly non-separable cases





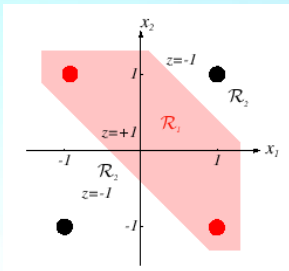


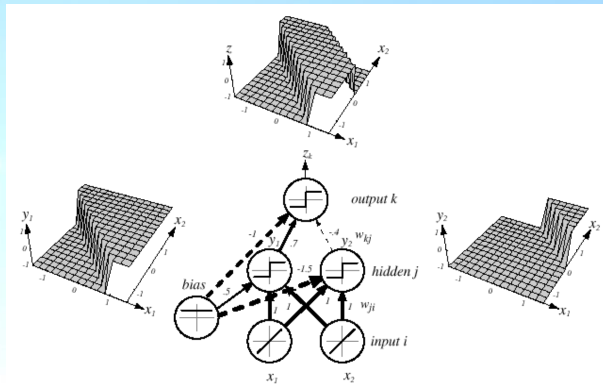
Linearly separable

Not perfectly separable, but can achieve some reasonable separation with a linear classifier.

No reasonable linear solution or approximation.

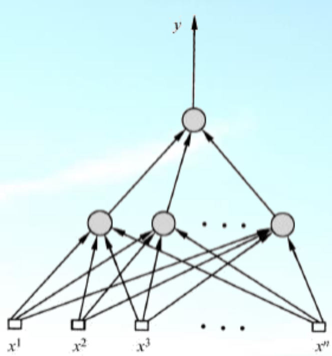
Implement nonlinear functions with multiple neurons

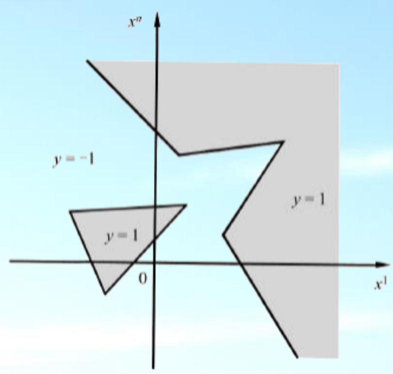




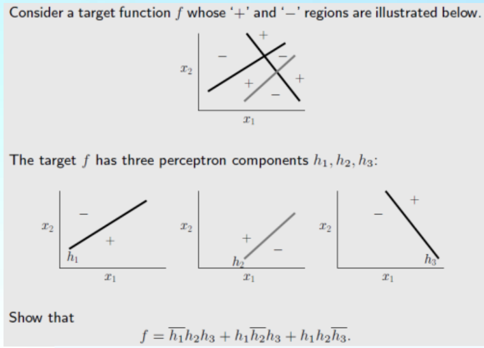
Duda, Hart & Stork, Pattern Classification (2nd edition), John Wiley & Sons, 2001

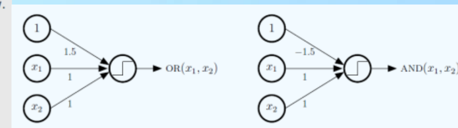
Early forms of multiple-layered perceptron (neural networks)

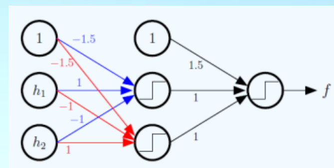


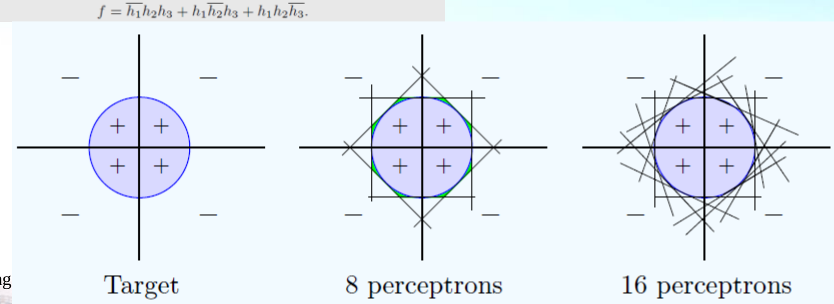


Boolean decomposition of perceptrons



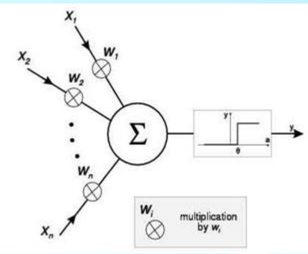


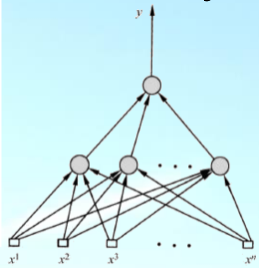


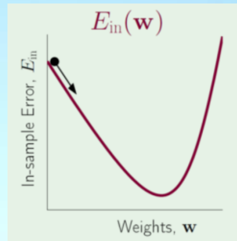


However, this field was closed for ~20 years (1960s-1980s)

Why didn't it work?







- The old trick for learning?
  - Cannot get gradient

Discriminant function

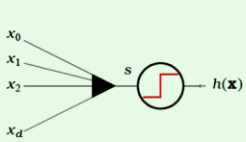
$$g_k(\mathbf{x}) \equiv y_k = \text{Sign}\left(\sum_j w_{jk}\text{Sign}\left(\sum_i w_{ij}x_i + w_{j0}\right) + w_{k0}\right)$$

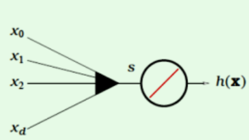
D

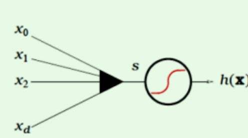
Recall: Three major types of linear machines

$$s = \sum_{i=0}^{d} w_i x_i$$

linear classification	linear regression	logistic regression
$h(\mathbf{x}) = \text{sign}(s)$	$h(\mathbf{x}) = s$	$h(\mathbf{x}) = \theta(s)$

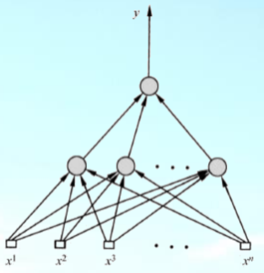


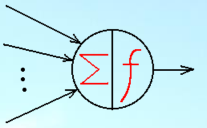




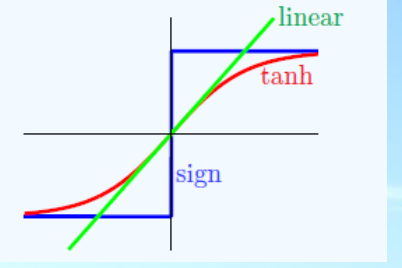
Abu-Mostafa, Magdon-Ismail, Lin, *Learning from Data, Lecture 9*

Solution?





f : activation function

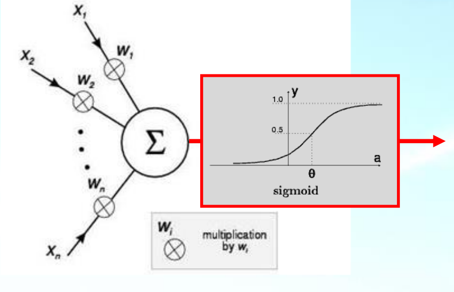


$$g_k(\mathbf{x}) \equiv y_k = \text{Sign}\left(\sum_j w_{jk}\text{Sign}\left(\sum_i w_{ij}x_i + w_{j0}\right) + w_{k0}\right)$$

Homework:
Show that nonlinearity cannot be achieved if we use linear activation function for all nodes in a multiple layer perceptron model.

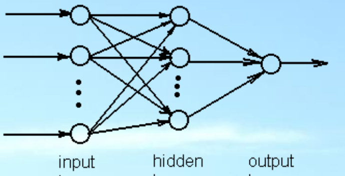
Sec 6.2 Problem 1, Duda et al, Pattern Classification, p. 335

MLP (Multi-layer Perceptron)
--- The most popular Artificial Neural Network (NN)



$$f(\alpha) = \frac{1}{1+e^{-(\alpha-\theta)}}$$

Sigmoid function



a 3-layer MLP

$$g_k(\mathbf{x}) \equiv y_k = f\left(\sum_j w_{jk} f\left(\sum_i w_{ij}x_i + w_{j0}\right) + w_{k0}\right)$$

Nonlinearity with multiple layers

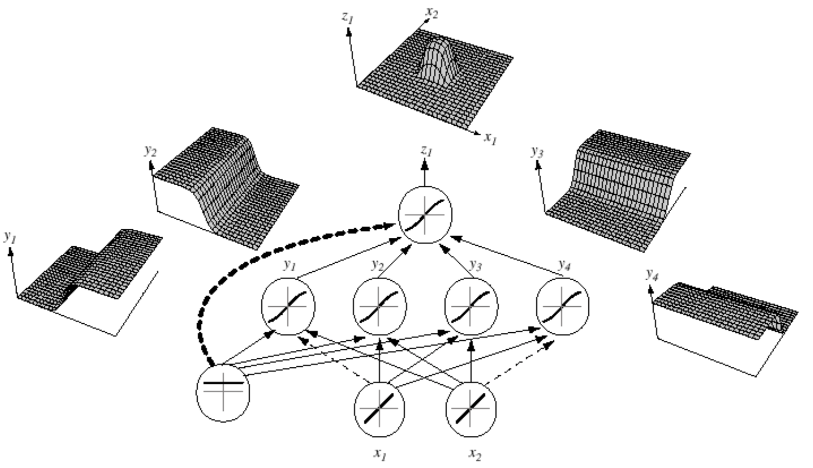


FIGURE 6.2. A 2-4-1 network (with bias) along with the response functions at different units; each hidden output unit has sigmoidal activation function $f(\cdot)$. In the case shown, the hidden unit outputs are paired in opposition thereby producing a "bump" at the output unit. Given a sufficiently large number of hidden units, any continuous function from input to output can be approximated arbitrarily well by such a network. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

The Kolmogorov theorem

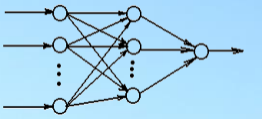
Any continuous function $g(\boldsymbol{x})$ defined on the unit hypercube I^n ($I = [0,1]$ and $n \geq 2$) can be represented in the form

$$g(\mathbf{x}) = \sum_{j=1}^{2n+1} \Xi_j \left(\sum_{i=1}^{d} \Psi_{ij}(x_i) \right)$$

for properly chosen functions Ξ_j and Ψ_{ij}.

The representative power of MLP

$$g_k(\mathbf{x}) \equiv y_k = f\left(\sum_j w_{jk} f\left(\sum_i w_{ij} x_i + w_{j0}\right) + w_{k0}\right)$$



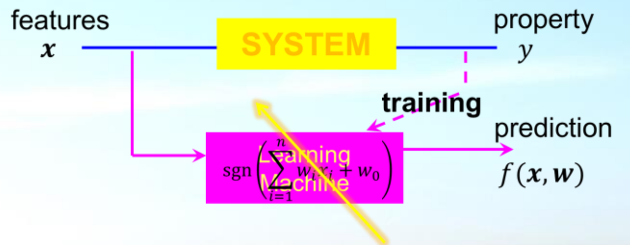
Conclusion:

MLP is capable of “implementing” any continuous function from input to output, given sufficient number of hidden units, proper activation function and weights.

1-minute break

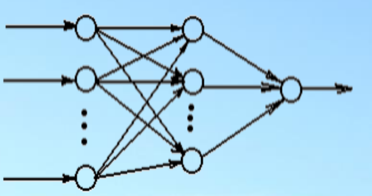
The basic elements of supervised learning

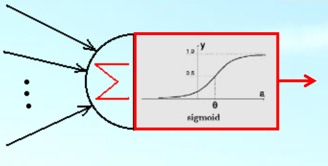
- It needs a teacher.
  - → We (people) ***design*** it (features, the model) and ***train*** it.
- We need materials to train it. / It needs materials to learn from.
  - → Training data
- We need to tell what is the goal of the learning.
  - → Objective function
- We need to tell it how to learn.
  - → Learning algorithm

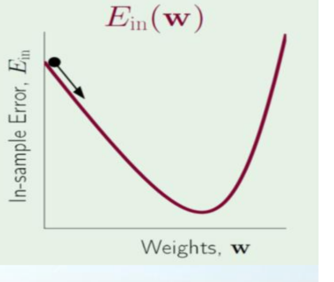


How to design a MLP?

- The NN structure:
  - # of layers and # of nodes at each layer
  - Input: features to use
  - Output: real values
    - Coding according to the task
- Choose the activation function
  - Sigmoid
- Algorithm for training the weights
  - Gradient Descent







Basic concepts of ML: Perceptron

- How can we make a learning machine?
  - It needs a teacher.
    - → The model: $y = \mathrm{sgn}(\sum_{i=1}^{d} w_i x_i + w_0)$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\}$, $\boldsymbol{x}_j \in R^{d+1}, y_j \in \{-1,1\}$
  - We need to tell what is the goal of the learning.
    - → Objective function: $\min J_P(\boldsymbol{\alpha}) = \sum_{\mathbf{y}_j \in \mathcal{Y}^k} (-\boldsymbol{\alpha}^T \mathbf{y}_j)$
  - We need to tell it how to learn.
    - → Learning algorithm: $\boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) - \rho_k \nabla J = \boldsymbol{\alpha}(k) + \rho_k \sum_{\mathbf{y}_j \in Y^k} \mathbf{y}_j$

Basic concepts of ML: Logistic Regression

- How can we make a learning machine?
  - It needs a teacher.
    - → The model: $h(\boldsymbol{x}) = \theta(\boldsymbol{w}^T \boldsymbol{x})$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\}$, $\boldsymbol{x}_j \in R^{d+1}, y_j \in \{-1,1\}$
  - We need to tell what is the goal of the learning.
    - → Objective function: $\min E(\boldsymbol{w}) = \frac{1}{N} \sum_{j=1}^{N} \ln\left(1 + e^{-y_j \boldsymbol{w}^T \boldsymbol{x}_j}\right)$
  - We need to tell it how to learn.
    - → Learning algorithm: $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \rho_k \nabla E$

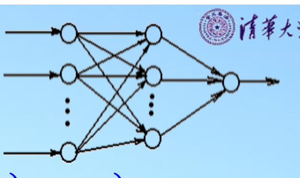
Basic concepts of ML: Linear Regression

- How can we make a learning machine?
  - It needs a teacher.
    - → The model: $f(\boldsymbol{x}) = \sum_{i=0}^{d} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\}$, $\boldsymbol{x}_j \in R^{d+1}, y_j \in R$
  - We need to tell what is the goal of the learning.
    - → Objective function: $\min E = \frac{1}{N} \sum_{j=1}^{N} \left(f(\boldsymbol{x}_j) - y_j\right)^2$
  - We need to tell it how to learn.
    - → Learning algorithm: $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \rho_k \nabla E$

Multilayer Perceptron ?

Xuegong Zhang 21

Basic concepts of ML: MLP



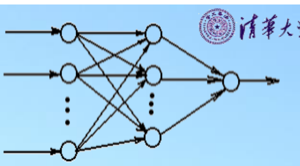
- How can we make a learning machine?
  - It needs a teacher.
    - → The model: $g(\mathbf{x}) = f\left(\sum_j w_{jk} f\left(\sum_i w_{ij} x_i + w_{j0}\right) + w_{k0}\right)$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\}$, $\boldsymbol{x}_j \in R^{d+1}, y_j \in R$
  - We need to tell what is the goal of the learning.
    - → Objective function: $\min E = \frac{1}{2} \sum_{j=1}^{N} \left(g(\boldsymbol{x}_j) - y_j\right)^2$
  - We need to tell it how to learn.
    - → Learning algorithm: $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \rho_k \nabla E$

D

$\nabla E = \partial E / \partial \boldsymbol{w}$?

Xuegong Zhang 22

The challenge



$$\min E = \frac{1}{2}\sum_{j=1}^{N}\left(g(\boldsymbol{x}_j) - y_j\right)^2$$

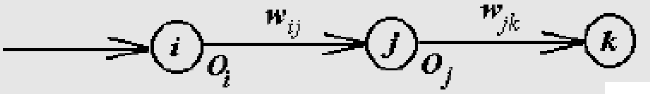
$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \rho_k \nabla E$$

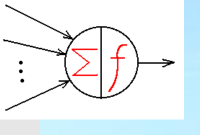
$$g(\mathbf{x}) = f\left(\sum_j w_{jk} f\left(\sum_i w_{ij} x_i + w_{j0}\right) + w_{k0}\right)$$

$$\nabla E = \partial E / \partial \boldsymbol{w}$$

Back-Propagation (BP) of the Error

LeCun, 1986; Rumelhart, Hinton & Williams, 1986; Parker, 1985

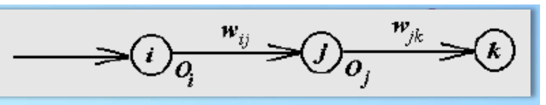




for node j, $net_j = \sum_i w_{ij} O_i$, $O_j = f(net_j)$

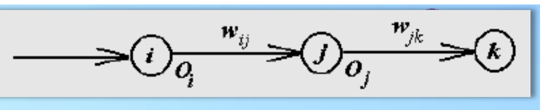
for output node: $\hat{y}_j = O_j$ is the actual output, y_j is the desired output, error

$$g(\mathbf{x}) = f\left(\sum_j w_{jk} f\left(\sum_i w_{ij} x_i + w_{j0}\right) + w_{k0}\right)$$



error: $$E = \frac{1}{2}\sum_j (y_j - \hat{y}_j)^2$$ Objective function

gradients: $$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} = \delta_j O_i, \qquad \delta_j \equiv \frac{\partial E}{\partial net_j}$$



$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} = \delta_j O_i, \qquad \delta_j \equiv \frac{\partial E}{\partial net_j}$$

$$net_j = \sum_i w_{ij} O_i, \quad O_j = f(net_j)$$

The Chain Rule for Derivatives

H

for output nodes: $$O_j = \hat{y}_j, \qquad \delta_j = \frac{\partial E}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial net_j} = \boxed{-(y_j - \hat{y}_j) f'(net_j)}$$

for hidden nodes (back-propagation):

$$\delta_j = \frac{\partial E}{\partial net_j} = \sum_k \frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial O_j} \cdot \frac{\partial O_j}{\partial net_j} = \boxed{\sum_k \delta_k w_{jk} f'(net_j)}$$

for output nodes: $O_j = \hat{y}_j$, $\delta_j = \frac{\partial E}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial net_j} = \boxed{-(y_j - \hat{y}_j) f'(net_j)}$

for hidden nodes (back-propagation):

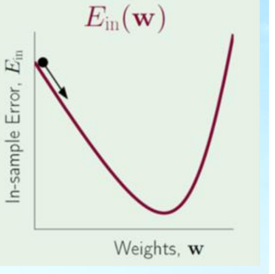
$$\delta_j = \frac{\partial E}{\partial net_j} = \sum_k \frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial O_j} \cdot \frac{\partial O_j}{\partial net_j} = \boxed{\sum_k \delta_k w_{jk} f'(net_j)}$$

adaption (at iteration t):

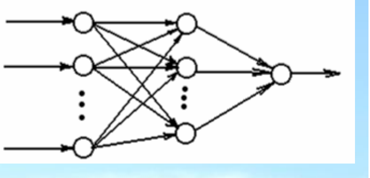
$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t), \qquad \boxed{\Delta w_{ij}(t) = -\eta \delta_j(t) O_i(t)}$$

where: η - step length (learning rate)

H

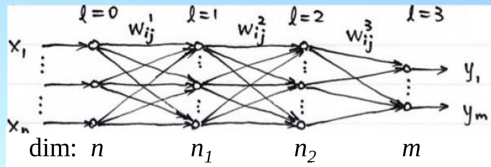


The Back-Propagation (BP) Algorithm



- Purpose: to train MLP weights with data
- Goal: to minimize the error (MSE)
- Algorithm: *gradient-decreasing*
  - forward: compute the output with certain input
    input nodes >>> hidden nodes >>> output nodes
  - compare: discrepancy between output and desired
  - backward: propagate the error back through the network
    input nodes <<< hidden nodes <<< output nodes
  - adaptation: adapt the weights according to the errors

Pseudo-codes for BP with sigmoid nodes

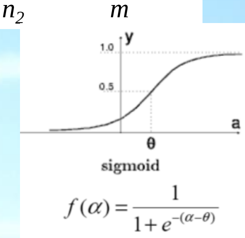


1º. Initialize weights (with small random values), $t = 0$

2º. Apply a training sample $\boldsymbol{x} = [x_1, \cdots, x_n]^{\mathrm{T}} \in R^n$ with desired output $D = [d_1, \cdots, d_m]^T \in R^m$

3º. Forward calculation: $Y = [y_1, \cdots, y_m]^{\mathrm{T}} \in R^m$,

$$y_l = f\left(\sum_{k=1}^{n_2} w_{kl} \, f\left(\sum_{j=1}^{n_1} w_{jk} f\left(\sum_{i=1}^{n} w_{ij} x_i\right)\right)\right), \ l = 1, \cdots, m$$



4º. Adjust weights from the output layer. For layer l,

$$w_{ij}^l(t+1) = w_{ij}^l(t) - \eta \delta_j^l x_i^{l-1}, \ j = 1, \cdots, n_l, i = 1, \cdots, n_{l-1}$$

where for the output layer

$$\delta_j^l = -y_j(1 - y_j)(d_j - y_j), j = 1, \cdots, m$$

and for hidden layers

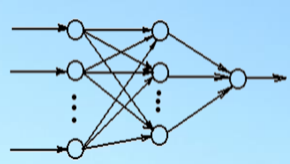
$$\delta_j^l = x_j^l(1 - x_j^l) \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} w_{jk}^{l+1}(t), j = 1, \cdots, n_l$$

$f(\alpha) = 1/(1 + e^{-\alpha})$

$f'(\alpha) = f(\alpha)(1 - f(\alpha))$

5º. Loop: if stop criterion not met, set $t = t + 1$ and go to 2º with another sample.

Basic concepts of ML: MLP



- How can we make a learning machine?
  - It needs a teacher.
    - → The model: $g(\mathbf{x}) = f\left(\sum_j w_{jk} f\left(\sum_i w_{ij} x_i + w_{j0}\right) + w_{k0}\right)$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\}, \ \boldsymbol{x}_j \in R^{d+1}, y_j \in R$
  - We need to tell what is the goal of the learning.
    - → Objective function: $\min E = \frac{1}{2} \sum_{j=1}^{N} \left(g(\boldsymbol{x}_j) - y_j\right)^2$
  - We need to tell it how to learn.
    - → Learning algorithm: $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \rho_k \nabla E$ via the BP algorithm

Basic concepts of ML: Perceptron

- How can we make a learning machine?
  - It needs a teacher.
    - → The model: $y = \mathrm{sgn}(\sum_{i=1}^{d} w_i x_i + w_0)$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\},\ \boldsymbol{x}_j \in R^{d+1}, y_j \in \{-1,1\}$
  - We need to tell what is the goal of the learning.
    - → Objective function: $\min J_P(\boldsymbol{\alpha}) = \sum_{\mathbf{y}_j \in \mathcal{Y}^k} (-\boldsymbol{\alpha}^T \mathbf{y}_j)$
  - We need to tell it how to learn.
    - → Learning algorithm: $\boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) - \rho_k \nabla J = \boldsymbol{\alpha}(k) + \rho_k \sum_{\mathbf{y}_j \in Y^k} \mathbf{y}_j$

Basic concepts of ML: Logistic Regression

- How can we make a learning machine?
  - It needs a teacher.
    - → The model: $h(\boldsymbol{x}) = \theta(\boldsymbol{w}^T \boldsymbol{x})$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\},\ \boldsymbol{x}_j \in R^{d+1}, y_j \in \{-1,1\}$
  - We need to tell what is the goal of the learning.
    - → Objective function: $\min E(\boldsymbol{w}) = \frac{1}{N} \sum_{j=1}^{N} \ln\left(1 + e^{-y_j \boldsymbol{w}^T \boldsymbol{x}_j}\right)$
  - We need to tell it how to learn.
    - → Learning algorithm: $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \rho_k \nabla E$

Basic concepts of ML: Linear Regression

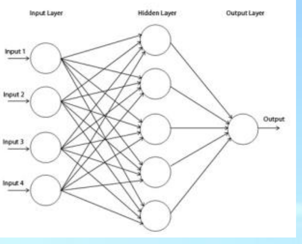
- How can we make a learning machine?
  - It needs a teacher.
    - → The model: $f(\boldsymbol{x}) = \sum_{i=0}^{d} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\},\ \boldsymbol{x}_j \in R^{d+1}, y_j \in R$
  - We need to tell what is the goal of the learning.
    - → Objective function: $\min E = \frac{1}{N} \sum_{j=1}^{N} \left(f(\boldsymbol{x}_j) - y_j\right)^2$
  - We need to tell it how to learn.
    - → Learning algorithm: $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \rho_k \nabla E$

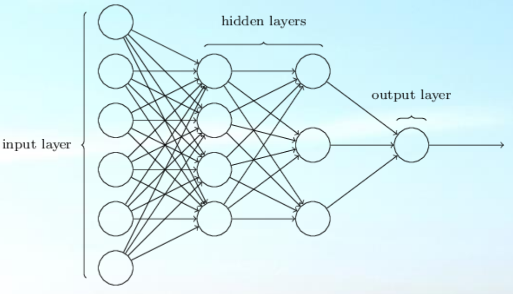
Basic concepts of ML: MLP

- How can we make a learning machine?
  - It needs a teacher.
    - → The model: $g(\mathbf{x}) = f\left(\sum_j w_{jk} f\left(\sum_i w_{ij} x_i + w_{j0}\right) + w_{k0}\right)$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\},\ \boldsymbol{x}_j \in R^{d+1}, y_j \in R$
  - We need to tell what is the goal of the learning.
    - → Objective function: $\min E = \frac{1}{2} \sum_{j=1}^{N} \left(g(\boldsymbol{x}_j) - y_j\right)^2$
  - We need to tell it how to learn.
    - → Learning algorithm: $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \rho_k \nabla E$ via the BP algorithm

What can MLP do?

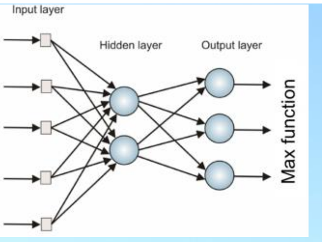
- As a general nonlinear classifier
  - Coding class labels
    - Two-class: 0 vs. 1
  - Network structure
    - 3-layer MLP (with 1 hidden layer)
      - any convex decision regions
    - 4-layer MLP (with 2 hidden layers)
      - any connected/disconnected decision regions

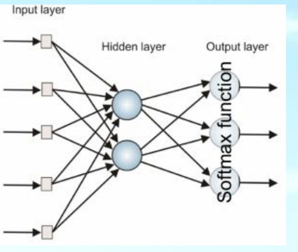




- Multi-class classification with MLP
  - Multi-class:
    - One-hot coding (1-of-C coding)
      e.g., 3 classes coded as: [1,0,0], [0,1,0], [0,0,1]
    - Softmax output

$$y_j = \frac{e^{\boldsymbol{w}_j \cdot \boldsymbol{o}_h}}{\sum_{k=1}^{K} e^{\boldsymbol{w}_k \cdot \boldsymbol{o}_h}}, j = 1, \cdots, K$$



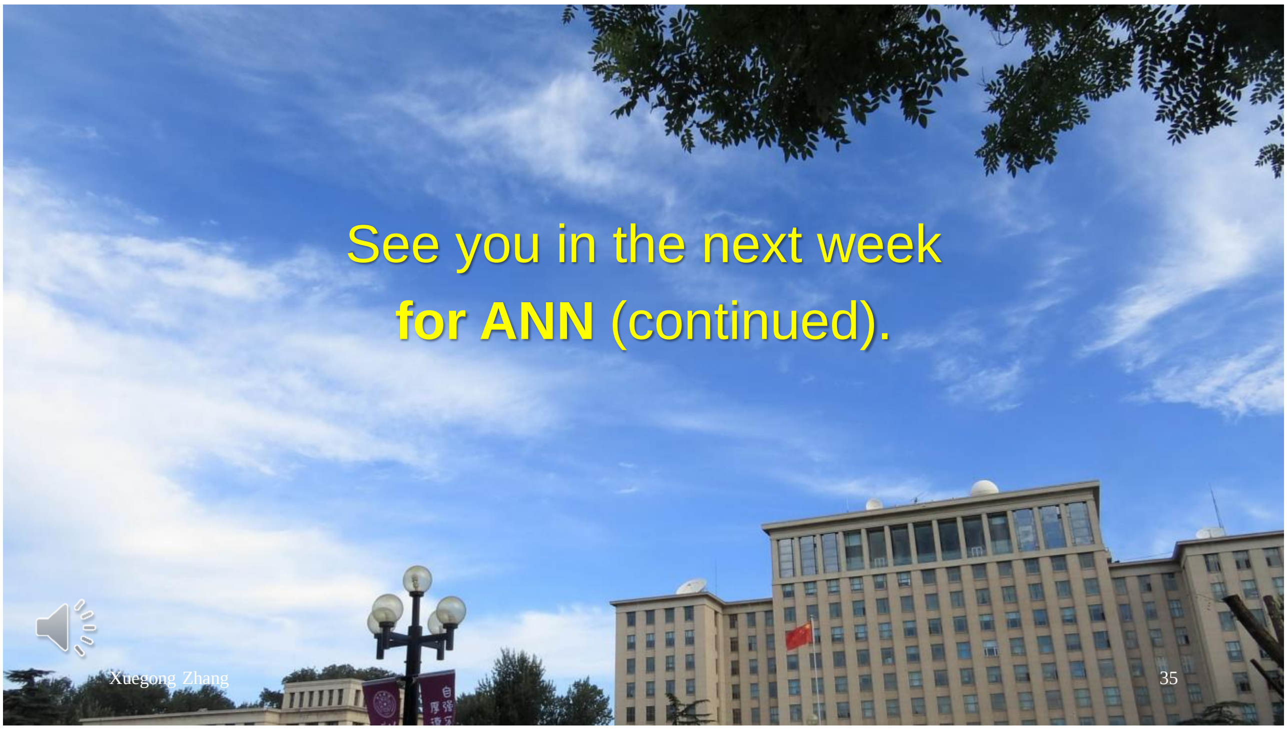


Homework

- Problems (Pr3)
  1. Prove that multilayer neural network with linear activation functions cannot achieve nonlinearity.
  2. Derive the BP algorithms for tanh() activation function, and for SoftMax output function.
- Deadline:
  - Oct. 6 (Wednesday), 23:00

D

- Computer exercises (Ex2)
  - Find a package of KNN
  - Describe its algorithm
  - Code your own MLP program with BP learning.
  - Experiment on the medical dataset
- Deadline:
  - Oct. 13 (Wednesday), 23:00


See you in the next week
for ANN (continued).
Xuegong Zhang
35