COEN281 -- Introduction to Pattern Recognition and Data Mining

Lecture 7: Neural Networks

Instructor: Dr. Giovanni Seni
GSeni@scu.edu

Department of Computer Engineering
Santa Clara University

Fall/18

Syllabus

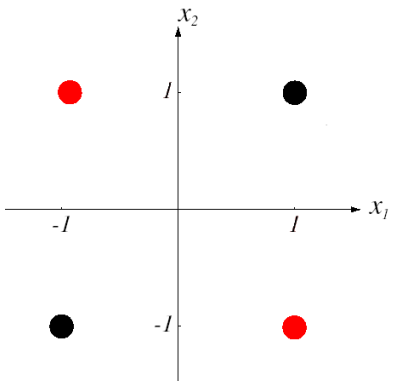
Week 1	Introduction; R (Ch.1)
Week 2	Bayesian Decision Theory (Ch.2; DHS: 2.1-2.6, 2.9) Parameter Estimation (DHS: 3.1-3.4)
Week 3	Linear Discriminant Functions (Ch.3&4; DHS: 3.8.2, 5.1-5.8) Regularization (Ch.6; SE: Ch.3)
Week 4	**Neural Networks** (DHS: 6.1-6.6, 6.8); Deep Learning
Week 5	Support Vector Machines (Ch.9)
Week 6	Decision Trees (Ch. 8.1; DHS: 8.3; Ch 2 SE)
Week 7	Ensemble Methods (Ch. 8.2; SE: Ch 4, 5)
Week 8	Clustering (Ch. 10; DHS: 10.6, 10.7) Clustering (DHS: 10.9); How many clusters are there? (DHS: 10.10)
Week 9	Non-metric: Association Rules Collaborative Filtering
Week 10	Text Retrieval; Other topics

Overview

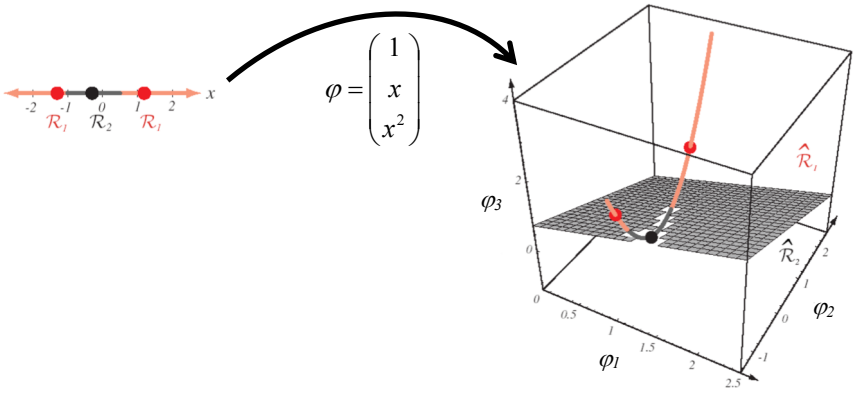
- Introduction
  - Generalized linear discriminant functions
  - Feedforward neural networks
  - Biological motivation
- Feedforward Networks
  - Net and output activation; activation functions
  - Relationship to Logistic Regression
  - Expressive power
- Network Learning
  - Overview
  - *Backpropagation* algorithm

Introduction
Generalized Linear Discriminant

- There are many problems for which linear discriminants are insufficient
  - XOR



Neural Networks
Generalized Linear Models (2)



- Idea: use functions $\varphi(\mathbf{x})$ to map points in $\mathbf{x}$-space to points in φ-space where a linear function suffices

$$g(\mathbf{x}) = \sum_{i=1}^{\hat{d}} w_i \varphi_i(\mathbf{x})$$

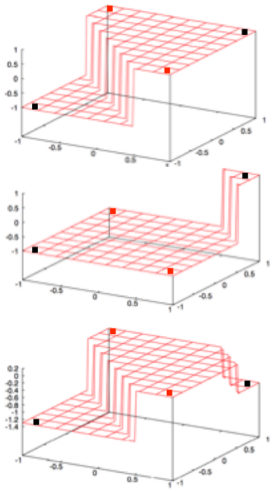
Neural Networks
Generalized Linear Models (3)

- NN idea - employ multiple copies of a single nonlinear function$^{\mp}$

$$y_1 = \varphi_1(\mathbf{x}) = \begin{cases} 1 & \text{if } (x_1 + x_2 + 0.5) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$y_2 = \varphi_2(\mathbf{x}) = \begin{cases} 1 & \text{if } (x_1 + x_2 - 1.5) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

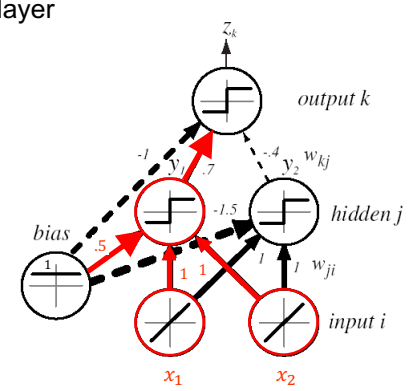
$$g(\mathbf{x}) = 0.7 \times y_1 - 0.4 \times y_2 - 1$$



$\mp$: $\varphi(\mathbf{x})$ = *sign* function

Introduction
Feedforward Neural Networks

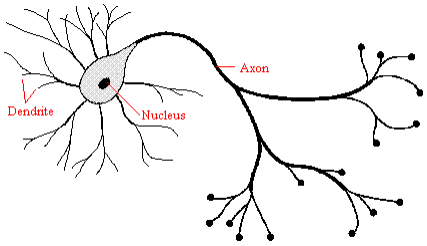
- Each φ() function is associated with a "unit" in a directed graph
  - units are arranged into "layers" – e.g., input layer, hidden layer, and output layer



$$\boldsymbol{w} = \begin{bmatrix} w_{10} = 0.5 \\ w_{11} = 1 \\ w_{12} = 1 \\ w_{20} = -1.5 \\ \dots \end{bmatrix}$$

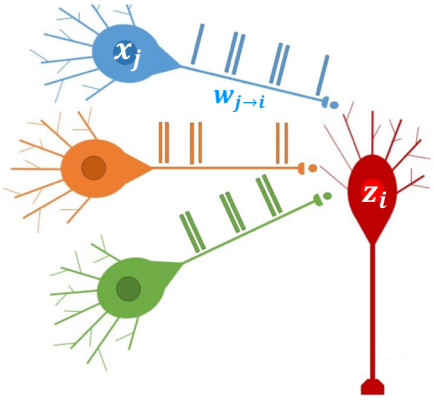
Neural Networks
Biological Inspiration

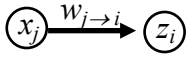
- Human brain contains ~100 billion neurons, each connected to ~10,000 other cells



  - Cell body (CPU): combines signals
  - Dendrite: input bus
  - Axon (cable): will transport the activation signal to neurons at different locations

Neural Networks
Simple Computation





$$x_j = \begin{cases} 1 & \text{neuron fires} \\ 0 & \text{else} \end{cases}$$

$$w_{j\to i} = \begin{cases} > 0 & \text{stimuli} \\ < 0 & \text{inhibit} \end{cases}$$

$$z_i = \begin{cases} 1 & \sum w_{j\to i} x_j > w_{j0} \\ 0 & \text{else} \end{cases}$$

Feedforward Network
Net and Output Activation

- Hidden layer:

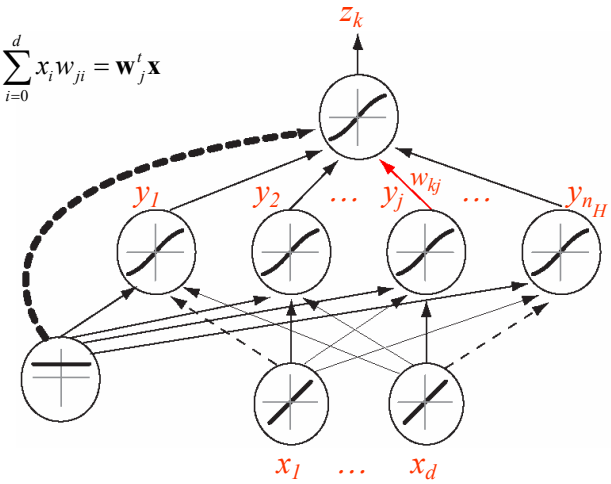
$$net_j = \sum_{i=1}^{d} x_i w_{ji} + w_{j0} = \sum_{i=0}^{d} x_i w_{ji} = \mathbf{w}_j^t \mathbf{x}$$

$$y_j = f(net_j)$$

- Output layer:

$$net_k = \sum_{j=0}^{n_H} y_j w_{kj} = \mathbf{w}_k^t \mathbf{y}$$

$$z_k = f(net_k)$$



Feedforward Network

Activation Function $f() = \varphi()$

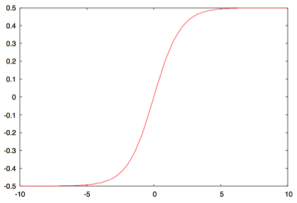
- Required characteristics
  - Nonlinear
  - Continuous and differentiable
- Desirable
  - Saturates
    - Keeps the weights and activations bounded
  - Monotonic
    - Avoids additional and unnecessary local extrema in error surface
  - Linear for small values of *net*
    - Allows linear model if appropriate

Feedforward Network

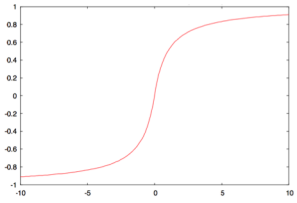
Activation Function (2)

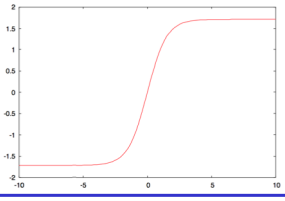
- Centered on zero and anti-symmetric preferred for "faster" learning

$$f(net) = \frac{1}{1+e^{-net}} - 0.5$$



$$f(net) = \frac{net}{1+|net|}$$





$$f(net) = 1.716 \times \tanh(\frac{2}{3} \times net)$$

Feedforward Network
Relationship to Logistic Regression

- Can view NN as a "recursive" logistic regression
  - Each of the predictors in our logistic regression is itself a logisitc regression on the data

$$P(\omega_1 \mid \mathbf{x}) = \phi(\boldsymbol{\beta}^t \mathbf{x}) \quad \Rightarrow \quad P(\omega_1 \mid \mathbf{x}) = \phi(\boldsymbol{\beta}^t \boldsymbol{\Theta}(\mathbf{x}))$$

where $\Theta_1(\mathbf{x}) = \phi(\boldsymbol{\beta}_1^t \mathbf{x})$

$\cdots$

$\Theta_{n_H}(\mathbf{x}) = \phi(\boldsymbol{\beta}_{n_H}^t \mathbf{x})$

and

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

Feedforward Network
Expressive Power

- Class of functions computed by three-layer NN

$$g_k(x) \equiv z_k = f\left(\sum_{j=1}^{n_H} w_{kj} f\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$$

- Can every discriminant function be implemented by a network as described by this equation?
  - Yes (due to A. Kolmogorov). Any continuous function $g(\mathbf{x})$ defined on the unit hypercube I^n ($I=[0,1]$ and $n \geq 2$) can be represented in the form:

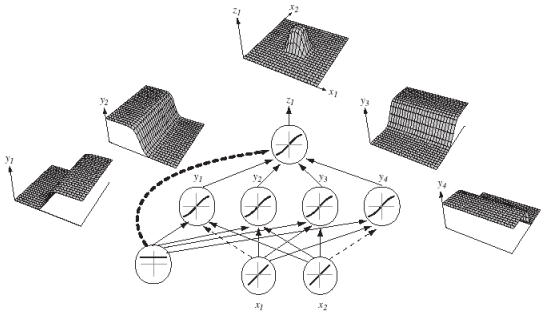
$$g(\mathbf{x}) = \sum_{j=1}^{2n+1} \Xi_j \left(\sum_{i=1}^{d} \Psi_{ij}(x_i)\right)$$

  for properly chosen functions Ξ_j and Ψ_{ij}

Feedforward Network

Expressive Power (2)

- Kolmogorov's Th. is of little practical use
  - Ξ_j and Ψ_{ij} can be extremely complex and not smooth
  - Doesn't tell us how to find the nonlinearities based on the data
- A more intuitive (Fourier-like) proof



Network Learning

Overview

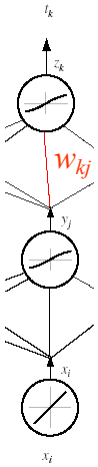
- Goal: set weights w_{kj} and w_{ji} based on the training patterns and the desired outputs
- Supervised training: for every input pattern there is a target vector ***t***
  - E.g., $t_k(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \omega_k \\ 0 & \text{otherwise} \end{cases}$
- Criterion function (on a single pattern)

$$J(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 = \frac{1}{2}\|\mathbf{t} - \mathbf{z}\|^2$$

- Gradient descent step: $\mathbf{w}(k+1) = \mathbf{w}(k) - \eta \nabla J(\mathbf{w})$

Network Learning
Backpropagation

- Hidden-to-output weights w_{kj}



$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial net_k}.\frac{\partial net_k}{\partial w_{kj}} = -\delta_k \frac{\partial net_k}{\partial w_{kj}}$$

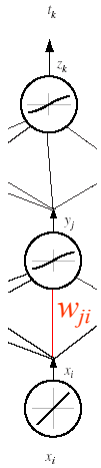
$$\delta_k = -\frac{\partial J}{\partial net_k} = -\frac{\partial J}{\partial z_k}.\frac{\partial z_k}{\partial net_k} = (t_k - z_k) f'(net_k)$$

$$\frac{\partial net_k}{\partial w_{kj}} = y_j$$

$$\Delta w_{kj} = -\eta \frac{\partial J}{\partial w_{kj}} = \eta \delta_k y_j = \eta (t_k - z_k) f'(net_k) y_j$$

Network Learning
Backpropagation (2)

- Input-to-hidden weights w_{ji}



$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}.\frac{\partial y_j}{\partial net_j}.\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}.f'(net_j).x_i$$

$$\frac{\partial J}{\partial y_j} = \frac{\partial}{\partial y_j}\left[\frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2\right] = -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial y_j}$$

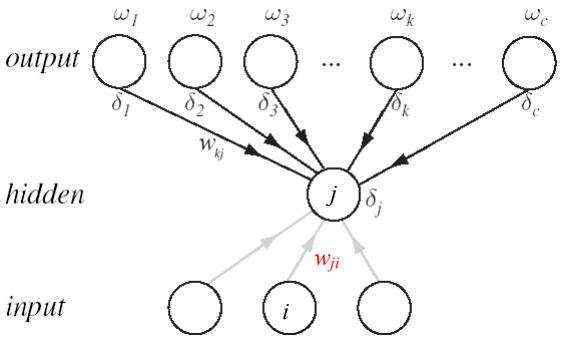
$$= -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial net_k}\frac{\partial net_k}{\partial y_j} = -\sum_{k=1}^{c}\underbrace{(t_k - z_k) f'(net_k)}_{\delta_k} w_{kj}$$

$$\text{define} \quad \delta_j = \frac{\partial J}{\partial y_j}.f'(net_j) = f'(net_j)\sum_{k=1}^{c} w_{kj}\delta_k$$

Network Learning
Backpropagation (3)

- Input-to-hidden weights w_{ji}

$$\Delta w_{ji} = -\eta \frac{\partial J}{\partial w_{ji}} = \eta \delta_j x_i = \eta \left[\sum_{k=1}^{c} w_{kj} \delta_k \right] f'(net_j) x_i$$



Network Learning
Backpropagation (4)

- Initialization
  - Small random non-zero values… why?
- Stochastic version

Initialize $\{n_H, \mathbf{w}, \theta, \eta, k \leftarrow 0\}$
do {
  $k \leftarrow k+1$
  $\mathbf{x}^k \leftarrow$ randomly chosen pattern
  $w_{kj} \leftarrow w_{kj} + \eta \delta_k y_j$
  $w_{ji} \leftarrow w_{ji} + \eta \delta_j x_i$
} **until** ($\|\nabla J(\mathbf{w})\| < \theta$)

- Batch version: $J = \sum_{p=1}^{n} J_p$

Network Learning

Backpropagation (5)

- Stopping criterion
  - stop training at a minimum of the error on the test or validation set