Data Mining

Neural Network
and its applications

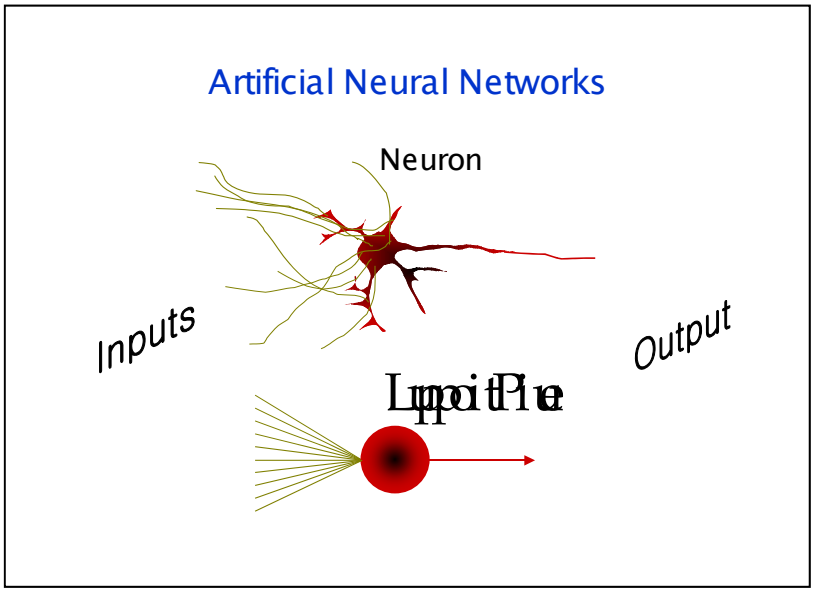
최 대 우

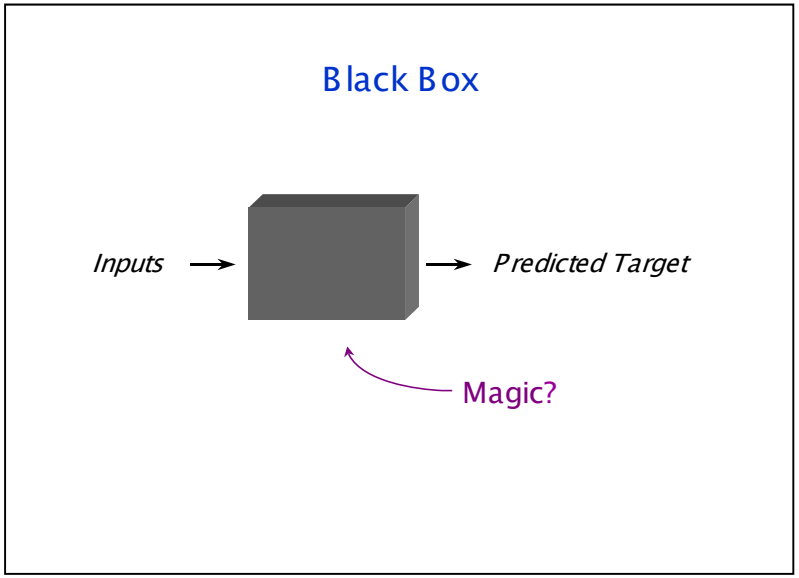
한국외국어대학교 통계학과

Introduction

- Neural Network은 인간 두뇌의 신경망을 흉내 내어 확보한 데이터로부터 반복적인 학습 과정을 거쳐 데이터에 숨어 있는 패턴을 찾아내는 모델링 기법이다.
- Neural Network은 매우 복잡한 구조를 가진 방대한 데이터 사이의 연관관계나 패턴을 찾아내고 이를 이용하여 향후를 예측하는 경우에 유용하다.
- 신경망은 패턴인식분야에서 발전되어 왔다. 다양한 형태의 신경망이 있으나 (예를 들어, bayesian network 등) 가장 흔히 쓰이는 multilayer perceptron (MLP) (또는 feed forward network)를 고려한다

Illustration

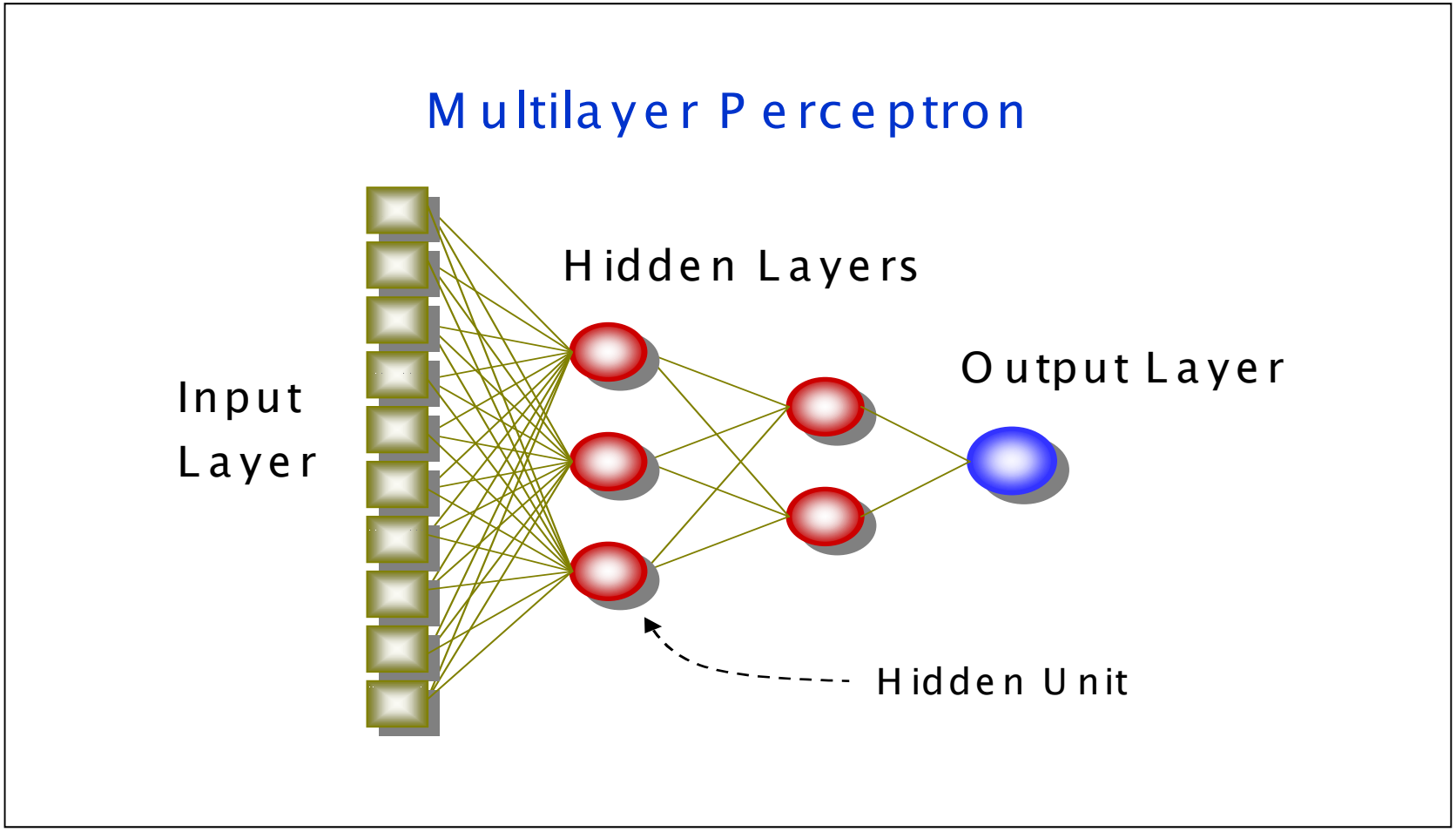




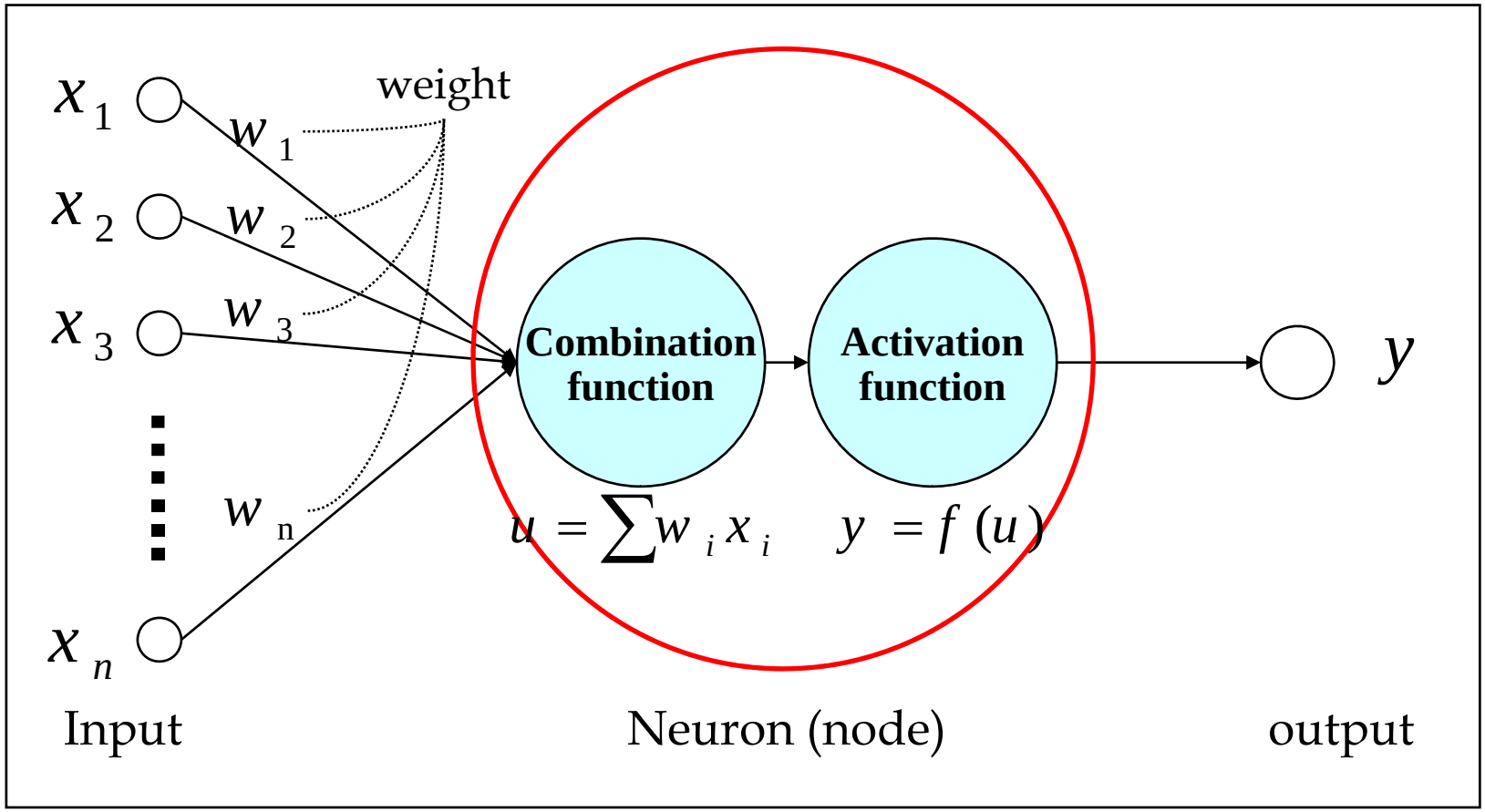
Properties & Usages

- Properties
  - Many Simple processing elements
  - Highly interconnected
  - Parallel processing
  - Activated beyond threshold
  - Adaptive (learning, training)
- Usages
  - Pattern Recognition
  - Classification
  - Clustering
  - Associative Memory
  - Data Compression
  - Combinatorial Optimization

Structure of multilayer Perceptron



Structure of Neural Network



Neural Network의 구성요소

- Input
- Weight
  - 각 input은 해당 node에 입력되며 고유의 weight와 결합됨
- Combination function
  - Weight와 결합된 input들은 combination function에 의해 단일 값을 표현됨
- Activation function
  - Combination function에 의해 단일 값을 표현된 값은 activation function에 의해 scaling 됨
- Output

Training

- NN은 알려진 결과를 가지고 있는 data와 관련된 요소들로부터 시작하여 이 data들의 관계를 modeling하는 것이다 . 이 과정을 network을 "학습(training)한다" 라고 부른다.
- NN에서 combination function과 activation function은 사용자에 의해 주어지고 weight는 target 변수의 예측 값과 실제 값의 차이를 최소화하도록 training 과정에서 추정된다.
- Hidden layer의 수와 hidden layer에서의 node (neuron)의 수를 사용자가 정한다.
- 이렇게 추정된 모형에 의하여 target 변수의 값이 알려지지 않은 새 data가 주어졌을 때 target 변수의 값을 "예측(predict)" 하게 된다.

Combination functions

- Input variable**들과** weight**를 결합하는 함수**
- **대표적인 결합 함수들**
  - Linear

$$bias_j + \sum_i w_{ij} x_i$$

  - Equal slopes

$$bias_j + \sum_i w_i x_i$$

  - Add

$$\sum_i x_i$$

  - Radial

$$-bias_j^2 \sum_i (w_{ij} - x_i)^2$$

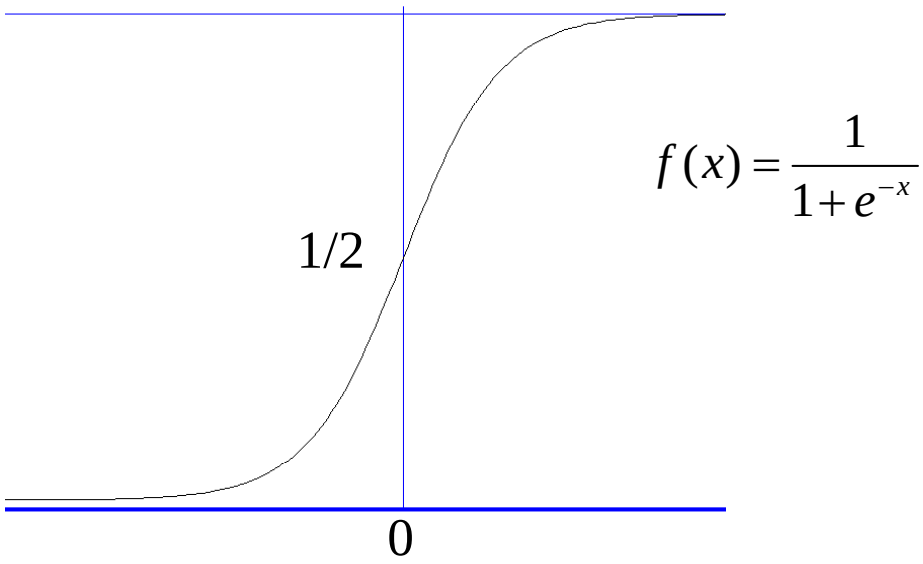
- Bias**는** input **이외의 외부에서 오는 영향력** (intercept)

Activation function

- **결합함수에 의한 단일 값을 일정** range**로** mapping/scaling **해주는 함수**
- **보통** activation function**에 의해 산출된 값**(output value)**은 특정 범위 내의 (** **보통** 0**과** 1**사이의**) **값이다**.

- **대표적인 활성함수들** (activation functions)
  - Logistic (Sigmoid) $\frac{1}{1+e^{-x}}$ range=(0,1)
  - tanh $\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ range=(-1,1)
  - Linear (identity) x range=(-∞,∞)
  - Gauss $e^{-x^2/2}$ range=(0,1)

Logistic (Sigmoid) functions

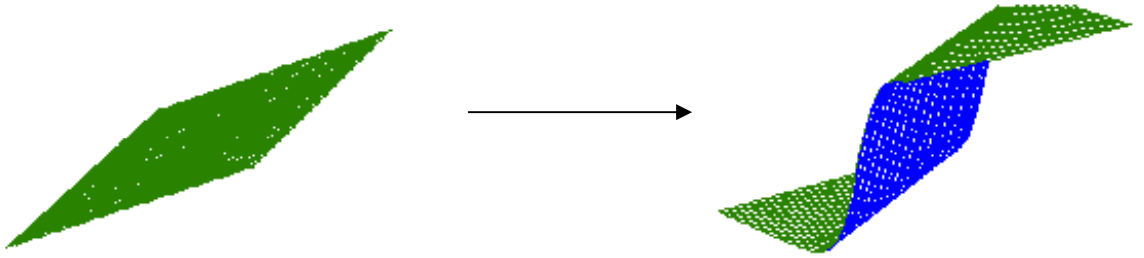


$$f(x) = \frac{1}{1+e^{-x}}$$

- Logistic function은 0 근처의 값들을 구별할 수 있다. 즉, x 값이 작은 범위에서는 input에 대한 작은 변화도 영향이 크다.
- x 값이 아주 크거나 아주 작을 때 input에 대한 작은 weight의 변화는 output에 거의 영향(변화)을 주지 않는다.

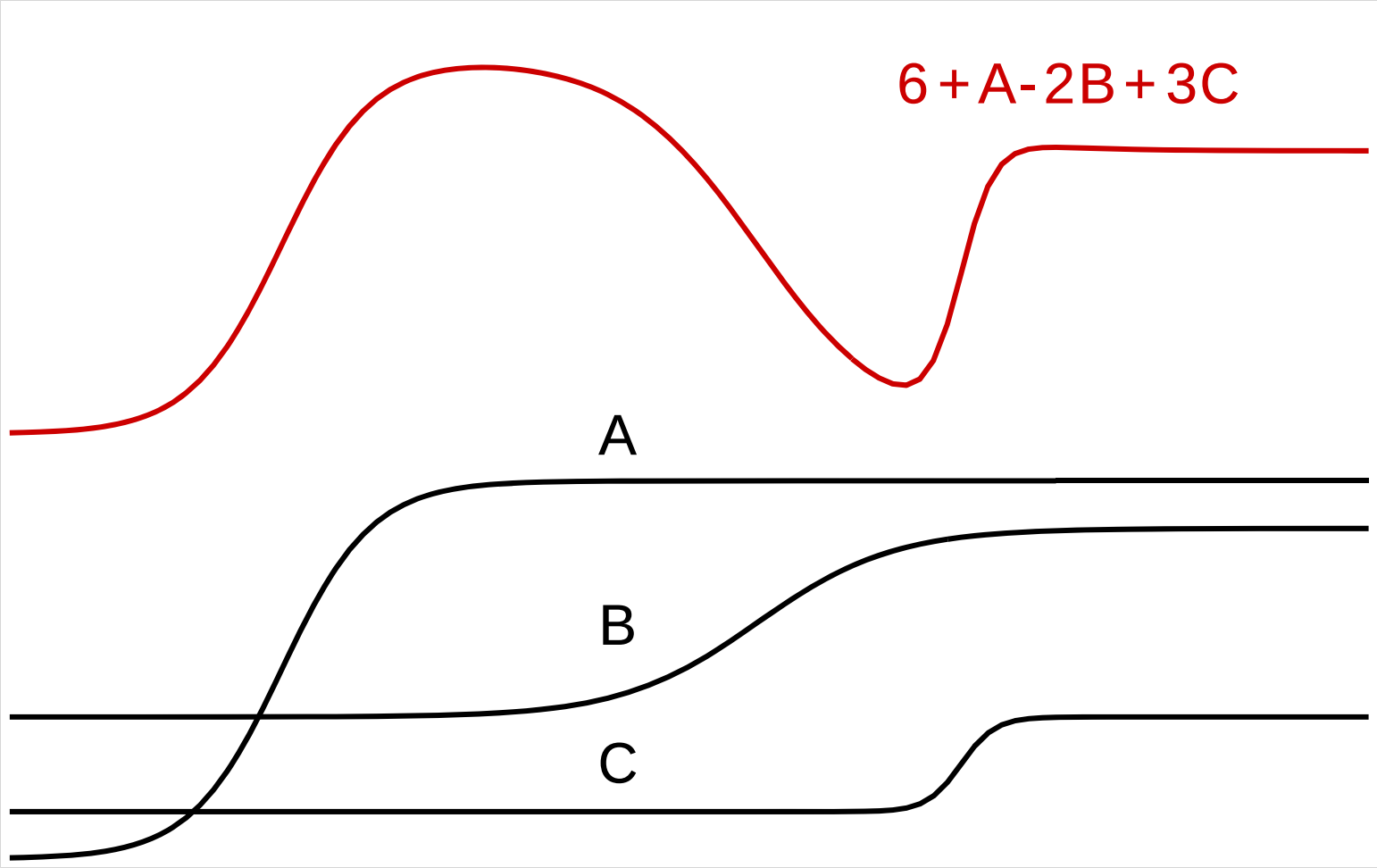
Illustration

(x_1, x_2) ⟶ combination function $bias + w_1 x_1 + w_2 x_2$ ⟶ activation function $\frac{1}{1+e^{-bias-w_1 x_1 - w_2 x_2}}$



입력 값들의 선형결합 출력 : S-자형의 곡면

Universal approximator



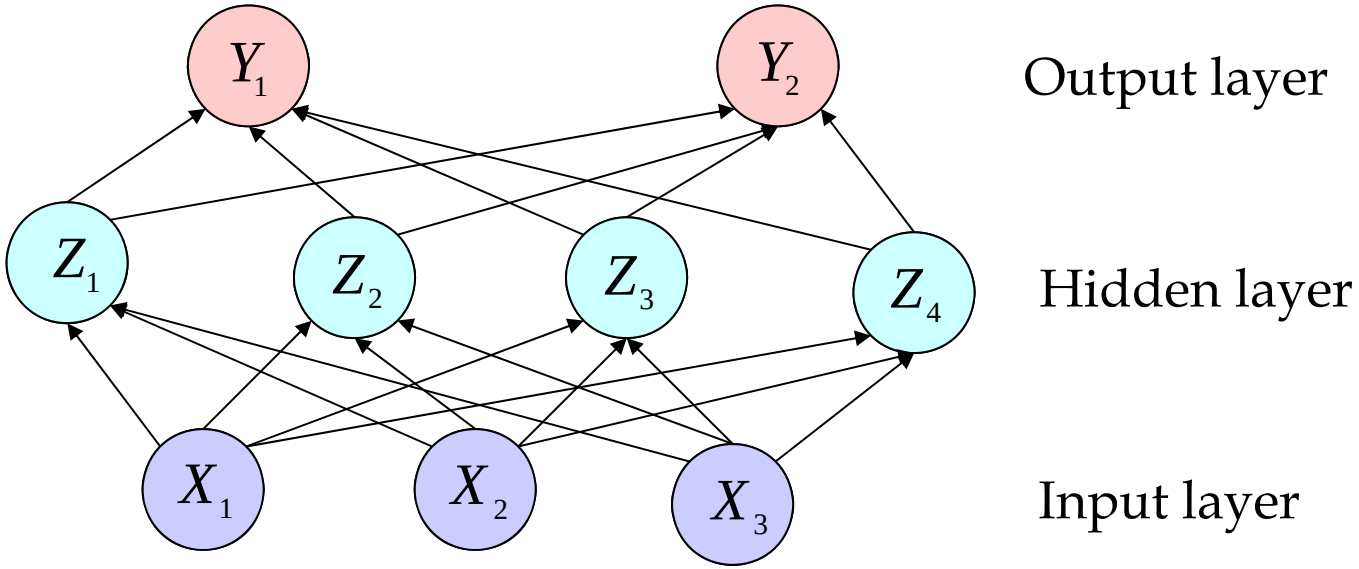
Another representation of neural network

- 아래 그림에서

$$z_j = \sigma(\alpha_{0j} + \alpha_j^T \mathbf{x}), \quad j = 1, \ldots, m = 4$$

$$\hat{y}_k = f_k(\beta_{0k} + \beta_k^T \mathbf{z}), \quad k = 1, \ldots, q = 2$$

여기서 $\sigma(z) = 1/(1+e^{-z})$ (Sigmoid function) 이다.



Another representation of neural network

- For regression

$$f_k(t) = t$$

- For *K*-class classification

$$f_k(\mathbf{t}) = e^{t_k} / \Sigma_l e^{t_l}$$

and each output y_k is a 0-1 variable for the k th class.

Fitting neural networks -1

- Error function $E = \Sigma_{cases} (\hat{y} - y)^2$, entropy for classification
- Generic approach to minimizing E over weights $w = (\beta_j, \alpha_k)$ is *back-propagation*.
- Idea
  - minimize E by moving down its gradient. Because of compositional form of model, gradient can be computed by the chain rule.
  - This can be computed by a forward and backward sweep over the network, keeping track of quantities only local to each unit.
- Advantage
  - Simple
  - efficient on parallel architecture
  - can be carried out online-processing each observation one at a time, updating gradient after each
  - This allows the network to handle very large training sets and also to update parameters as new observation come in

Fitting neural networks -2

- Disadvantage
  - Can be very slow

- Starting values for weights
  - Usually random values near zero are used.
- Regularization
  - Often neural networks have too many parameters (weights) and will overfit the data at the global minimum of E.
  - Two solutions are common:
    - Early stopping (adhoc but effective)
    - Weight decay : adding a ridge-regression style penalty $\lambda \sum w^2$ to E

Back-propagation in detail

$$R(\alpha,\beta)=\sum_{i=1}^{N}\sum_{k=1}^{K}(y_k^i-\hat{y}_k^i)^2=\sum_{i=1}^{N}R^i$$

$$\frac{\partial R^i}{\partial \beta_{kj}}=-2(y_k^i-\hat{y}_k^i)\,f_k'(\beta_k^T\mathbf{z}^i)z_j^i$$

$$\frac{\partial R^i}{\partial \alpha_{jl}}=-\sum_{k=1}^{K}2(y_k^i-\hat{y}_k^i)\,f_k'(\beta_k^T\mathbf{z}^i)\beta_{kj}\sigma'(\alpha_j^T\mathbf{x}^i)x_l^i$$

- Given these derivatives, a gradient update at the *r+1* st iteration has the form

$$\beta_{kj}^{(r+1)}\leftarrow\beta_{kj}^{(r)}-\gamma_r\sum_i\frac{\partial R^i}{\partial \beta_{kj}^{(r)}}$$

$$\alpha_{jl}^{(r+1)}\leftarrow\alpha_{jl}^{(r)}-\gamma_r\sum_i\frac{\partial R^i}{\partial \alpha_{jl}^{(r)}}$$

where γ_r is the learning rate which can change with iteration number *r*.

Local minima

- Neural network은 복잡한 모형이므로 object function의 표현이 매끈하지 않을 가능성이 높다.
- 따라서, 추정과정에서 object function의 global minima 대신local minima를 찾을 가능성이 높다.
- Local minima를 찾지 않기 위해서는 초기값을 설정하는 것이 중요하다.
- 초기값을 찾는 방법
  - Random Search
  - Genetic Algorithm
  - Simulated Annealing

Random search

- 각 모수에 대하여 난수를 발생시켜 모수의 값으로 한다.
- 이 모수의 값에 대하여 object function 값을 계산한다.
- 이렇게 여러 번 object function 값을 계산하여 가장 성능이 좋은 값을 초기값으로 하여 모수를 추정한다.
- 모수의 난수값을 가지고 몇 단계 추정하여 이 값을 비교할 수 있다.
- 난수의 분포로는 보통 균일분포 (uniform distribution)이나 정규분포(normal distribution)이 사용된다.

Q&A