Neural Network Fundamental

Single Layer Perceptron

Piji Li

pjli@se.cuhk.edu.hk

Outline

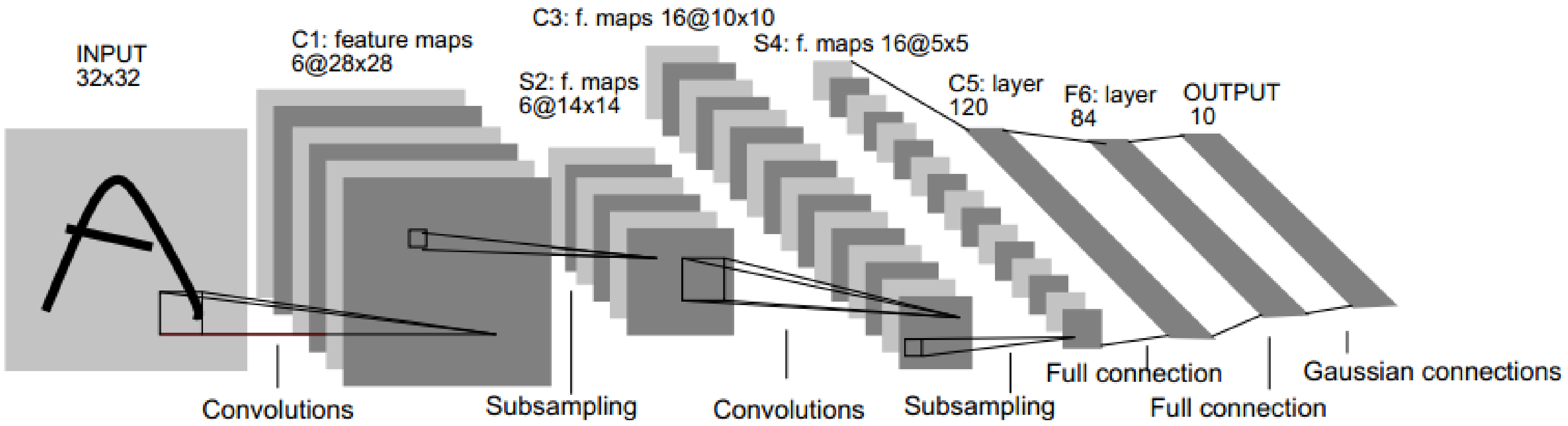
- Introduction
- Structure
- Learning
- Tricks

Outline

- **Introduction**
- Structure
- Learning
- Tricks

Introduction

- **Deep Learning**
  - Nowadays, deep learning techniques have been applied to fields like **computer vision, speech recognition, natural language processing and bioinformatics** where they have been shown to produce **state-of-the-art** results on various tasks.
- **Neural Network**
  - Deep neural networks, convolutional deep neural networks, deep recurrent neural networks.



Outline

- Introduction
- **Structure**
- Learning
- Tricks

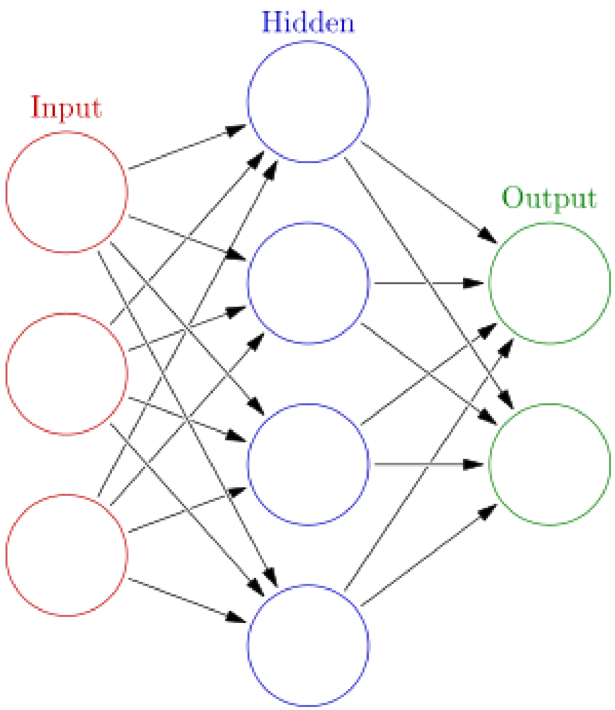
Structure-single layer perceptron

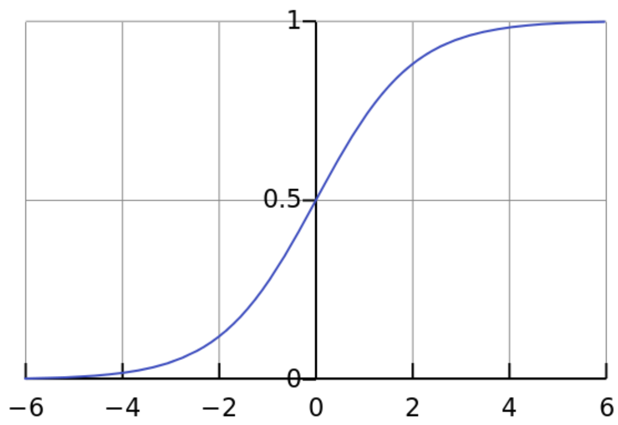
- Single hidden layer FF and BP
- Regression(K = 1)
- Classification(K-class)
- Linear combinations
- Activation Function $\sigma(v)$

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X),\ m = 1, \ldots, M$$
$$T_k = \beta_{0k} + \beta_k^T Z,\ k = 1, \ldots, K$$
$$f_k(X) = g_k(T),\ k = 1, \ldots, K$$
$$\sigma(v) = 1/(1 + e^{-v})$$

$$g_k(T) = T_k$$

$$g_k(T) = \frac{e^{T_k}}{\sum_{\ell=1}^{K} e^{T_\ell}}$$ Softmax function

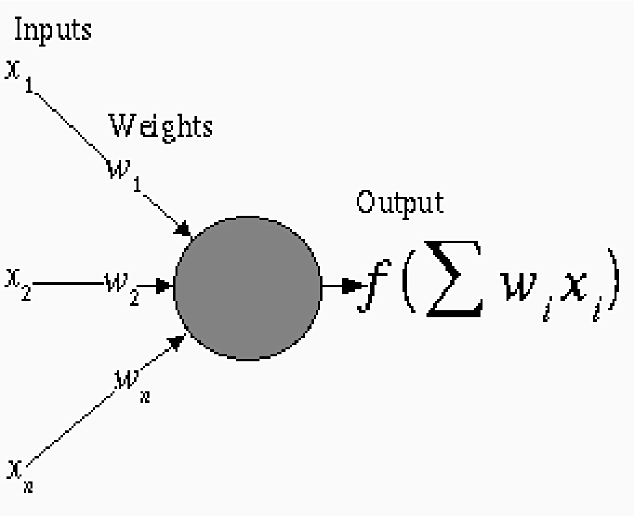




Piji Li

Non-linear Model

- Notice that if σ is the identity function, then the entire model collapses to a linear model in the inputs.
- Hence a neural network can be thought of as a nonlinear generalization of the linear model, both for regression and classification.
- By introducing the nonlinear transformation , it greatly enlarges the class of linear models.



Outline

- Introduction
- Structure
- **Learning**
- Tricks

Parameters

$$\{\alpha_{0m}, \alpha_m;\ m = 1, 2, \ldots, M\}\ \ M(p+1) \text{ weights}$$
$$\{\beta_{0k}, \beta_k;\ k = 1, 2, \ldots, K\}\ \ K(M+1) \text{ weights}$$

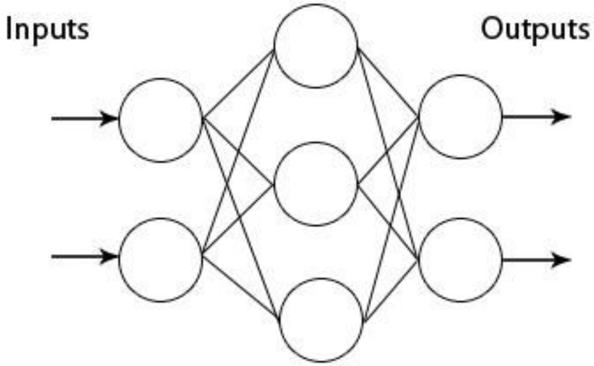
Loss functions:

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2$$

Sum-of-squared errors

$$R(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k(x_i)$$

Cross-entropy



Minimize by GD

Back-Propagation(BP)

Back-Propagation(BP)

1986



D.E. Rumelhart, G.E. Hinton, R.J. Williams
Learning representation by back-propagating errors. *Nature*, 323 (1986), pp. 533–536

- Solved learning problem
- Biological system
- …

- Hard to train (non-convex, tricks)
- Hard to do theoretical analysis
- Small training sets ...

Back-Propagation(BP)

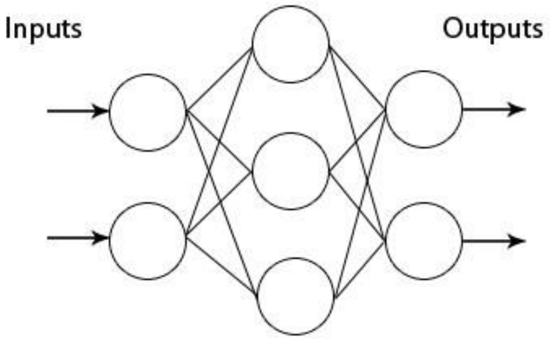
Here is back-propagation in detail for squared error loss:

$$
\begin{aligned}
R(\theta) &\equiv \sum_{i=1}^{N} R_i \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K}(y_{ik} - f_k(x_i))^2
\end{aligned}
$$

$$
\begin{aligned}
Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X),\ m = 1,\dots,M \\
T_k &= \beta_{0k} + \beta_k^T Z,\ k = 1,\dots,K \\
f_k(X) &= g_k(T),\ k = 1,\dots,K
\end{aligned}
$$

Take derivatives:

$$
\begin{aligned}
\frac{\partial R_i}{\partial \beta_{km}} &= -2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)z_{mi}, \\
\frac{\partial R_i}{\partial \alpha_{m\ell}} &= -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{i\ell}.
\end{aligned}
$$



Given these derivatives, a gradient descent update methoc

$$
\begin{aligned}
\beta_{km}^{(r+1)} &= \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}} \\
\alpha_{m\ell}^{(r+1)} &= \alpha_{m\ell}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{m\ell}^{(r)}}
\end{aligned}
$$

Piji Li

Back-Propagation(BP)

Rewriting:

$$\frac{\partial R_i}{\partial \beta_{km}} = \delta_{ki} z_{mi}$$

$$\frac{\partial R_i}{\partial \alpha_{m\ell}} = s_{mi} x_{i\ell}$$

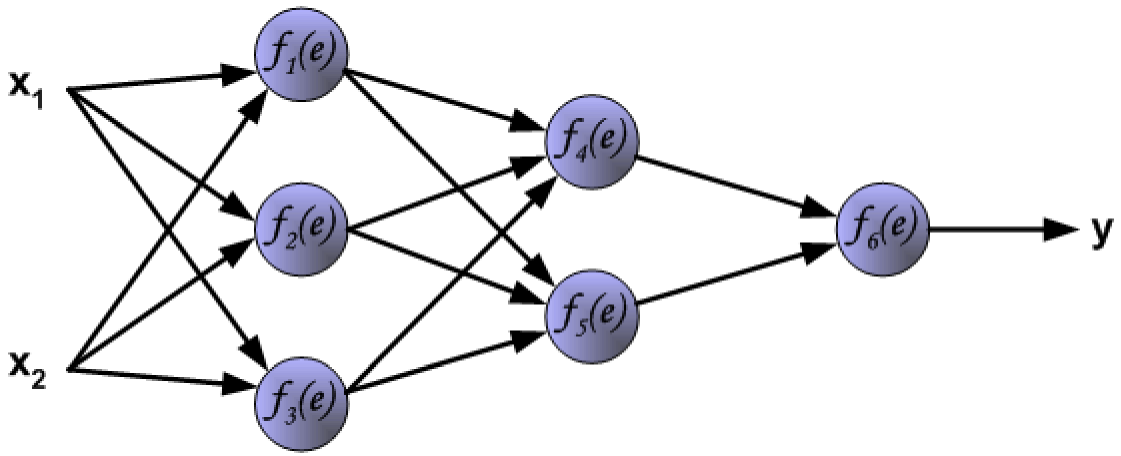
errors and

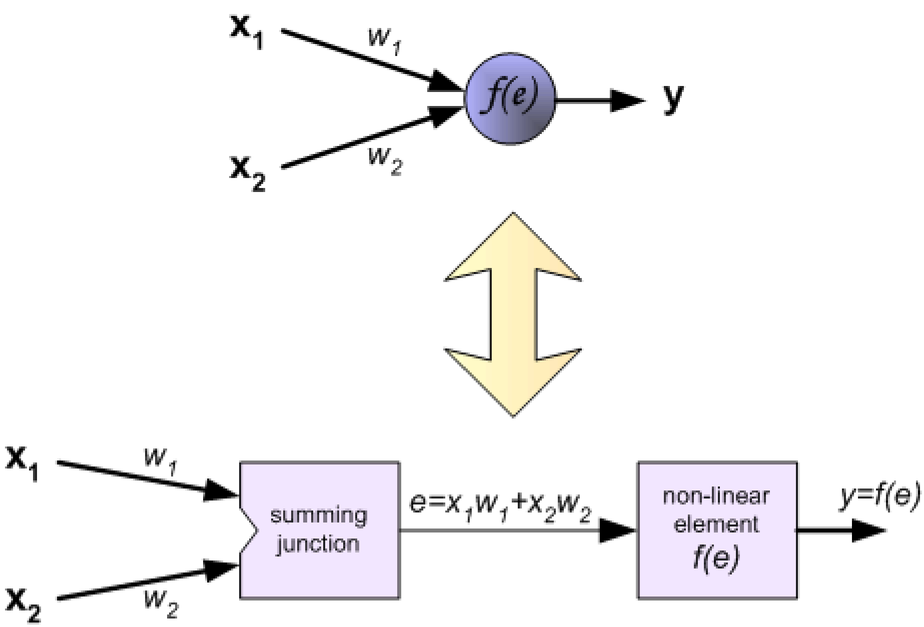
$$s_{mi} = \sigma'(\alpha_m^T x_i) \sum_{k=1}^{K} \beta_{km} \delta_{ki}$$

Two-pass algorithm:

- In the forward pass, the current weights are fixed and the predicted values are computed.
- In the backward pass, the output-layer errors are computed, and then back propagated to the hidden-layer errors.

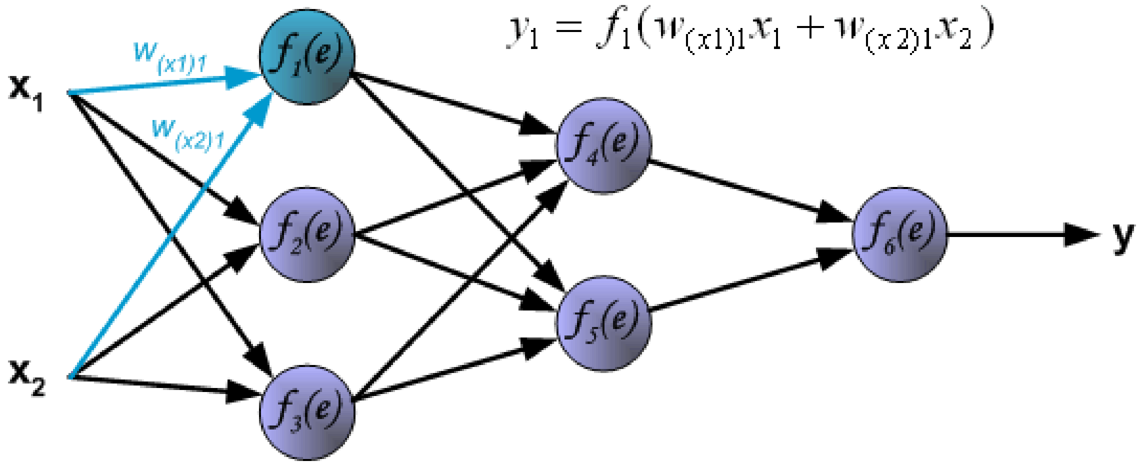
Back-Propagation(BP)-Examples





Back-Propagation(BP)-Examples

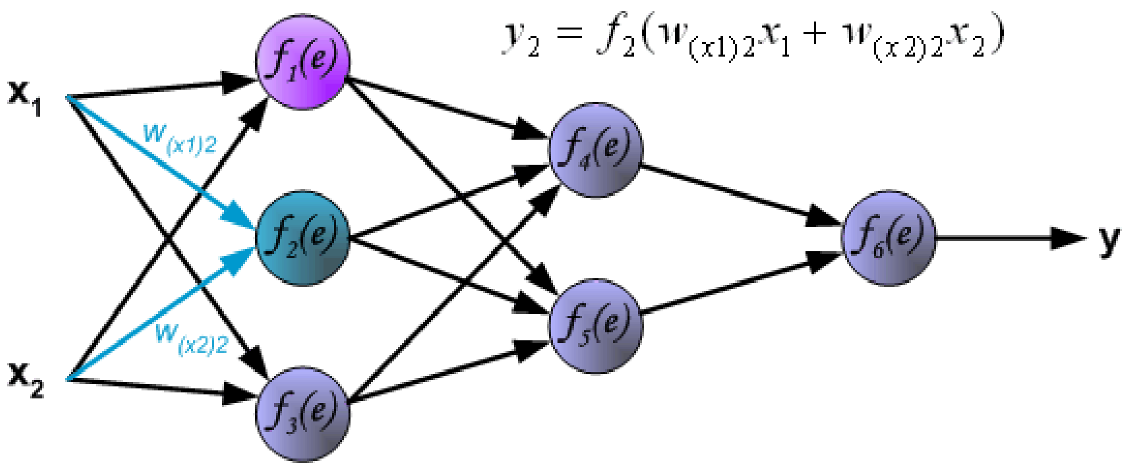
Forward pass:



Piji Li

Back-Propagation(BP)-Examples

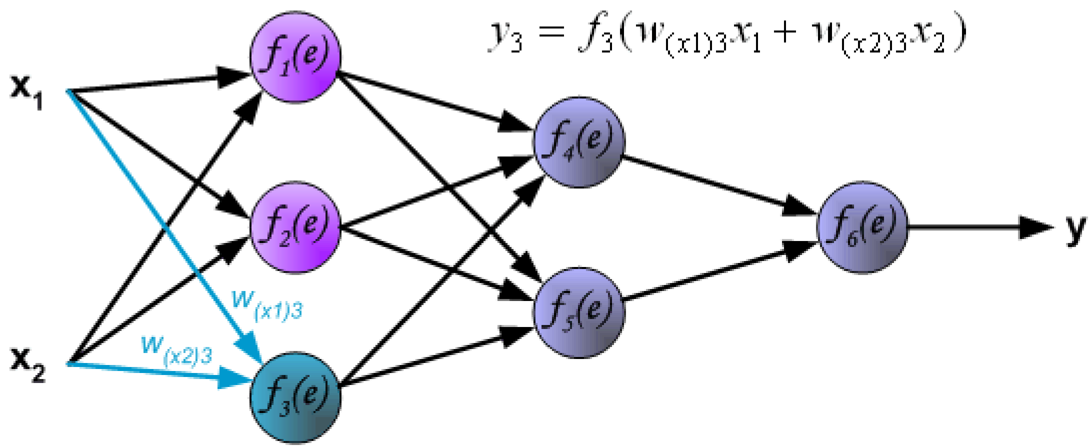
Forward pass:



Piji Li

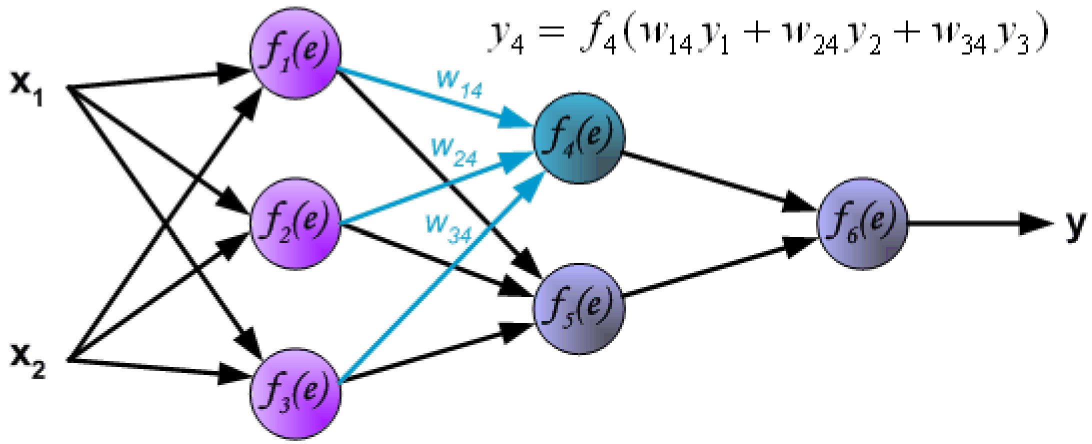
Back-Propagation(BP)-Examples

Forward pass:



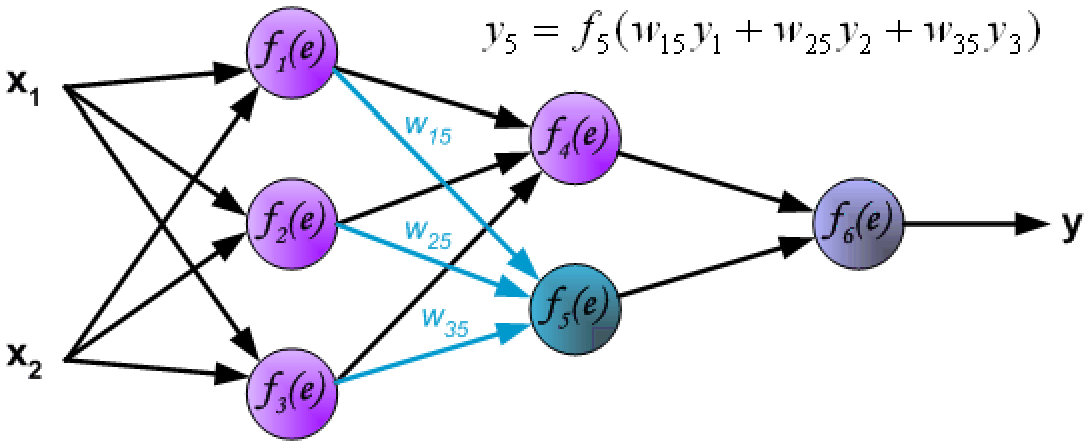
Back-Propagation(BP)-Examples

Forward pass:



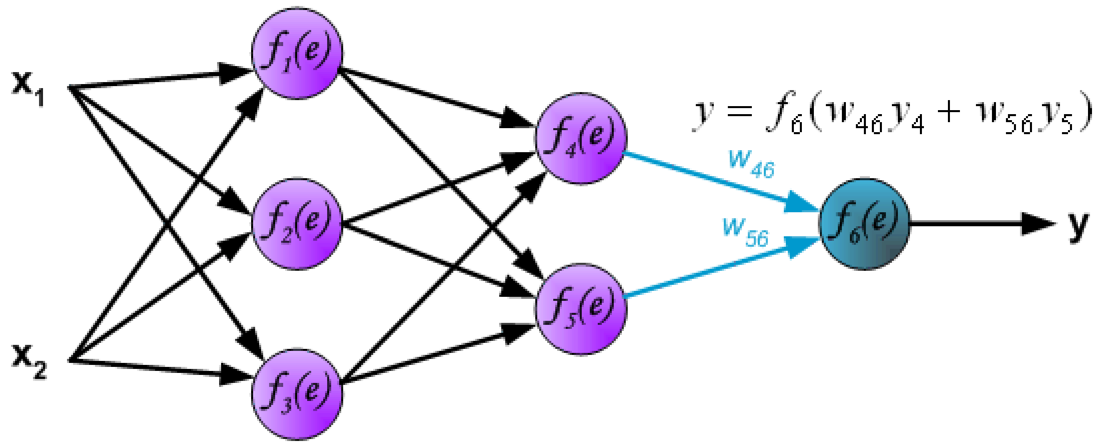
Back-Propagation(BP)-Examples

Forward pass:



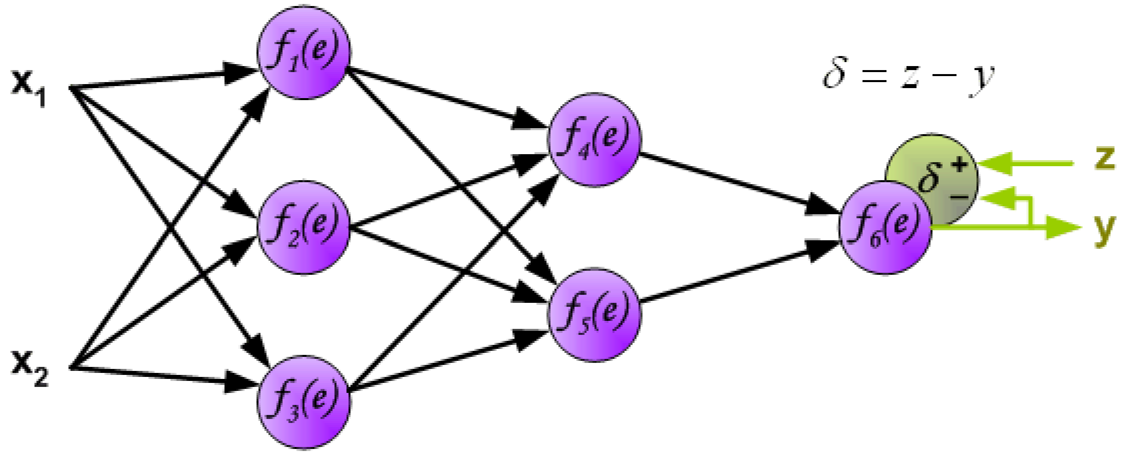
Back-Propagation(BP)-Examples

Forward pass:



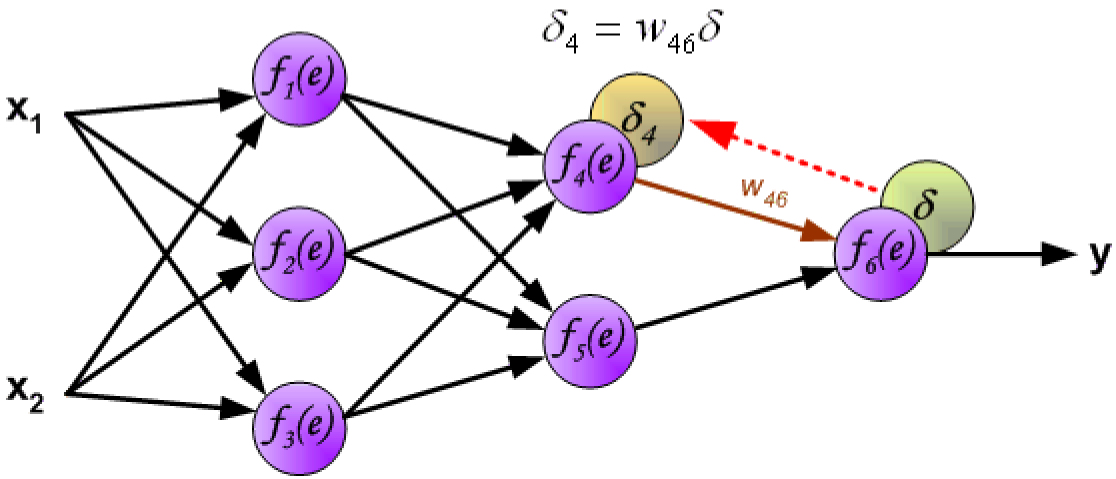
Back-Propagation(BP)-Examples

Backward pass:



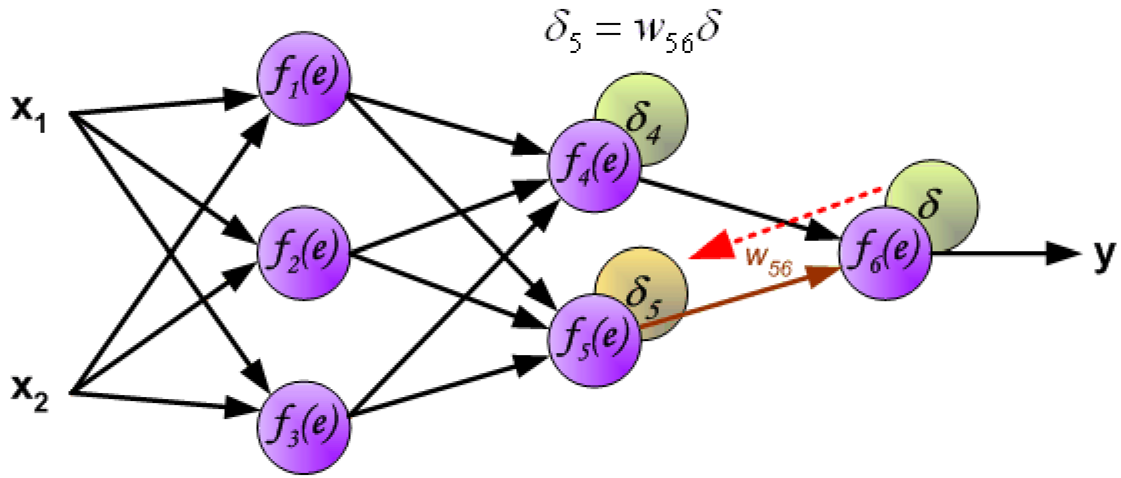
Back-Propagation(BP)-Examples

Backward pass:



Back-Propagation(BP)-Examples

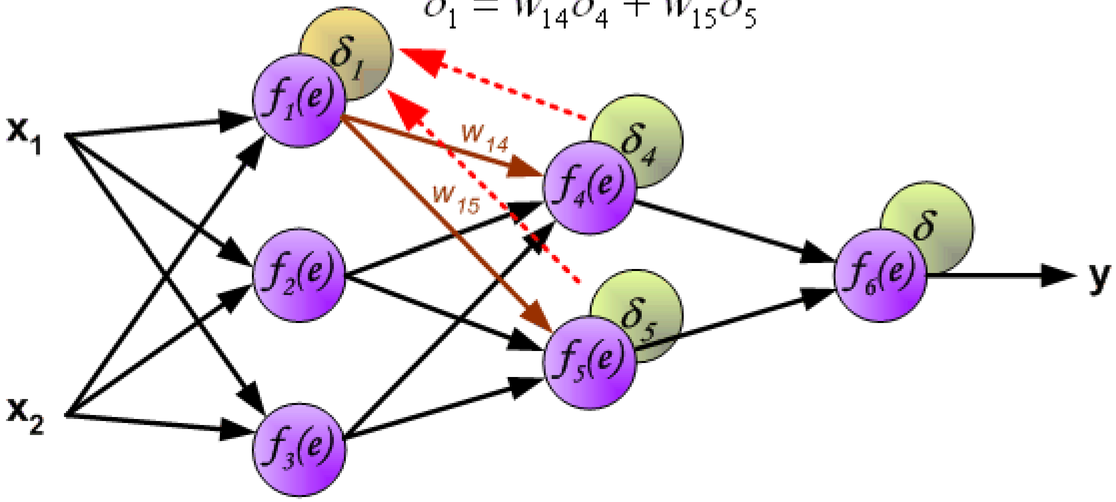
Backward pass:



Back-Propagation(BP)-Examples

Backward pass:

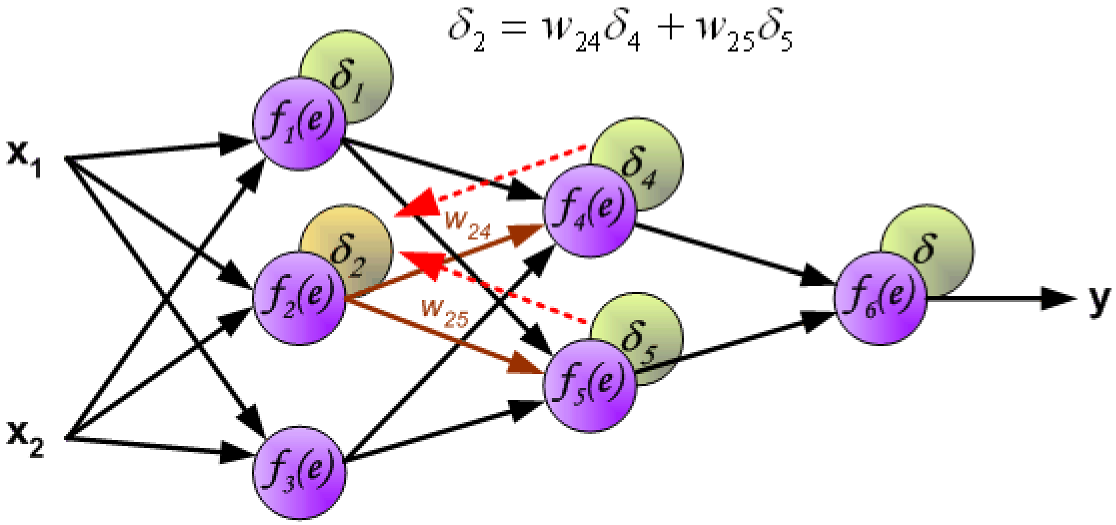
$$\delta_1 = w_{14}\delta_4 + w_{15}\delta_5$$



Piji Li

Back-Propagation(BP)-Examples

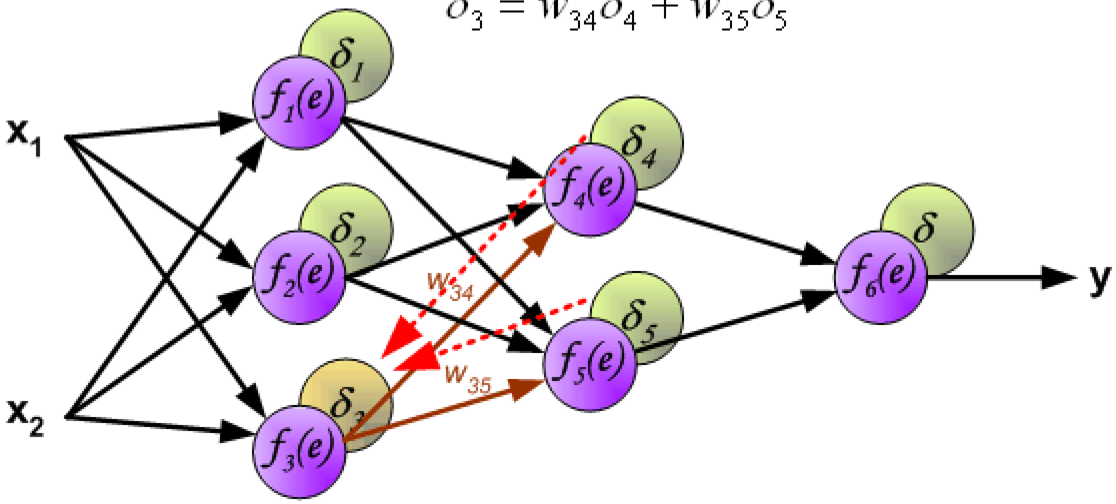
Backward pass:



Back-Propagation(BP)-Examples

Backward pass:

$$\delta_3 = w_{34}\delta_4 + w_{35}\delta_5$$

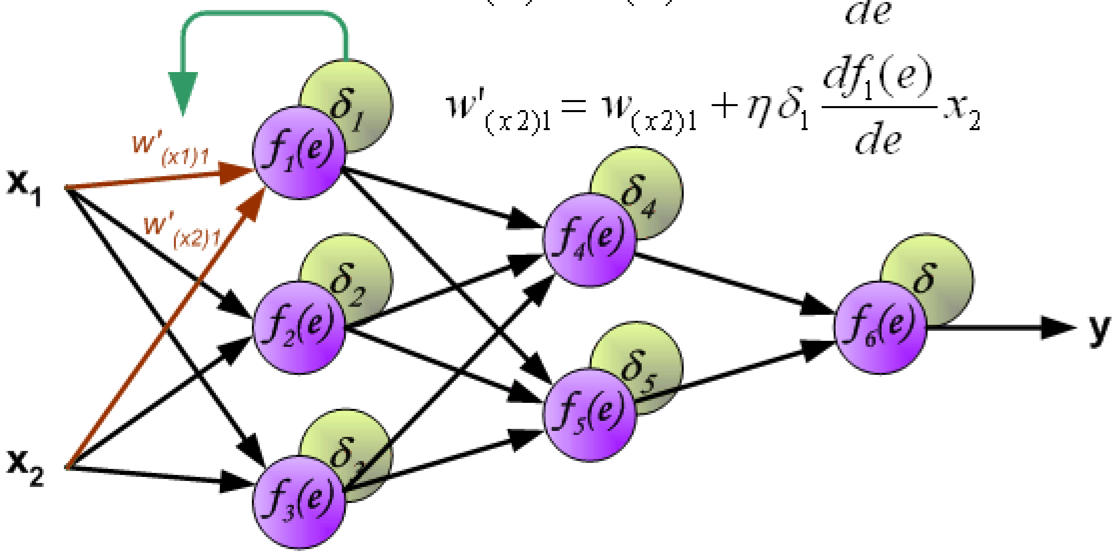


Back-Propagation(BP)-Examples

Update parameters:

$$w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$

$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$

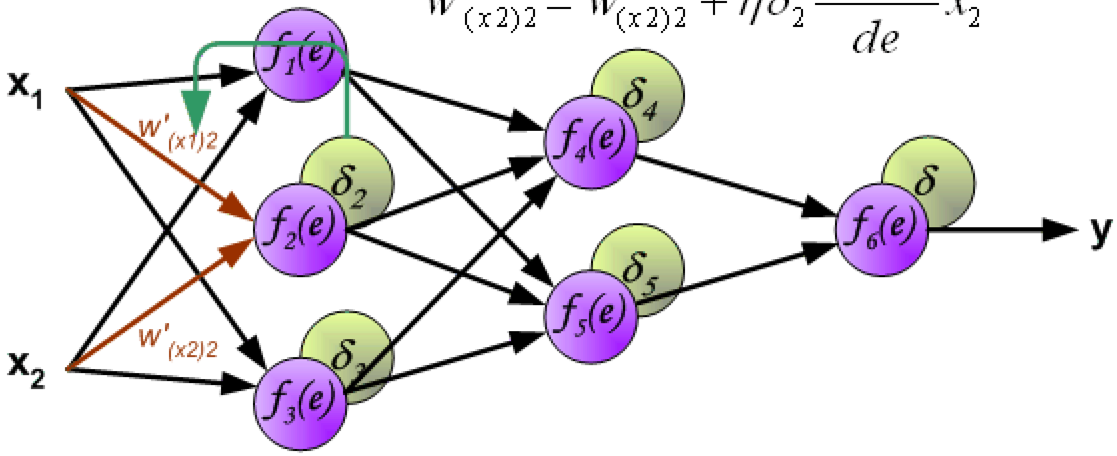


Back-Propagation(BP)-Examples

Update parameters:

$$w'_{(x1)2} = w_{(x1)2} + \eta \delta_2 \frac{df_2(e)}{de} x_1$$

$$w'_{(x2)2} = w_{(x2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2$$

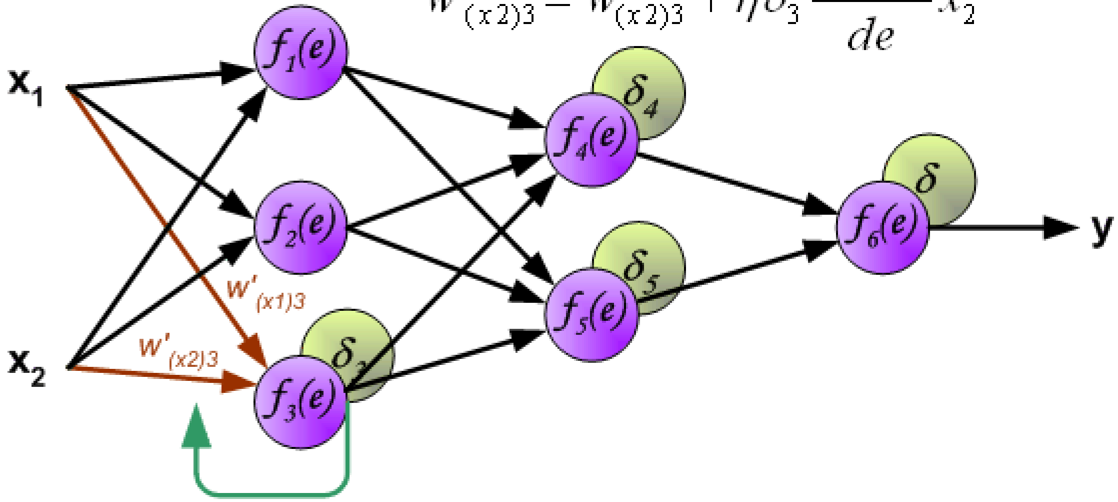


Back-Propagation(BP)-Examples

Update parameters:

$$w'_{(x1)3} = w_{(x1)3} + \eta \delta_3 \frac{df_3(e)}{de} x_1$$

$$w'_{(x2)3} = w_{(x2)3} + \eta \delta_3 \frac{df_3(e)}{de} x_2$$



Piji Li

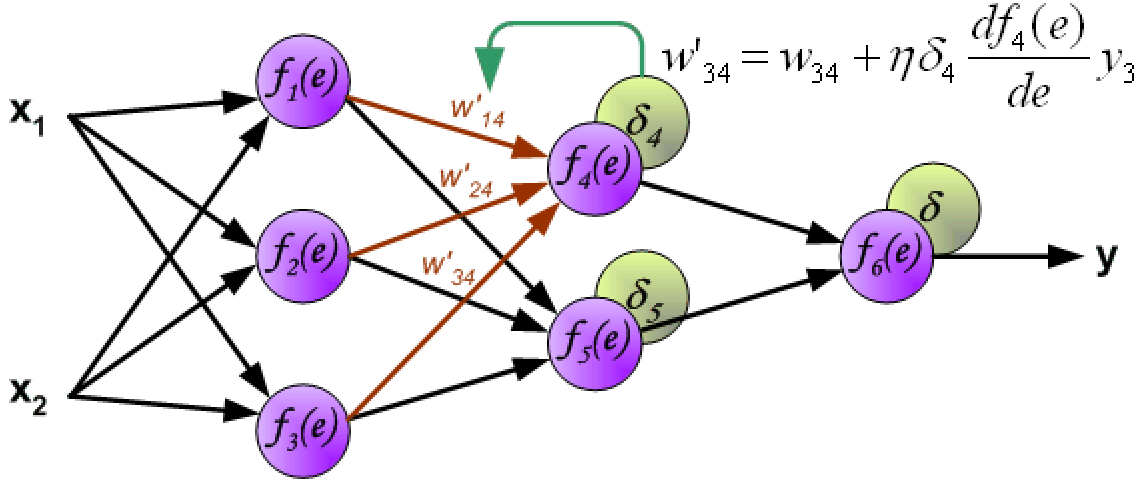
Back-Propagation(BP)-Examples

Update parameters:

$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2$$

$$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$$



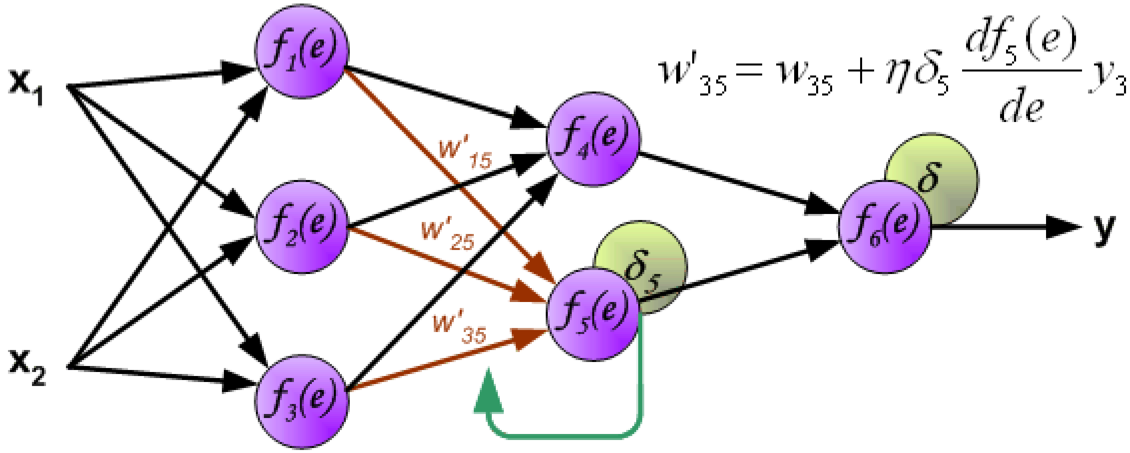
Back-Propagation(BP)-Examples

Update parameters:

$$w'_{15} = w_{15} + \eta \delta_5 \frac{df_5(e)}{de} y_1$$

$$w'_{25} = w_{25} + \eta \delta_5 \frac{df_5(e)}{de} y_2$$

$$w'_{35} = w_{35} + \eta \delta_5 \frac{df_5(e)}{de} y_3$$

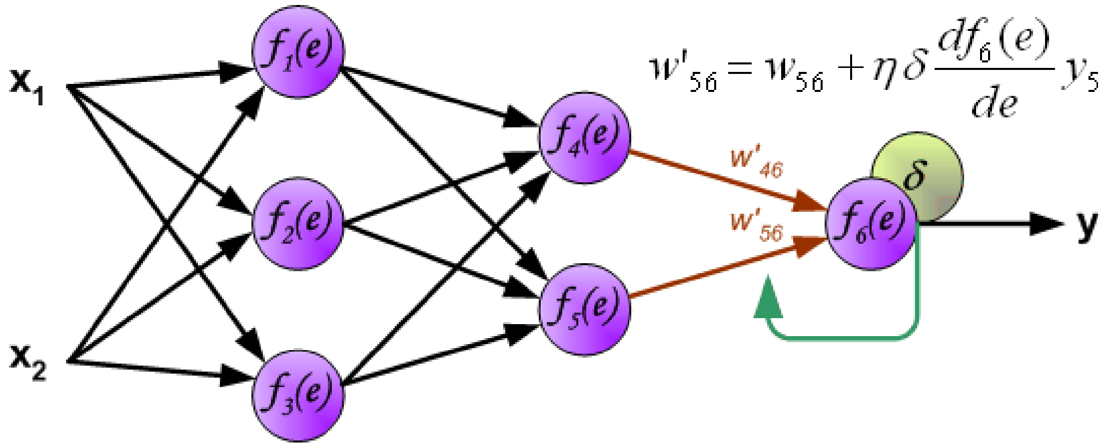


Back-Propagation(BP)-Examples

Update parameters:

$$w'_{46} = w_{46} + \eta\delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta\delta \frac{df_6(e)}{de} y_5$$



Outline

- Introduction
- Structure
- Learning
- **Tricks**

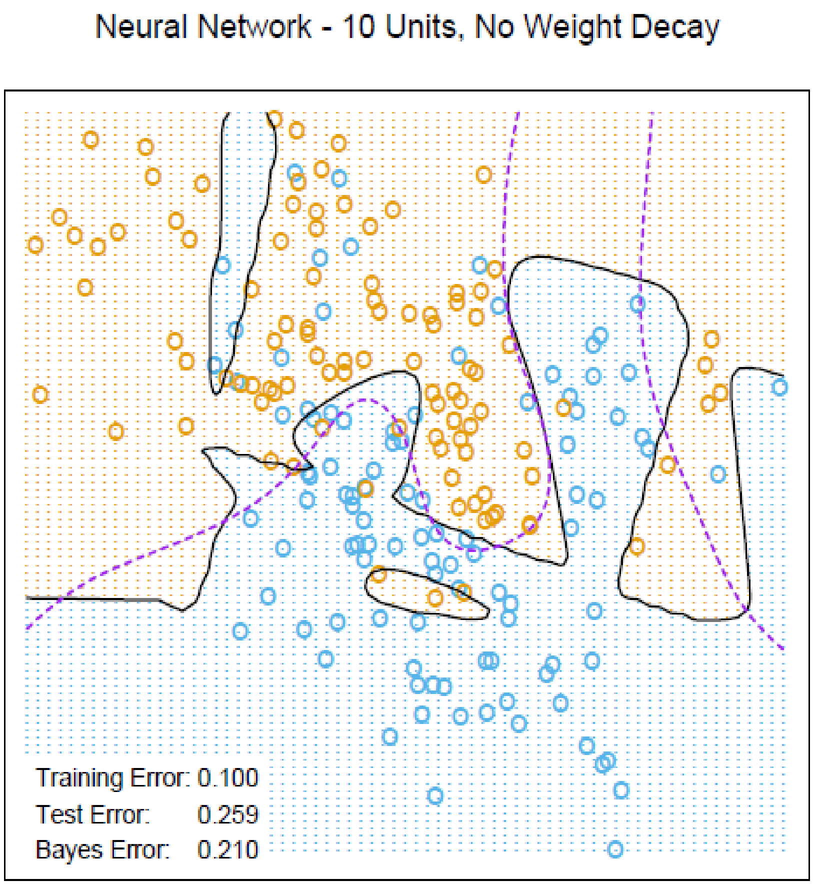
Tricks-Initial values

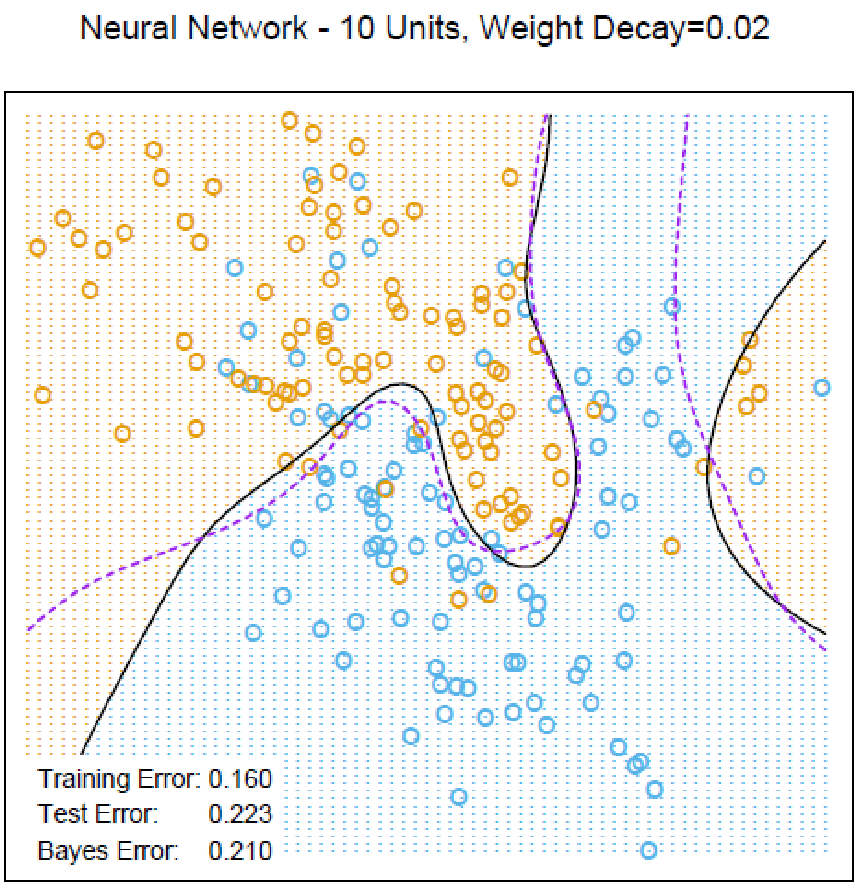
- The weights are typically initialized to small random values chosen from a zero-mean Gaussian with a standard deviation of about 0.01.
- Note that if the weights are near zero, then the operative part of the sigmoid is roughly linear, and hence the neural network collapses into an approximately linear model.
- Use of exact zero weights leads to zero derivatives and perfect symmetry, and the algorithm never moves.
- Starting instead with large weights often leads to poor solutions.

Tricks-Regularization

Add a penalty to the error function

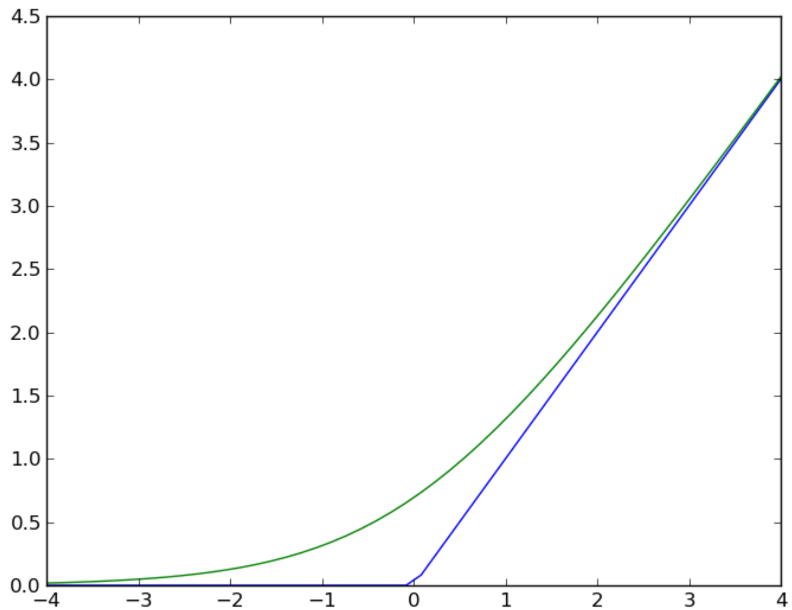
$$R(\theta) + \lambda J(\theta)$$





Tricks-Tuning

- Learning Rate
- Mini-batch size for batch learning
- Activation function
  - Rectifier function $f(x) = \max(0, x)$



Piji Li

Thanks a lot!