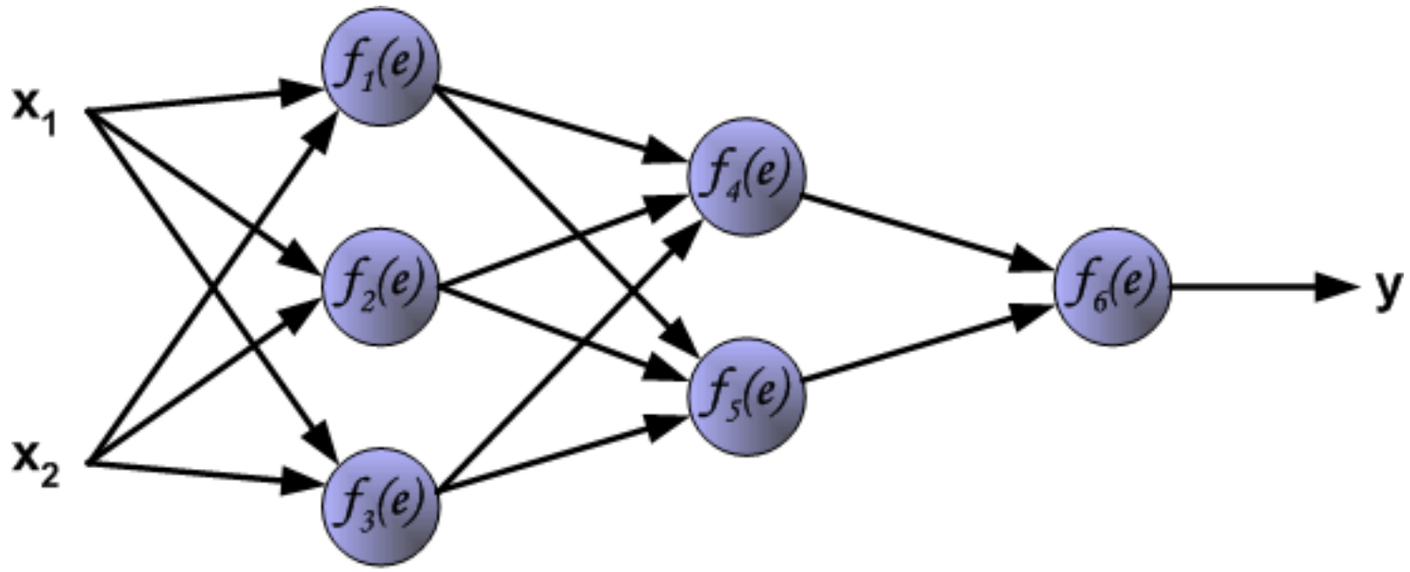
Principle of Back-propagation Algorithm

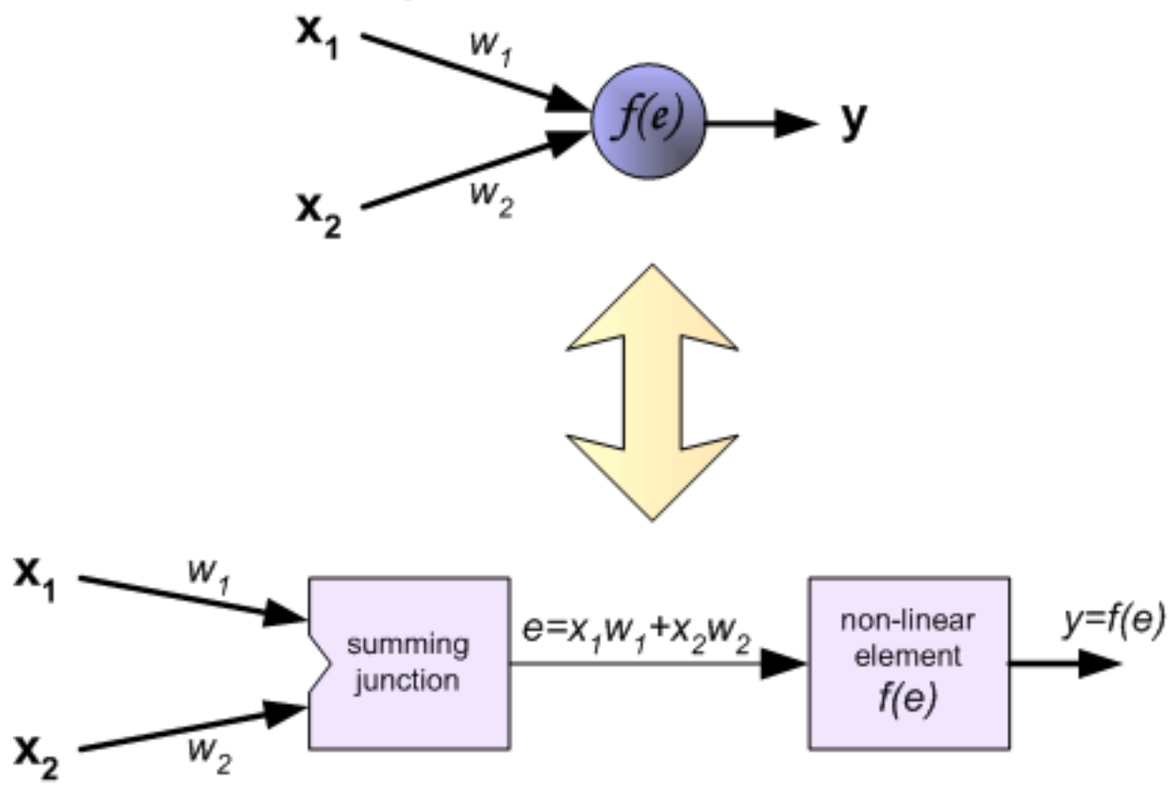
In multi-layer neural networks

Back-propagation Algorithm



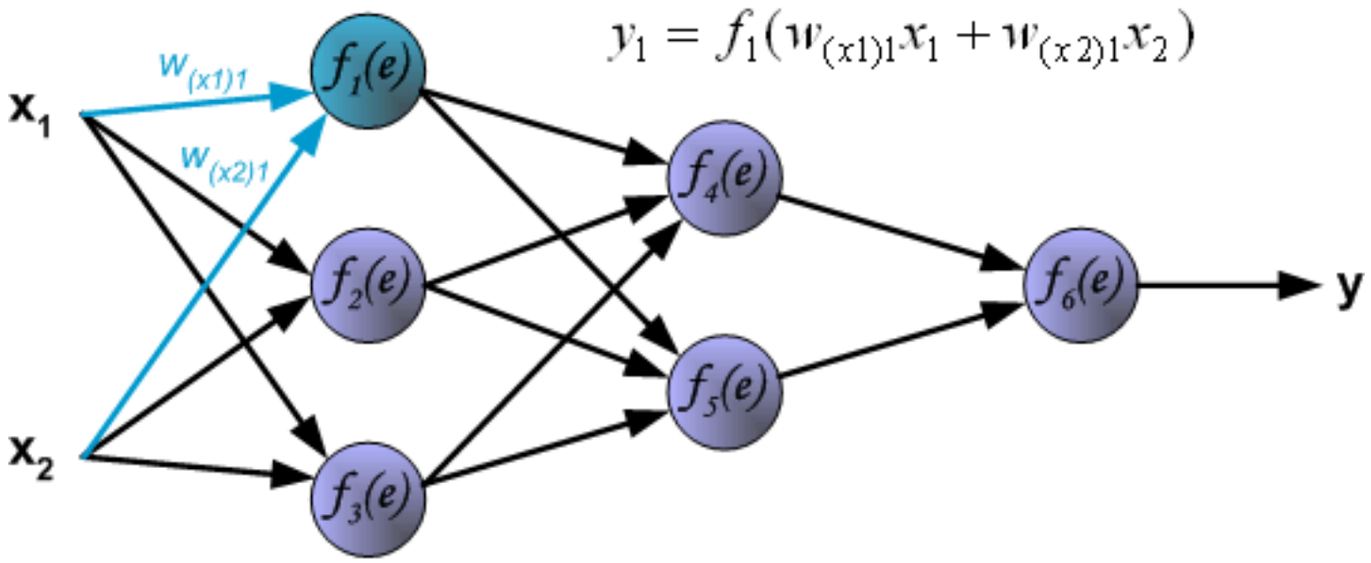
- Principles of training multi-layer neural network using back-propagation:
  To illustrate this process the three layer neural network with two inputs and one output is used

Back-propagation Algorithm



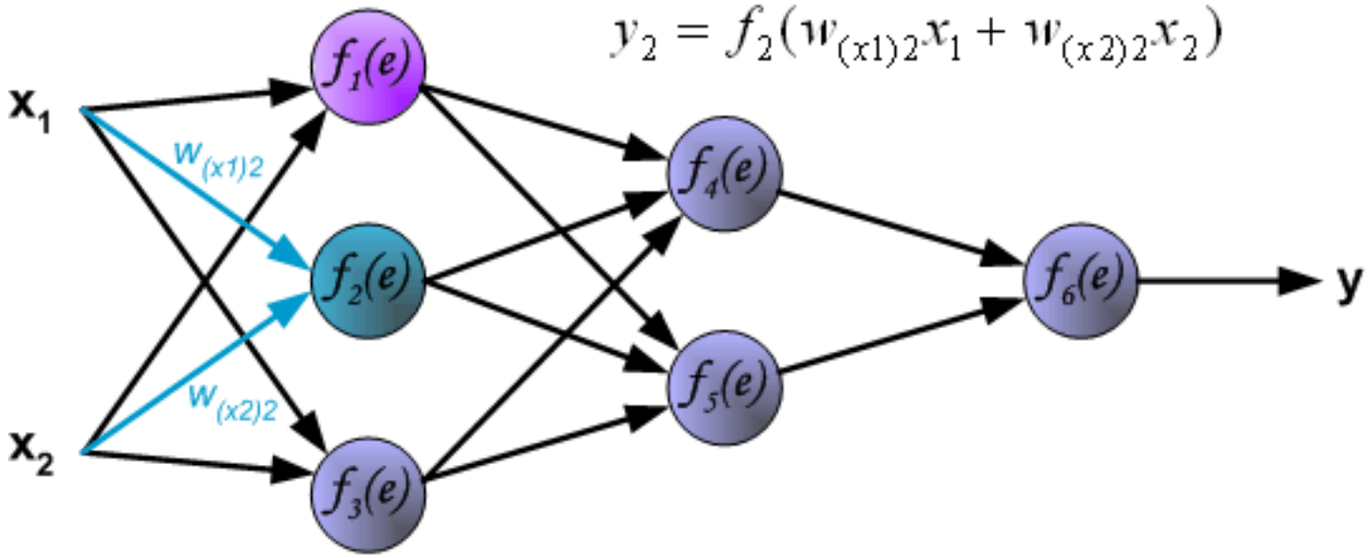
- Each neuron is composed of two units. First unit adds products of weights coefficients and input signals. The second unit realize nonlinear function, called neuron activation function. Signal **y** is an output signal of neuron

Back-propagation Algorithm



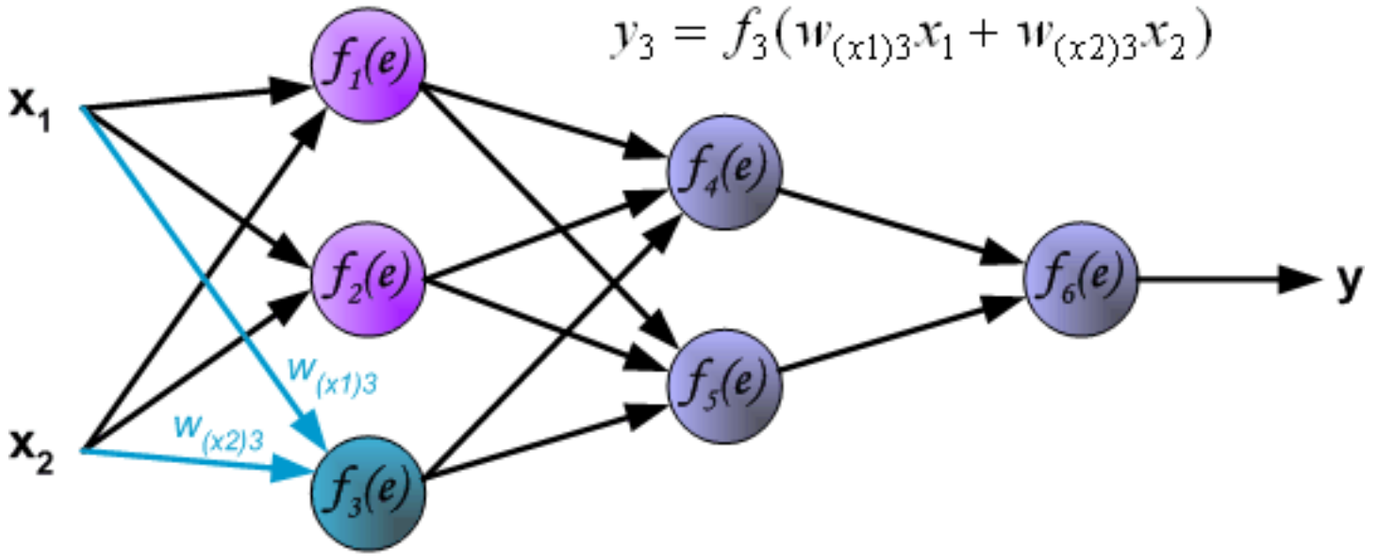
- The training data set consists of input signals (x1 and x2) assigned with corresponding target (desired output) z. Each teaching step starts with forcing both input signals from training set. After this stage we can determine output signals values for each neuron in each network layer. Pictures illustrate how signal is propagating through the network, Symbols w(xm)n represent weights of connections between network input xm and neuron n in input layer. Symbols yn represents output signal of neuron n.

Back-propagation Algorithm



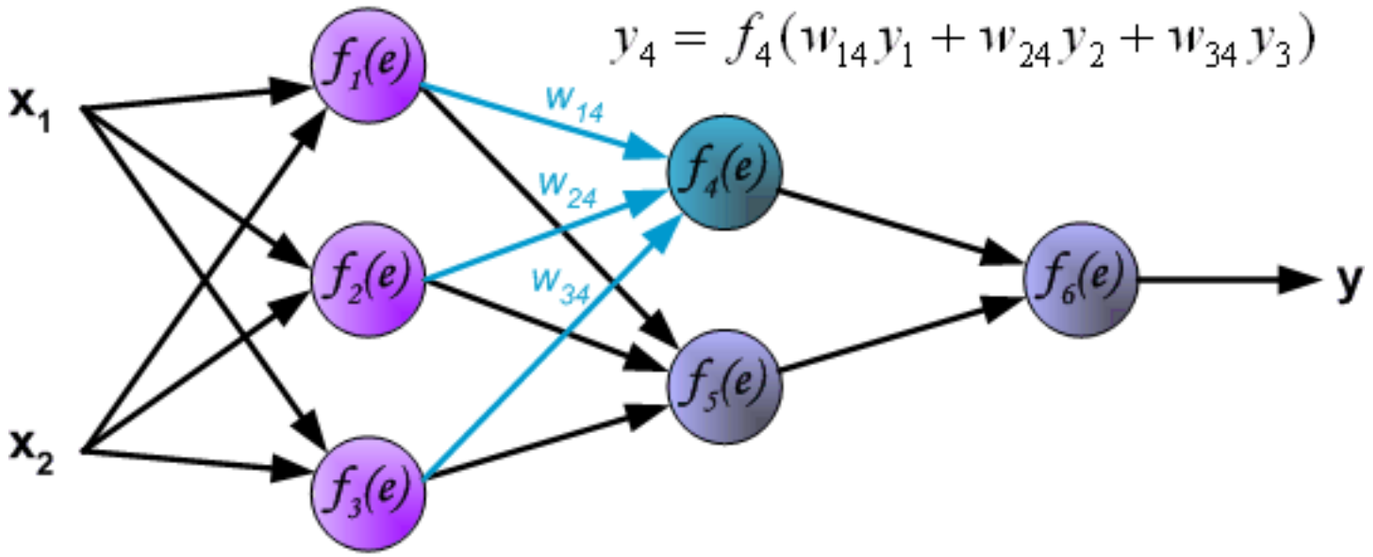
- The training data set consists of input signals (x1 and x2) assigned with corresponding target (desired output) z. Each teaching step starts with forcing both input signals from training set. After this stage we can determine output signals values for each neuron in each network layer. Pictures illustrate how signal is propagating through the network, Symbols w(xm)n represent weights of connections between network input xm and neuron n in input layer. Symbols yn represents output signal of neuron n.

Back-propagation Algorithm



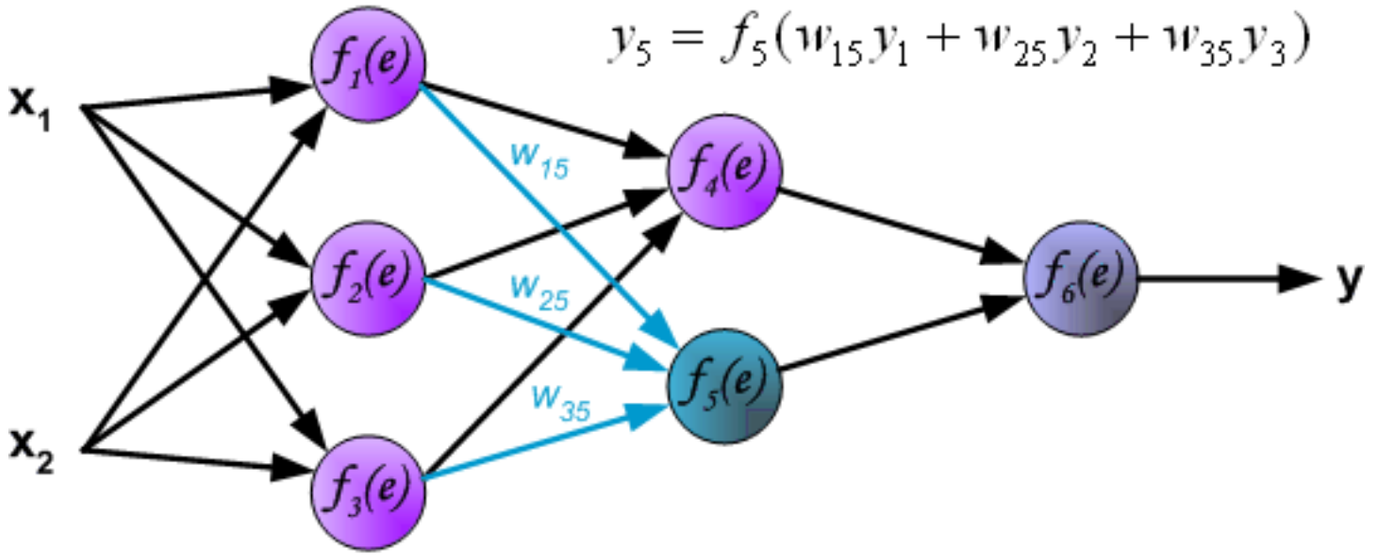
- The training data set consists of input signals (x1 and x2) assigned with corresponding target (desired output) z. Each teaching step starts with forcing both input signals from training set. After this stage we can determine output signals values for each neuron in each network layer. Pictures illustrate how signal is propagating through the network, Symbols w(xm)n represent weights of connections between network input xm and neuron n in input layer. Symbols yn represents output signal of neuron n.

Back-propagation Algorithm



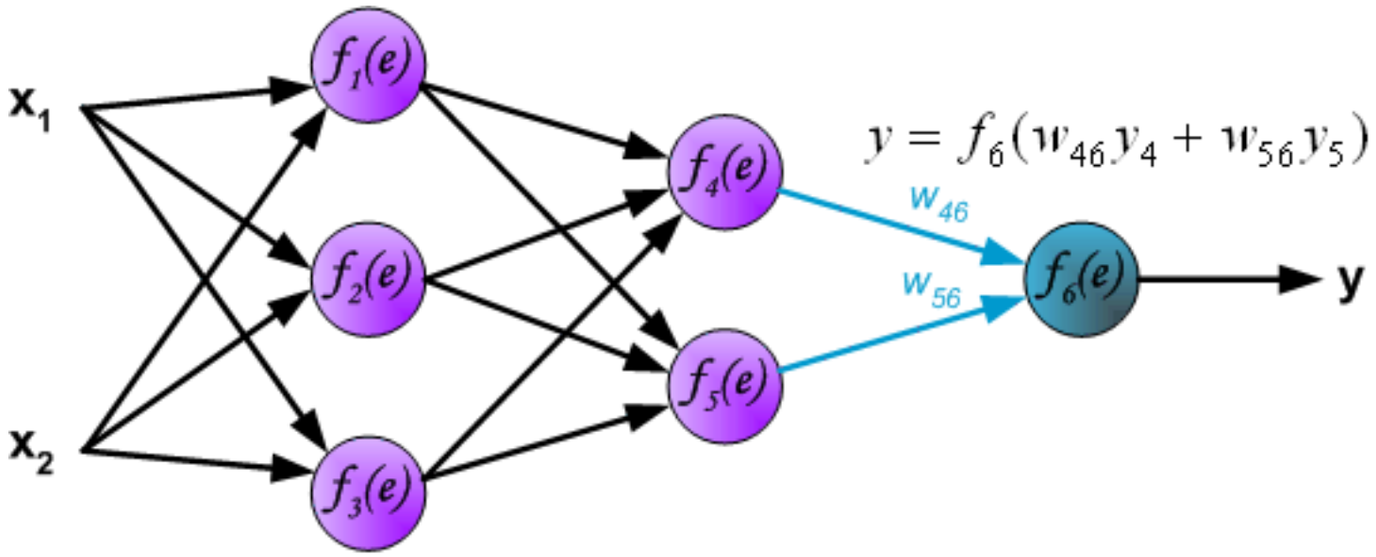
- Propagation of signals through the hidden layer. Symbols **w** represent weights of connections between output of neuron m and input of neuron n in the next layer.

Back-propagation Algorithm



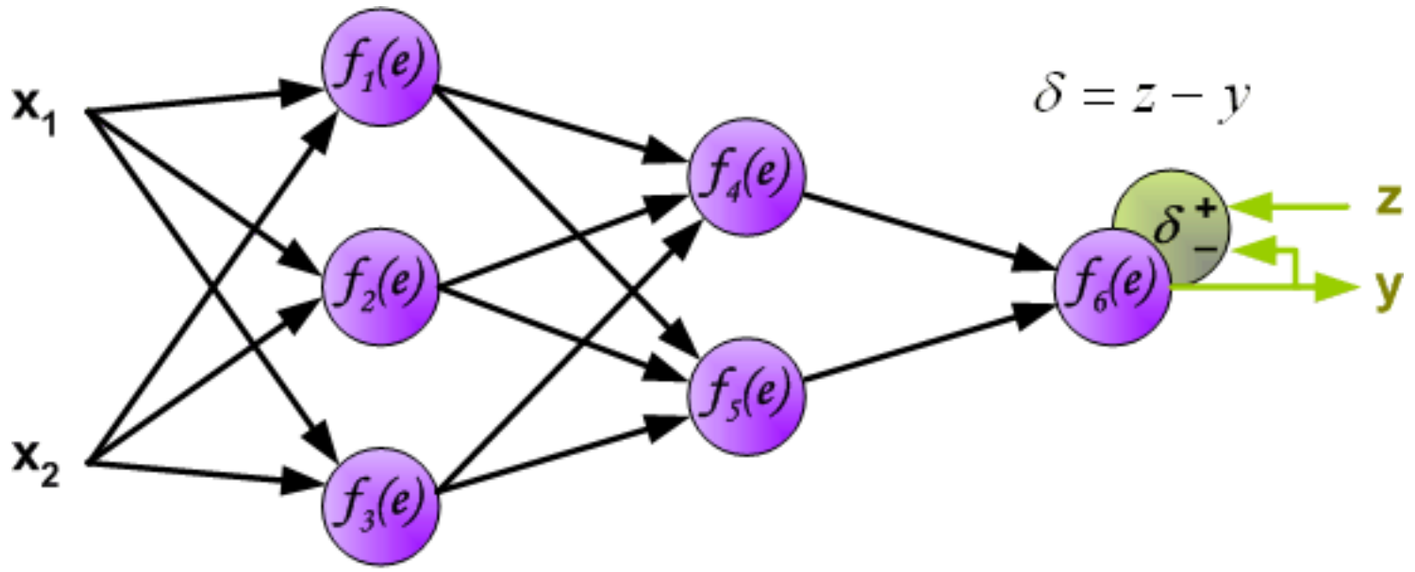
- Propagation of signals through the hidden layer. Symbols **w** represent weights of connections between output of neuron m and input of neuron n in the next layer.

Back-propagation Algorithm



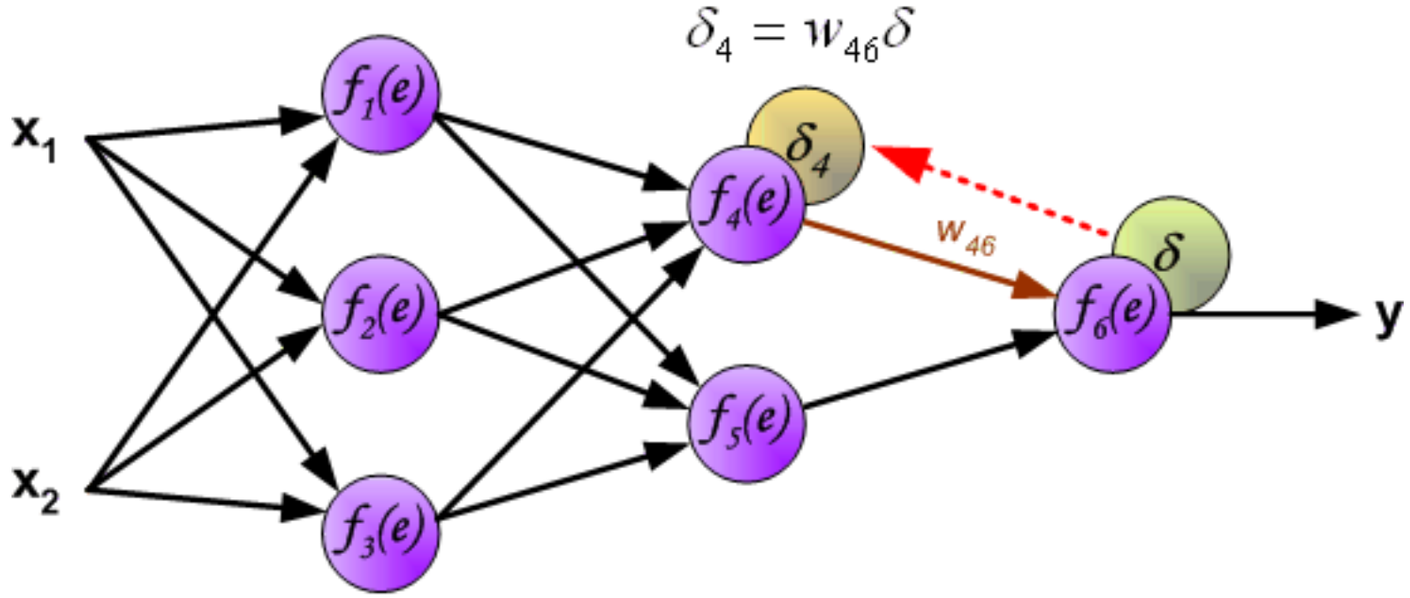
- Propagation of signals through the output layer.

Back-propagation Algorithm



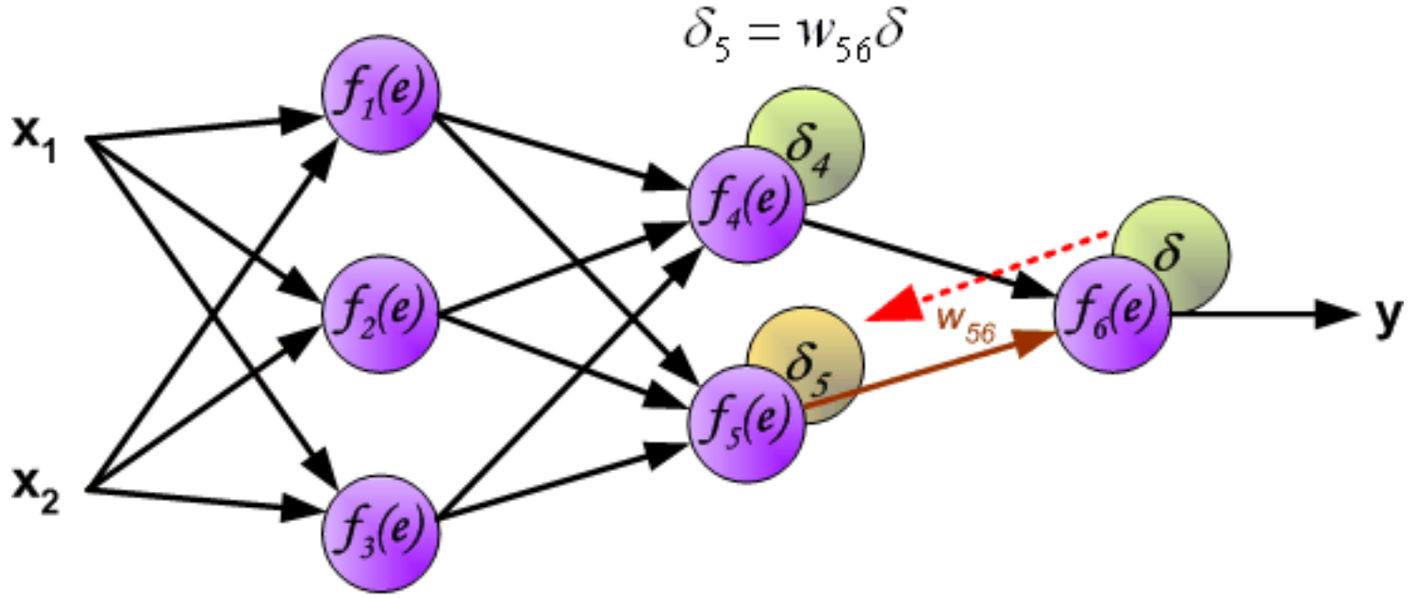
- In the next algorithm step the output signal of the network y is compared with the desired output value (the target), which is found in training data set. The difference is called error signal d of output layer neuron.

Back-propagation Algorithm



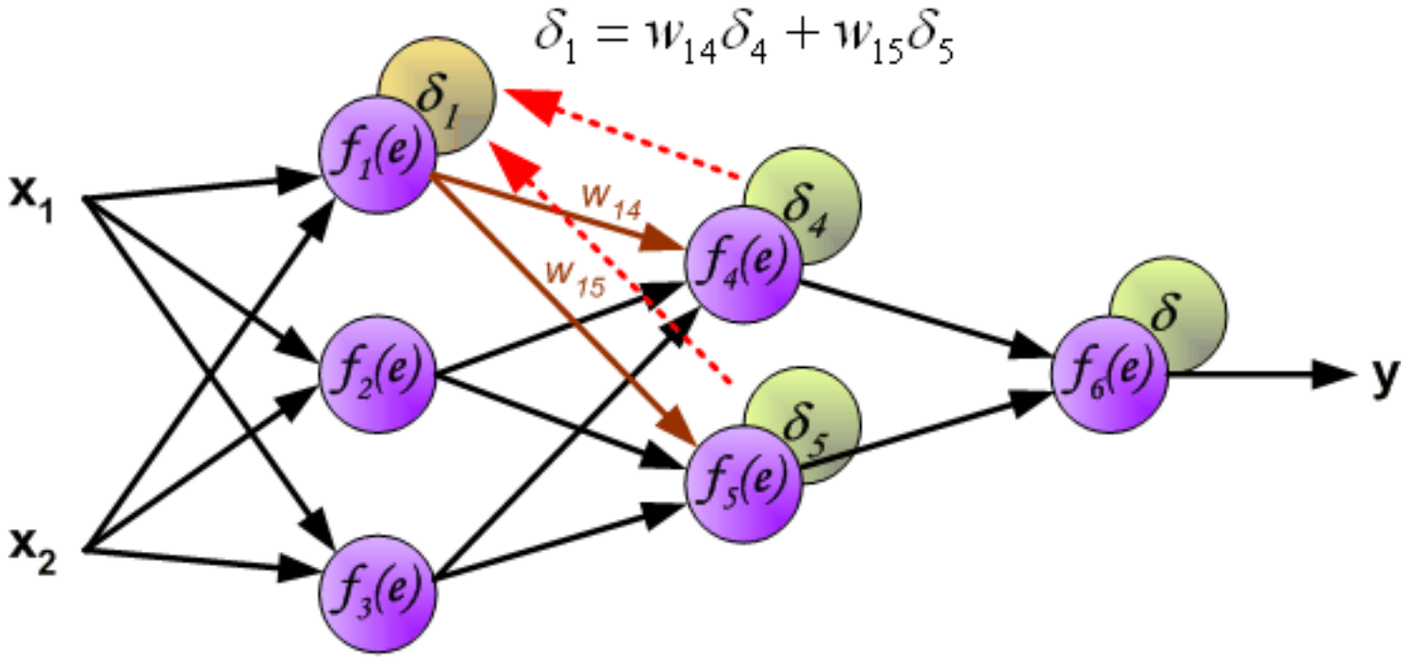
- It is very difficult to compute error signal for internal neurons directly. The back-propagation algorithm is to propagate error signal **d** (computed in single teaching step) back to all neurons, which output signals were input for discussed neuron.

Back-propagation Algorithm



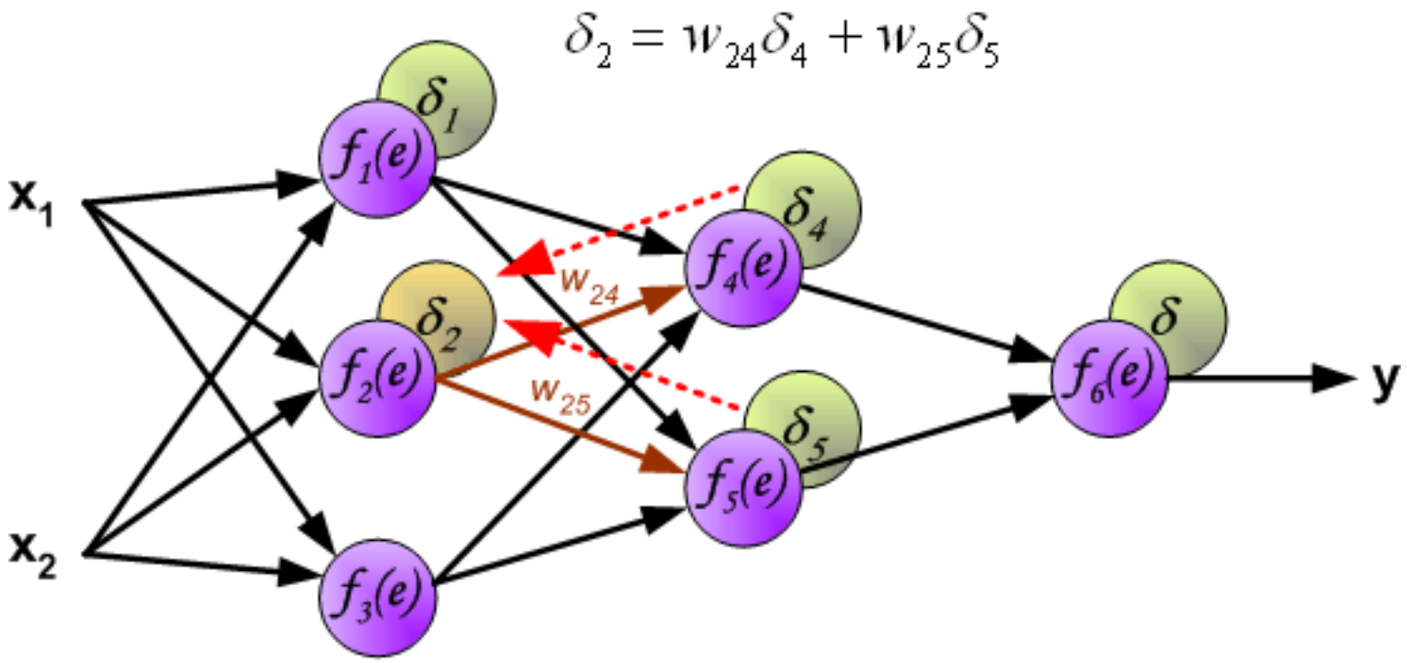
- It is very difficult to compute error signal for internal neurons directly. The back-propagation algorithm is to propagate error signal **d** (computed in single teaching step) back to all neurons, which output signals were input for discussed neuron.

Back-propagation Algorithm



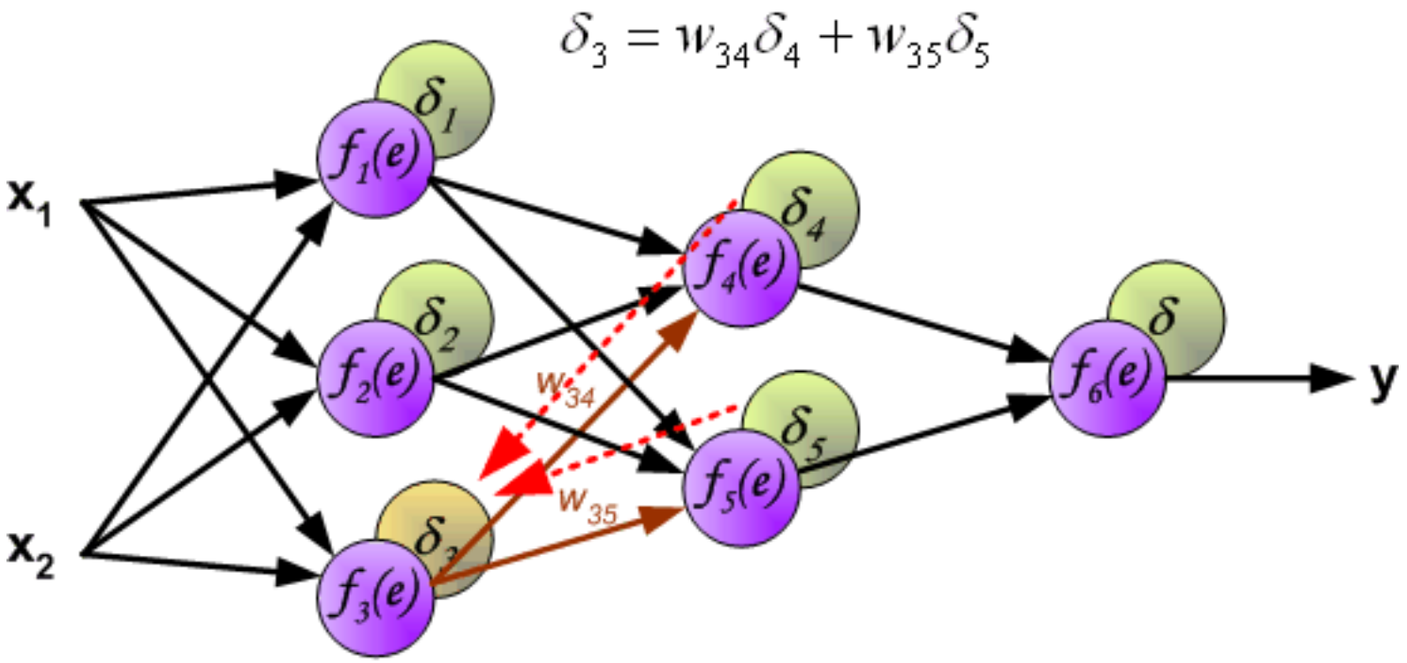
- The weights' coefficients **w** used to propagate errors back are equal to this used during computing output value. Signals are propagated from output to inputs one after the other. This technique is used for all network layers. If propagated errors came from few neurons they are added.

Back-propagation Algorithm



- The weights' coefficients **w** used to propagate errors back are equal to this used during computing output value. Signals are propagated from output to inputs one after the other. This technique is used for all network layers. If propagated errors came from few neurons they are added.

Back-propagation Algorithm

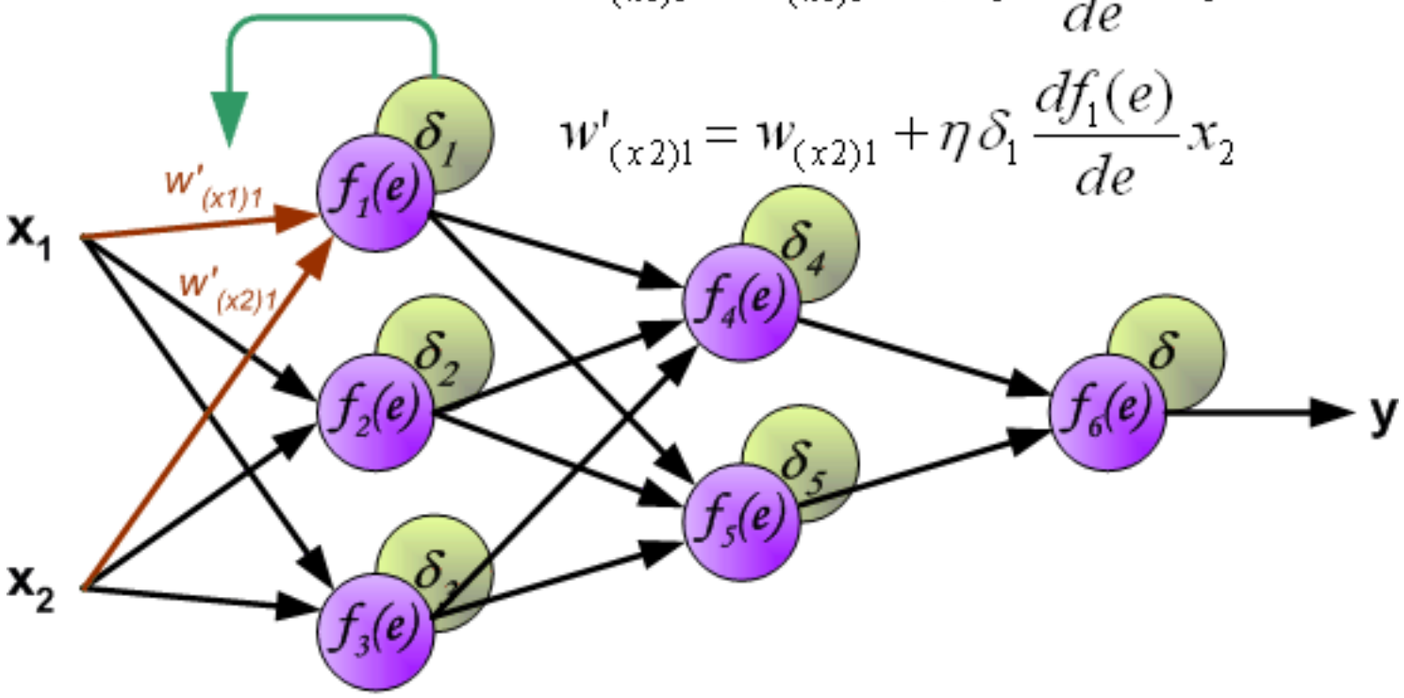


- The weights' coefficients **w** used to propagate errors back are equal to this used during computing output value. Signals are propagated from output to inputs one after the other. This technique is used for all network layers. If propagated errors came from few neurons they are added.

Back-propagation Algorithm

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

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

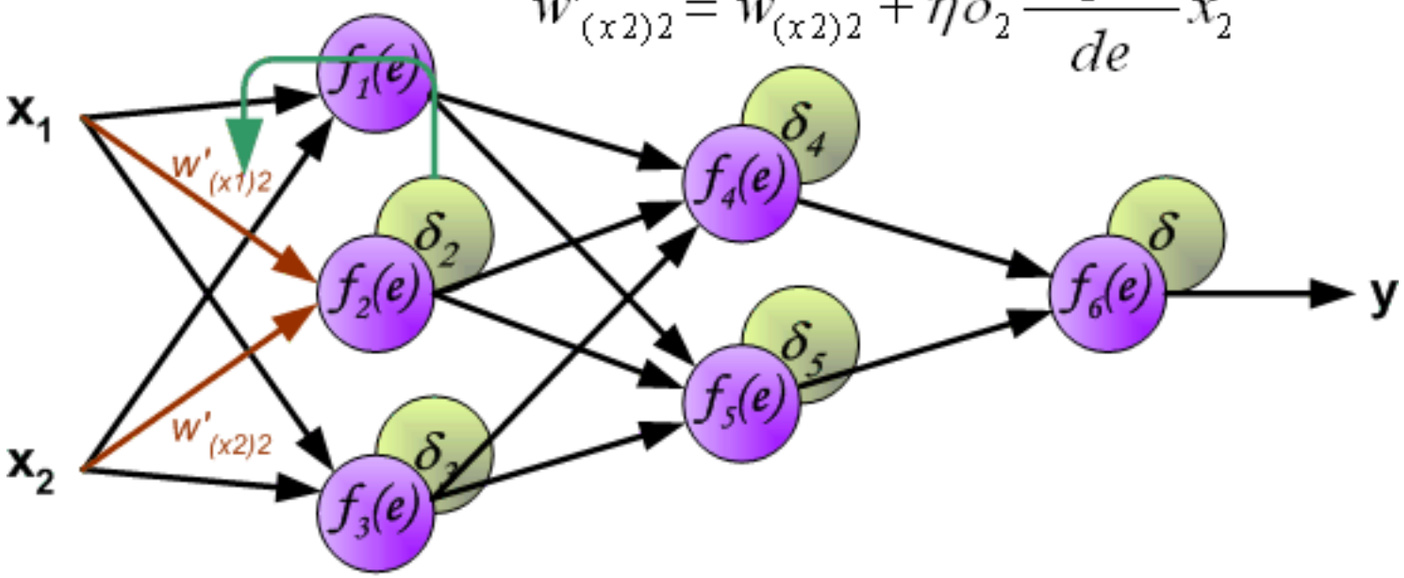


- When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified.

Back-propagation Algorithm

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

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

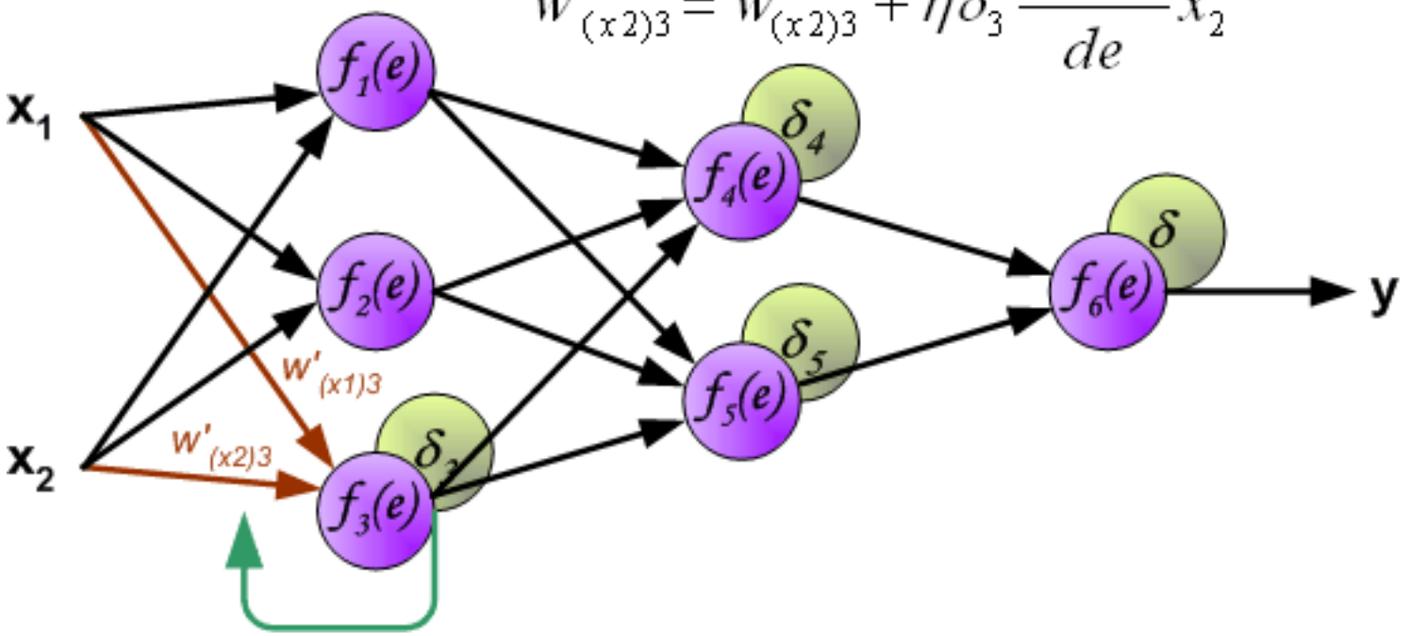


- When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified.

Back-propagation Algorithm

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

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



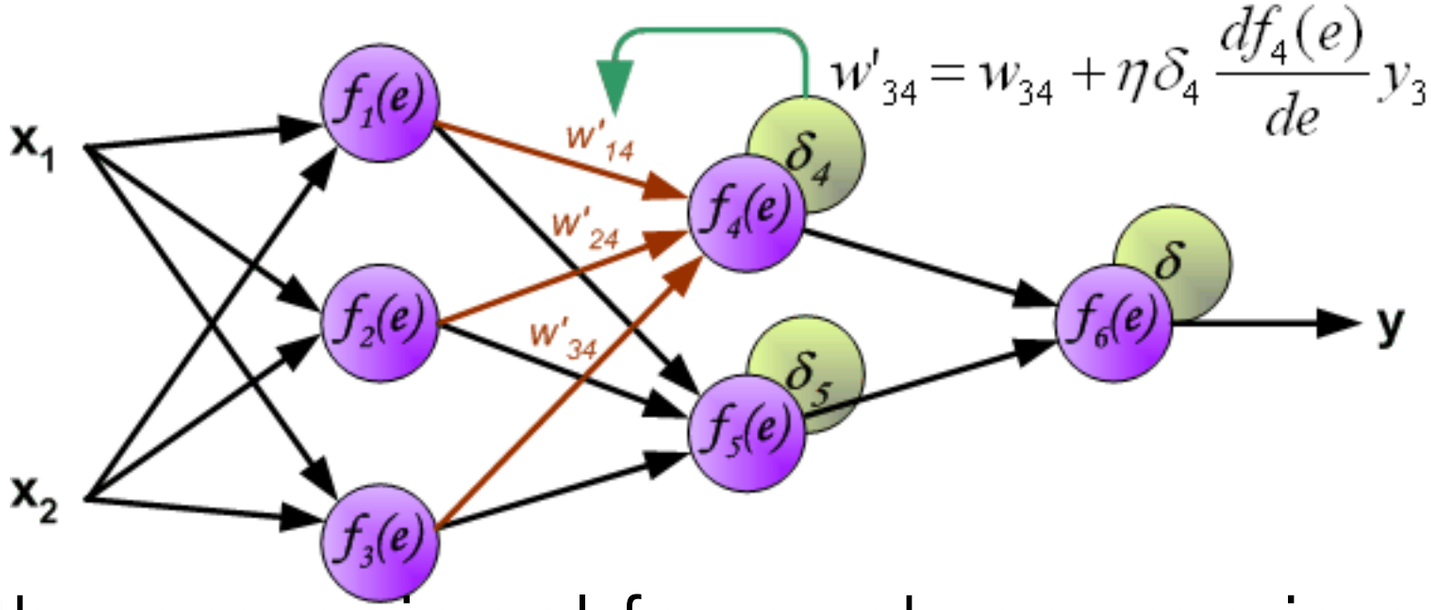
- When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified.

Back-propagation Algorithm

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

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

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



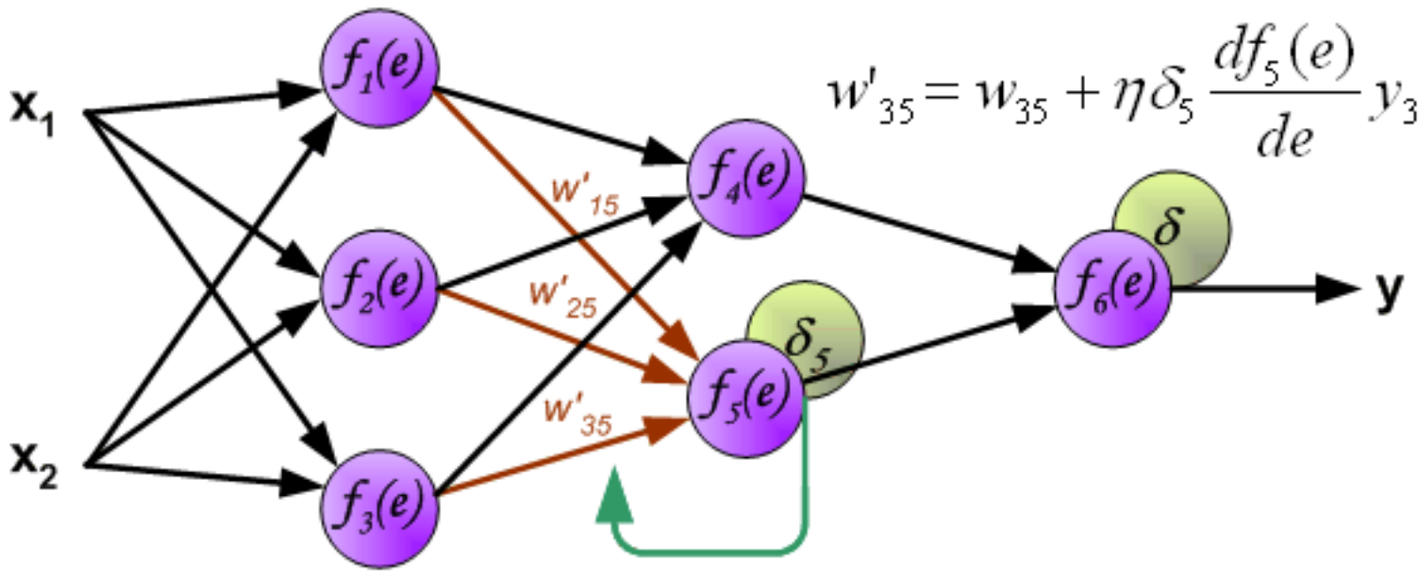
- When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified.

Back-propagation Algorithm

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

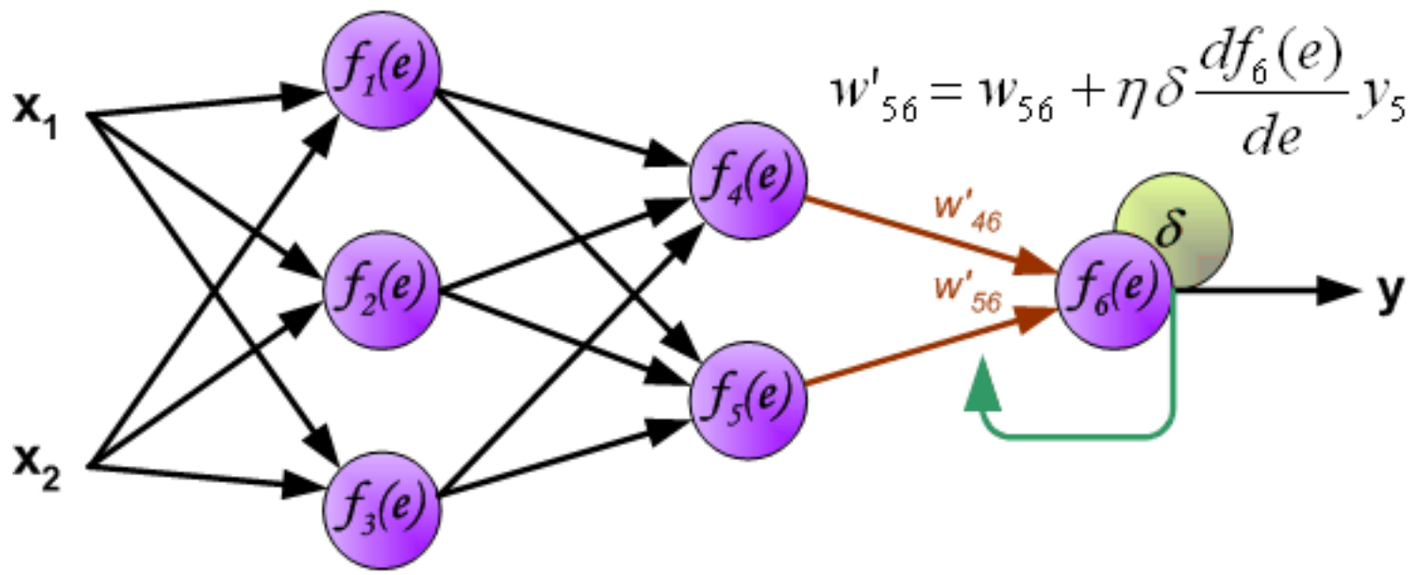
$$w'_{25} = w_{25} + \eta \delta_5 \frac{df_5(e)}{de} y_2$$

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



- When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified.

Back-propagation Algorithm



- When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified.

Back-propagation Algorithm

- Easy to implement (Relatively)
- Too many parameters to be tuned
  (Training cost: time and datasets)
- Work as a black box
- Deterministic model
  (Not a probabilistic model)

References

- Ryszard Tadeusiewcz "Sieci neuronowe", Kraków 1992

- Principles of training multi-layer neural network using back propagation (http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html)