Going Backwards

Back-Propogation of Error

Error Function

Sum-of-squares errros is used.

$$E(\mathbf{t}, \mathbf{y}) = \frac{1}{2} \sum_{k=1}^{N} (y_k - t_k)^2$$

Activation Function

As an example; consider the sigmoid function:

$$a = g(h) = \frac{1}{1 + exp(-\beta h)}$$

DIFFERENTIABILITY is key!

Back Propagation

The problems

1. For the **output** neurons, we don't know the inputs.
2. For the **hidden** neurons, we don't know the targets.
3. For **extra** hidden layers, we don't know the inputs or targets.

The Mulit-layer Perceptron Algorithm

Intialization

- Initialize all weights to small random values.

The Multi-layer Perceptron Algorithm

Training: Forwads phase

- repeat

  For each input vector:

  Forwads phase:

  - Compute the activation of each neuron j in the hidden layer.
  - Work through the network until you get to the output layer neuron.

The Multi-layer Perceptron Algorithm

Training: Backwards phase

Backwards phase:

- Compute the error at the output using:

$$\delta_o(\kappa) = (y_\kappa - t_\kappa)y_\kappa(1 - y_\kappa)$$

- Compute the error in the hidded layer(s) using:

$$\delta_h(\zeta) = a_\zeta(1 - a_\zeta)\sum_{\kappa=1}^{N} w_\zeta \delta_o(\kappa)$$

- Update the output layer weights using:

$$w_{\zeta\kappa} \leftarrow w_{\zeta\kappa} - \eta\delta_o(\kappa)a_\zeta^{\text{hidden}}$$

- Update the hidden layer weights using:

$$v_I \leftarrow v_I - \eta\delta_h(\kappa)x_I$$

MLP: Example

CNNs

References I

[Mar14] Stephen Marsland. *Machine Learning, An Algorithmic Perspective*. CRC Press, 2014.

[Ian17] Aaron Courville Ian Goodfellow Yoshua Bengio. *Deep Learning*. MIT Press, 2017.