

Multilayer Perceptrons

“Neural networks” usually refers to backpropagation neural nets with a hidden layer and an output layer.

In general, the output of a neuron might be an output of the NN or an input to other neurons.

An input to a neuron might be a data value or the output of another neuron.

Learning is achieved by the generalized delta rule, i.e., the calculation of $\partial E / \partial w$.

Variations

Difficulties include slow convergence, no optimality guarantees, and bias-variance dilemma.

Data: Data are scaled to stabilize updating

Architecture: Size of hidden layer can be varied.

Initialization: Weights are initialized randomly.

Learning: Momentum can speed up convergence.

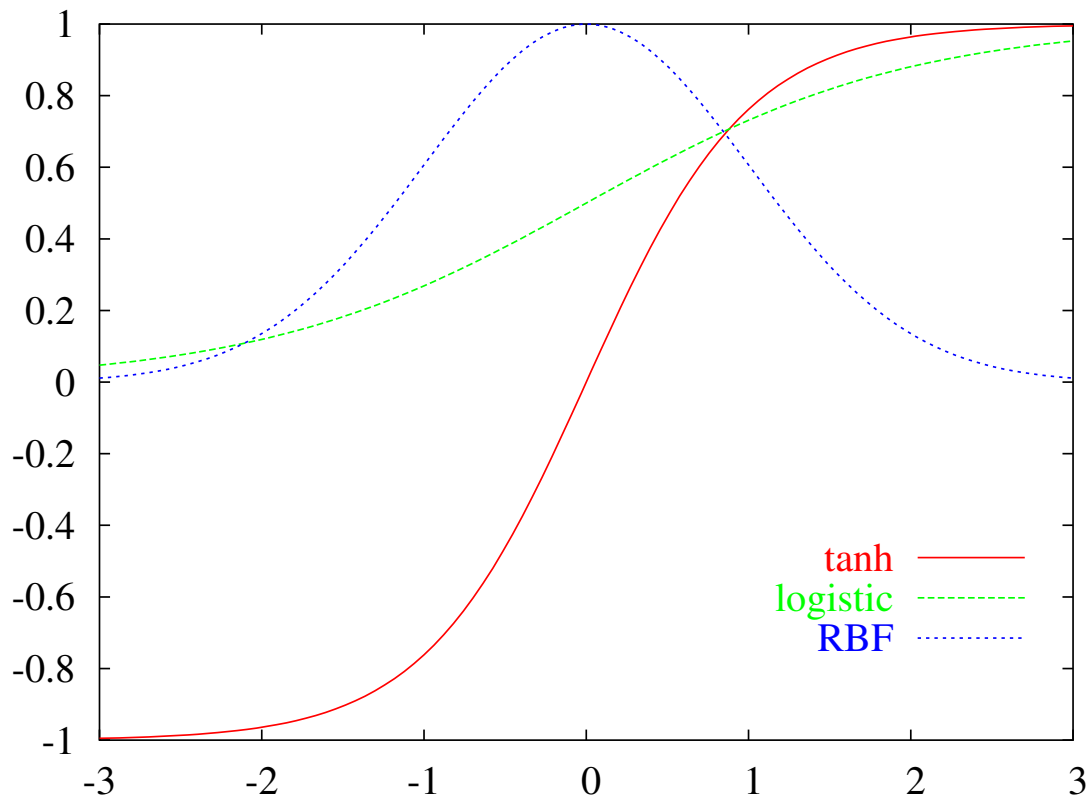
Generalization: Use the validation set method to avoid overfitting.

Generalized Delta Rule for a Single Neuron

Let \mathbf{x} be the inputs and \mathbf{w} be the weights. Again, as a mathematical convenience, let $x_{n+1} = 1$, so that w_{n+1} becomes the bias weight. The output is $o = a(u) = a(\mathbf{w} \cdot \mathbf{x})$.

$\tanh(u) = (e^u - e^{-u}) / (e^u + e^{-u})$ and $\text{logistic}(u) = 1 / (1 + e^{-u})$ are common activation functions.

For a Gaussian RBF neuron, $o = e^{-r^2 / (2\sigma^2)}$ where $r^2 = (\mathbf{x} - \mathbf{c}) \cdot (\mathbf{x} - \mathbf{c})$, and σ and \mathbf{c} are the parameters to be learned.



Generalized Delta Rule for a Single Neuron

First, we need to know $\partial E/\partial o$. There are two cases.

1. If o is an output of the NN, and d is the desired output, then use $\partial E(d, o)/\partial o$. E.g., if $E(d, o) = (d - o)^2/2$, then $\partial E/\partial o = o - d$.
2. If o is an input y to other neurons, each of these neurons will have a $\partial E/\partial y$. Then $\partial E/\partial o = \Sigma \partial E/\partial y$.

Next, we need to determine $\partial o/\partial u$, $\partial u/\partial w_i$, and $\partial u/\partial x_i$.

For the tanh activation function,

$$\frac{\partial \tanh(u)}{\partial u} = (1 - \tanh(u))(1 + \tanh(u))$$

For the logistic activation function,

$$\frac{\partial \text{logistic}(u)}{\partial u} = \text{logistic}(u)(1 - \text{logistic}(u))$$

$$\partial u/\partial w_i = x_i \text{ and } \partial u/\partial x_i = w_i.$$

Finally, we can use the chain rule to determine $\partial E / \partial w_i$ and $\partial E / \partial x_i$.

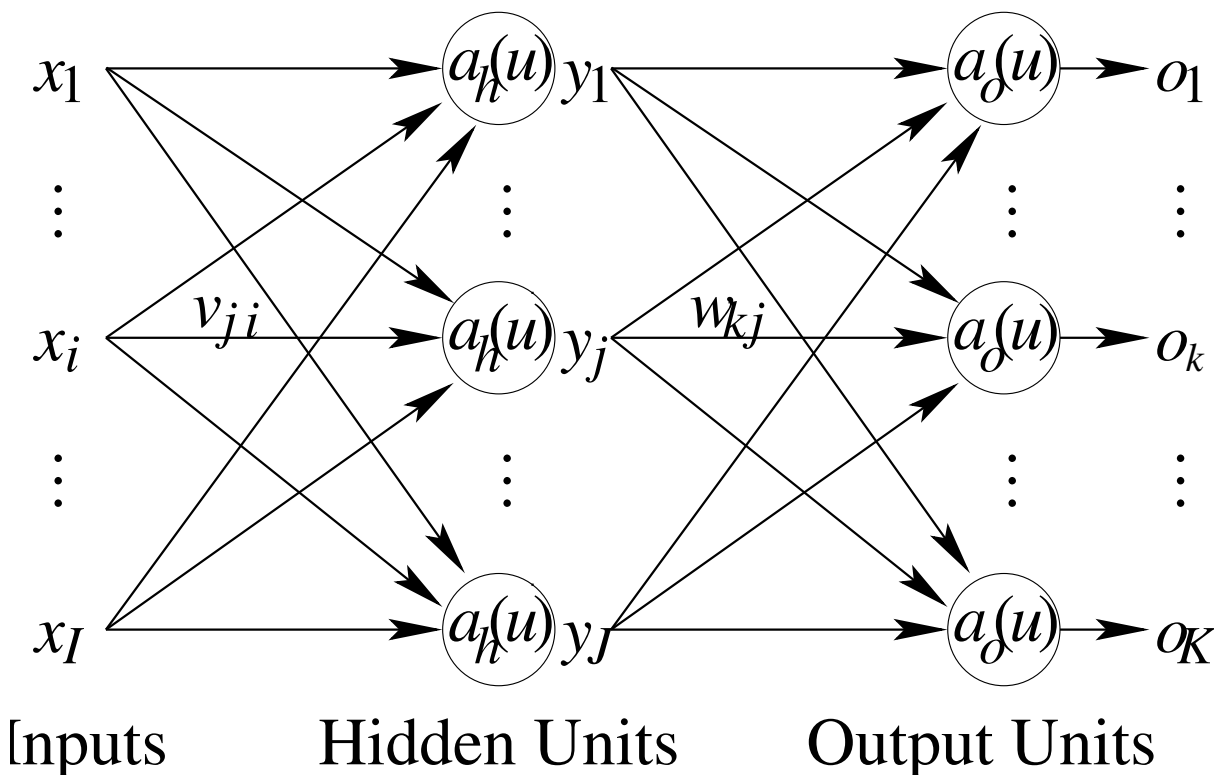
$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial o} \frac{\partial o}{\partial u} \frac{\partial u}{\partial w_i} \quad \text{and} \quad \frac{\partial E}{\partial x_i} = \frac{\partial E}{\partial o} \frac{\partial o}{\partial u} \frac{\partial u}{\partial x_i}$$

E.g., if \tanh is activation, o is a NN output, d is desired, and $E(d, o) = (d - o)^2 / 2$, then:

$$\frac{\partial E}{\partial w_i} = (o - d)(1 - \tanh(u))(1 + \tanh(u))x_i$$

$$\frac{\partial E}{\partial x_i} = (o - d)(1 - \tanh(u))(1 + \tanh(u))w_i$$

Multilayer Neural Network



Backpropagation Algorithm

Let $a'_h(u)$ and $a'_o(u)$ be the derivatives of activation functions a_h and a_o . Use squared error.

$$\frac{\partial E}{\partial w_{kj}} = (o_k - d_k) a'_o(\mathbf{w}_k \cdot \mathbf{y}) y_j$$

$$\frac{\partial E}{\partial y_j} = \sum_{k=1}^K (o_k - d_k) a'_o(\mathbf{w}_k \cdot \mathbf{y}) w_{kj}$$

$$\frac{\partial E}{\partial v_{ji}} = \frac{\partial E}{\partial y_j} a'_h(\mathbf{v}_j \cdot \mathbf{x}) x_i$$

For incremental (online) version, update by:

$$w_{kj} \leftarrow w_{kj} - \eta \frac{\partial E}{\partial w_{kj}}$$

$$v_{kj} \leftarrow v_{kj} - \eta \frac{\partial E}{\partial v_{kj}}$$

For batch (offline) version, update weights by:

$$w_{kj} \leftarrow w_{kj} - \eta \sum_{p=1}^P \frac{\partial E_p}{\partial w_{kj}}$$

$$v_{kj} \leftarrow v_{kj} - \eta \sum_{p=1}^P \frac{\partial E_p}{\partial v_{kj}}$$

Gaussian RBF Derivatives

If $o = e^{-r^2/(2\sigma^2)}$ where $r^2 = (\mathbf{x} - \mathbf{c}) \cdot (\mathbf{x} - \mathbf{c})$, then

$$\frac{\partial o}{\partial c_i} = \frac{(x_i - c_i) o}{\sigma^2}$$

$$\frac{\partial o}{\partial x_i} = \frac{(c_i - x_i) o}{\sigma^2}$$

$$\frac{\partial o}{\partial \sigma^2} = \frac{r^2 o}{2\sigma^4}$$

Example Neural Network

