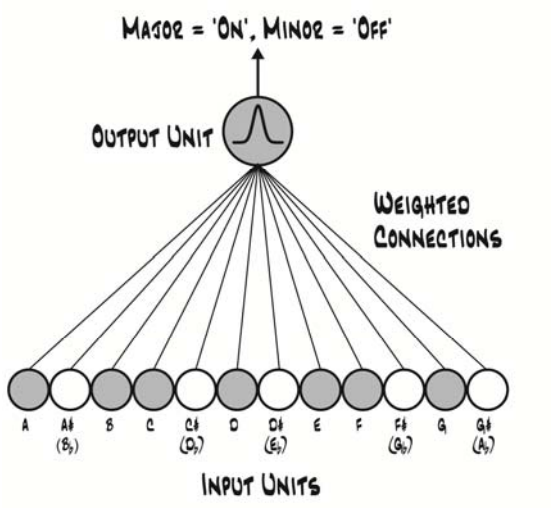
Quick Review PSYCO 452 Week 1 (January 2016)

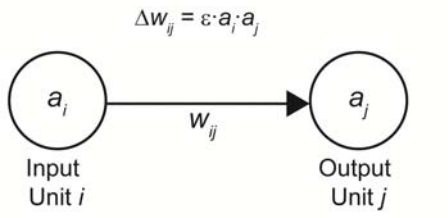
Key Architecture: Perceptron



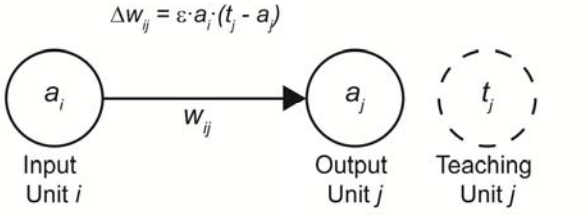
Easiest way to think about it: a distributed associative memory that uses nonlinear activation functions in the output unit.

Perceptron Learning

Any learning rule that we are considering involves computing a weight change, and then adding that weight change to the existing weight of the connection. Any connection has an input end and an output end. The Hebb rule involves multiplying a learning rate times activity at the input end times activity at the output end.



The delta rule involves multiplying the learning rate times the activity at the input end times the error at the output end.



The delta rule, as in the figure above, is used when that output unit activity is defined by a threshold function.

When an integration device is used in the output, then we can use calculus to define a gradient descent learning rule. This tries to find the steepest downhill slope in the error space. For an integration device, it

is identical to the delta rule, with the exception that error is also multiplied by the derivative of the output unit's activation function:

$$W_{ij(new)} = W_{ij(old)} + \varepsilon a_i (t_j - o_j)f'(net) = W_{ij(old)} + \varepsilon a_i (t_j - o_j)(a_j)(1 - a_j)$$

For the gradient descent rule for a value unit, which uses the Gaussian activation function, two different kinds of error are computed. The first is the difference between desired and actual output unit activity. The second is the difference between net input and the mean of the Gaussian, but just for those patterns that are supposed to turn the output unit on:

$$E = \sum E_p = \sum \sum \left(t_{pi} - o_{pi}\right)^2 + \sum \sum t_{pi} \left(net_{pi} - \mu_i\right)^2$$

This leads to a slightly more elaborate learning rule that includes both types of error, and scales each with the derivative of the Gaussian:

$$W_{ij(new)} = W_{ij(old)} + \eta(t_j - o_j)G'(net)a_i + \eta(t_j * net)G'(net)a_i$$