Abstract Neuron

__init_(activation function,sigma*)

calc_ouput(inputs, weights)

Layer

Variables:

Number of neurons
Vector of outputs
Weight matrix
Vector of delta
Neuron object

__init__(initial_weights, layer_number, number of neurons)

calc_output(inputs of previous layer)

returns vector of layer's outputs

backprop(last output vector, downstream delta values)

returns deltas for neurons in current layer
update weights

MLP Network

Variables:

Vector of layers
Matrix of layer outputs

__init__(number of layers, neurons/layer, training data, 'String' MLP/radial, # outputs, # of training runs)

train() ???? calls forward and back propagation

forward_propagation(inputs)

loop over layers
return (individual layer outputs, last layer output)

back_propagation(deltas from previous layer)

loop over layers
layer.backprop(deltas from

last_layer_update(true training output)

Adeline rule to calculate first delta

Radial Basis