

Perceptron Network classes: documentation – Neuron Class

CLASS NAME		Cneuron		
Data Members		Access	Description	
<code>struct</code> dendrite		<code>private</code>	Contains to <code>double</code> variables 'x' (the dendrite's input) and 'w' (the weight modifier).	
<code>dendrite*</code> dendrites		<code>private</code>	Dendrites are the input elements that feed into the neuron	
<code>unsigned int</code> dendriteTotal		<code>private</code>	Total # of dendrites attached to the neuron	
<code>double</code> activation		<code>private</code>	The sum of the weighted inputs (or dendrites)	
<code>double</code> y	γ	<code>private</code>	The actual output (after the activation has been passed through the 'sigmoid function')	
<code>double</code> d	δ	<code>private</code>	delta (or ERROR) between the desired and actual outputs	
<code>double</code> dy		<code>Private</code>	The Desired Output - only applies to output layer neurons (to calculating their delta)	
Return Value	Function Name	Parameters	Description	Notes
	<i>default constructor</i>			Called in <i>layer</i> object in creating an array of <i>neurons</i>
	<i>custom constructor</i>	<code>unsigned int</code> newDendriteTotal	Total # of dendrites (including bias input)	
	<i>copy constructor</i>			
	<i>destructor</i>			
<code>void</code>	CalcOutput			Calculates and sets the net activation of the neuron, then passes this through a 'Sigmoid Function' to get a non-linear output.
<code>void</code>	CreateDendrites	<code>unsigned int</code> newDendriteTotal	The total # of dendrites to be created for the neuron.	Sets up and DMAs the dendrites of a neuron (if the default constructor was used this needs to be called before neuron is used).
<code>double</code> returns the value of <i>activation</i>	GetActivation			Get the current <i>activation</i> level of the neuron.
<code>double</code> returns the value of <i>d</i>	GetDelta			Retrieve the Delta (or Error) in the neuron's output
<code>double</code> returns the value of <i>dendrites[*].w</i>	GetDendrite_w			Returns the current weight modifier applied to the dendrite
<code>double</code> returns the value of <i>dendrites[*].x</i>	GetDendrite_x			Returns the input value currently applied to the dendrite
<code>double</code> returns the value of <i>dy</i>	GetDesiredOutput			Get the Desired Output of an output layer neuron.
<code>double</code> returns the value of <i>y</i>	GetOutput			Get the actual output of the neuron after it has passed through the 'sigmoid function'.
<code>void</code>	info			Display info about the neuron
<code>void</code>	SetDelta	<code>double</code> new_d		Sets the Delta (or Error) in the neuron's output used to adjust the dendrite weights of the neuron.
<code>bool</code> 0 = ERROR 1 = SUCCESS	SetDendrite_w	<code>unsigned int</code> dendriteElem <code>double</code> new_w	Index in the <i>dendrites</i> array weight modifier to be applied to dendrite	Sets the weight modifier applied to a dendrite (usually random values are assigned before training begins).

bool 0 = ERROR 1 = SUCCESS	SetDendrite_x	unsigned int dendriteElem	Index in the <i>dendrites</i> array	Set the input value applied to a dendrite. Any attempt to set the bias input(x) will be rejected (dendrites[0]) as this is ALWAYS +1.
		double new_x	new input value to be applied to dendrite	
void	SetDesiredOutput	double new_dy	The new value of <i>dy</i>	Set the Desired Output of an output layer neuron.
void	SetOutput	double new_y	The new output value of the neuron	Explicitly set the output value: This is used to set up the input layer neurons. No processing takes place at input layer neurons. They are used simply to supply input values to be processed by subsequent layers in the net.

Perceptron Network classes: documentation – Layer Class

CLASS NAME		Clayer		
Data Members		Access	Description	
Cneuron* neurons		private	Array of neurons belonging to the layer	
unsigned int neuronTotal		private		
unsigned int dendriteTotal		private	The total # of dendrites belonging to each neuron in the layer	
Unsigned int type		private	The type of layer (set during initialisation). 0 = input layer, 1 = hidden layer, 2 = output layer There is only ever one input & output layer. There can be any number of hidden layers in-between.	
		private		
Return Value	Function Name	Parameters	Description	Notes
	<i>default constructor</i>			default values: type = 1, neuronTotal = 0, dendriteTotal = 2
	<i>custom constructor</i>	unsigned int newNeuronTotal unsigned int newDendriteTotal unsigned int newType	# of neurons within the layer The # of dendrites required by each neuron in a processing layer = # of neurons in preceding layer The type of layer (input, hidden or output)	
	<i>copy constructor</i>			
	<i>destructor</i>			deletes dynamically allocated array to which 'neurons' is a pointer
void	CalcOutput			Calculates the outputs of all the neurons in the layer (will have no effect on 'input layer' neurons)
void	CreateNeurons	unsigned int newNeuronTotal unsigned int newDendriteTotal unsigned int newType	Sets <i>neuronTotal</i> Sets <i>dendriteTotal</i> Sets the layer <i>type</i>	Creates the neuron elements for this layer (each with dendrites determined by the number of neurons in the previous layer) – also sets layer <i>type</i> .
double returns the value of a neuron's <i>activation</i>	GetActivation	unsigned int neuronElem	Index of a neuron within the <i>neurons</i> array	Get the current <i>activation</i> level of a neuron.
double returns the value of <i>d</i> (of a particular neuron element in the layer)	GetDelta	unsigned int neuronElem		Retrieve the Delta (or Error) in the neuron's output
double returns the value of <i>dendrites[*].w</i> (of a neuron element)	GetDendrite_w	unsigned int neuronElem unsigned int dendriteElem	Index of a particular dendrite belonging to the neuron	Returns the current weight modifier applied to a neuron's dendrite.
double returns the value of <i>dendrites[*].x</i> (of a neuron element)	GetDendrite_x	unsigned int neuronElem unsigned int dendriteElem		Returns the input value currently applied to a neuron's dendrite

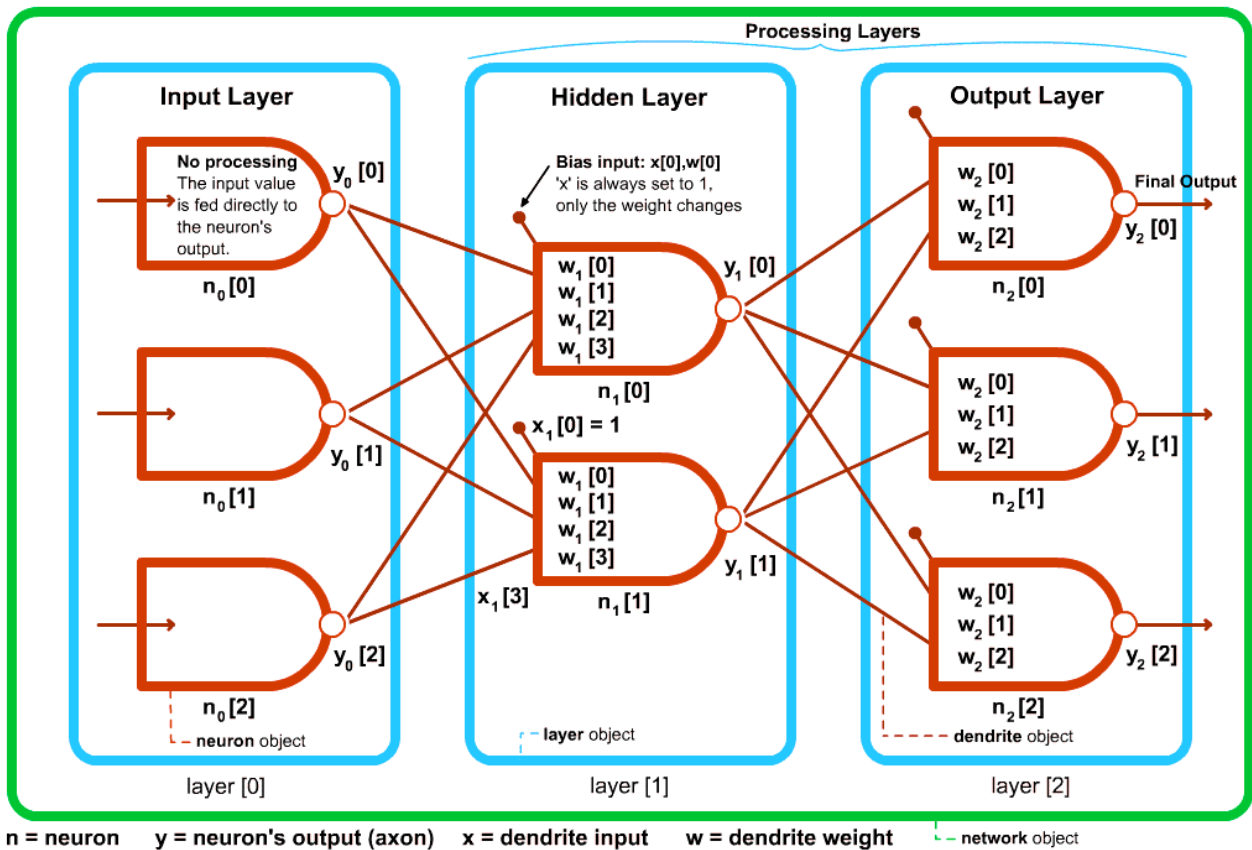
double returns the value of <i>dy</i> (of a neuron element)	GetDesiredOutput	unsigned int neuronElem		Get the Desired Output of an output layer neuron
unsigned int returns the value of <i>neuronTotal</i>	GetNeuronTotal			Returns the total number of neurons in the layer
double returns the value of <i>y</i> (of a neuron element)	GetOutput	unsigned int neuronElem		Get the actual output of the neuron after it has passed through the 'sigmoid function'.
void	Info			Display into on the layer (DOS only)
bool 0 = ERROR 1 = SUCCESS	SetDelta	unsigned int neuronElem		Set the delta (or Error) in the neuron's output.
		double new_d	The new delta (ERROR) value.	
bool 0 = ERROR 1 = SUCCESS	SetDendrite_w	unsigned int neuronElem		Set the weight modifier of a neuron's dendrite.
		unsigned int dendriteElem		
		double new_w		
bool 0 = ERROR 1 = SUCCESS	SetDendrite_x	unsigned int neuronElem		Set the input value of a neuron's dendrite.
		unsigned int dendriteElem		
		double new_x		
bool 0 = ERROR 1 = SUCCESS	SetDesiredOutput	unsigned int neuronElem		Set the Desired Output of an output layer neuron.
		double new_dy		
void	SetType	unsigned int newType		Set the layer's <i>type</i> ()

Perceptron Network classes: documentation – Network Class

CLASS NAME		Cnetwork		
Data Members		Access	Description	
<code>Clayer*</code> layers		private	0 (first element) => input layer 1 (inbetween elements) => hidden layers 2 (last element) => output layer	
<code>unsigned int</code> layerTotal		private	The number of layers in the entire network	
<code>unsigned int</code> pass		private	The number of passes through the training algorithm	
<code>unsigned int</code> epoch		private	The number of times the training algorithm has been applied to the ENTIRE training/pattern set. Incremented by 'IsTrained()'.	
<code>double</code> eta	η	private	(n) also know as the 'learning co-efficient' (eta x delta x input(i))	
<code>double</code> epsilon	ϵ	private	The netError must be be below this value to terminate training	
<code>double</code> alpha	α	private	[not yet implemented] modulates 'momentum' – which is proportional to the change in weight from the previous training pass?	
<code>double</code> rmsDelta		private	The net 'Root-Mean-Square' of all the delta errors in the net.	
Return Value	Function Name	Parameters	Description	Notes
	<i>default constructor</i>			
	<i>custom constructor</i>	<code>unsigned int</code> newlayerTotal <code>double</code> newEta <code>double</code> newEpsilon	Number of layers in the network Value used to decide whether to the net is trained.	The network object encapsulates / co-ordinates all neural net objects.
	<i>copy constructor</i>			
	<i>destructor</i>			
<code>void</code>	createLayers	<code>unsigned int</code> newlayerTotal		DMAs layer objects to the <i>layers</i> array (only required if the default constructor is called).
<code>void</code>	defineLayers	<code>unsigned int</code> layerElem <code>unsigned int</code> newNeuronTotal	The index of the layer element within the <i>layers</i> array. The first element is defined as a <i>input layer</i> , the last element becomes the <i>output layer</i> . All elements in between are <i>hidden layers</i> . Total # of neurons within the layer.	After the net's <i>layers</i> have been created they need to be defined before they are useable within the network. (The # of dendrites required by each neuron in a processing layer = # of neurons in the preceding layer +1).
<code>double</code>	GetDendrite_w	<code>unsigned int</code> layerElem <code>unsigned int</code> neuronElem <code>unsigned int</code> dendriteElem	Index of a layer element Index of a neuron within the layer Index of a particular dendrite belonging to the neuron	
<code>double</code>	GetDendrite_x	<code>unsigned int</code> layerElem <code>unsigned int</code> neuronElem <code>unsigned int</code> dendriteElem		Returns the input value currently applied to a neuron's dendrite.

double	GetDesiredOutput	unsigned int layerElem		Get the Desired Output of an output layer neuron
		unsigned int neuronElem		
unsigned int	GetPass		<i>pass</i> is incremented by the <i>train()</i> function	Get the current # of passes through the network training algorithm.
double	GetRMS			Get the net <i>Root-Mean-Square</i> of the all the neuron deltas (errors) in the network.
void	info			Display info on the current state of the network. (DOS-only)
bool 0 = FAIL 1 = SUCCESS (network trained)	IsTrained			If the Root-Mean-Square of all deltas is below <i>epsilon</i> then the network IS TRAINED, return success to stop training.
bool 0 = ERROR 1 = SUCCESS	SetDelta	unsigned int layerElem		Set the delta (or Error) in a neuron's output.
		unsigned int neuronElem		
		double new_d	Neuron's delta	
bool 0 = ERROR 1 = SUCCESS	SetDendrite_w	unsigned int layerElem		Set the weight modifier of a neuron's dendrite.
		unsigned int neuronElem		
		unsigned int dendriteElem		
		double new_w	Weight modifier	
bool 0 = ERROR 1 = SUCCESS	SetDendrite_x	unsigned int layerElem		Set the input value of a neuron's dendrite.
		unsigned int neuronElem		
		unsigned int dendriteElem		
		double new_x	Input value	
bool 0 = ERROR 1 = SUCCESS	SetDesiredOutput	unsigned int neuronElem		Set the Desired Output of an output layer neuron (used by the training algorithm).
		double new_dy	Desired Output	
void	test			Gets the inputs from the <i>input layer</i> and forward-propagates through the net to produce the corresponding output values at the <i>output layer</i> .
void	train			Given a particular <i>pattern</i> of input values at the <i>input layer</i> , apply the training algorithm and adjust the weight modifiers of all dendrites accordingly. Finally, calculate the <i>Root-Mean-Square</i> error of the current training pass.

MLP Network (created using my perceptron classes)



Equation 1:

To find the total **activation** of a neuron:

a = activation

n = total number of dendrites

w = dendrite's *weight* modifier

x = dendrite's *input*

$$a = \sum_{i=0}^n w_i x_i$$

Equation 2:

To find the change in a dendrites *weight* (after finding all neuron deltas):

Δw = change in *weight* of the dendrite

η = (*eta*) learning co-efficient

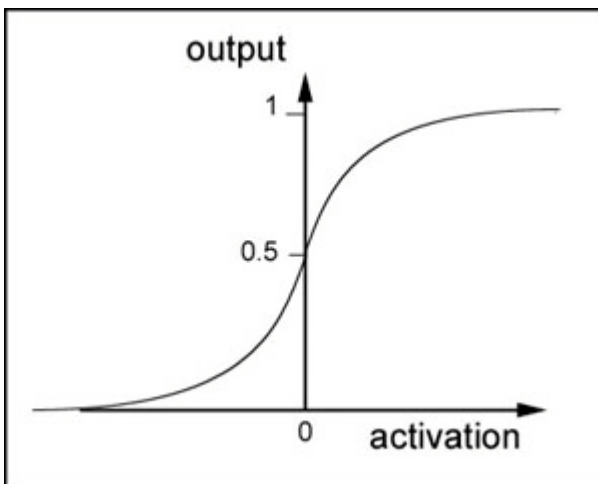
x = *input* applied to the dendrite

δ = *delta* error of the neuron

$$\Delta w = \eta x \delta$$

Equation 3:

To find the neuron's actual output (squashing / sigmoid function):



y = the actual output

a = neuron's *activation*

$$y = \frac{1}{1 + e^{-a}}$$

Equation 4:

To find an output layer neuron's *delta* (or error) value:

δ = *delta* error of the neuron

y = the actual output

dy = the desired output

$$\delta = y(1-y)(dy-y)$$

Equation 5:

To find an hidden layer neuron's *delta* (or error) value:

δ = *delta* error of the neuron

y = the actual output

dy = the desired output

λ = the current *layer* index

ρ = the current neuron element

τ = total number of neurons in the next layer up

$$\delta_{\lambda}(\rho) = x_{\lambda}(\rho)[1-x_{\lambda}(\rho)] \sum_{i=0}^{\tau} w_{\lambda+1}(\rho, i+1) \delta_{\lambda+1}(i)$$