

MicroMLP / README.md



jczic Fix derivative functions problems ✓



1 contributor

MicroMLP is a micro artificial neural network multilayer perceptron (principally used on ESP32 and [Pycom](#) modules)



Very easy to integrate and very light with one file only :

- "microMLP.py"

MicroMLP features :

- Modifiable multilayer and connections structure
- Integrated bias on neurons
- Plasticity of the connections included
- Activation functions by layer
- Parameters Alpha, Eta and Gain
- Managing set of examples and learning
- QLearning functions to use reinforcement learning

- Save and load all structure to/from json file
- Various activation functions :
 - Heaviside binary step
 - Logistic (sigmoid or soft step)
 - Hyperbolic tangent
 - SoftPlus rectifier
 - ReLU (rectified linear unit)
 - Gaussian function

Use deep learning for :

- Signal processing (speech processing, identification, filtering)
- Image processing (compression, recognition, patterns)
- Control (diagnosis, quality control, robotics)
- Optimization (planning, traffic regulation, finance)
- Simulation (black box simulation)
- Classification (DNA analysis)
- Approximation (unknown function, complex function)



Using *MicroMLP* static functions :

Name	Function
Create	<code>mlp = MicroMLP.Create(neuronsByLayers, activationFuncName, layersAutoConnectFunction=None, useBiasValue=1.0)</code>
LoadFromFile	<code>mlp = MicroMLP.LoadFromFile(filename)</code>

Using *MicroMLP* speedy creation of a neural network :

```
from microMLP import MicroMLP
mlp = MicroMLP.Create([3, 10, 2], "Sigmoid", MicroMLP.LayersFullConnect)
```

Using *MicroMLP* main class :

Name	Function
Constructor	<code>mlp = MicroMLP()</code>
GetLayer	<code>layer = mlp.GetLayer(layerIndex)</code>
GetLayerIndex	<code>idx = mlp.GetLayerIndex(layer)</code>
RemoveLayer	<code>mlp.RemoveLayer(layer)</code>
GetInputLayer	<code>inputLayer = mlp.GetInputLayer()</code>
GetOutputLayer	<code>outputLayer = mlp.GetOutputLayer()</code>

Learn	<code>ok = mlp.Learn(inputVectorNNValues, targetVectorNNValues)</code>
Test	<code>ok = mlp.Test(inputVectorNNValues, targetVectorNNValues)</code>
Predict	<code>outputVectorNNValues = mlp.Predict(inputVecto</code>
QLearningLearnForChosenAction	<code>ok = mlp.QLearningLearnForChosenAction(stateVectorN rewardNNValue, pastStateVectorNNValues, chosenActionIndex, terminalState=True, discountFactorNNValue=None)</code>
QLearningPredictBestActionIndex	<code>bestActionIndex = mlp.QLearningPredictBestActionIndex(stateVecto</code>
SaveToFile	<code>ok = mlp.SaveToFile(filename)</code>
AddExample	<code>ok = mlp.AddExample(inputVectorNNValues, targetVectorNNValues)</code>
ClearExamples	<code>mlp.ClearExamples()</code>
LearnExamples	<code>learnCount = mlp.LearnExamples(maxSeconds=30, maxCount=None, stopWhenLearned=True, printMAEAverage=True)</code>

Property	Example	Read/Write
Layers	mlp.Layers	get
LayersCount	mlp.LayersCount	get
IsNetworkComplete	mlp.IsNetworkComplete	get
MSE	mlp.MSE	get
MAE	mlp.MAE	get
MSEPercent	mlp.MSEPercent	get
MAEPercent	mlp.MAEPercent	get
ExamplesCount	mlp.ExamplesCount	get

Using *MicroMLP* to learn the XOr problem (with hyperbolic tangent) :

```
from microMLP import MicroMLP

mlp = MicroMLP.Create( neuronsByLayers          = [2, 2, 1],
                       activationFuncName       = MicroMLP.ACTFUNC_TANH,
                       layersAutoConnectFunction = MicroMLP.LayersFullConnect )

nnFalse = MicroMLP.NNValue.FromBool(False)
nnTrue  = MicroMLP.NNValue.FromBool(True)

mlp.AddExample( [nnFalse, nnFalse], [nnFalse] )
mlp.AddExample( [nnFalse, nnTrue ], [nnTrue  ] )
mlp.AddExample( [nnTrue  , nnTrue  ], [nnFalse] )
mlp.AddExample( [nnTrue  , nnFalse], [nnTrue  ] )


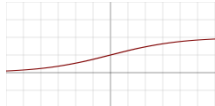

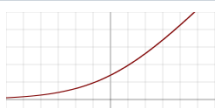

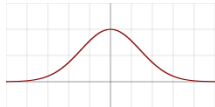
learnCount = mlp.LearnExamples()

print( "LEARNED :" )
print( " - False xor False = %s" % mlp.Predict([nnFalse, nnFalse])[0].AsBool )
print( " - False xor True  = %s" % mlp.Predict([nnFalse, nnTrue ])[0].AsBool )
print( " - True  xor True   = %s" % mlp.Predict([nnTrue  , nnTrue  ])[0].AsBool )
print( " - True  xor False = %s" % mlp.Predict([nnTrue  , nnFalse])[0].AsBool )

if mlp.SaveToFile("mlp.json") :
    print( "MicroMLP structure saved!" )
```

Variable	Description	Default
mlp.Eta	Weighting of the error correction	0.30

Variable	Description	Default
mlp.Alpha	Strength of connections plasticity	0.75
mlp.Gain	Network learning gain	0.99
mlp.CorrectLearnedMAE	Threshold of self-learning error	0.02

Graphe	Activation function name	Const	Detail
	"Heaviside"	MicroMLP.ACTFUNC_HEAVISIDE	Heaviside binary step
	"Sigmoid"	MicroMLP.ACTFUNC_SIGMOID	Logistic (sigmoid or soft step)
	"TanH"	MicroMLP.ACTFUNC_TANH	Hyperbolic tangent
	"SoftPlus"	MicroMLP.ACTFUNC_SOFTPLUS	SoftPlus rectifier
	"ReLU"	MicroMLP.ACTFUNC_RELU	Rectified linear unit
	"Gaussian"	MicroMLP.ACTFUNC_GAUSSIAN	Gaussian function

Layers auto-connect function	Detail
MicroMLP.LayersFullConnect	Network fully connected

Using *MicroMLP.Layer* class :

Name	Function
Constructor	<code>layer = MicroMLP.Layer(parentMicroMLP, activationFuncName=None, neuronsCount=0)</code>
GetLayerIndex	<code>idx = layer.GetLayerIndex()</code>

Name	Function
GetNeuron	<code>neuron = layer.GetNeuron(neuronIndex)</code>
GetNeuronIndex	<code>idx = layer.GetNeuronIndex(neuron)</code>
AddNeuron	<code>layer.AddNeuron(neuron)</code>
RemoveNeuron	<code>layer.RemoveNeuron(neuron)</code>
GetMeanSquareError	<code>mse = layer.GetMeanSquareError()</code>
GetMeanAbsoluteError	<code>mae = layer.GetMeanAbsoluteError()</code>
GetMeanSquareErrorAsPercent	<code>mseP = layer.GetMeanSquareErrorAsPercent()</code>
GetMeanAbsoluteErrorAsPercent	<code>maeP = layer.GetMeanAbsoluteErrorAsPercent()</code>
Remove	<code>layer.Remove()</code>

Property	Example	Read/Write
ParentMicroMLP	<code>layer.ParentMicroMLP</code>	get
ActivationFuncName	<code>layer.ActivationFuncName</code>	get
Neurons	<code>layer.Neurons</code>	get
NeuronsCount	<code>layer.NeuronsCount</code>	get

Using *MicroMLP.InputLayer(Layer)* class :

Name	Function
Constructor	<code>inputLayer = MicroMLP.InputLayer(parentMicroMLP, neuronsCount=0)</code>

SetInputVectorNNValues	<code>ok = inputLayer.SetInputVectorNNValues(inputVectorNNValues)</code>
------------------------	--

Using *MicroMLP.OutputLayer(Layer)* class :

Name	Function
------	----------

Constructor	<code>outputLayer = MicroMLP.OutputLayer(parentMicroMLP, activationFuncName, neuronsCount=0)</code>
GetOutputVectorNNValues	<code>outputVectorNNValues = outputLayer.GetOutputVectorNNValues()</code>
ComputeTargetLayerError	<code>ok = outputLayer.ComputeTargetLayerError(targetVectorNNVa</code>

Using *MicroMLP.Neuron* class :

Name	Function
Constructor	<code>neuron = MicroMLP.Neuron(parentLayer)</code>
GetNeuronIndex	<code>idx = neuron.GetNeuronIndex()</code>
GetInputConnections	<code>connections = neuron.GetInputConnections()</code>
GetOutputConnections	<code>connections = neuron.GetOutputConnections()</code>
AddInputConnection	<code>neuron.AddInputConnection(connection)</code>
AddOutputConnection	<code>neuron.AddOutputConnection(connection)</code>
RemoveInputConnection	<code>neuron.RemoveInputConnection(connection)</code>
RemoveOutputConnection	<code>neuron.RemoveOutputConnection(connection)</code>
SetBias	<code>neuron.SetBias(bias)</code>
GetBias	<code>neuron.GetBias()</code>
SetOutputNNValue	<code>neuron.SetOutputNNValue(nnvalue)</code>
ComputeValue	<code>neuron.ComputeValue()</code>
ComputeError	<code>neuron.ComputeError(targetNNValue=None)</code>
Remove	<code>neuron.Remove()</code>

Property	Example	Read/Write
ParentLayer	<code>neuron.ParentLayer</code>	get
ComputedOutput	<code>neuron.ComputedOutput</code>	get
ComputedDeltaError	<code>neuron.ComputedDeltaError</code>	get
ComputedSignalError	<code>neuron.ComputedSignalError</code>	get

Using *MicroMLP.Connection* class :

Name	Function
Constructor	<code>connection = MicroMLP.Connection(neuronSrc, neuronDst, weight=None)</code>
UpdateWeight	<code>connection.UpdateWeight(eta, alpha)</code>

☰ 254 lines (208 sloc) | 9.96 KB

...

Property	Example	Read/Write
NeuronSrc	<code>connection.NeuronSrc</code>	get
NeuronDst	<code>connection.NeuronDst</code>	get
Weight	<code>connection.Weight</code>	get

Using *MicroMLP.Bias* class :

Name	Function
Constructor	<code>bias = MicroMLP.Bias(neuronDst, value=1.0, weight=None)</code>
UpdateWeight	<code>bias.UpdateWeight(eta, alpha)</code>
Remove	<code>bias.Remove()</code>

Property	Example	Read/Write
NeuronDst	<code>bias.NeuronDst</code>	get
Value	<code>bias.Value</code>	get
Weight	<code>bias.Weight</code>	get

Using *MicroMLP.NNValue* static functions :

Name	Function
FromPercent	<code>nnvalue = MicroMLP.NNValue.FromPercent(value)</code>
NewPercent	<code>nnvalue = MicroMLP.NNValue.NewPercent()</code>
FromByte	<code>nnvalue = MicroMLP.NNValue.FromByte(value)</code>
NewByte	<code>nnvalue = MicroMLP.NNValue.NewByte()</code>
FromBool	<code>nnvalue = MicroMLP.NNValue.FromBool(value)</code>

Name	Function
NewBool	<code>nnvalue = MicroMLP.NNValue.NewBool()</code>
FromAnalogSignal	<code>nnvalue = MicroMLP.NNValue.FromAnalogSignal(value)</code>
NewAnalogSignal	<code>nnvalue = MicroMLP.NNValue.NewAnalogSignal()</code>

Using *MicroMLP.NNValue* class :

Name	Function
Constructor	<code>nnvalue = MicroMLP.NNValue(minValue, maxValue, value)</code>

Property	Example	Read/Write
AsFloat	<code>nnvalue.AsFloat = 639.513</code>	get / set
AsInt	<code>nnvalue.AsInt = 12345</code>	get / set
AsPercent	<code>nnvalue.AsPercent = 65</code>	get / set
AsByte	<code>nnvalue.AsByte = b'\x75'</code>	get / set
AsBool	<code>nnvalue.AsBool = True</code>	get / set
AsAnalogSignal	<code>nnvalue.AsAnalogSignal = 0.39472</code>	get / set

By JC`zic for **HC²** ;')

Keep it simple, stupid 👍