Multiclass Neural Network

Updated: October 12, 2015

Creates a multiclass classification model using a neural network algorithm

Category: Machine Learning / Initialize Model / Classification (https://msdn.microsoft.com/en-us/library/azure/dn905808.aspx)

Module Overview

You can use the **Multiclass Neural Network** module to create a neural network model that can be used to predict a target that has multiple values. For example, neural networks are frequently used in complex computer vision tasks, such as digit or letter recognition, document classification, and pattern recognition.

Classification using neural networks is a supervised learning method, and therefore requires a *tagged dataset* that includes a label column.

You can train the model by providing the model and the tagged dataset as an input to Train Model (https://msdn.microsoft.com/en-us/library/azure/dn906044.aspx) or to Sweep Parameters (https://msdn.microsoft.com/en-us/library/azure/dn905810.aspx). The trained model can then be used to predict values for the new input examples.

Understanding Neural Networks

A neural network is a set of interconnected layers, in which the inputs lead to outputs by a series of weighted edges and nodes. The weights on the edges are learned when training the neural network on the input data. The direction of the graph proceeds from the inputs through the hidden layer, with all nodes of the graph connected by the weighted edges to nodes in the next layer.

Most predictive tasks can be accomplished easily with only one or a few hidden layers, although recent research has shown that deep neural networks (DNN) can be very effective for complex tasks (such as image or speech recognition), in which a successive layers model increases the levels of semantic depth.

To compute the output of the network for any given input, a value is calculated for each node in the hidden layers and in the output layer. For each node, the value is set by calculating the weighted sum of the values of the nodes in the previous layer and applying an activation function to that weighted sum.

How to Configure a Neural Network Model

1. Specify how you want the model to be trained, by setting the **Create trainer mode** option.

    - **Single Parameter**

      If you know how you want to configure the neural network, you can provide a specific set of values as arguments. You might have learned these values by experimentation or received them as guidance.

    - **Parameter Range**

      If you are not sure of the best parameters, you can find the optimal parameters by specifying multiple values and using a parameter sweep to find the optimal configuration.

      During training, all combinations of the settings you provided will be tested to determine the combination of settings that produces the optimal results.

2. You can further customize the neural network model in two ways:

    - Accept the default neural network architecture, and use the **Properties** pane to set other parameters that control the behavior of the neural network, such as learning rate and normalization. For more information, see the Options section.

    - If you want to fully customize the network architecture and its connections, use the Net# language (https://azure.microsoft.com/documentation/articles/machine-learning-azure-ml-netsharp-reference-guide/).

3. Connect a tagged dataset and train the model.

    - If you set **Create trainer mode** option to **Single Parameter**, train the model by using a tagged dataset and the Train Model (https://msdn.microsoft.com/en-us/library/azure/dn906044.aspx) module.

    - If you set **Create trainer mode** option to **Parameter Range**, train the model using Sweep Parameters (https://msdn.microsoft.com/en-us/library/azure/dn905810.aspx) and a tagged dataset.

      You can then use the trained model, or you can make a note of the parameter settings to use when configuring a learner.

Options

You can use these options to modify the behavior of the neural network.

Create trainer mode

Choose the method used for configuring and training the model:

- ***Single Parameter***

  Select this option to configure and train the model with a single set of parameter values that you supply.

If you choose this option, you should train the model by using the Train Model (https://msdn.microsoft.com/en-us/library/azure/dn906044.aspx) module.

- ***Parameter Range***

  Select this option to use the Range Builder and specify a range of possible values. You then train the model using a parameter sweep, to find the optimum configuration.

> **Warning**
>
> - If you pass a parameter range to Train Model (https://msdn.microsoft.com/en-us/library/azure/dn906044.aspx), it will use only the first value in the parameter range list.
> - If you pass a single set of parameter values to the Sweep Parameters (https://msdn.microsoft.com/en-us/library/azure/dn905810.aspx) module, when it expects a range of settings for each parameter, it ignores the values and using the default values for the learner.
> - If you select the **Parameter Range** option and enter a single value for any parameter, that single value you specified will be used throughout the sweep, even if other parameters change across a range of values.

Hidden layer specification

Select the type of network architecture to create.

- ***Fully-connected case***

  Choose this option to use the default neural network architecture.

  The default multiclass neural network is defined as follows:

  - The neural network model has one hidden layer.
  - The output layer is fully connected to the hidden layer, and the hidden layer is fully connected to the input layer.
  - The number of nodes in the input layer is determined by the number of features in the training data.
  - The number of nodes in the hidden layer is determined by the user (with a default value of 100).
  - The number of nodes in the output layer depends on the number of classes.

- ***Custom definition script***

Choose this option to use the Net# language to create a custom neural network architecture.

Neural network definition

Type or paste Net# statements in the text box to customize the architecture of the neural network, including the number of hidden layers, their connections, and advanced options such as specifying the mappings between layers.

This option is available only if you select the **Custom definition script** option.

For script examples, see Customizing a Neural Network by Using Net#.

Number of hidden nodes

Type the number of hidden nodes.

By default, there is one hidden layer with 100 nodes.

Learning rate

Type a value that defines the step taken at each iteration, before correction.

A larger value for learning rate can cause the model to converge faster, but it can overshoot local minima.

Number of learning iterations

Specify the maximum number of times the algorithm processes the training cases.

The initial learning weights diameter

Specify the node weights at the start of the learning process.

The momentum

Specify a weight to apply during learning to nodes from previous iterations

The type of normalizer

Choose the method that is used for feature normalization, or use no normalization. The following normalization methods are supported:

- ***Binning normalizer***: The binning normalizer creates bins of equal size, and then normalizes every value in each bin to be divided by the total number of bins.

- ***Gaussian normalizer***: The Gaussian normalizer rescales the values of each feature to have mean 0 and variance 1. This is done by computing the mean and the variance of each feature, and then, for each instance, subtracting the mean value and dividing by the square root of the variance (the standard deviation).

- ***Min-max normalizer***: The min-max normalizer linearly rescales every feature to the [0,1] interval.

  Rescaling to the [0,1] interval is done by shifting the values of each feature so that the minimal value is 0, and then dividing by the new maximal value (which is the difference between the original maximal and minimal values).

- ***Do not normalize***: No normalization is performed.

Shuffle examples

Select this option to shuffle cases between iterations.

If you deselect this option, cases are processed in exactly the same order each time you run the experiment.

Random number seed

Type a value to use as the seed.

Specifying a seed value is useful when you want to ensure repeatability across runs of the same experiment.

Allow unknown categorical levels

When this option is selected, the model will create a grouping for Unknown values in the training and validation sets.

If you deselect it, the model can accept only the values contained in the training data. In the former case, the model might be less precise on known values but provide better predictions for new (unknown) values.

Customizing a Neural Network by Using Net#

A neural network model is defined by the structure of its graph, which includes these attributes:

- Number of hidden layers
- Number of nodes in each hidden layer
- Connections between the layers
- Choice of activation functions
- Weights on the graph edges

The structure of the graph and the activation function are determined by the user. The weights on the edges are learned when training the neural network on the input data.

- The first layer is always the input layer.
- The last layer is always the output layer. The number of nodes in the output layer should be equal to the number of classes.
- You can define any number of intermediate layers (sometimes called hidden layers, because they are contained within the model and are not directly exposed as endpoints).

You can change the architecture of neural network models that you create by using the Net# language:

- Creating hidden layers and controlling the number of nodes in each layer.
- Specifying how layers are to be connected to each other.

- Defining special connectivity structures, such as convolutions and weight sharing bundles.
- Specifying different activation functions.

The Net# reference guide explains the syntax and provides sample network definitions. It explains how you can use Net# to add hidden layers and define the way that the layers interact with each other. For more information, see:

Guide to the Net# Neural Networks Specification Language (http://go.microsoft.com/fwlink/?LinkId=402867)

Examples

For examples of how this learning algorithm is used, see these sample experiments in the Model Gallery (http://gallery.azureml.net/). The experiments are related and described in a single document that progresses from basic to advanced configurations:

- Deep Neural networks example (part A) (http://go.microsoft.com/fwlink/?LinkId=525278)
- Deep Neural networks example (part B) (http://go.microsoft.com/fwlink/?LinkId=525279)
- Deep Neural networks example (part C) (http://go.microsoft.com/fwlink/?LinkId=525280)
- Deep Neural networks example (part D) (http://go.microsoft.com/fwlink/?LinkId=525281)

Technical Notes

A neural network can be thought of as a weighted directed acyclic graph. The nodes of the graph are arranged in layers, and they are connected by weighted edges to nodes in the next layer. To compute the output of the network on a given input example, a value is calculated for each node in the hidden layers and in the output layer. For each node, the value is set by calculating the weighted sum of the values of the nodes in the previous layer and applying an activation function to that weighted sum.

Warning
Neural networks can be computationally expensive, due to a number of hyperparameters and the introduction of custom network topologies. Although in many cases neural networks produce better results than other algorithms, obtaining such results may involve fair amount of sweeping (iterations) over hyperparameters.

Module Parameters

Name	Range	Type	Default	Description
Hidden layer specification	List	Neural Network Topology	Fully-connected case	Specify the architecture of the hidden layer or layers
The initial learning weights diameter	>=double.Epsilon	Float	0.1	Specify the node weights at the start of the learning process
The learning rate	[double.Epsilon;1.0]	Float	0.1	Specify the size of each step in the learning process
The momentum	[0.0;1.0]	Float	0.0	Specify a weight to apply during learning to nodes from previous iterations
Neural network definition	Any	StreamReader		When you select **Custom definition script**, type a valid script expression on each line to define the layers, nodes, and behavior of a custom neural network
The type of normalizer	List	Normalization Method	Minimum-maximum normalizer	Select the type of normalization to apply to learning examples
Number of learning iterations	>=1	Integer	100	Specify the number of iterations while learning
Shuffle examples	Any	Boolean	True	Select this option to change the order of instances between learning iterations
Random number seed	Any	Integer		Specify a numeric seed to use for random number generation. Leave blank to use the default seed.

Allow unknown categorical levels	Any	Boolean	True	Indicate whether an additional level should be created for unknown categories. If the test dataset contains categories that are not present in the training dataset, they are mapped to this unknown level.

Output

Name	Type	Description
Untrained model	ILearner interface (https://msdn.microsoft.com/en-us/library/azure/dn905938.aspx)	An untrained multiclass classification model

See Also

Machine Learning / Initialize Model / Classification (https://msdn.microsoft.com/en-us/library/azure/dn905808.aspx)
Two-Class Neural Network (https://msdn.microsoft.com/en-us/library/azure/dn905947.aspx)
Neural Network Regression (https://msdn.microsoft.com/en-us/library/azure/dn905924.aspx)
A-Z List of Machine Learning Studio Modules (https://msdn.microsoft.com/en-us/library/azure/dn906033.aspx)