Cross-Validate Model

Updated: July 22, 2015

Cross-validates parameter estimates for classification or regression models by partitioning the data

Category: Machine Learning / Evaluate (https://msdn.microsoft.com/en-us/library/azure/dn906026.aspx)

Module Overview

You can use **Cross-Validate Model** to evaluate an untrained classification or regression model. Cross-validation is a standard technique used in machine learning to assess both the variability of a dataset and the reliability of any model trained using that data.

Cross-Validate Model takes an untrained classification or regression model and a dataset as an input. It divides the dataset into some number of subsets (*folds*), builds a model on each, and then returns a set of accuracy statistics for each. By comparing the accuracy statistics for each fold you can interpret the quality of the data set and understand whether the model is susceptible to variations in the data.

Cross-validate also returns scored labels and score probabilities for the outcome you are predicting with the model, so that you can assess the reliability of the predictions.

Understanding Cross-Validation

1. Cross validation randomly splits the training data into a number of partitions, or *folds*. The algorithm defaults to 10 folds if you have not previously partitioned the dataset.

   Tip

   To divide the dataset into a different number of folds, you can use the Partition and Sample (https://msdn.microsoft.com/en-us/library/azure/dn905960.aspx) module and indicate how many folds to use.

2. The module begins with fold 1, sets aside the data in fold 1 to use for validation (called the *holdout fold*), and uses the remaining folds to train a model.

   For example, if you use the default 10 folds, you would create 10 models during cross-validation, each model trained using 9/10th of the data, and tested on the remaining 1/10th.

   Note: You can use any of the machine learning algorithms provided in Azure Machine Learning Studio.

3. During testing of the model for each fold, multiple accuracy statistics are evaluated. Which statistics are used depends on whether you are evaluating a classification model or regression model.

4. When the building and evaluation process is complete for all folds, **Cross-Validate Model** generates a set of performance metrics and scored results for all folds.

Another common way of evaluating a model is to divide the data into a training and test set using Split (https://msdn.microsoft.com/en-us/library/azure/dn905969.aspx), and then validate the model on the training data. However, there are some advantages when using **Cross-Validate Model**.

- **Uses more test data**

  Cross-validation measures the performance of the model with the specified parameters in a bigger data space – basically it uses the entire training dataset for both training and evaluation, instead of some portion of it. In contrast, if you validate a model by using data generated from a random split, typically you evaluate the model only on 30% or less of the available data.

  However, because **Cross-Validate Model** trains and validates the model multiple times over a larger dataset, it is much more computationally intensive and takes much longer than validating on a random split.

- **Evaluates dataset as well as model**

  Cross-validation doesn’t simply measure the accuracy of a model, but also gives you some idea of how representative the dataset is and how sensitive any model you create might be to variations in the data.

For these reasons, you might use **Cross-Validate Model** in the initial phase of building and testing your model, to evaluate the goodness of the model parameters (assuming that computation time is tolerable), and then train and evaluate your model using the established parameters with the Train Model (https://msdn.microsoft.com/en-us/library/azure/dn906044.aspx) and Evaluate Model (https://msdn.microsoft.com/en-us/library/azure/dn905915.aspx) modules.

How to Use Cross-Validate Model

Cross-Validate Model is simple to use, though it can take a long time to run if you use a lot of data.

1. First, connect the untrained model to evaluate. The model can be untrained because **Cross-Validate Model** automatically trains the model as part of evaluation.

2. Connect the training dataset.

3. If you want to be able to repeat the results exactly, set a value for the **Random seed** parameter.

4. **Cross-Validate Model** randomly splits the training data into 10 folds. A fold is a subset of the original dataset.

   If you want to use some other number of partitions, use Partition and Sample (https://msdn.microsoft.com/en-us/library/azure/dn905960.aspx) to create the desired number of partitions.

5. In each iteration over the dataset, **Cross-Validate Model** uses one fold as a validation dataset, and uses the remaining folds to train a model.

6. Each of the 10 models is tested against the data in all the other folds.

7. After all iterations are complete, **Cross-Validate Model** creates a set of performance metrics for all models, as well as scores for the entire dataset.

8. To view the results, in the experiment, right-click the output port of the **Cross-Validate Model** module and select **View Results**.

Results

Cross-Validate Model returns a set of evaluation metrics for each fold, and a summary for the entire dataset.

The first output dataset contains scores and probabilities for the model, as well as the source data for each row.

- **Fold assignment.** Indicates which fold each row of data was assigned to during cross-validation.

- **Scored Labels.** The predicted value for each row.

- **Scored Probabiities.** A probability score for the current predicted value.

The second output contains a summary report. A careful review and comparison of fold results can help you identify irregularities in the dataset or the model.

- **Fold number.** An identifier for the current fold results.

- **Number of examples in fold.** The count of rows assigned to each fold.

- **Model.** The type of learner that was used in creating the models.

- **Metrics.** The metrics that are provided depend on the type of model that you are evaluating.

  Classification models: Precision, recall, F-score, AUC, average log loss, and training log loss metrics.

  Regression models: Mean absolute error, root mean squared error, relative absolute error, and relative squared error metrics

If you would like to see additional examples of how to use cross-validation and interpret the results, see the examples in the Model Gallery.

By looking at the results for each of the hold-out folds, you can learn these tihngs:

1. **Fold 7** is in the first row. From the name, Fold 7, you can tell that the model was trained on the data in folds 0-6 and folds 8-9. The model was then evaluated using the data in Fold 7.

   For Fold 7, the **Scored Probability** is 0.132534, and the **Scored Labels** is <=50K.

   This means that the Fold 7 model predicts the label of this particular row as <=50K, with a predicted probability of 0.132534.

2. Similarly, the second row is assigned to **Fold 1**. Therefore, this particular model was trained on the data in folds 0 and folds 2-9.

   This model also predicts the value of **Scored Labels** for this row as <=50K, but in this model, the **Score Probability** is 0.074428.

Examples

For examples of how cross-validation is used in machine learning, see these sample experiments in the Model Gallery (http://gallery.azureml.net/):

- The Cross validation for binary classifier (http://go.microsoft.com/fwlink/?LinkId=525734) sample demonstrates how to use cross-validation with a binary classification model.
- The Cross Validation Regression: Auto Imports Dataset (http://go.microsoft.com/fwlink/?LinkId=525735) sample demonstrates how to use cross validation with regression models and how to interpret the results.

Technical Notes

It is a best practice to normalize datasets before using them for cross-validation.

Because **Cross-Validate Model** trains and validates the model multiple times, it is much more computationally intensive and takes longer time than validating on random split to complete. We recommend that you use **Cross-Validate Model** to establish the goodness of the model given the specified parameters. We also recommend that you use Sweep Parameters (https://msdn.microsoft.com/en-us/library/azure/dn905810.aspx) to identify the optimal parameters.

Expected Inputs

Name	Type	Description
Untrained model	ILearner interface (https://msdn.microsoft.com/en-us/library/azure/dn905938.aspx)	Untrained model for cross validation on dataset
Dataset	Data Table (https://msdn.microsoft.com/en-us/library/azure/dn905851.aspx)	Input dataset

Module Parameters

Name	Range	Type	Default	Description
Label column	any	ColumnSelection		Select the column that contains the label to use for validation
Random seed	any	Integer	0	Seed value for random number generator This value is optional. If not specified

Outputs

Name	Type	Description
Scored results	Data Table (https://msdn.microsoft.com/en-us/library/azure/dn905851.aspx)	Results of scoring
Evaluation results by fold	Data Table (https://msdn.microsoft.com/en-us/library/azure/dn905851.aspx)	Results of evaluation (by fold and entire)

Exceptions

Exception	Description
Error 0035 (https://msdn.microsoft.com/en-us/library/azure/dn906038.aspx)	Exception occurs if no features were provided for a given user or item.
Error 0032 (https://msdn.microsoft.com/en-us/library/azure/dn905829.aspx)	Exception occurs if argument is not a number.
Error 0033 (https://msdn.microsoft.com/en-us/library/azure/dn905872.aspx)	Exception occurs if argument is Infinity.
Error 0001 (https://msdn.microsoft.com/en-us/library/azure/dn905993.aspx)	Exception occurs if one or more specified columns of data set couldn't be found.
Error 0003 (https://msdn.microsoft.com/en-us/library/azure/dn906003.aspx)	Exception occurs if one or more of inputs are null or empty.
Error 0006 (https://msdn.microsoft.com/en-us/library/azure/dn905885.aspx)	Exception occurs if parameter is greater than or equal to the specified value.
Error 0008 (https://msdn.microsoft.com/en-us/library/azure/dn905856.aspx)	Exception occurs if parameter is not in range.
Error 0013 (https://msdn.microsoft.com/en-us/library/azure/dn906041.aspx)	Exception occurs if the learner that is passed to the module has an invalid type.

See Also

Machine Learning / Evaluate (https://msdn.microsoft.com/en-us/library/azure/dn906026.aspx)
Evaluate Recommender (https://msdn.microsoft.com/en-us/library/azure/dn905954.aspx)
A-Z List of Machine Learning Studio Modules (https://msdn.microsoft.com/en-us/library/azure/dn906033.aspx)