Initializing & Optimizing Machine Learning Models

Eng Teong Cheah
MVP Visual Studio &
Development Technologies

Agenda

Using hyperparemeters

Using multiple algorithms & models

Scoring and evaluating models

Using hyperparameters

Choose parameters to optimize your algorithms

There are 4 steps in the process of finding the best parameter set:

Define the parameter space

For the algorithm, first decide the exact parameter values you want to consider.

Define the cross-validation settings

Decide how to choose cross-validation folds for the datasets

Define the metric

Decide what metric to use for determining the best set of parameters, such as accuracy, root mean squared, precision, recall, or f-score.

Train, evaluate, and compare

For each unique combination of the parameter values, cross-validation is carried out by and based on the error metric you define. After evaluation and comparison, you can choose the best-performing model.

Tune Model Hyperparameters

The terms ***parameter*** and ***hyperparamete***r can be confusing.

The model's ***parameters*** are what you set in the properties pane.

Basically, this module performs a ***parameter sweep*** over the specified parameter settings, and learns an optimal set of ***hyperparameters***, which might be different for each specific decision tree, dataset, or regression method.

Tune Model Hyperparameters

The module support 2 methods for finding optimum settings for a model:

Integrated train and tune

You configure a set of parameters to use, and then let the module iterate over multiple combinations, measuring accuracy until it finds a “best” model. With most learner modules, you can choose which parameters should be changed during the training process, and which should remain fixed.

Tune Model Hyperparameters

Depending on how long you want the tuning process to run, you might decide to exhaustively test all combinations, or you could shorten the process by establishing a grid of parameter combinations and testing a randomized subset of the parameter grid.

Tune Model Hyperparameters

Cross validation with tuning

With this option, you divide your data into some number of folds and then build test models on each fold. This method provides the best accuracy and can help find problems with the dataset; however, it takes longer to train.

Using multiple algorithms & model

Considerations when choosing an algorithm

Accuracy

Getting the most accurate possible isn't always necessary. Sometimes an approximation is adequate, depending on what you want to use it for.

If that's the case, you may be able to cut your processing time dramatically by sticking with more approximate methods. Another advantage of more approximate methods is that they naturally tend to avoid overfitting.

Considerations when choosing an algorithm

Training time

The number of minutes or hours necessary to train a model varies a great deal between algorithms.

Training time is often closely tied to accuracy – one typically accompanies the other.

In addition, some algorithms are more sensitive to the number of data points than others.

When time is limited it can drive the choice of algorithm, especially when the data set is large.

Considerations when choosing an algorithm

Linearity

Lots of machine learning algorithms make use of linearity.

Linear classification algorithms assume that classes can be separated by a straight line (or its higher-dimensional analog).

These include logistic regression and support vector machines(as implemented in Azure ML).

Linear regression algorithms assume that data trends follow a straight line.

These assumptions aren't bad for some problems, but on others they bring accuracy down.

Considerations when choosing an algorithm

Number of parameters

Parameters are the knobs a data scientist gets to turn when setting up an algorithm.

They are numbers that affect the algorithm's behavior, such as error tolerance or number of iterations or options between variants of how the algorithm behaves.

The training time and accuracy of the algorithm can sometimes be quite sensitive to getting just the right settings.

Typically, algorithms with large numbers parameters require the most trial and error to find a good combination.

Considerations when choosing an algorithm

Number of features

For certain types of data, the number of features can be very large compared to the number of data points.

This is often the case with genetics or textual data.

The large number of features can bog down some learning algorithms, making training time unfeasibly long.

Support Vector Machines are particularly well suited to this case

Scoring and evaluating models

Scoring models

Scoring is widely used in machine learning to mean the process generating new values, given a model and some new input.

The generic term “score” is used, rather than “prediction”, because the scoring process can generate so many different types of values:

Scoring models

- A list of recommended items and a similarity score.
- Numeric values, for time series models and regression models.
- A probability value, indicating the likelihood that a new input belongs to some existing category.
- The name of a category or cluster to which a new item is most similar.
- A predicted class or outcome, for classification models.

List of scoring models

Machine Learning Studio provides many different scoring modules.

You select one depending on the type of model you are using, or type of scoring task you are performing:

Apply Transformation

Applies a well-specified data transformation to a dataset.
Use this model to apply a saved process to a set of data.

Evaluating models

The Machine Learning – Evaluate category includes the following modules:

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

Evaluating models

Evaluate Model

Evaluates a score classification or regression model by using standard metrics.

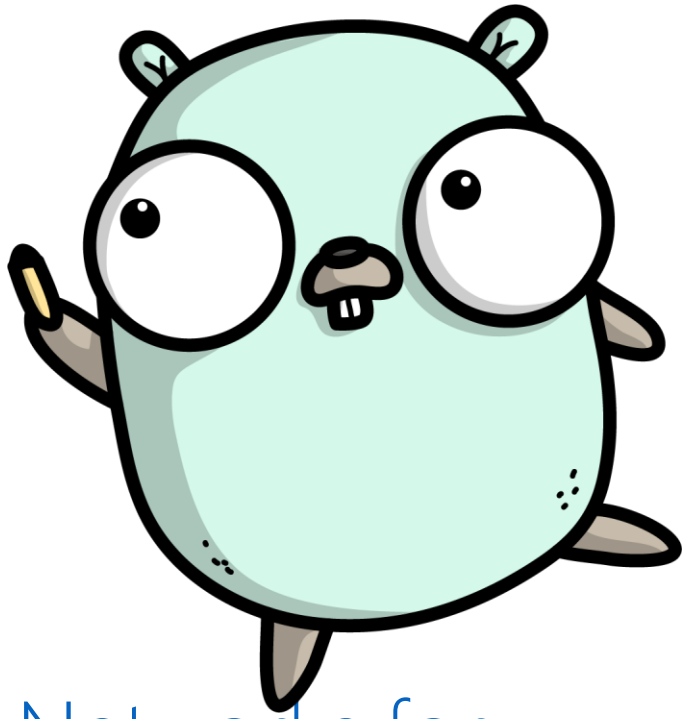
Evaluate Recommender

Evaluates a score classification or regression model by using standard metrics.

Demo

Create a tuned model

Resources



TutorialsPoint

Microsoft Docs

Lecture Collection | Convolutional Neural Networks for Visual Recognition(Spring 2017)

Python Numpy Tutorial

Image Credits: @ashleymcnamara

Thank you

Eng Teong Cheah
Microsoft MVP Visual Studio & Development Technologies
Twitter: @walkercet
Github: https://github.com/ceteongvanness
Blog: https://ceteongvanness.wordpress.com/
Youtube: http://bit.ly/etyoutubechannel