Machine Learning - Lecture 5: Cross-validation

Chris Thornton

October 18, 2011

Introduction

In general, ML involves deriving models from data, with the aim of achieving some kind of desired behaviour, e.g., prediction or classification.

But this generic task is broken down into a number of special cases.

We also have a range of ways in which the performance of methods is assessed and described.

This lecture will examine some of the concepts involved.

Class/output variables and values

In the majority of cases, ML methods model data with the aim of predicting the value of one of the variables of the dataset.

The to-be-predicted variable is called the **output variable**.

Correct values of the output variable are called **target output values**, or just **targets**.

If the output variable takes categorical values, it may be called the **class variable**, in which case targets may be called **target classes**.

In some cases, there may be multiple output variables, but this is quite unusual.

Attribute/input variables and values

The other variables are called **input variables**.

Their values are **input values**.

If they take categorical values, they may be called **attributes** or **features**.

Their values are then **attribute values** or **feature values**.

A complete set of input values may be called a **vector**, **attribute vector** or **feature vector**.

Confusingly, input vectors are also sometimes just called **inputs**.

Supervised v. unsupervised learning

Where the datapoints are examples associating particular output value(s) with particular input vectors, the method is said to being doing **supervised learning**.

Any other scenario is then some form of **unsupervised learning**.

However, the latter term is usually reserved for the case where a model is built without any pre-classification of variables.

Classification v. regression

If the values of the output variable are names of categories, supervised learning is also called **classification**.

If the values are real numbers, the task is called **regression**.

A simple but popular case of classification is **concept learning**.

This is where the aim is to predict whether an input vector is or is not a member of a particular class.

Values of the output variable in this case are usually given as +/-, 1/0, or yes/no.

Seens v. Unseens

In any supervised learning experiment, the set of input/output examples provided is called the **training set**.

We then usually have a second set of examples, called the **testing set**, which is used solely for testing generalization performance, i.e., ability of the model to produce correct output values for input vectors that do not appear in the training set.

Cases in the training set may be called **training examples**.

Statisticians are more likely to call them **seen cases**, or just **seens**.

Cases in the testing set may be called **testing examples**, **unseen cases** or just **unseens**.

Error v. Error rate

One of the advantages of supervised learning is that we can use testing sets to get an objective measurement of learning performance.

The inaccuracy of predicted output values is termed the **error** of the method.

If target values are categorical, the error is expressed as an **error rate**.

This is the proportion of cases where the prediction is wrong.

Off-training set error

Error-rate is normally measured on the testing set only. In this case, it may also be called the **off-training-set error-rate** or **OTS error**.

It may also be called the **prediction error-rate**, or **generalization error-rate**.

If error is calculated on the training set, then it would be called the **training error-rate**.

Error-rate example

Let's say a machine-learning method is provided with this training set.

```
petrol   hatchback   FW-drive   Ford
diesel   saloon      FW-drive   Ford
petrol   formula-1   FW-drive   Ferrari
petrol   convertible FW-drive   Ford
```

The testing set has two cases:

```
petrol   convertible RW-drive
diesel   hardtop     FW-drive
```

Using a 1-NN method, both inputs are classified as Fords.

However, it turns out that the first case is in fact a Ferrari.

The model gets 1 out of 2 classifications wrong.

The error rate is 50%.

IID assumption

For error measurements to make any sense, it is vital we have no overlap between training and testing examples.

On the other hand, we need to be sure that both sets of data originate from the same source or domain.

If they don't, there's no reason to expect that a model built for one will apply to the other.

In ML, we normally handle this by requiring the training and testing data to be **identically and independently distributed**.

It is a requirement that the testing data show the same statistical distribution as the training data.

But they must also be completely independent of the training data.

This is known as the **IID** assumption.

Holdout sets

In addition to training and testing sets, we can also have **holdout sets**.

A holdout set is a (usually) small set of input/output examples held back for purposes of tuning the modeling.

The modeling process gets to see all the training data in the usual way.

But it then gets tested on the cases held back and the performance measurements obtained are used to control the modeling in some way (e.g., set a parameter).

Note that this is completely separate from use of a testing set, which is used for obtaining a final evaluation.

For example, we might hold back 10% of the training data and try to find the optimal value of k in k-means clustering by seeing which value gives the lowest error-rate on the holdout data.

Cross-validation

In the simplest case, holdout (or testing) sets are constructed just by splitting some original dataset into more than one part.

But the evaluations obtained in this case tend to reflect the particular way the data are divided up.

The solution is to use statistical sampling to get a more accurate measurments.

This is called **cross-validation**.

Cross-validation strategies

The aim in cross-validation is to ensure that every example from the original dataset has the same chance of appearing in the training and testing set.

The basic protocols are

Cross-validation strategies

The aim in cross-validation is to ensure that every example from the original dataset has the same chance of appearing in the training and testing set.

The basic protocols are

- n-fold cross-validation: divide the data up into n chunks and train n times, treating a different chunk as the holdout set each time.

Cross-validation strategies

The aim in cross-validation is to ensure that every example from the original dataset has the same chance of appearing in the training and testing set.

The basic protocols are

- n-fold cross-validation: divide the data up into n chunks and train n times, treating a different chunk as the holdout set each time.
- leave-one-out validation: just like n-fold cross-validation except that chunks contain a single datapoint.

Cross-validation strategies

The aim in cross-validation is to ensure that every example from the original dataset has the same chance of appearing in the training and testing set.

The basic protocols are

- n-fold cross-validation: divide the data up into n chunks and train n times, treating a different chunk as the holdout set each time.
- leave-one-out validation: just like n-fold cross-validation except that chunks contain a single datapoint.

Summary

Summary

- Inputs, outputs, classes, attributes

Summary

- Inputs, outputs, classes, attributes
- Supervised v. unsupervised learning

Summary

- Inputs, outputs, classes, attributes
- Supervised v. unsupervised learning
- Training sets, testing sets, seens and unseens

Summary

- Inputs, outputs, classes, attributes
- Supervised v. unsupervised learning
- Training sets, testing sets, seens and unseens
- Various types of error

Summary

- Inputs, outputs, classes, attributes
- Supervised v. unsupervised learning
- Training sets, testing sets, seens and unseens
- Various types of error
- IID assumption

Summary

- Inputs, outputs, classes, attributes
- Supervised v. unsupervised learning
- Training sets, testing sets, seens and unseens
- Various types of error
- IID assumption
- Holdout sets

Summary

- Inputs, outputs, classes, attributes
- Supervised v. unsupervised learning
- Training sets, testing sets, seens and unseens
- Various types of error
- IID assumption
- Holdout sets
- Cross-validation protocols

Summary

- Inputs, outputs, classes, attributes
- Supervised v. unsupervised learning
- Training sets, testing sets, seens and unseens
- Various types of error
- IID assumption
- Holdout sets
- Cross-validation protocols

Questions

Questions

- How many input variables are allowed in a training example?

Questions

- How many input variables are allowed in a training example?
- In the context of a training problem, what is the difference between an input and an input *value*?

Questions

- How many input variables are allowed in a training example?
- In the context of a training problem, what is the difference between an input and an input *value*?
- What is the solution to the 'problem' specified by a set of training data?

Questions

- How many input variables are allowed in a training example?
- In the context of a training problem, what is the difference between an input and an input *value*?
- What is the solution to the 'problem' specified by a set of training data?
- In the context of a training problem, what set of inputs should the derived model be able to handle?

Questions

- How many input variables are allowed in a training example?
- In the context of a training problem, what is the difference between an input and an input *value*?
- What is the solution to the 'problem' specified by a set of training data?
- In the context of a training problem, what set of inputs should the derived model be able to handle?
- What syntactic differences are there between training and uunseen cases?

Questions

- How many input variables are allowed in a training example?
- In the context of a training problem, what is the difference between an input and an input *value*?
- What is the solution to the 'problem' specified by a set of training data?
- In the context of a training problem, what set of inputs should the derived model be able to handle?
- What syntactic differences are there between training and uunseen cases?
- How is the error rate affected if the model produced from some training examples yields an incorrect output for a test case?

Questions

- How many input variables are allowed in a training example?
- In the context of a training problem, what is the difference between an input and an input *value*?
- What is the solution to the 'problem' specified by a set of training data?
- In the context of a training problem, what set of inputs should the derived model be able to handle?
- What syntactic differences are there between training and uunseen cases?
- How is the error rate affected if the model produced from some training examples yields an incorrect output for a test case?
- What would the error-rate be in the Ford/Ferrari classification task if both cases were classified correctly?

Questions

- How many input variables are allowed in a training example?
- In the context of a training problem, what is the difference between an input and an input *value*?
- What is the solution to the 'problem' specified by a set of training data?
- In the context of a training problem, what set of inputs should the derived model be able to handle?
- What syntactic differences are there between training and uunseen cases?
- How is the error rate affected if the model produced from some training examples yields an incorrect output for a test case?
- What would the error-rate be in the Ford/Ferrari classification task if both cases were classified correctly?