Cse352 Lecture Notes

Classification: Testing Classifier Accuracy

Professor Anita Wasilewska
Computer Science Department
Stony Brook University

Overview

- **Introduction**
- **Basic Concept** on **training** and **testing**
- **Main Methods** of **predictive accuracy** evaluations

Predictive Accuracy Evaluation

The **main methods** of **predictive accuracy** evaluations are:

- **Resubstitution** (N ; N)
- **Holdout** (2N/3 ; N/3)
- **k-fold cross-validation** (N- N/k ; N/k)
- **Leave-one-out** (N-1 ; 1)

where N is the number of records (instances) in the dataset

Predictive Accuracy

- **REMEMBER:** we must know the classification (class attribute values) of **all instances** (records) used in the test procedure
- **Basic Concepts**

Success: instance (record) **class** is classified **correctly**

Error: instance **class** is classified **incorrectly**

Error rate: a **percentage of errors** made over the **whole set** of instances (records) used for **testing**

Predictive Accuracy: a percentage of **well classified** data in the testing data set.

Correctly and Not Correctly Classified

- A **record is correctly classified** if and only if the following conditions hold:

(1) we **can classify** the record, i.e. **there is a rule** such that its LEFT side **matches** the record,

(2) **classification determined by the rule** is **correct**, i.e. the RIGHT side of the rule **matches** the value of the record's class attribute

OTHERWISE

- **the record is not correctly classified**
- Words used:
- not correctly = incorrectly = misclassified

Predictive Accuracy

- **Example:**

Testing Rules (testing record #1) = record #1.class - Succ
Testing Rules (testing record #2) not= record #2.class - Error
Testing Rules (testing record #3) = record #3.class - Succ
Testing Rules (testing record #4) = instance #4.class - Succ
Testing Rules (testing record #5) not= record #5.class - Error

Error rate:

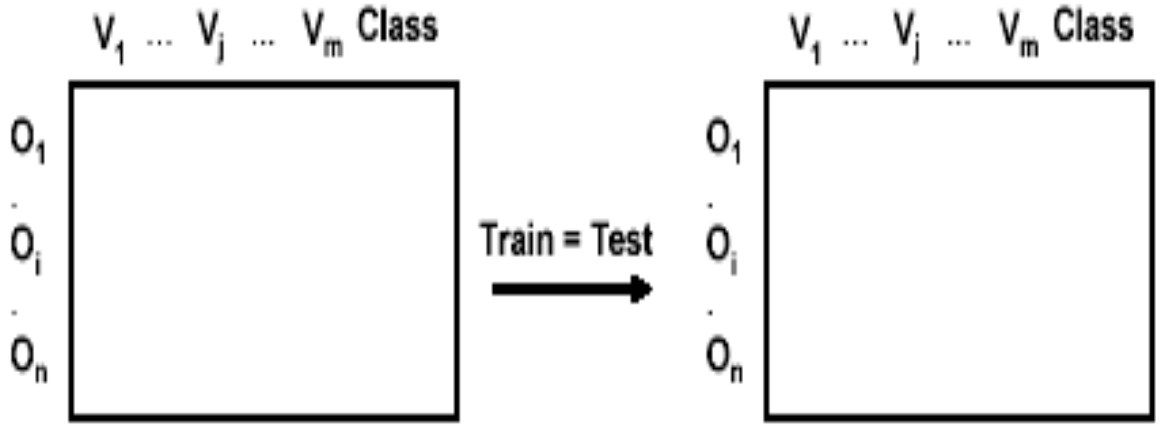
2 errors: #2 and #5

Error rate = 2/5=40%

Predictive Accuracy: 3/5 = 60%

Resubstitution (N ; N)

Testing the classification model by using the given data set („already used for „training")



Re-substitution Error Rate

- **Re-substitution** error rate is obtained from **training data**
- **Training Data Error:** uncertainty of **the rules**
- **The error rate is not always 0%**, but usually (and hopefully) very low!
- **Re-substitution error rate** indicates only how **good (bad)** are our **results** (rules, patterns, NN) on the TRAINING data
- It expresses some **knowledge about the algorithm** used

Re-substitution Error Rate

- **Re-substitution error rate** is usually used as the **performance measure**:

The **training error rate** reflects **imprecision** of the training results

The lower training error rate the better

In the case of **rules** it is called **rules accuracy**

Predictive Accuracy

Predictive accuracy reflects how **good** are the training results with respect to the **test data**

The higher predictive accuracy **the better**

(N:N) re-substitution **does not** compute predictive accuracy

- **Re-substitution** error rate = **training data error rate**

Why not always 0%?

- The **error rate** on the training data is **not always 0%** because **algorithms** involve different (often statistical) parameters and measures that lead to **uncertainties**

- It is used for "**parameters tuning**"

- The **error** on the training data is NOT a good **indicator of performance** on **future data** since it **does not** measure any **not yet seen data**
- **Solution:**

  Split data into **training** and **test** set

Training and test set

- **Training** and **Test** data may differ in nature, but **must have** the same format

Example:

Given customer data from two different towns A and B.

We train the classifier with the data

from town A and we test it on data from town B, and vice-versa

Learning Process

- It is important that the **test data is not used** in any way to create the training **rules**

- In fact, **learning process** operate in three stages:

  Stage 1: build the **basic patterns** structure

  -training

  Stage 2: optimize **parameter settings**; can use (N:N) re-substitution

  - **parameter tuning**

  Stage 3: use **test data** to compute

  predictive accuracy/error rate

Validation Data

- Proper **learning process** uses three sets of data:
- **training data, validation data** and **test data**
  validation data is **used** for parameter tuning
- validation data **is not** a test data
- validation data can be the training data, or a subset of training data
- The **test data** can not be used for parameter tuning!

Training and testing

- Generally, the **larger is** the training set, the **better is** the classifier

- **Larger** test data assures more **accurate** predictive accuracy, or error estimation

- **Remember:**
- the error rate of re-substitution (N;N) can tell us **ONLY** whether the algorithm used in training is good or not good or how good it is

Training and testing

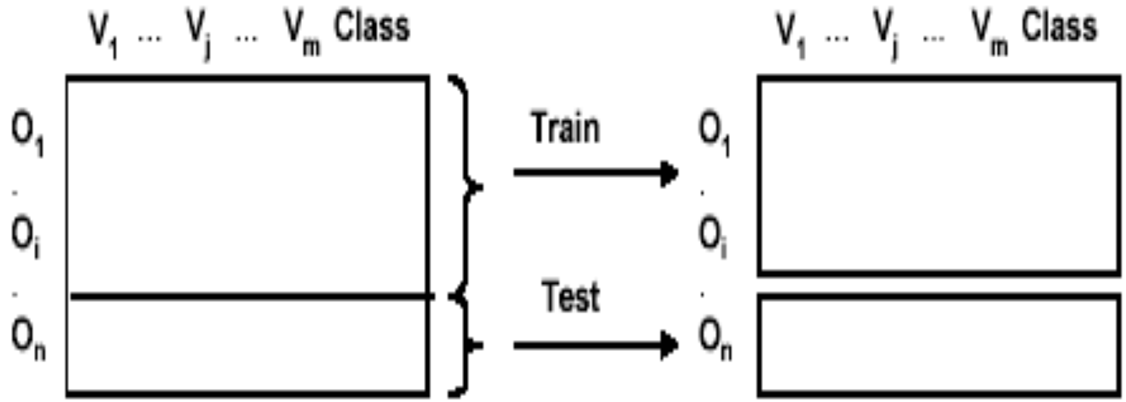
- **Holdout procedure**
  is a **method** of **splitting** original data into
  training and test data sets

- **Dilemma:**
- ideally **both training** and **test data** should be large
- What to do if the **amount of data** is limited?
- How to **split** the data into training and test subsets?
- **Disjoint sets** - in the best way

Holdout

Train-and-Test (for large sample sizes) (> 1000))
dividing the given data set in

- a **training sample** for generating the classification model
- a **test sample** to test the model on independent objects with given classifications (randomly selected, 20-30% of the complete data set)



Holdout (N- N/3 ; N/3)

- The **holdout method reserves** a certain amount of data for testing and uses the **remainder** for training – so they are **disjoint!**

- **Usually**, one third **(N/3)** of data is used for **testing**, and **the rest (N -N/3) = (2N/3)** for **training**

- **The choice** of records for **train** and **test** data is **essential**

We usually perform a **cycle:**

Train-and-test; repeat

Repeated Holdout

- **Holdout** can be made more reliable by repeating the process with **different sub-samples** (subsets of data):

  1. In each iteration, a **certain portion** is randomly selected for training, the **rest of data** is used for testing

  2. The **error rates** or **predictive accuracy** on different **iterations** are **averaged** to yield an overall error rate, or overall **predictive accuracy**
- Repeated holdout still is not **optimal**: the different test sets **overlap**

k-fold cross-validation (N - N/k ; N/k)

- This is a **cross-validation** used to **prevent** the **overlap** of the test sets

- **First step:** split data into k **disjoint subsets**
- D1, … Dk, of **equal size**, called **folds**

- **Second step:** use each subset in turn for **testing**, the remainder for **training**

- **Trainin**g and **testing** is performed **k times**

k-fold cross-validation predictive accuracy computation

- The predictive accuracy estimate is the overall number of correct classifications from all iterations, **divided** by the total number of records in the **initial data**

Stratified cross-validation

- In the **stratified cross-validation**
- the folds are stratified; i.e.
- the **class distribution** of the tuples
- (records) in **each fold** is
- approximately **the same as** in the
- initial data

10 folds cross-validation

- In general,
- 10-fold cross-validation or
  stratified 10-fold cross-validation
- is **recommended** and
- **widely used** even if computational power allows using more folds
- **Why 10?**

Extensive experiments have shown that this is the best choice to get an accurate estimate due to its relatively low bias and variance

So interesting!

Improve cross-validation

- Even better:

  repeated cross-validation

  Example:

  10-fold cross-validation is repeated 10 times and results are averaged;
  We adopt the union of rules as the **new set** of rules

A particular form of cross-validation

- k-fold cross-validation: (N -N/k ; N/k)
- If k = N, what happens?
- We get (N-1; 1)

  It is called "leave –one –out"

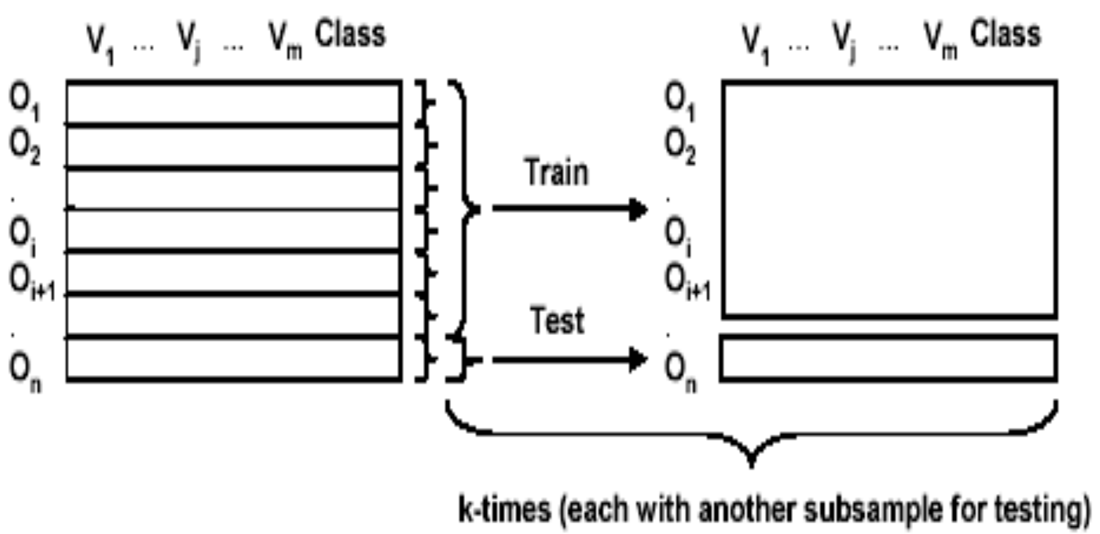
  Each sample (record) is used the same number of times for training and once for testing

Leave-one-out (N-1 ; 1)

Cross-Validation (for moderated sample sizes) → Sampling without replacemen

- Dividing the given data set into ***m* subsamples of equal size**
- Each subsample is tested by using a model generated from the remaining *(m-1)* subsamples

  → **Leave-One-Out**: m = Number of objects



Leave-one-out (N-1 ; 1)

- **Leave-one-out** is a particular form of cross-validation

We set number of folds to number of training instances, i.e. k= N

For N instances we build classifier (repeat the training - testing) n times

Leave-one-out Procedure

- Let C(i) be the classifier (rules, patterns) built on all data **excep**t record x_i
- Evaluate C(i) on x_i
- Determine if it is **correc**t or in **error**
- Repeat for all i=1,2,…,n
- The **total error** is the **proportion** of all the incorrectly classified x_i
- The final set of rules (patterns) is a union of all rules obtained in the process

Leave-one-out (N-1 ; 1)

- Makes the **best** use of the data
- Involves no random sub-sampling
- Stratification is not possible
- **Computationally** expensive
- MOST commonly used