UNIVERSITAT DE BARCELONA

Course. Introduction to Machine Learning

Work 3. Lazy Learning Exercise

Session 1

Dr. Maria Salamó Llorente

Dept. Mathematics and Informatics,

Faculty of Mathematics and Informatics,

University of Barcelona

Contents

Introduction

Introduction

- Many lazy learning techniques exist
  - This course we will concentrate on instance-based learning
- In lazy learning,
  - storing and using specific instances may improve its performance
  - Removing specific features may improve its performance
- IBL algorithms usually use all the training set but this causes:
  - A large storage is needed
  - The generalization process is slow
  - The data may contain inconsistencies and noise
- To deal with these problems,
  - Reduction techniques are used
  - Feature selection techniques are used

Introduction

The **goal** of Work 3 is to…

1. Implement 3 IBL algorithms: ib1, ib2, ib3 (Week 1)
2. Analyze **best ibl** algorithm (Week 2)
3. Implement k-IBL algorithm with analysis of parameters
   1. Distance metrics (Week 2)
   2. Voting policies (Week 2)
4. Analyze **best k-IBL** algorithm (Week 2)
5. Implement feature selection techniques (Week 3)
6. Compare best k-IBL with and without feature selection techniques using different metrics: accuracy and efficiency (Week 4)
7. Perform statistical analysis and write report (Week 4)

Introduction

- Four things make a lazy learner:
  - A distance metric
  - How many nearby neighbors to look at?
  - A weighting function (optional)
  - How to fit with the local points?

Introduction

- 1- Nearest Neighbor
  - A distance metric: **Euclidean distance**
  - How many nearby neighbors to look at? **One**
  - A weighting function (optional): **Unused**
  - How to fit with the local points?: **Just predict the same output as the nearest neighbor**

Introduction

- In this work you implement and analyze **Instance-based learning** algorithms
- Special attention to:
  - Distance metrics: **Euclidean, Manhattan or cosine, Clark or Canberra, HVDM or IVDM**
  - How many nearby neighbors to look at? **One or K**
  - A weighting function (optional): **two feature selection algorithms**
  - How to fit with the local points?: **Define a voting rule**

What to do in work 3?

- Adapt the parser to read the class and the 10 fold cross-validation sets
- Find the best IBL algorithm
  - ib1, ib2, ib3
  - Use Euclidean distance
  - Feature weights are set up to 1.0
- Find the best k-IBL algorithm
  - K = 1, 3, 5, 7
  - Distance metric
    - Euclidean, Manhattan or cosine, Clark or Canberra, HVDM or IVDM
  - Voting policies
    - Most Voted, Modified Plurality, Borda Count
- Apply feature selection technique to the best k-IBL algorithm
- Evaluate the results with an statistical analysis
- Write the report

Instance-based learning

Implement your own code

Instance-based learning

- IBL algorithms are derived from the nearest neighbor pattern classifier
- They only use selected instances to generate classification predictions
- IBL algorithms are incremental and their goals include:
  - maximizing classification accuracy on subsequently presented instances
- Instance-base learning is a carefully focused case-based learning approach that contributes evaluated algorithms for:
  - selecting good cases for classification,
  - reducing storage requirements,
  - tolerating noise, and
  - learning attribute relevance

What makes an instance-based learner?

- A distance measure
  - Nearest neighbour: typically Euclidean
- Number of neighbours to consider
  - Nearest neighbour: one
- A weighting function (optional)
  - Nearest neighbour: unused (equal weights)
- How to fit with neighbours
  - Nearest neighbour: same output as nearest neighbour

IBL algorithms

- Assume that "***similar instances have similar classifications***"

- **IB1** is identical to the nearest neighbor algorithm, except that
  - it normalizes its attributes' ranges,
  - processes instances incrementally, and
  - has a simple policy for tolerating missing values.
- **IB2** is identical to IB1, except that
  - it saves only misclassified instances
- **IB3** is an extension of IB2 that employs a "*wait and see*" evidence gathering method to determine which of the saved instances are expected to perform well during classification

IB-Algorithms

(David W. Aha, Dennis Kibler, and Marc K. Albert. 1991)

- **IB1**: store all examples
  - High noise tolerance
  - High memory demands
- **IB2**: Store examples that are misclassified by current example set
  - Low noise tolerance
  - Low memory demands
- **IB3**: like IB2 but,
  - – Maintain a counter for the number of times the example participated in correct and incorrect classifications
  - – Use a significant test for filtering noisy examples
    - Improved noise tolerance
    - Low memory demands

IB1

- IBL algorithms assume that "***similar instances have similar classifications***"

Table 1. The IB1 algorithm (CD = Concept Description).

$CD \leftarrow \emptyset$
for each $x \in$ Training Set **do**
1. **for each** $y \in CD$ **do**
   Sim[y] $\leftarrow$ Similarity(x, y)
2. $y_{max} \leftarrow$ some $y \in CD$ with maximal Sim[y]
3. **if** class(x) = class(y_{max})
   then classification $\leftarrow$ **correct**
   else classification $\leftarrow$ **incorrect**
4. $CD \leftarrow CD \cup \{x\}$

IB2

- IBL algorithms assume that "***similar instances have similar classifications***"

Table 2. The IB2 algorithm (*CD* = Concept Description).

$CD \leftarrow \emptyset$
for each $x \in$ Training Set **do**
  1. **for each** $y \in CD$ **do**
     Sim[y] $\leftarrow$ Similarity(x, y)
  2. $y_{max} \leftarrow$ some $y \in CD$ with maximal Sim[y]
  3. **if** class(x) = class(y_{max})
     then classification $\leftarrow$ **correct**
     else
       3.1 classification $\leftarrow$ **incorrect**
       3.2 $CD \leftarrow CD \cup \{x\}$

IB2: Speed up

- IB2 is an extension to IB1 algorithm
  - Save memory and speed up classification
  - Unnecessary to use all data points for classification
- Algorithm
  - Work with data points incrementally
  - For each newly received data point apply NN using already saved points to predict its class
  - Only remember misclassified instances for future predictions
  - Problem:
    - Important instances in the early moments of learning are discarded
    - Noisy data gets incorporated

IB3

UNIVERSITAT DE BARCELONA

IBL algorithms assume that "***similar instances have similar classifications***"

Table 5. The IB3 algorithm (CD = Concept Description).

$CD \leftarrow \emptyset$
for each x in Training Set **do**
1. **for each** $y \in CD$ **do**
   Sim[y] $\leftarrow$ Similarity(x, y)
2. **if** $\exists\{y \in CD |$ acceptable(y)$\}$
   then $y_{max} \leftarrow$ some acceptable $y \in Cd$ with maximal Sim[y]
   else
   2.1 $i \leftarrow$ a randomly-selected value in $[1, |CD|]$
   2.2 $y_{max} \leftarrow$ some $y \in CD$ that is the i-th most similar instance to x
3. **if** class(x) $\neq$ class(y_{max})
   then classification $\leftarrow$ **correct**
   else
   3.1 classification $\leftarrow$ **incorrect**
   3.2 $CD \leftarrow CD \cup \{x\}$
4. **for each** y in CD **do**
   if Sim[y] $\geq$ Sim[y_{max}]
   then
   4.1 Update y's classification record
   4.2 **if** y's record is significantly poor
   then $CD \leftarrow C - \{y\}$

IB3: Handle noise

- IB3 is an extension of IB2
  - Deal with noise, keep only good classifier data points
  - Discard instances that do not perform well
- Algorithm:
  - Keep a record of the number of correct and incorrect classification decisions that each saved data point makes
  - Two predetermined thresholds are set on success ratio
  - An instance is selected to be used for training:
    - *If the number of incorrect classifications is ≤ the first (lower) threshold and,*
    - *If the number of correct classifications is ≥ the second (upper) threshold*

IBL reference

Instance-based learning algorithms

D.W. Aha, D. Kibler, and M.K. Albert

Machine Learning , 6, 37- 66 (1991)