Handwritten Digit Recognition using Edit Distance-Based KNN

Marc Bernard marc.bernard@univ-st-etienne.fr
Elisa Fromont elisa.fromont@univ-st-etienne.fr
Amaury Habard amaury.habrard@univ-st-etienne.fr
Marc Sebban marc.sebban@univ-st-etienne.fr
Laboratoire Hubert Curien, UMR CNRS 5516,
18 Rue du Professeur Benoît Lauras, 42000 Saint-Etienne, France

Abstract

We discuss the project given for the last 5 years to the 1st year Master students who follow the Machine Learning lecture (60h) at the University Jean Monnet in Saint Etienne, France. The goal of this project is to develop a GUI that can recognize digits and/or letters drawn manually. The system is based on a string representation of the digits using Freeman codes and on the use of an edit-distance-based K-Nearest Neighbors classifier. In addition to the machine learning knowledge about the KNN classifier (and its optimizations to make it efficient) and about the edit distance, some programming skills on how to develop such a GUI are required.

1. Introduction

The Machine Learning (ML) course is given to the 1st year Master students at the University Jean Monnet in Saint Etienne. The course's duration is 60 hours for which 45 hours are dedicated to theory and desk exercises and 15 hours are dedicated to lab sessions on WEKA[1] or R[2] and on starting ($\sim$ 8 hours) the student project on handwritten digit/letters recognition which also requires about 12 hours of personal work at home.

These students already have a bachelor in computer science and thus have reasonable programming skills (for example they know how to use Java SWING[3] to create a nice GUI) and some experience in working in small (4-8 persons) groups. They had a few hours of mathematics and in particular of statistics ($\sim$ 60h) since high school but they cannot be considered as having a very strong mathematical and, in general, theoretical background.

The ML course covers topics related to supervised classification. The introductory part of this course ($\sim$ 15h) covers the general definition of supervised classification, the error estimations, the non-parametric Bayesian methods and their link to the class estimations using Parzen windows and K-nearest neighbors (KNN). When focusing on the KNN part, a couple of hours are spent on the possible distances used to compare two instances and, in particular for string-based data, on the edit distance. The other parts of the course cover different symbolic and numerical algorithms and models from Inductive Logic Programming to Hidden Markov Models through Decision Trees, Neural Networks etc. Given their background, this course is usually perceived as theoretical and the project is usually very well welcomed by the students.

The project is proposed to groups of up to 5 students. The goal of the project is to make students work on data acquisition and data storage, the implementation and optimization of a machine learning algorithm (here KNN and its cleaned and condensed versions), the evaluation of the parameters (e.g. the weights of the edit distance, the number of nearest neighbors, etc.) and of the system (cross validation, test set, etc.). A nice Graphical User Interface (GUI) is required to be able to give a live demo of the system at the end of the project.

Section 2 gives an overview of the project and of the theoretical prerequisites. Section 3 gives more details and a description of the steps the students go through when working on the project. We conclude in Sect. 4.

[1] http://www.cs.waikato.ac.nz/ml/weka/
[2] http://www.r-project.org/
[3] http://docs.oracle.com/javase/tutorial/uiswing/

2. Project Description

In the following, everything which concerns a digit can also be applied to a letter.

2.1. Overview

The goal of the project is to design a handwritten digit recognition system based on a KNN classifier. At the end of the project, each group of students should be able to present a GUI on which a user can draw (for example with a mouse) a black and white digit on a screen. This drawing could be either, saved as an image (or directly as a Freeman code in a text file) to populate a database or, could be recognized online (and efficiently) by the system.

2.2. Freeman codes

There are multiple ways to encode a digit drawn on a screen for further processing. One of them consists in saving the digit and the frame in which it has been drawn as an image. From this image, a successful string representation has been proposed by Freeman (Freeman, 1961) to encode the digit. The computation of the Freeman codes is presented to the students during a lab session dedicated to the initiation of the project.

To compute the Freeman code of an image, one needs to move along a digital curve or a sequence of border pixels based on eight-connectivities. The direction of each movement is encoded by using a numbering scheme $\{i \| i = 0, 1, .., 7\}$ denoting an angle of $45 * i$ (counter-clockwise) from the positive y-axis. The chain codes can be viewed as a connected sequence of straight-line segments with specified lengths and directions as showed in Figure 1.

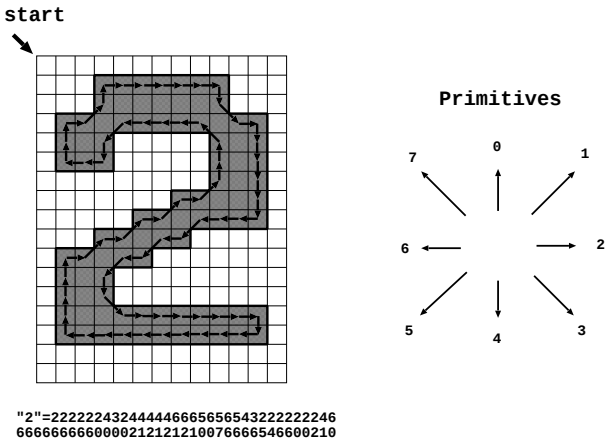

Figure 1. Representation of the digit 2 drawn on the left using a Freeman code.

2.3. KNN

We suggest the students to use the K-nearest neighbors algorithm for the digit recognition. From experience, we know that this is a simple yet effective method to automatically identify the drawn digit independently of the writing style or the scale (but the digit has to be drawn in a bounded-size window).

The KNN algorithm is presented during the lecture. Algorithm 1 explains how to classify an unknown example $\mathbf{x}$ using a dataset S.

Algorithm 1 KNN algorithm

Input: $\mathbf{x}, S, d$
Output: class of $\mathbf{x}$
for $(\mathbf{x}', l') \in S$ **do**
  Compute the distance $\mathrm{d}(\mathbf{x}', \mathbf{x})$
end for
Sort the $|S|$ distances by increasing order
Count the number of occurrences of each class l_j among the k nearest neighbors
Assign to $\mathbf{x}$ the most frequent class

To be able to identify the digit online (with a real time constraint), the students have to optimize their original algorithm (e.g. by reducing the size of the database to reduce the computations). Two algorithms are proposed to speed up the classification process: one (see Algo. 2) which removes from the original dataset S the outliers and the examples located in the Bayesian error region, smoothing the decision boundaries, and one (see Algo. 3) which deletes the irrelevant examples (e.g. the centers of the homogeneous clusters).

Algorithm 2 Step1 : Remove from the original dataset S the outliers and the examples of the Bayesian error region

Input: S
Output: $S_{cleaned}$
Split randomly S into two subsets S_1 and S_2
repeat
  Classify S_1 with S_2 using the 1-NN rule
  Remove from S_1 the misclassified instances
  Classify S_2 with the new set S_1 using the 1-NN rule
  Remove from S_2 the misclassified instances
until stabilization of S_1 and S_2
$S_{cleaned} = S_1 \cup S_2$

Another optimization can be made to speed up the original algorithm by speeding up the search for the nearest neighbor using the triangle inequality property

Algorithm 3 Step2 : Remove the irrelevant examples

Input:S
Output: STORAGE
STORAGE $\leftarrow \emptyset$; DUSTBIN $\leftarrow \emptyset$
Draw randomly a training example from S and put it in STORAGE
repeat
  for $\mathbf{x_i} \in S$ **do**
    if $\mathbf{x_i}$ is correctly classified with STORAGE using the 1-NN rule **then**
      STORAGE DUSTBIN $\leftarrow \mathbf{x_i}$
      STORAGE $\leftarrow \mathbf{x_i}$
    end if
  end for
  STORAGE $\leftarrow$ STORAGE $\setminus$ DUSTBIN
until stabilization of STORAGE
Return STORAGE

of the distance function (see (Vidal, 1994) for more details on the techniques).

- Let $\mathbf{x}$ be the example to classify by the nearest-neighbor (NN) rule. Let us consider that the current NN of $\mathbf{x}$ is $\mathbf{y}$ which is at a distance δ from $\mathbf{x}$.
- Let $\mathbf{z}$ be the next training example. If $d(\mathbf{x}, \mathbf{z}) \leq \delta$ then update the current NN. Otherwise, remove the following two categories of examples located:
  1. in the sphere centered at $\mathbf{z}$ and of radius $d(\mathbf{x}, \mathbf{z}) - \delta$,
  2. out of the sphere centered at $\mathbf{z}$ and of radius $d(\mathbf{x}, \mathbf{z}) + \delta$,

2.4. Distances

The distance used to compare two Freeman codes in the database is the edit distance (explained below). A variant of the project could consist in testing the performance of other distances such as a L1, L2 or a Mahalanobis distance on other types of image representation (quad tree, 0-1 pixel matrix, interest point detectors,...).

Definition 1 *The Levenshtein distance (or Edit Distance) between two strings* $\mathbf{x} = x_1...x_T$ *and* $\mathbf{y} = y_1...y_V$ *is given by the minimum number of edit operations needed to transform* $\mathbf{x}$ *into* $\mathbf{y}$*, where an operation is an insertion, deletion, or substitution of a single character.*

Rather than counting the number of edit operations, one can assign an edit cost to each edit operation and search for the less costly transformation:

- $c(x_i, y_j)$ is the cost of the substitution of x_i by y_j,
- $c(x_i, \lambda)$ is the cost of the deletion (x_i into the empty symbol λ),
- $c(\lambda, y_j)$ is the cost of the insertion of y_j.

Input: Two strings $\mathbf{x}(T)$ and $\mathbf{y}(V)$
Output: Edit Distance $D(T, V)$ between $\mathbf{x}$ and $\mathbf{y}$
$D(0,0) \leftarrow 0$
for r=1 to T **do**
  $D(r, 0) \leftarrow D(r-1, 0) + c(x_r, \lambda)$
end for
for k=1 to V **do**
  $D(0, k) \leftarrow D(0, k-1) + c(\lambda, y_k)$
end for
for r=1 to T **do**
  for k=1 to V **do**
    $d_1 \leftarrow D(r-1, k-1) + c(x_r, y_k)$
    $d_2 \leftarrow D(r-1, k) + c(x_r, \lambda)$
    $d_3 \leftarrow D(r, k-1) + c(\lambda, y_k)$
    $D(r, k) \leftarrow min(d_1, d_2, d_3)$
  end for
end for
Return $D(T, V)$

3. Project Implementation

The first job of the group of students is to divide the different tasks between all the members of the group and organize the communication to be able to integrate all the parts before the evaluation of the project. A student version of the project can be found online at his address: http://labh-curien.univ-st-etienne.fr/~sebban/AA-RDF/applet.php?lang=en.

3.1. GUI

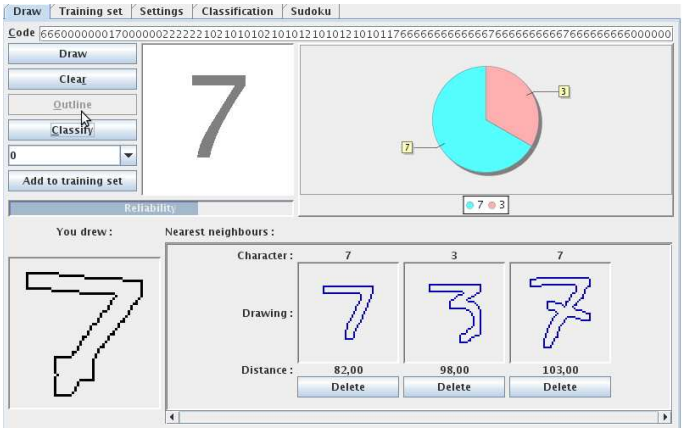

Figure 2. Recognition of the digit 7 in the final GUI.

The project focuses on machine learning. The aim is not to test the programming skills of the students. So, the choice of the programming language and the design of the GUI is left to the students. However, in our case, most of the students use the JAVA Swing API. This

API provides tools to create a graphical user interface (GUI) for Java programs. An example of a resulting GUI is given Figure 2.

3.2. Dataset

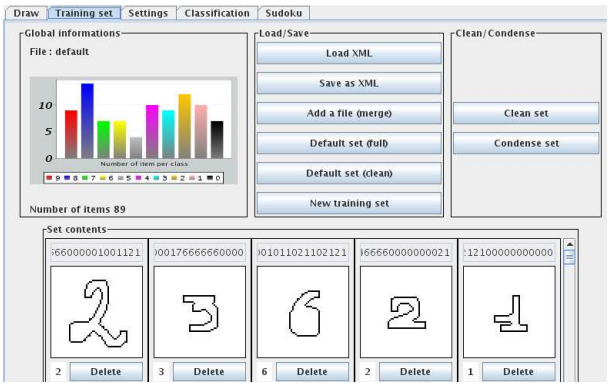

Figure 3. Example of a dataset and its statistics computed by the students' system

The students have to create their own dataset and a structure to store the data. Some students decide to create a real database system (using for example PostgreSQL[4]) and store the images which contain a digit. With this kind of method, the student can use and compare multiple representations of the image. Some students decide to store only the Freeman code as textual information or as XML files as shown in Figure 3. Whatever the structure, the students find out early in their experimental process that more than one person should populate the dataset with some digits to make sure that the instances of each class are diverse enough (in term of writing style) to have a sufficiently good classification accuracy.

3.3. Algorithms

The students can choose any programming language but it is usually done according to the choice of the API for the GUI (so most of the time, it is JAVA). Most of the necessarily algorithms are given (as pseudo code) during the lecture and have to be implemented. However, the part concerning the preprocessing of the digit (for example the transformation into a Freeman code independently of the position of the digit in the drawing window) is left as the students' choice. Depending on their organization skills, the students may not have time to implement all the possible optimizations of the KNN algorithm. In this case, we expect them to be critical about the performance of their algorithm especially during the demo.

[4]http://www.postgresql.org/

3.4. Evaluation

Because of the limited size of their database, most of the students use a 5-fold or a leave-one-out cross validation strategy to evaluate the global performance of their algorithm. They keep increasing the size of their dataset until they reach a "sufficiently" good accuracy (around 80% with a representative enough database). Then, they try to use the dataset reduction techniques to evaluate the loss of accuracy.

They usually try to test different "K" for the KNN algorithm and keep the one that gives the best accuracy results by cross validation.

For the edit distance, they usually start by giving an equal weight to each of the three edit operations then they increase the weights of the insertion and deletion compared to the substitution. A better idea is to weight the substitution differently for each possible code. Indeed, it must seem easier to change a Freeman code 1 into a code 2 (because it is a small angle) than to change a Freeman code 1 into a code 4. (Rico-Juan & Micó, 2003) is a good reference to find the weights of the edit operations for this problem.

4. Conclusion

The proposed project shows the whole process of supervised classification from the data acquisition to the design of a classification system and its evaluation. The project is usually extremely successful with the students. Most of them are very proud of the performance of their system and most of the time, they implement side games such as Sudoku or letter games (such as scrabble) to show the performance of their system online. The project, besides giving practical applications to some machine learning techniques seen in class, gives them the opportunity to work as a group. As explained in the text, some changes can be made on the representations of the images and on the distances used to compare two elements in the database.

Acknowledgements

This work is supported by the Pascal 2 network of Excellence.

References

Freeman, Herbert. On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, EC-10(2):260–268, June 1961.

Rico-Juan, Juan Ramón and Micó, Luisa. Comparison of aesa and laesa search algorithms using string and

tree-edit-distances. *Pattern Recogn. Lett.*, 24(9-10): 1417–1426, June 2003. ISSN 0167-8655.

Vidal, Enrique. New formulation and improvements of the nearest-neighbour approximating and eliminating search algorithm (aesa). *Pattern Recognition Letters*, 15(1):1–7, 1994.