Hand Digit Recognition

Oussama Bouldjedri, Arunava Maulik, Renuka Chittimala

January 12, 2018

Abstract

The goal of this project is to develop a GUI which will able to recognize digits drawn manually on the canvas of the program. The recognition system is based on the two representations of the freeman's code generated from the digits drawn; structured and numerical . As an additional task, a sequence mining algorithm was implemented to generate the most frequent pattern from the digit to be recognized. Also, a basic calculator was built as the final task of the project.

1. Introduction

Handwritten Digit Recognition is a fundamental classification and image recognition task in machine learning. Some of its applications include digit recognition in pin codes of postal mails, bank check processing. The various input formats can be photographs, paper documents or even touch screen devices. The task is to build a GUI to recognize handwritten digits in real time. The primary challenge of this task is to account for the style of writing of different people and the distortions associated with them. For example, the french way of writing the digits 1 and 7 are different from the international way of doing it. We have used the MNIST database which has been a reliable resource of hand written digits since its release in 1999. This dataset has been used as the benchmark for all testing purposes. The image from canvas of GUI is taken as input. Before generating the freeman code, the image is pre processed using techinques which include binarization,smoothing and de-noising. This freeman's code is used directly for structured representation of features to be used in the classification task. For the numerical representation of the features, a histogram of frequencies of digits in the freeman's code. We tried various techniques to speed up the computation for both the representations of the data. We performed the additional task of finding the frequent patterns for each partial digit drawn on the and used it for classification of the the digit. This project combines the knowledge of machine learning and data mining techniques and helps us putting into practice our theoretical knowledge.

2. Literature review

2.0.1. Handwritten Digit Recognition Using K Nearest-Neighbor, Radial-Basis Function, and Backpropagation Neural Networks

In (1), a comparative study was done among 3 different classification algorithms; multi layer neural networks, radial basis function(RBF) networks and the KNN classifier. The neural network was found to be the best in terms of computation speed and memory management but took the most training time. The RBF classifier was however more accurate but reauired more memory with less training time. The KNN classifier was found to reauire more memory, longer classification time but it's simplicity made it a good candidate for this task.

2.0.2. Handwritten digit recognition with a back-propagation network

In (2), a large back propagation(BP) network was implemented to recognize zipcode digits provided by the U.S. Postal Service. The central idea is to reduce the number of free parameters considerably to be computed by the learning algorithm. However, care has to be taken to not the reduce the computational power of the network too much. This results in a network architecture with reduced entropy and increases the probability to correctly generalize.

2.0.3. Handwritten digit recognition by neural networks with single-layer training

In (3),this paper image databases with digits like CENPARMI, CEDAR, and MNIST were used. Different feature vectors like chaincode feature, gradient feature, profile structure feature, and peripheral direction contributivity are tested .The classifiers used include KNN,three neural classifiers, a learning vector quantization classifier, DLQDF classifier, and two support vector classifiers (SVCs).Out of the feature vectors used chain code and gradient feature have some advantages.RBF kernel in SVC gave the best result and among non-SV, polynomial classifiers and DLQDF gave highest results

2.0.4. Handwritten digit recognition: benchmarking of state-of-the-art techniques

In (4),neural network classifiers with single-layer is used for recognition of handwritten digits.Introduced STEPNET procedure that beaks problems into subproblems which are solved by linear separator.Two databases European database, Postal Service comprising were used. Performance obtained by this method are comparable to that of more complex networks.

3. Dataset

The MNIST dataset was used for training of the KNN classifier. The dataset has a training set of 60,000 examples, a test set of 10,000 examples and is a subset of a larger set available from NIST. The digits in the datasets are size-normalized and centered to a fixed size. The dataset requires minimal preprocessing and formatting and is a good resource for machine learners to practise different pattern recognitions methods on realistic data. To build this dataset, the images from the original NIST dataset were normalized to a 20x20 pixel box whilst maintaining the aspect ratio. The images were finally centered in a 28x28 image by computing the center of mass of the pixels.
We also created our own dataset using the GUI to draw images, save them, extract features and use it to test our algorithms.

3.1. Dataset Visualization

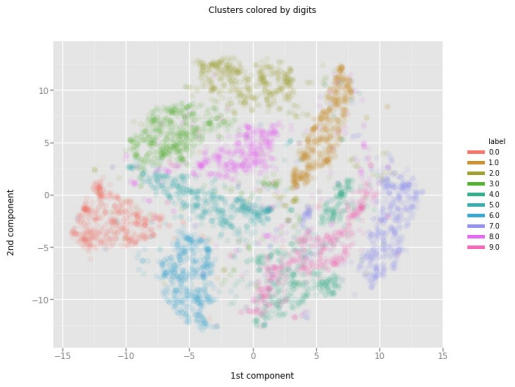
At first we tried to visualize the MNIST dataset after reducing it's dimensionality using the PCA algorithm. We used 5000 examples randomly from the dataset for the purpose.

However, the first two Eigen vectors have a combined variance of only about 16% which quite clearly indicates that the digits cannot be separated in the plot above.

Next, we used the t-distributed Stochastic Neighbor Embedding(t-SNE) algorithm to plot the data for better visualization. T-sne is a useful tool for visualizing high dimensional data. It takes into account similar datapoints and formulates a joint probability between them. It then tries to minimize the Kullbeck Liebler Divergence between the joint probabilities of the high-dimensional data and the low-dimensional embedding.

We get the KL divergence as 2.13 in this case with 5000 samples.

[1]



4. Preprocessing of Dataset

The image that we take as inputs i.e., test sample to the function is of size 200x200.We used cv2 is used to read the image .Initially we trained our model on MNIST dataset where each sample is a 28x28 image. To begin with the preprocessing on the data ,the test sample is resized to 28x28 in order to make the it compatible with the training data. By doing this we can also use own images as training samples along with MNIST data. Pixel range of the test sample lies between 0-255 even though the image is a black white. To handle this issue we applied thresholding on the image. Simple thresholding, Adaptive thresholding, Otsu's thresholding are some of the methods of thresholding. We used Simple thresholding in which if pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black),here in our case we used white as all our samples are on black background. In most of the cases the image may contain noise and can be treated as one of the component of the image which further create a mess in classification. To avoid this there should be an approach to take into account all the components and then decide to which among them are noise. For this we used connectedComponentsWithStats which computes the connected components labeled image of boolean image with 4 or 8 way connectivity and returns N, the total number of labels [0, N-1] where 0 represents the back-

[1] https://medium.com/@luckylwk/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b

ground label. In the process of generating freeman's code we also have to take care of the selecting the bounding box of the image. Region of Interest ,ROI, lets to allow to perform operation on certain region of the image. It improves accuracy and performance. Here we handled this by making the top, bottom, left, right most rows and columns respectively to zero.

Figure 1. Image on canvas

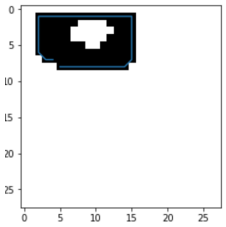


Figure 2. After dilation

There may be chances of loss in data if the object touches the border as we are replacing borders with zeros.In order to handle this issue we dilated the image using cv2.dilate. This also address the problem of loss in data by drastically resizing the image from 200x200 to 28x28 . Also when there are disconnected components in the image which can be notable features when taken into consideration and when not considered may change the meaning of the object. For example ,in most of the cases 1 and 7 have the same sequence of code, by adding a horizontal bar makes the entire sequence different.

In the following case when the horizontal bar is ignored its classified as 1.

Figure 3. Image on canvas

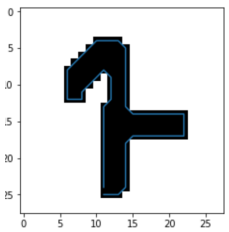


Figure 4. After dilation

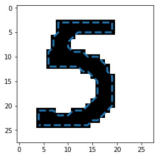
As dilation can handle the issue of disconnected components if exists as it enlarges the boundaries of regions of foreground pixels. Also with dilation the length freeman's code can be reduced which can speedup the classification process.Addressing these issues by dilating the image raised some other issues.Dilation has its own disadvantages, like if the object in the image is very small then dilation on top of it makes the object loose its significance.For example, too small 8 when dilated can be classified as 0 as the areas of foreground pixels grow in size while holes within those regions become smaller. One way to handle the issues raised by dilating the image is controlling the dilation by applying some conditions.For example , a thin and and long 8 can be misclassified as 1.

5. Freeman's code generation

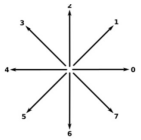
Some popular chain codes include the Freeman Chain Code of Eight Direction (FCCE), Vertex Chain Code (VCC), Three OrThogonal symbol chain code (3OT), Directional Freeman Chain Code of Eight Directions (DFCCE) and Unsigned Manhattan Chain Code (UMCC). The concept of freeman chain code is to separately encode each connected component in the image.For each such components in the region of interest, a starting point is selected along with its coordinates. Then the traversal of the code moves along the boundary of the region. At each step it records the symbol representing the direction ,0,1,. . . ,7, of its movement. This continues until the encoder returns to the starting point. The following pseudocode is used for generating freeman's chain code.

```
/* Directions  0 1 2    Change in    -1  0  1   Change in    -1 -1 -1
               7 x 3    x-dimension  -1  0  1   y-dimension   0  0  0
               6 5 4                 -1  0  1                 1  1  1
 */
// Find the first pixel on the boundary of the object.
for all rows, r
   for all columns, c
      if image[r][c] is in the region of interest
         write starting point
         start_y = r
         start_x = c
         break  //starting point found

//write chain code

for each direction
   if image[r+change_y[direction]][c+change_x[direction]] is
         in the region of interest
      write direction
      r = r + change_y[direction]  //set new current point (c, r)
      c = c + change_x[direction]
      break

while the coordinates of the current pixel are not equal to those of the
      starting pixel
   //figure direction to start search
   b_direction = (direction + 5) % 8
   for each direction (from b_direction to 7  and  0 to b_direction-1)
      if image[r+change_x[direction]][c+change_y[direction]] is
         in the region of interest
      write direction
      r = r + change_x[direction]  //set new current point
      c = c + change_y[direction]
      break
```

The chain code generated for a image is a connected sequence of lines along the border of the object in the image as shown below.



```
111333433233322211100707777771112121133333333
333557777777665433433445455555566676777777777
```



Directions in Freeman chain code
8-connected

[2]

6. Methods used

Two methods were used for the generation of features from the freeman's code and subse-

[2] http://www.cs.unca.edu/~reiser/imaging/chaincode.html

quent use of the KNN Algorithm for the Classification Task.

6.1. Method 1

6.2. Numerical Features

The freeman's code was converted to numerical features to be used in the knn for classification. For each image, the respective freeman's code was considered and a histogram of the counts of the respective digits was built. This was done for each image and a matrix of features was thus built to be used in the classifier.

6.3. Removing Bayesian error

In order to optimize the training process of the KNN algorithm we used a lot of strategies: we used a famous way to remove the outliers and the examples of the bayesian error region.and this way consist of doing the next

- we split randomly S(the set of our samples) into S1 and S2
- we will repeat the next process till the stabilisation(the set is the same as the last iteration of this process) of S1 and S2:
- classify the set S1 with the set S2 using knn with k=1 then remove the misclassified examples from S1 after the last classification
- classify the set S2 with the set S1 using knn with k=1 then remove the miss classified examples from S2 and by the end after stabilization of S1 and S2 the clean set will be the union of S1 and S2

Also we used a famous way to remove the irrelevant examples.

- we initialize 2 empty sets ST and DUSTBIN
- we draw in a random way a sample from S and put it in ST
- we repeat the next process till the stabilization of the ST set for every x in S : if x_i is correctly classified with st using the 1NN rule then dustbin<-x else ST<-x

  return ST

6.4. Speeding Up Nearest Neighbor Search

The task is to remove irrelevant examples with the help of the triangle inequality property. Let x be the example to be classified and y be its current nearest neighbor at a distance of γ. If the distance between x and a new example z is lesser than γ, we update the nearest distance. If not, we remove examples that are outside the sphere centering z and a radius equal to d(x,y)-γ. Also, the examples outside the sphere centering z and having a radius equal to d(x,y)+γ are also removed.

[3]

7. Few results

Before removing the Bayesian error, we get the following result after tuning 'K'. We get the best value of K as 25.

	Training	Val.	Test
in %	51.9	50.73	50.81

After removal of the Bayesian error, we get the following results with K=40(the best value of K)

	Training	Val.	Test
in %	67.79	65.48	57.8

8. Method 2

8.1. Structured Features

The freeman's code generated in the previous steps was used as structured data to be used directly in the knn algorithm with edit distance as the distance measure for classification. The sequences that we obtain after using the freeman's code were directly used to find the nearest neighbor using edit distance as the distance metric.

8.2. Speed up of computation

To speed up the calculation of the edit distance, firstly we implemented a custom function to find the nearest neighbor instead of using the scikit learn implementation of the nearest neighbor function. This considerably increased the computation speed by upto 5 times. Further more, we experimented with 2 other similar distance metrics:

8.3. Jaro Algorithm

The Jaro algorithm is used as a measure of characters that are in common. It is no more than half the length of the longer string , with consideration for transpositions. The higher the Jaro distance for two strings is, the more similar the strings are. The score is normalized such that 0 equates to no similarity and 1 is an

[3] https://hal.archives-ouvertes.fr/hal-00714509/document

exact match.

8.4. Jaro-Winkler Algorithm

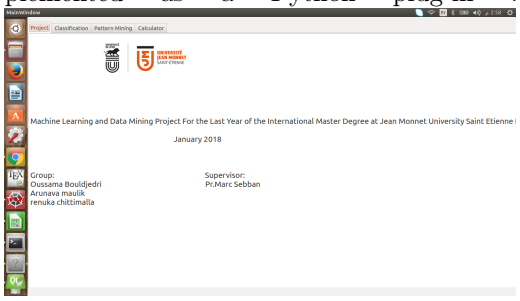
Jaro-Winkler Algorithm : Winkler modified the Jaro algorithm to support the idea that differences near the start of the string are more significant than differences near the end of the string. The lower the Jaro–Winkler distance for two strings is, the more similar the strings are. The score is normalized such that 0 equates to no similarity and 1 is an exact match. The Jaro–Winkler similarity is given by 1 − Jaro–Winkler distance. The efficiency of bounded Jaro Winkler was tested in (5).

We used the 'Levenshtein' package to test our results. Although both the Jaro and Jaro-Winkler algorithms were considerably faster than the regular edit distance algorithm, their accuracy of predictions was slightly worse. It shows that both these algorithms are suited for shorter strings. The following illustrates the respective computation times with 1000 datapoints in the training dataset and 1 test example.

	edit_dist	Jaro	Jaro-Winkler
comp. time(in secs)	2.669	0.586	0.571

8.5. The GUI

To implement the GUI we used PyQt5 which is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in .....



PyQt brings together the Qt C++ cross-platform application framework and the cross-platform interpreted language Python.

Qt is more than a GUI toolkit. It includes abstractions of network sockets, threads, Unicode, regular expressions, SQL databases, SVG, OpenGL, XML, a fully functional web browser, a help system, a multimedia framework, as well as a rich collection of GUI widgets.

Qt classes employ a signal/slot mechanism for communicating between objects that is type safe but loosely coupled making it easy to create re-usable software components.

Qt also includes Qt Designer, a graphical user interface designer. PyQt is able to generate Python code from Qt Designer. It is also possible to add new GUI controls written in Python to Qt Designer.

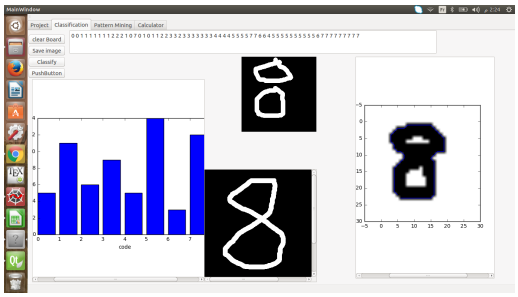
snap shot for the GUI our GUI contain mainly 3 tabs: the first tab is presentation of this project

the second tab is for the main taks classification task: in which the user can draw on the canvas the digit he want to recognize then using a button he can get the result which will be expressed in many outputs the freeman code generated will be printed in text the contour of the digit will be printed in blue in one of the canvases in that tab the histogram of the frequencies of the freeman code will be also plotted in another canvas and finally the recognized digit will be plotted in a another canvas

the most import widget of this tab is the drawing canvas which is defined as class Draw in the source code

the third one is for the sequence mining task in which the user can draw a digit on the canvas

the fourth tab is a calculator tab in were the user draw the digits on the canvas and then use the buttons to perform an arithmetic operation and get the result



8.6. Large Margin Nearest Neighbor

A Metric learning(similarity learning), Large margin nearest neighbor LMNN (from metric learning package) was implemented. It consists of learning a pseudo metric (Mahanalobis distance metric) under which all data instances in the training set are surrounded by at least k instances that share the same class label. In other words the metric is trained with the goal that the k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin. We used this metric to optimize the results. We got a very slight improvement of 1% in the results despite much tuning.

This algorithm failed to improved the score considerably. Sometimes when the classification

problem is not linearly separable the LMNN will learn a close matrix to the Identity matrix so the result will be almost the same, we can solve this problem by using the kernel PCA to project them in another space so the LMNN can perform better

[4]

8.7. Sequence mining

:

For this task we used the prefix span algorithm through the GUI Implemented in java on our generated freeman codes for every single digit in order to recognize the frequent sub sequences for every digit following these steps:

- we provided a text file that contain free man code sequences to the algorithm that convert them from a sequence data base to a SPMF format
- we used the prefix span algorithm with minimum support as :0.8
- next we processed these frequent patterns and converted it to a format which is similar to what we had earlier used as training data for digit recognition. We maintain this database for every digit.
- when an incomplete digit is drawn in the GUI, the freeman code generated searches this pattern in the database that we had created in the above step and our model tries to classify this incomplete pattern.

[5]

9. Conclusion

We successfully implemented a Hand written digit recognition System. For the structural representation of our data, we used Freeman's chain code and for numerical representation, we built a histogram of features from the Freeman' chain code of the digit. We trained our system on both MNIST and our own dataset and tested on images drawn on the canvas of the GUI. Our classification algorithm works better on structural representation than on numerical representation as in the case of former we preserve the sequential information but in the latter we lose it. We tried to solve the issue of disconnected components in the images drawn, to reduce the length of the generated freeman's code and to handle ROI by dilating the image. We faced difficulties in classifying an image if it has small digit in it as it is dilated and loses quite a few of its significant features. We tried to increase the performance of our model by using LMNN, however the algorithm is not suited to our data. Lastly, we implemented a frequent pattern mining algorithm which tries to classify a partially drawn digit drawn on the canvas.

[4] https://www.cs.cornell.edu/~kilian/code/lmnn/lmnn.html

[5] http://www.philippe-fournier-viger.com/spmf/

Bibliography

[1] Yuchun Lee. Handwritten digit recognition using k nearest-neighbor, radial-basis function, and backpropagation neural networks. 3:440–449, 09 1991.

[2] B Boser Le Cun, John S Denker, D Henderson, Richard E Howard, W Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, 1990.

[3] S. Knerr, L. Personnaz, and G. Dreyfus. Handwritten digit recognition by neural networks with single-layer training. *IEEE Transactions on Neural Networks*, 3(6):962–968, Nov 1992.

[4] Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako, and Hiromichi Fujisawa. Handwritten digit recognition: Benchmarking of state-of-the-art techniques. 36:2271–2285, 10 2003.

[5] Kevin Dreßler and Axel-Cyrille Ngonga Ngomo. On the efficient execution of bounded jaro-winkler distances. In *Proceedings of Ontology Matching Workshop*, 2014.