Digit Recognizer

Shaily
Btech IT 6th Sem
Indira Gandhi Delhi Technical University
for Women
Delhi,India
tayalshaily20@gmail.com

Veena Rawat
Btech IT 6th Sem
Indira Gandhi Delhi Technical University
for Women
Delhi,India
iamveenarawat@gmail.com

***Abstract*—In this work, we use SVM(Support Vector Machine) classifiers for handwritten digit recognition. According to input variables, the classifier was trained and tested. The algorithm outputs an optimal hyper plane which categorizes new example. In this work, using the converted grayscale image as input,approximately 98% handwritten digit recognition accuracy was obtained in the MNIST database.**

***Index Terms*—digit recognition,svm, support vector machine, classificaion**

I. Introduction

In recent times, character or digit recognition technology has been driven by the increasing demand of converting a large amount of printed or handwritten information to a digital format. Billions of checks, mail correspondence (including pin codes), etc. required large number of humans to process it and convert them to digital format. The process was time consuming, error prone and very tiresome, motivating the development of digit recognition systems.Character recognition is a very difficult problem, due to the great variability in writing styles, that is the same character can be written in different sizes and angles. In this paper, we discuss classification systems for handwritten digit recognition using four different fea- ture sets and SVM classifiers In this paper, we discuss classification systems for handwritten digit recognition using four different fea- ture sets and SVM classifiers In this paper, we discuss classification systems for handwritten digit recognition using four different fea- ture sets and SVM classifiers There are many applications for handwriting recognition. There are many techniques that have been developed to recognize the handwriting. One of them is Optical Character Recognition (OCR). OCR will read text from scanned document and translating the images into a form that computer can manipulate it. In this paper we will use Support Vector Machine (SVM) classification algorithm to recognize the handwritten digits.

II. Related Work

There are many research that have been done regarding the handwriting recognition in various field.Handwritten digit recognition by combining SVM classifiers,a zip code recognition system,wavelet based feature extraction , shape descriptor based on trainable COSFIRE filters. Youssouf Chherawala, Partha Pratim Roy and Mohamed Cheriet in their paper Feature Set Evaluation for Offline Handwriting Recognition Systems: Application to the Recurrent Neural Network stated that handwriting recognition system is dependent on the features extracted from the word image. There are various method to extract the features but there are no method that have been proposed to identify the most promising of these other than a straightforward comparison based on the recognition rate.

III. The System Architecture

The recognition system is constructed around a modular architecture of feature extraction and digit classification units. The preprocessed digit images are given as input to the feature extractor that transfers the extracted features toward SVM classifiers.

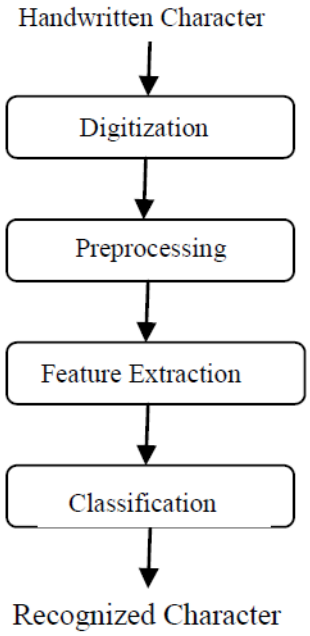


Fig. 1. Flow Diagram

A. Digitization and Preprocessing

The training/test digit image is converted to a graysacle image to make all the pixel values either 0 or 1. The image is then reduced to a symmetric size of 28x28pixel for easy identification. Hence, there are 784 pixels for each image.

B. Feature Extraction

- Every 784 pixel values for an image is the feature assosciated with that digit.
- These features are extracted and used for classification by SVM.

C. Classification

Support Vector Machine Classifier is used to perform its algorithm on the features extracted at the above step. These are then compared with the dataset used.

IV. Support Vector Machines

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for classification. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well.

A. Linear classification

In very simple terms what an SVM tries to accomplish is calculate ”something” which separates points labeled in one of two ways into two distinct groups.

In the figure below (source: wikipedia) we have white points and black points and a bunch of lines are drawn to try and separate the points. Of the three lines, only the red and blue ones do their job. The green line is not acceptable because on the right hand side of it we can find both black and white points.

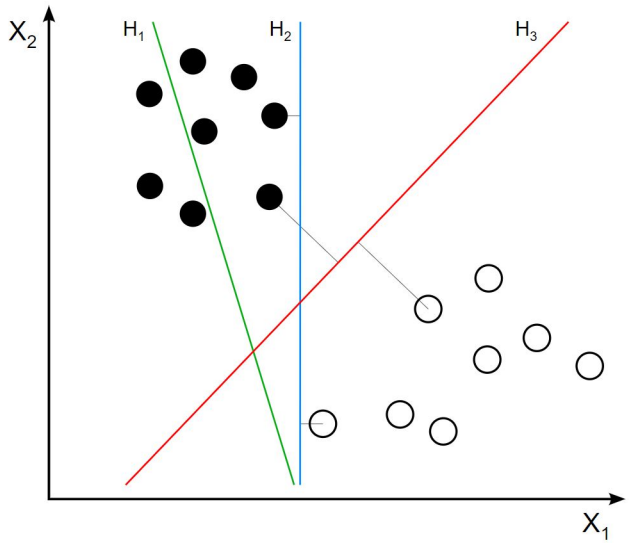


Fig. 2. Linear Classifier.

The blue one gets awfully close to some of the points and that raises the question: if another white point is observed close to the border will it be classified together with the other white points or misclassified as being black? The red line generalizes better.

B. HyperPlane

Well, first of all, the concept of a separating line is only valid if what we are trying to separate are 2D points. The ”something” I mentioned earlier is not necessarily a line. More generally, it is called a hyperplane. In 1D this is a point. In 2D, a line. In 3D a plane (not an aircraft). In n dimensions a hyperplane. Note that in general we try to classify n-dimensional points. A hyperplane is defined as

$$f(x) = \vec{w} \cdot \vec{x} + b = 0 \tag{1}$$

where

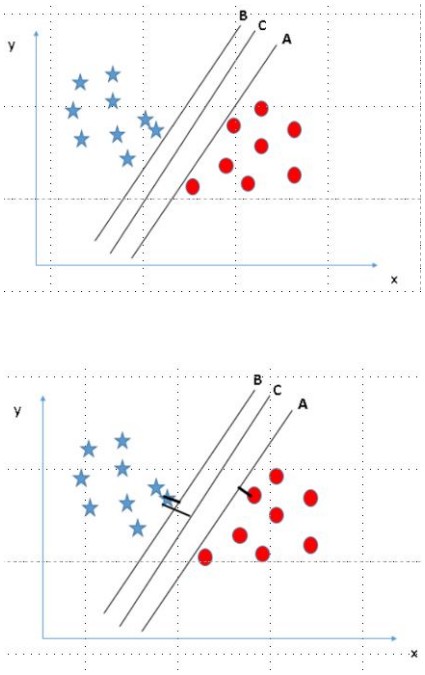
is the vector normal to the hyperplane (i.e. perpendicular to the hyperplane) and bb is something called a bias. To better understand where this is coming from go back to the 2D space and you'll see this is just another representation for a line.Take(-a, 1) and (x, y) and you get

$$\vec{w} \cdot \vec{x} = y - ax \tag{2}$$

$$\vec{w} \cdot \vec{x} - b = y - ax - b \tag{3}$$

Note that both +b and -b are used depending on who is doing the writing. Both are correct, as long as you are consistent with your notation. I will be using plus.

Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyperplane? Here, maximizing the distances between



nearest data point (either class) and hyper-plane willhelp us to decide the right hyper-plane. This distance is called asMargin. Hence, we name theright hyper-plane asC. If we select a hyper-plane having low margin then there is high chance of miss-classification.

C. Kernel Functions

Something else to do with the x vectors not being linearly separable is that while they may not be separable in their current space, they may be separable in another space.

For example some points may look one way (inseparable) in 2D but look differently (separable) when projected into 3D. Remember how I said earlier what's cool about that Lagrangian is we can replace that dot product? What we replace that dot product with is called a kernel function. These kernel functions transform the feature vectors (our x values) into a different space and separate them there.

See the figure below for a representation of this idea.

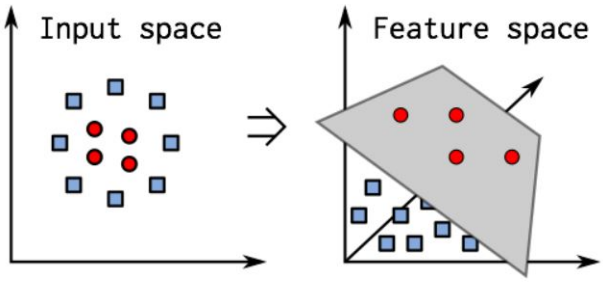


Fig. 3. Example of a figure caption.

So using a kernel function our Lagrangian becomes and is solved just like before. The difference is this time our separators will probably be more "creative", i.e. not as boring as a simple straight line, but curves, spheres etc.

Some popular kernel choices are: Note that not any function can be a kernel function and choosing a kernel function seems to be a matter of experimentation, cross-validation, and domain-specific knowledge related to the data the algorithm is trained on.

V. Database

	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	...	pixel772	pixel773	pixel774
0	0.0	0.0	0.0	0.333333	0.777778	0.769231	0.666667	0.8	0.705882	0.761905	...	0.16	0.358491	0.6250
1	0.0	0.0	0.0	0.333333	0.777778	0.769231	0.666667	0.8	0.705882	0.761905	...	0.16	0.377358	0.6875
2	0.0	0.0	0.0	0.333333	0.777778	0.769231	0.666667	0.8	0.705882	0.761905	...	0.16	0.377358	0.6875
3	0.0	0.0	0.0	0.333333	0.777778	0.769231	0.666667	0.8	0.705882	0.761905	...	0.16	0.377358	0.6875
4	0.0	0.0	0.0	0.333333	0.777778	0.769231	0.666667	0.8	0.705882	0.761905	...	0.16	0.377358	0.6875

Fig. 4. DataSet

MNIST(Modified National Institute of Standards and Technology) is a large database of handwritten digits.The images of handwritten digits are converted into gray scale image(i.e black and white) ans the size is reduced to 28x28 pixels. Each pixel has a number associated with it where 0 is a dark pixel and 1 is a white pixel. Every image consist of 784 pixels. The training and testing data contains 785 columns where first column represents the label. Other 784 columns represents the pixel value(0 or 1) of individual image. The train and test dataset contains 60,000 and 10,000 samples respectively.

VI. Conclusion

The digit recognition experiment was using a linear kernel to training set to obtain SVM classifier, and then to test and verify classifier with the test set.This was able to produce an accuracy of about 98%.

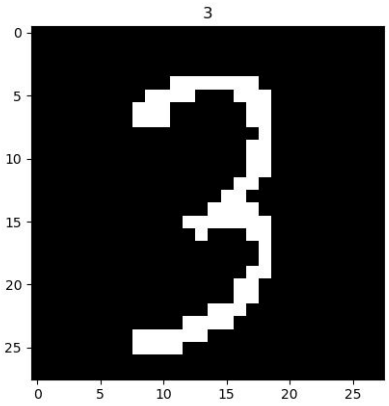


Fig. 5. Example Image

```
(base) C:\Users\tayal\Desktop\digit\digit_recog>python classification_compelete.py
Fitting this might take some time .....
predicting .....
Getting Accuracy .....
Score 0.98773006135
```

Fig. 6. Accuracy

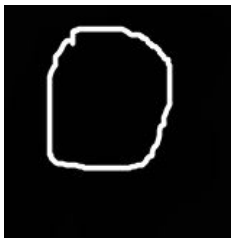


Fig. 7. Test/Drawn Image

Fig. 8. Predicted Image

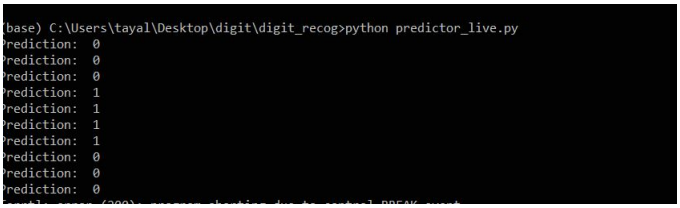


Fig. 9. Result

REFERENCES

[1] http://www.cristiandima.com/basics-of-support-vector-machines/
[2] https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
[3] https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
[4] https://www.kaggle.com/sanesanyo/digit-recognition-using-svm-with-98-accuracy
[5] https://en.wikipedia.org/wiki/Support-vector_machine