Digit Recognizer Report

Xiang Zhang
zxiang4@binghamton.edu

Siyu Liu
sliu121@binghamton.edu

Na Li
nli19@binghamton.edu

Abstract

Handwritten recognition is the ability of a computer to receive and interpret intelligible handwritten input from source such as paper documents, photographs, touch-screens and other devices. Our paper is based on features extraction methods. In this paper we will use HOG to extract features and then use SVM classification to recognize the handwritten digits. HOG is a very efficient feature descriptor for handwritten digits which is stable on illumination variation because it is a gradient-based descriptor. Moreover, linear SVM has better performance than polynomial, RBF and sigmoid kernels.

1. Introduction

Handwriting recognition is the ability of a computer or device to take as input handwriting from source such as printed physical documents, pictures and other devices. It also can use handwriting as a direct input to a touch screen and then interpret it as text. There are many devices now can take handwriting as an input such as smart phones, tablets and PDA to a touch screen through a stylus or finger. This is useful as it allows the user to quickly write down number and text to the devices. There are many applications for handwriting recognition are available this day. There are many technique that have been developed to recognize the handwriting. In this paper we will use SVM classification to recognize the handwritten digits.

2. Background and Motivation

Hand writing recognition of characters has been around since the 1980s.The task of handwritten digit recognition, using a classifier, has great importance and use such as online handwriting recognition on computer tablets, recognize zip codes on mail for postal mail sorting, processing bank check amounts, numeric entries in forms filled up by hand (for example tax forms) and so on. There are different challenges faced while attempting to solve this problem. The handwritten digits are not always of the same size, thickness, or orientation and position relative to the margins. Our goal was to implement a pattern classification method to recognize the handwritten digits provided in the MINIST data set of images of hand written digits (09). Each image is a 28 x 28 grayscale (0255) labeled representation of an individual digit.

The general problem we predicted we would face in this digit classification problem was the similarity between the digits like 1 and 7, 5 and 6, 3 and 8, 9 and 8 etc.

People write the same digit in many different ways, which we can see in the Figure 1. Finally the uniqueness and variety in the handwriting of different individuals also influences the formation and appearance of the digits.

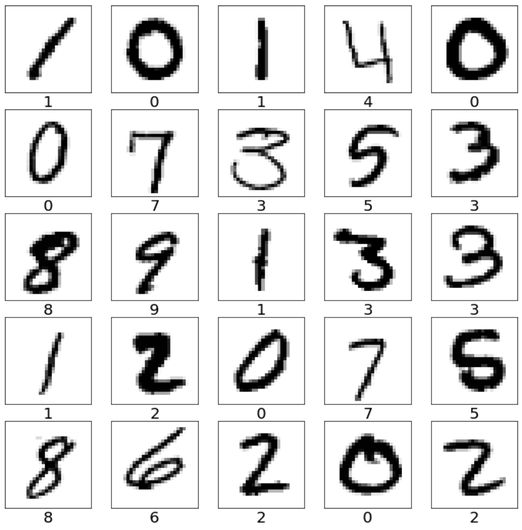


Figure 1. Handwritten Digits Samples

3. Related Work

3.1. Convolutional Neural Network(CNN)

This paper [5] is the original inspiration and basis for the improvements in accuracy we made to the character dataset. The paper describes the process they used to achieve up to a 99.1% accuracy on the MNIST dataset, using both a 3-layer convolutional network and a 5-layer network that failed to outperform the former. Additionally, the paper discusses the problem of segmentation, and how it cannot be decoupled from the recognition of isolated characters. Essentially, making the decision to segment an image with multiple characters before the recognition of individual characters is not optimal, as the process should be parallelized to test multiple hypotheses at the same time. In terms of feature extraction, LeCun et. al. argues that humans cannot possibly capture all relevant information from images and even coming close requires expert knowledge on the subject, so instead they resorted to utilizing GradientBased Learning to learn the useful features for the images.

3.2. K-nearest neighbor(KNN)

KNN [4] is one of the simplest classifier to implement because it is a brute force method. The algorithm is used to find the nearest match for a given case to the known cases stored in memory. The K is used to signify how many votes are used for decision making. It is most optimal to choose a value of K that is odd so it eliminates a tie between two sets. There are three steps to implement for this classifier: First step: compare known samples with the test sample using a distance metric. The distance metric is used to find the nearest neighbor. The most widely used distance metric is the Euclidean distance, because it gives a normalized value.

3.3. Logistic Regression

Logistic regression is a probabilistic, linear classifier. It is parametrized by a weight matrix W and a bias vector b. Classification is done by projecting an input vector onto a set of hyperplanes, each of which corresponds to a class. The distance from the input to a hyperplane reflects the probability that the input is a member of the corresponding class.

4. Feature Extraction

Feature extraction is a type of dimensionality reduction that efficiently represents interesting parts of an image as a compact feature vector. This approach is useful when image sizes are large and a reduced feature representation is required to quickly complete tasks such as image matching and retrieval.

Feature detection, feature extraction, and matching are often combined to solve common computer vision problems such as object detection and recognition, content-based image retrieval, face detection and recognition, and texture classification.

The following features extraction methods are often used in training classifier:

- Histogram of oriented gradients (HOG)
- Speeded-up robust features (SURF)
- Local binary patterns (LBP)
- Haar wavelets
- Color histograms

We use HOG in our project, which provide us good results.

5. Approach

We have proposed a Deskew + HOG + SVM approach which process deskewed image data using Histogram of Oriented Gradients (HOG), and then train the feature data by SVM model. Moreover, linear SVM has been employed as classifier which has better responses than polynomial, RBF and sigmoid kernels.

We have analyzed our model on MNIST dataset and 99.085% accuracy rate has been achieved which is comparable with the state of the art. The block diagram for our approach is shown below Figure 2

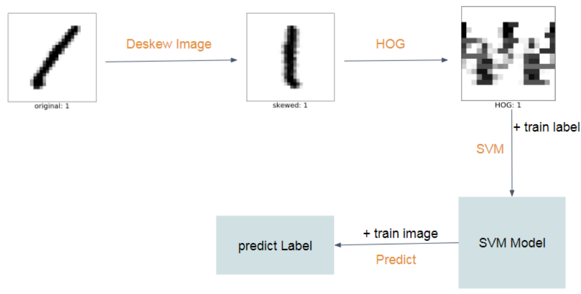


Figure 2. Our approach pip-line

5.1. Deskew

The basic idea of the algorithm is:

- Find reference lines in the image
- Calculate the angle of the lines
- Calculate the skew angle as an average of the angles
- Rotate the image

The lines are detected with the Hough algorithm [3]. Each point in the image can lie on an infinite number of lines. To find the reference lines, we let each point vote for all the lines that pass through the point. The lines with the highest number of points are our reference lines. The result shows in Figure 3.

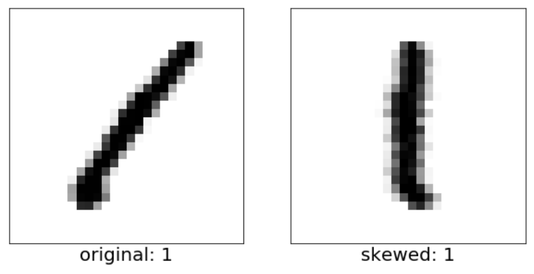


Figure 3. Image Deskew

5.2. Histogram of Oriented Gradients

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

Robert K. McConnell of Wayland Research Inc. first described the concepts behind HOG without using the term HOG in a patent application in 1986. In 1994 the concepts were used by Mitsubishi Electric Research Laboratories. However, usage only became widespread in 2005 when Navneet Dalal and Bill Triggs [2], researchers for the French National Institute for Research in Computer Science and Automation (INRIA), presented their supplementary work on HOG descriptors at the Conference on Computer Vision and Pattern Recognition (CVPR). In this work they focused on pedestrian detection in static images, although since then they expanded their tests to include human detection in videos, as well as to a variety of common animals and vehicles in static imagery.

Figure 4 shows the pipeline of HOG.

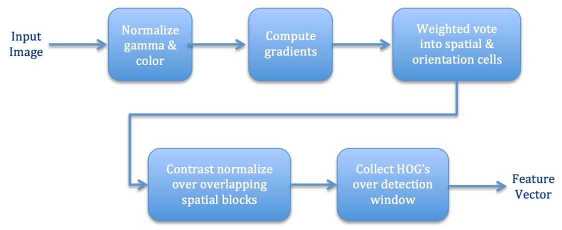

Input Image
Normalize gamma & color
Compute gradients
Weighted vote into spatial & orientation cells
Contrast normalize over overlapping spatial blocks
Collect HOG's over detection window
Feature Vector

Figure 4. Pipeline of HOG

5.2.1 Algorithm implementation

Gradient computation The first step of calculation in many feature detectors in image pre-processing is to ensure normalized color and gamma values. As Dalal and Triggs point out, however, this step can be omitted in HOG descriptor computation, as the ensuing descriptor normalization essentially achieves the same result. Image pre-processing thus provides little impact on performance. Instead, the first step of calculation is the computation of the gradient values. The most common method is to apply the 1-D centered, point discrete derivative mask in one or both of the horizontal and vertical directions. Specifically, this method requires filtering the color or intensity data of the image with the following filter kernels:
$[-1, 0, 1]$ and $[-1, 0, 1]^T$

Dalal and Triggs tested other, more complex masks, such as the 3x3 Sobel mask or diagonal masks, but these masks generally performed more poorly in detecting humans in images. They also experimented with Gaussian smoothing before applying the derivative mask, but similarly found that omission of any smoothing performed better in practice.

Orientation binning The second step of calculation is creating the cell histograms. Each pixel within the cell casts a weighted vote for an orientation-based histogram channel based on the values found in the gradient computation. The cells themselves can either be rectangular or radial in shape, and the histogram channels are evenly spread over 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is unsigned or signed. Dalal and Triggs found that unsigned gradients used in conjunction with 9 histogram channels performed best in their human detection experiments. As for the vote weight, pixel contribution can either be the gradient magnitude itself, or some function of the magnitude. In tests, the gradient magnitude itself generally produces the best results. Other options for the vote weight could include the square root or square of the gradient magnitude, or some clipped version of the magnitude.

Descriptor blocks To account for changes in illumination and contrast, the gradient strengths must be locally normalized, which requires grouping the cells together into larger, spatially connected blocks. The HOG descriptor is then the concatenated vector of the components of the normalized cell histograms from all of the block regions. These blocks typically overlap, meaning that each cell contributes more than once to the final descriptor. Two main block geometries exist: rectangular R-HOG blocks and circular C-HOG blocks. R-HOG blocks are generally square grids, represented by three parameters: the number of cells per block, the number of pixels per cell, and the number of channels per cell histogram. In the Dalal and Triggs human detection experiment, the optimal parameters were found to be four 8x8 pixels cells per block (16x16 pixels per block) with 9 histogram channels. Moreover, they found that some minor improvement in performance could be gained by applying a Gaussian spatial window within each block before tabulating histogram votes in order to weight pixels around the edge of the blocks less. The R-HOG blocks appear quite similar to the scale-invariant feature transform (SIFT) descriptors; however, despite their similar formation, R-HOG blocks are computed in dense grids at some single scale without orientation alignment, whereas SIFT descriptors are usually computed at sparse, scale-invariant key image points and are rotated to align orientation. In addition, the R-HOG blocks are used in conjunction to encode spatial form information, while SIFT descriptors are used singly. Circular HOG blocks (C-HOG) can be found in two variants: those with a single, central cell and those with an angularly divided central cell. In addition, these C-HOG blocks can be described with four parameters: the number of angular and radial bins, the radius of the center bin, and the expansion factor for the radius of additional radial bins. Dalal and Triggs found that the two main variants provided equal performance, and that two radial bins with four angular bins, a center radius of 4 pixels, and an expansion factor of 2 provided the best performance in their experimentation(to achieve a good performance, at last use this configure). Also, Gaussian weighting provided no benefit when used in conjunction with the C-HOG blocks. C-HOG blocks appear similar to shape contextdescriptors, but differ strongly in that C-HOG blocks contain cells with several orientation channels, while shape contexts only make use of a single edge presence count in their formulation.

Block normalization Dalal and Triggs explored four different methods for block normalization. Let v be the non-normalized vector containing all histograms in a given block, $||v||k$ be its k-norm for $k = 1, 2$ and e be some small constant (the exact value, hopefully, is unimportant). Then the normalization factor can be one of the following:

$L2 - norm : f = \frac{v}{\sqrt{||v||_2^2 + e^2}}$
L2-hys: L2-norm followed by clipping (limiting the maximum values of v to 0.2) and renormalizing
$L1 - nor : f = \frac{v}{||v||_1 + e}$
$L1 - sqrt : f = \sqrt{\frac{v}{||v||_1 + e^2}}$

In addition, the scheme L2-hys can be computed by first taking the L2-norm, clipping the result, and then renormalizing. In their experiments, Dalal and Triggs found the L2-hys, L2-norm, and L1-sqrt schemes provide similar performance, while the L1-norm provides slightly less reliable performance; however, all four methods showed very significant improvement over the non-normalized data.

One sample of HOG feature extraction from a digit number in MNIST shows in Figure5.

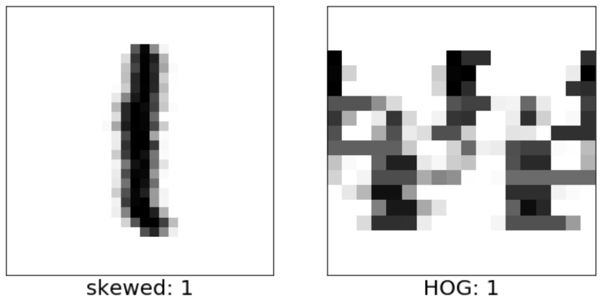


Figure 5. HOG

5.3. Support Vector Machine

In machine learning, support vector machines [1] (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data are not labeled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The clustering algorithm which provides an improvement to the support vector machines is called support vector clustering and is often[citation needed] used in industrial applications either when data are not labeled or when only some data are labeled as a preprocessing for a classification pass.

For a given set of points belonging to two classes, an SVM tries to find a decision function for an optimal separating hyperplane (OSH) which maximizes the margin between the two sets of data points. The solutions to such optimization problems are derived by training the SVM with sets of data similar to what it may encounter during its application. It is shown in Figure6

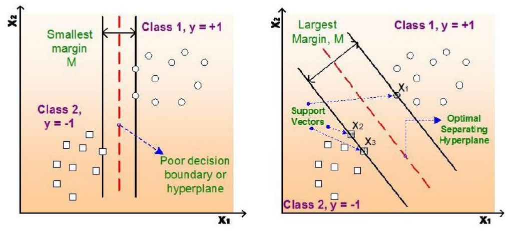


Figure 6. SVM

Digit recognition is a multi-class problem, but normal SVM can only separate two classes. So, one v.s. all method is applied here to realize a multi-SVM classifier [6].

One v.s. all is to set one class as class 1 and the others as class 2, then a SVM classifier is trained using the redefined dataset. So, there will be 10 different SVM classifiers since there are 10 digits (09). In prediction process, a new sample will be tested on each classifier, and each classifier will yield a estimated label and corresponding probability (confidence of estimation). Finally, the estimation with the highest confidence will be accepted as the finally output of the multi-SVM classifier.

6. Experiments

6.1. Dataset

The MNIST (modifed NIST) database [5] was extracted from the NIST special databases SD3 and SD7. SD3 and SD7 were released as the training and test data sets, respectively, for the First Census OCR Systems Conference. We use it on Kaggle, there's 42k images for training and 27k images for testing. Figure 1 is some sample of MNIST dataset.

6.2. Compare Method

We compare several methods from open kernels on Kaggle. We compare our Kaggle score with these methods. We tried:

- KNN
- Logistic Classifier
- PAC + SVM
- CNN

6.3. Validation

Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it.

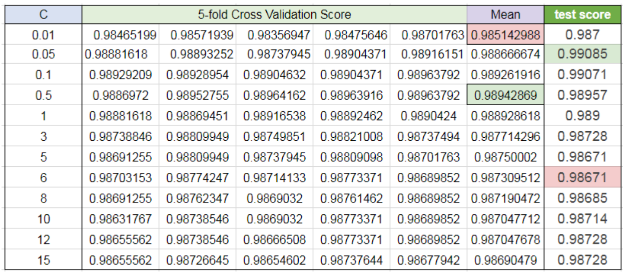
In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.

In our project, We applied 5-fold cross-validation and calculate its mean value to evaluate our training model.

Kernels	Linear	Polynomial	RBF	Sigmoid
Accuracy	0.99	0.89	0.95	0.94

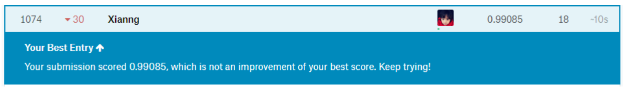
Table 1. Accuracy from different SVM kernels

7. Results



C	5-fold Cross Validation Score					Mean	test score
0.01	0.98465199	0.98571939	0.98356947	0.98475646	0.98701763	0.985142988	0.987
0.05	0.98881618	0.98893252	0.98737945	0.98904371	0.98916151	0.988666674	0.99085
0.1	0.98929209	0.98928954	0.98904632	0.98904371	0.98963792	0.989261916	0.99071
0.5	0.9886972	0.98952755	0.98964162	0.98963916	0.98963792	0.98942869	0.98957
1	0.98881618	0.98869451	0.98916538	0.98892462	0.9890424	0.988928618	0.989
3	0.98738846	0.98809949	0.98749851	0.98821008	0.98737494	0.987714296	0.98728
5	0.98691255	0.98809949	0.98737945	0.98809098	0.98701763	0.98750002	0.98671
6	0.98703153	0.98774247	0.98714133	0.98773371	0.98689852	0.987309512	0.98671
8	0.98691255	0.98762347	0.9869032	0.98761462	0.98689852	0.987190472	0.98685
10	0.98631767	0.98738546	0.9869032	0.98773371	0.98689852	0.987047712	0.98714
12	0.98655562	0.98738546	0.98666508	0.98773371	0.98689852	0.987047678	0.98728
15	0.98655562	0.98726645	0.98654602	0.98737644	0.98677942	0.98690479	0.98728

Figure 7. Result



1074 ▾30 Xianng 0.99085 18 ~10s

Your Best Entry ↑

Your submission scored 0.99085, which is not an improvement of your best score. Keep trying!

Figure 8. Rank and Score On Kaggle

From Figure 7 we find the best cross validation we get when we set C with 0.5, and we get the test score 0.98957 on Kaggle. However, If we set C with 0.005, we get our best test score on Kaggle, which is 0.99085. The rank and score we get on Kaggle are showing on Figure 8

We also analyzed polynomial, RBF and sigmoid kernels that there was not any improvement than linear SVM; their accuracy has been shown in table 1.

Compare to other methods, our result is also kind of good. Our result perform better than KNN, Logistic Classifier, PCA+SVM, but worse than CNN. Right now, the highest score on Kaggle are all Deep learning models. Someone even implement the 100% accuracy. The compare result shows in the Table 2.

Method	Accuracy
KNN	0.971
Logistic Classifier	0.916
PCA+SVM	0.984
CNN	0.9975
Our Method	0.99085

Table 2. Compare result

References

[1] C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society.

[3] R. Duda, P. Hart, S. I. M. P. C. A. I. CENTER., and S. R. I. A. I. Group. *Use of the Hough Transformation to Detect Lines and Curves in Pictures*. AD-a457 992. sri international menlo park ca artificial intelligence center, 1971.

[4] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer. Knn model-based approach in classification. In R. Meersman, Z. Tari, and D. C. Schmidt, editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 986–996, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[5] Y. Lecun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In F. Fogelman and P. Gallinari, editors, *International Conference on Artificial Neural Networks, Paris*, pages 53–60. EC2 Cie, 1995.

[6] M. Pal. Multiclass approaches for support vector machine based land cover classification. *CoRR*, abs/0802.2411, 2008.