Machine Learning Final Project

Handwritten Formula Competition

0386032 – Jan Kristanto

Introduction

The Handwriting digit recognition has been studied for long time and many researchers already published some papers. Character recognition has similar well-studied. There are two categories for general handwritten recognition. First is offline where the text was written in a non-digital medium and online recognition, where the text was written on a special digitzer where the sensor picks up the pen-tip movements. Although the offline recognition still lags behind the online handwritten accuracy.

The handwriting recognition is to input data of mathematical expressions into a computer or tablet. Handwriting recognition is a process that aims to identify the most appropriate character from handwritten strokes. It is done by extracting features from a set of handwritten strokes, then apply a machine learning technique to build a model that classifies a stroke into a class of symbols.

The main goal of this project is to recognize handwritten formula and compute the answers. It is fundamentally a supervised learning problem in that a training set of labelled images is provided. From the given a set of handwritten formula images, we write an algorithm to recognize these images and try to compute the correct answers which equals to the image labels. In this Machine Learning Final Project, I am going to implement the handwritten math formula recognition (offline).

Data

I am given datasets consist of two parts. First is the data training (the symbols) and data testing (the formulas). In the data training, there are 22 symbols, each symbol has 515 image samples. The testing is different. Not just exactly with the data training. But in form of formula. I need to do some detection and segmentation to get the symbols like the training data. There are 5 level data testing (formulas). Each level consists of 30 testing formula. All of the data is in image (PNG) format.

Classifier Model

The first and the most important (because it’s embodies a Machine Learning Technique) part of the project is model selection and training. I was thinking of Convolutional Neural Networks for no other Machine Learning model performs better on images than CNN. Neural Networks are always hard to train, especially because they require a lot of time, memory. Besides, it is always difficult to find the optimized parameters which perform well both on training and test data. Because the difficulties of CNN, I try three model in here, Support Vector Machine (SVM) , V-SVM, C-SVM and Neural Network. I am using LIBSVM to implement SVM. Before build model need to do preprocessing and feature extraction. Here I use Histogram of Oriented Gradients (HOG) as feature extraction.

Preprocessing

The purpose of this step is to convert the image data to matrix Matlab format and also do some adjustment. There are two preprocessing in here. Preprocessing for training data and preprocessing for testing data (formula).

Preprocessing for training data

1. Transform image (100 x 100) pixels to vector

2. For the image with 100 x 200 pixels. Resize the image to 50 x 100, then add padding 25 x 100 pixels on the up and down images.

Preprocessing for testing data (formula)

1. Convert the RGB image to grayscale image.

2. Because each image has different size, but still around 1030 x 240. So resize the image to that size.

3. Convert image pixels to vector.

Feature extraction

I use Hog for feature extraction. Histogram of oriented gradients is a point descriptor used to discover ends in computer act or power of seeing and image processing. The HOG descriptor expert way of art and so on counts events of degree of slope adjustment in made near, not general divisions of an image - discovery window, or field, range of interest.

V-SVM Training

I use LIBSVM to implement V-SVM on the training phase. I use the V-SVM and Radial basis function (RBF) kernel. V-SVM is used when optimizing the parameters ν included to replace the original C, ν can be regarded as margin error in ξ_n > 0 part of the upper bound, but also has support vector of the lower bound.

$$\begin{aligned} &\mathit{maximize} && \tilde{L}(a) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m) \\ &\mathit{subject\ to} && 0 \le a_n \le 1/N \\ & && \sum_{n=1}^{N} a_n\, t_n = 0 \quad \mathit{and} \quad \sum_{n=1}^{N} a_n \ge \nu \end{aligned}$$

Radial basis function (RBF) is a term that describes the function of real value that output depends exclusively on the distance input from several origins. Typically, radial basis functions are defined in terms of input vectors standard Euclidean norm, but technically one can use other norms as well. In machine learning, radial basis functions are most often used as the kernel for classification by support vector machine (SVM). For this application of RBFS, Gaussian radial basis functions are almost always selected.

$$k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\gamma|\boldsymbol{x}-\boldsymbol{x}'|^2}$$

In this case I am using parameter v = 0.06. I got **99%** accuracy for training data.

C-SVM Training

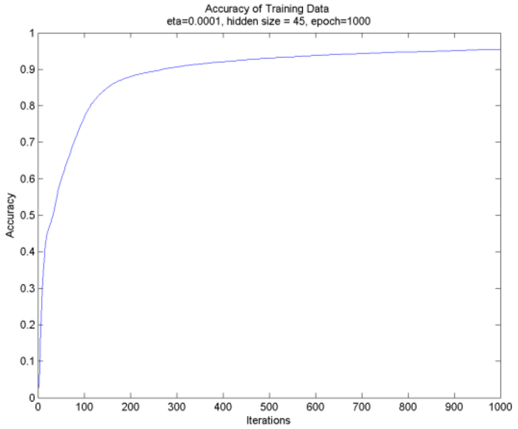
I use LIBSVM to implement C-SVM on the training phase. I use the C-SVM and Radial basis function (RBF) kernel. The C - SVM method used when Data distribution is overlapping, that is, when inseparable data, given the error margin in the range of data corresponding to the error distance penalty. According to the lecture note, the problem of SVM can be transferred into the dual representation.

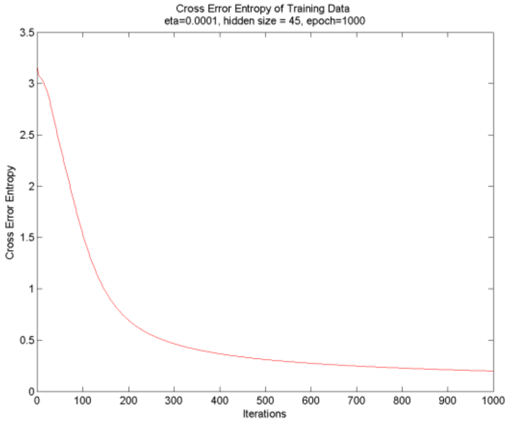
$$\begin{aligned} &\mathit{maximize} && \tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m) \\ &\mathit{subject\ to} && 0 \le a_n \le C \\ & && \sum_{n=1}^{N} a_n\, t_n = 0 \end{aligned}$$

In this case I am using parameter C = 10. I got **98%** accuracy for training data.

Neural Network Training

I use code from the Homework 3 to implement this NN training. LIBSVM to implement C-SVM on the training phase. I am using 1 hidden layer Neural network with parameters eta = 0.0001, hidden size = 45 and iterations = 1000. The accuracy regarding with training data is around **95%**. In here my input is only 81 features because I use HOG feature extractor before throw the training data to Neural Network training. The figure below is the NN training. The left one is plot the accuracy versus epoch and the right one is the plot Cross Error Entropy versus epoch.





Detection and Segmentation

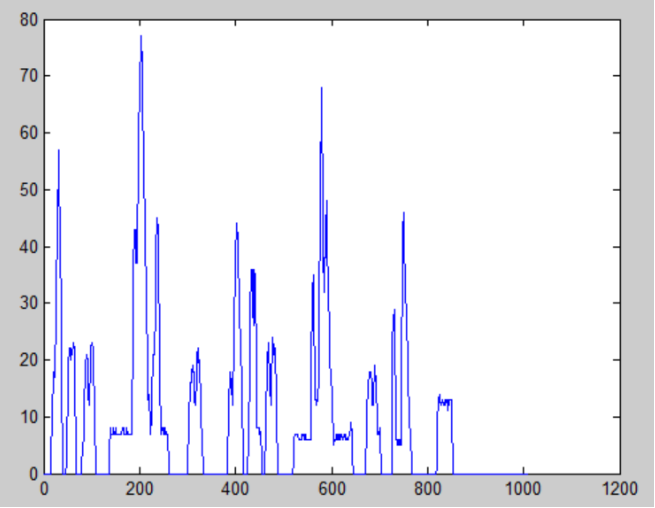
After the model has been trained, the second step of the project will be to transform the given test images to a string so that they could be solved by some algorithm. In this problem, I use Vertical Horizontal Projection and Component Connected labeling.

Vertical Projection is to find the sum of pixels in each column of an image. This is called the vertical projection. Where we can use the vertical projection to vertically separate connected characters. Horizontal Projection is to find the sum of pixels in each row of an image. This is called the horizontal projection. Where we can use the horizontal projection to horizontally separate connected characters.

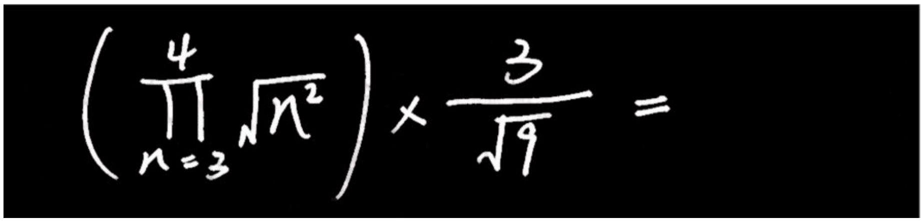
Component Connected labeling is just simply Finding connected components in images. it allows to find adjacent pixels that have the same input value.

I use those techniques to extract the symbols from the image formula. Here is my algorithm :

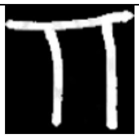
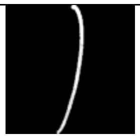
1. Recursive Vertically and Horizontally projection until convergence (the image can not be subdivided).
2. Perform component connected labelling. Just in case any symbols that can not separate by projection.
3. Cut the symbols until the smallest size, but without change the dimension of the symbols.
4. Before I throw to the classifier. I adjust the symbols as much as like the training data. In term of size. So I add padding to make the symbols square and resize the symbol to 100x100 but without ruin the proportionality.



The figure above is the histogram of Vertical Projection. Based on the histogram we can cut the image and extract the symbols. This process occurs recursively until convergence.



Here, I try to show a sample formula and the segmentation results.

1	2	3	4	5	6	7	8
							
9	10	11	12	13	14	15	16
							
17							

Interpretation & Evaluation

The last step of the project will be to compute the answer for the given mathematical expression. This step is not related to Machine Learning or Computer Vision. This step will get more difficult as we move on to higher levels. I just use the “eval” function in Matlab, but there are validation and transformation before that for Summations, Products and their sub and super-scripts and powers.

Sigma and Product

1. Find sigma/product symbols
2. Find the digit before sigma/product symbol and recognize as digit ending.
3. Find the m or n after sigma/product symbols
4. Find the digit after m or n character
5. Find m or n character until non digit symbol.

Square Root

1. Find sqrt symbols
2. Find the small digit before sqrt symbol and
3. Find the m or n or digit after sqrt symbols until operator symbols

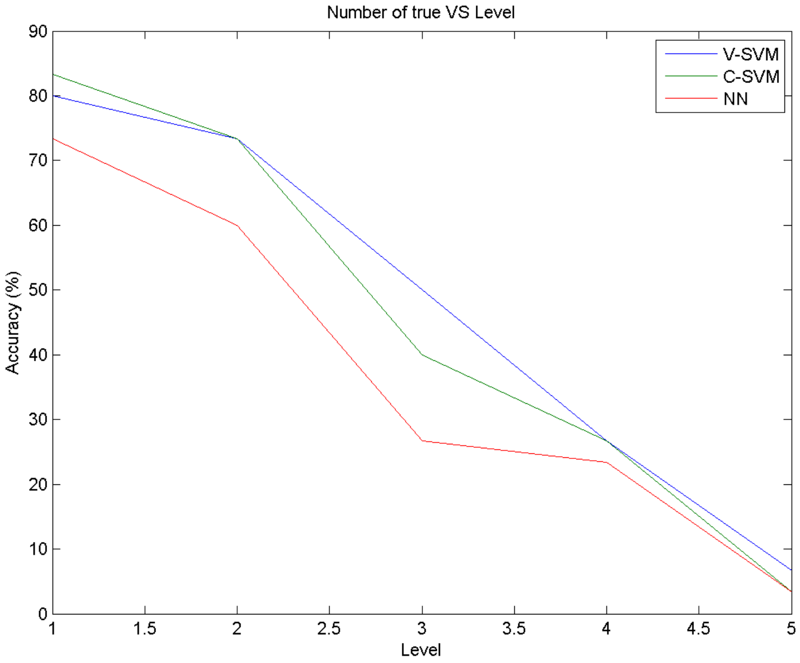
Power

1. Take a look the size of the digit and compare to the previous digit.

The table below is the accuracy and correct prediction for those three model, V-SVM , C-SVM and Neural Network regarding with the testing data (formula).

Level	V-SVM	C-SVM	NN	The best Model
Level 1	24 (0.8)	25 (0.833)	22 (0.733)	C-SVM
Level 2	22 (0.733)	22 (0.733)	18 (0.6)	V-SVM & C-SVM
Level 3	15 (0.5)	12 (0.4)	8 (0.266)	V-SVM
Level 4	8 (0.266)	8 (0.266)	7 (0.233)	V-SVM & C-SVM
Level 5	2 (0.066)	1 (0.033)	1 (0.033)	V-SVM
Accuracy	**71 / 150 (0.473)**	**68 / 150 (0.453)**	**56 / 150 (0.373)**	**V-SVM**

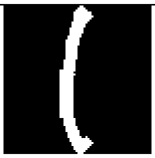
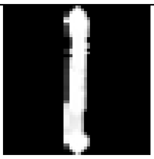
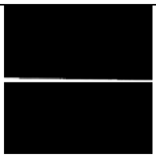
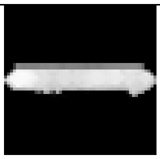
Based on the table above, the V-SVM win almost in all level except in the level 1, where C-SVM perform better than V-SVM. Whereas the Neural Network performs not so good in here. The figure below is the plot of comparison among the model. Blue line is V-SVM, green line is C-SVM and red line is Neural Network.



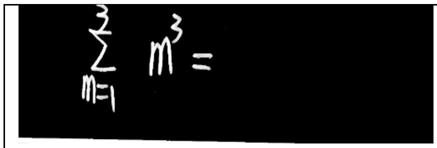
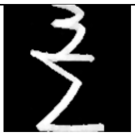
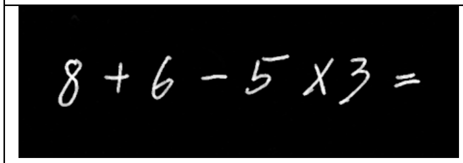
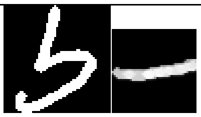
Limitation & Future Work

I realize that my machine still far away from satisfied performance. I am trying to point out the drawback and my proposed solution that I wish could overcome the drawback. Basically There three important elements in this task.

First is the Model / Classifier. If we have more robust classifier, it can reduce the miss classify symbols and avoid miss calculation as well. In my opinion the V-SVM is quite good although it very sensitive and we need to adjust the test data as much as similar with the training data. In my case, my classifier often missed classify to recognize "(" and "1" , and also "fraction" and "subtract". My idea to solve this problem is to try other model than HOG and SVM, or at least use more robust feature extraction like CNN.

(	1	Fraction	Subtract
			

Because the hand writing is complicated. There are a lot of style and also sometimes people accidentally "write" improper symbols. It makes the detection and segmentation harder. To solve this problem, my idea is to exploit the Morphology opening closing image, although it will damage the image too. The other idea, we can try to use window sliding and classifier to detect the symbols.

		Sigma and 3 failed separated
		5 accidentally separated

Last is to compute the answer for the given mathematical expression. In this machine I just use string to store the equations. In my opinion, we have to use the standard format, like MathML, Latex or at least Tree.