# Machine Learning Final Report

Chu-en Ho, *B00902006,* Hao-en Sung, *B00902064,* and Hsuan-ting Chen, *B00902108*

***Abstract*—In this report, we will show our machine learning framework, preprocessing procedures and experiments on several models, including kernel-based, tree-based, and deep learning models. Furthermore, a deeply discussion is given at the end of this work.**

***Keywords*—Picture Preprocessing, Machine Learning Models, Model Ensemble**

## I. INTRODUCTION

In this era of *Big Data*, machine learning skills become more and more essential for each person. After learning these skills in *Machine Learning* Course held by Professor *Hsuan-Tien* this semester, now is time for us to put them into practice.

The formal definition of two tracks in final project is given in section 2. In section 3, we elaborate all methods that we have tried, including *Preprocessing, Models, Ensemble, and Cost-sensitive Error Matrix*. Section 4 introduces our design for experiments. There are detailed records of our experiments in section 5. We have a complete discussion of our observation and a conclusion in section 6.

## II. PROBLEM DEFINITION

This final project is a handwriting recognition problem. Each instance is a figure of 12810 (122 $\times$ 105) pixels. There are total of 32 labels, including 12 Chinese Animal Zodiacs, 10 lowercase Chinese figures, and 10 uppercase Chinese figures.

This project has two tracks and two phases. In the first phase, train set and test set without label are given. In the second phase, the ground truth label of previous test set is announced and a new test set without label is given. The difference of track 0 and track 1 will be demonstrated as following:

*1) Track 0:* In this track, given $x$, $y$, and hypothesis $g$, the error function is $[\![y \neq g(x)]\!]$

*2) Track 1:* There are five possible error circumstances for each instance in this track.

| | |
|---|---|
| • no error: | error of 0 |
| • error between *Number* and *Zodiac*: | error of 4 |
| • error between *Zodiacs*: | error of 2 |
| • error between *Number cases*: | error of 1 |
| • error between *Same-cased Numbers*: | error of 2 |

## III. METHODOLOGY

### A. Framework

We have a robust chain to analyze and process these handwriting data. It is depicted completely by [Fig. 1].

First Edited January 17, 2015.

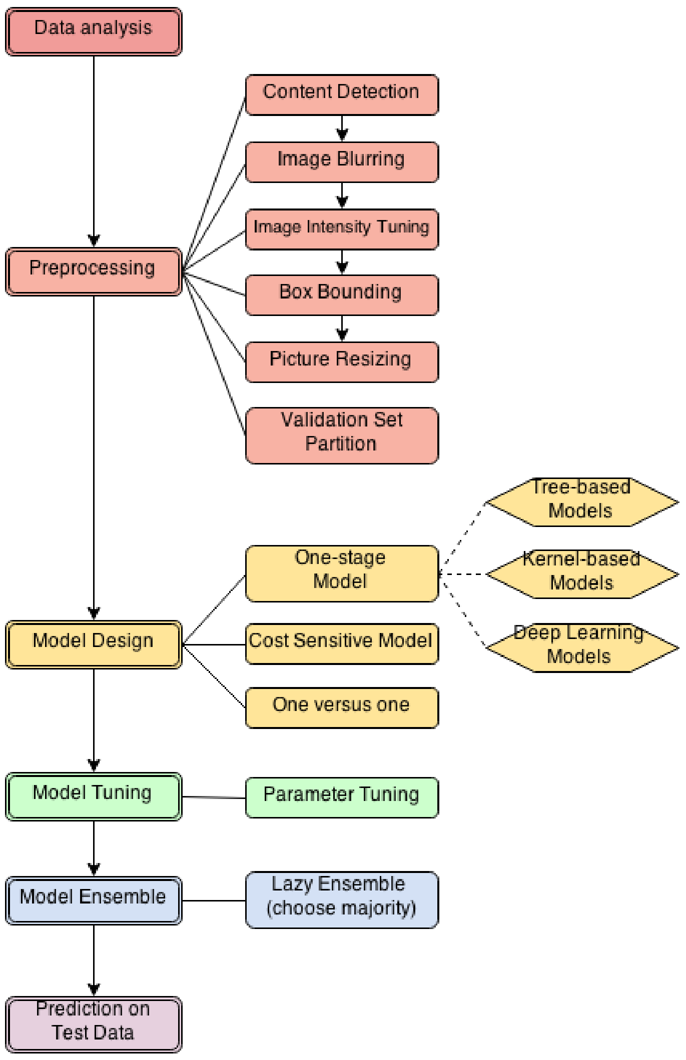

**Fig. 1:** Our framework

### B. Observing

The very first step of our work is to render these pictures. After using human intelligence, we find out there are some known issues for these figures.

1) Some figures are destroyed by ranodm circle-like noise with radius $1/4$ figure size.
2) Some figures have straight-line-like noise. They are probably created when scanning.
3) Some figures have barely no information.
4) Some characters are not in the central of figures.

Issue 1 and 2 are hard to deal with if we know few about image recovery algorithms. We can solve issue 3 easily by performing Content Detection. Issue 4 can be solved by Box bounding.

### C. Preprocessing

For those tasks that human have a better insight, e.g. image recognition or handwriting recognition, image preprocessing is the key to achieve good performance. Since we are new to image preprocessing, we only try a few intuitive methods and some given toolkits, e.g. MATLAB, in our preprocessing procedure. These methods include:

*1) Content Detection:* We test the summation of pixel values for each instance. If the summation exceeds certain boundary, e.g. 50, we will keep this instance, or just drop it. Our assumption is that: instances with barely no information are similar to random-like noise. An example that surpasses detection boundary is [Fig. 2].

**Fig. 2:** Origin Figure (122 × 105)

*2) Box Bounding:* We use a brute force way to detect the left, right, top, bottom boundary of each figure. To be clear, for left (right, top, bottom) boundary, we aim to find out a certain column (column, row, row) that is left-most (right-most, top-most, bottom-most) and it has non-zero summation.

*3) Picture Resizing:* Since box-bounded figures may have different sizes, we need to perform normalization on them. Another reason for picture resizing is that the smaller the figure, the faster the model training and prediction. Furthermore, smaller figure will force the aggregation of features and enhance the chance of overlapping of strokes in figures. That is the reason why we ususally down-scale the figure size. The cropped and rescaled example is shown as [Fig. 3].

**Fig. 3:** Cropped and Resized Figure (122 × 105)

*4) Image Median Filtering:* Use midfilt2 to perform median filtering with 3-by-3 matrix. The mean value of 3-by-3 neighbor pixel will substitutes for the origin ones. It will dispel noises of the image to perform following processes. The median filtered figure is shown as [Fig. 4].

**Fig. 4:** 3 × 3 Median Figure (122 × 105)

*5) Image Blurring:* The motion mode of fspecial can blur images properly. This method blurs the image by mimic the linear motion of a camera. This can reduce the divergence between two handwrites which has different margins between strokes. The blurred figure is shown as [Fig. 5].

**Fig. 5:** Blurred Figure (122 × 105)

*6) Image Intensity Manage:* We use the method *imadjust* in MATLAB to do image intensity manage. Imadjust is different from histogram equalization, which increases the local contrast in an image. We use the most basic form of imadjust, which just linearly maps the lowest and highest value into [0 1]. This enhances the global contrast but doesnt change the relative relation of any two locations in the image. This method can reduce the influence of the strength of handwrites but we can still indicate that which part is vigorous then others. After adjust the intensity, the figure is shown as [Fig. 5].

**Fig. 6:** Blurred Figure (122 × 105)

### D. Models

For this task, we try one linear model, one kernel-based mode, two tree based model, and a deep learning model. Brief introduction of them is demonstrated as following:

*1) L2-regularized L2-loss Support Vector Machine Linear (SVML):* Support Vector Machine (SVM) is a supervised learning model that are widely used in solving Machine Learning problems. In practice, if we are not sure about the *difficulty* of problem, we usually use linear model first. Its objective function is listed as [Fig. 7].

*2) Support Vector Machine with Radial Kernel (SVMR):* The main concept of SVM is to use a transform matrix $\phi$ [Fig. 8] to transform each instance from the original space to a, in general,

$$\min_{w} \quad \frac{1}{2}w^T w + C\sum_{i=1}^{l}(\max(0, 1 - y_i w^T x_i))^2.$$

**Fig. 7:** L2-regularized L2-loss Support Vector Machine Objective Function

higher dimensional space. For Radial Kernel, it will even map each instance into an infinity space with $\phi = e^{-\gamma * |u-v|^2}$.

After transformation, the model will try solving an optimization problem in either primal form [Fig, 9] or dual form [Fig. 10] to separate instances with a hyperplane.

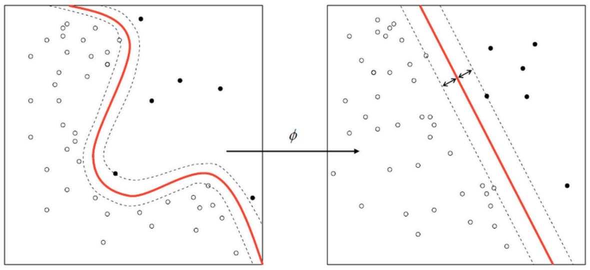

**Fig. 8:** Transform Matrxi $\phi$ (www.npmjs.com)

$$\begin{aligned} &min && \frac{1}{2}w^T w + C\sum_{i=1}^{m}\xi_i \\ &s.t. && y_i(x_i^T w + b) \geq 1 - \xi_i \qquad \xi_i \geq 0 \end{aligned}$$

**Fig. 9:** Primal form for soft margin SVM

$$\begin{aligned} &\max_{\alpha \geq 0} \mathcal{L}(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y^{(i)} y^{(j)} {x^{(i)}}^T x^{(j)} \\ &\text{s.t.} \quad \sum_i \alpha_i y^{(i)} = 0 \\ &0 \leq \alpha_i \leq c \quad \forall i \end{aligned}$$

**Fig. 10:** Dual form for soft margin SVM

*3) Gradient Boosting Model (GBM):* GBM is tree-based model whose main idea is to solve an optimization problem for new weak learners that minimizes the aggregated prediction. It is a powerful model in fitting data. In experiment we focus on two parameters: number of trees and shrinkage rate which both are related to covergence speed in GBM. With the usage of validation set, we can prevent GBM from overfitting. Its optimization formula is listed as [Fig. 11].

*4) Random Forest (RF):* RF is another well-known tree-based model for supervised learning probelm. It is called a forest because RF is an aggregation model, which consists of many individual, single decision trees, as shown in [Fig. 12] Instances will be sampled by random to build each decision tree to increase the diversity. This strategy is called Bootstrapping. Since RF uses the average prediction results from decision trees, it reduces the possibility of overfitting training data, and gives stable performance in most of real world problems.

$$\begin{aligned} F_m(x) &= F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla_f L(y_i, F_{m-1}(x_i)), \\ \gamma_m &= \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) - \gamma \frac{\partial L(y_i, F_{m-1}(x_i))}{\partial f(x_i)}\right) \end{aligned}$$

**Fig. 11:** GBM Optimization Formula

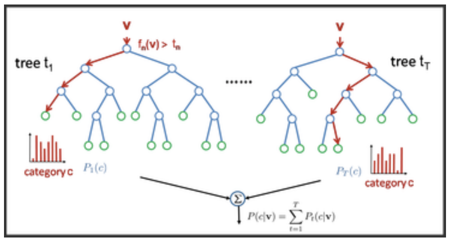

**Fig. 12:** Random Forest aggregates Decision Trees (http://www.iis.ee.ic.ac.uk/icvl/iccv09_tutorial.html)

*5) PCNet + SVML:* PCANet, invented by, is an unsupervised learning model inspired from Convolutional Neural Network(CNN). CNN is an well known model which is a power tool for image and video recognition, but hard to tune its parameters. PCANet provided an easy to train, efficiently model and meanwhile acquire good enough performance.

PCANet comprises three basic data processing components: cascaded principal component analysis (PCA) , binary hash, and block-wise histograms. PCA is employed to learn multi-stage filter banks. It is followed by simple binary hashing and block histograms for indexing and pooling.The output can be regarded as features trained from data.

After extracting feature by PCANet, same as PCANets inventor, we then use Liblinear to train and predict.

## *E. Ensemble*

In general, model ensemble will improve the performance. It can be explained as follows: if multiple models are weak, they can share their overlapped knowledge to get a better result; if one of the models outperforms the rest, model ensemble can be used to prevent overfitting.

In practice, there are usually two ways to do ensemble. One is grid search on weight, e.g. from 0 to 10, another one is using linear model to learn the weigting.

Nevertheless, in this multi-label task, it is hard to preserve model prediction probability on all labels. So, we just perform an lazy-ensemble instead, that is, choose the majority of labels predicted from models as our final prediction. If all labels given from the models are different, we just use the label from the model with best performance.

### F. Cost-sensitive One-sided Support Vector Regression (CSOSR)

The approach and implementation are proposed by *Han-Hsing Tu*, *Hsuan-Tien Lin* and it can be downloaded from http://www.csie.ntu.edu.tw/~htlin/program/cssvm/.

CSOSR is a reduction-based approach for cost-sensitive classification which reduce the origin problem to one-sided regression. The main idea is that if we can estimate cost for any instance and label pair, we can use the estimation to choose the best label prediction with least estimated cost.

$r(x, k)$ estimate cost for instance $x$ with predicting label $k$
classifier $gr(x) \equiv \text{argmin}_{1 \le k \le K}\, r(x, k)$.

### G. One versus One

We try to use One versus One linear model instead of One versus All (which Liblinear used) after PCANet and hopefully we can easily modify this to a Cost-sensitive model by inserting pseudo instances. However the performance does not improve.

## IV. Experiment Design

We care about two variants, one is the selection of model, and another one is the size of transformed figures.

First, we grid search different parameters for each model. Second, we use that certain parameter set to build models for five figures, they are: origin figure, $122 \times 105$ (scale = 1.00), $60 \times 50$ (scale = 0.50), $40 \times 35$ (scale = 0.33), $30 \times 25$ (scale = 0.25).

## V. Experiment Result

### A. Models

*1) SVML:* Parameter settings are: cost $= 0.001$ and $\epsilon = 0.01$. Its performance is shown in [Fig. 13].

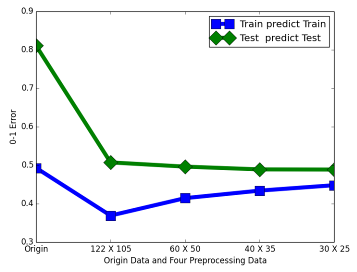

**Fig. 13:** SVMC Performance with Origin data and Preprocessing Data

*2) SVMR:* Parameter settings are: $\gamma = 0.01$, cost $= 20$ and $\epsilon = 0.01$. Its performance is shown in [Fig. 14].

*3) GBM:* Parameter settings are: 100 trees with shrinkage rate 0.1. Because of the high computational cost, its experiment results are not available currently.

*4) RF:* Parameter settings are: 200 trees, no maximum depth, minimum 2 node split, minimum 1 leaf, and no maximum leafs. Its preformance is shown in [Fig. 15].

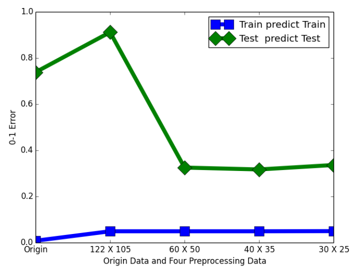

**Fig. 14:** SVMR Performance with Origin data and Preprocessing Data

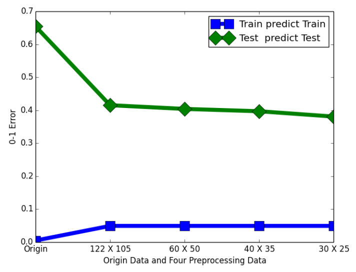

**Fig. 15:** RF Performance with Origin data and Preprocessing Data

*5) PCANet + SVML:* Parameter settings for PCANet are: number of PCA stage$=2$, patch size $= 7 \times 7$, number of filters in first stage $= 20$, number of filters in second stage $= 8$, histogram block size $= 7 \times 7$, histogram overlap ratio $= 0.5$, pyramid layer $= 421$.

Parameter settings for SVML are: cost $= 10$ and $\epsilon = 0.01$. Its performance is shown in [Fig. 16]

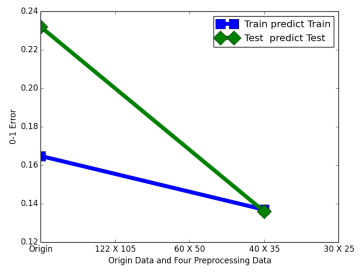

**Fig. 16:** PCANet Performance with Origin data and Preprocessing Data

### B. Ensemble

We first put an eye on the performance of our four models. Their performances on test data are shown in [Fig. 17]

Then, we try to perform a lazy ensemble describe in previous chapter. However, we find out that the performance drop from 0.136 to 0.271296798698. It means that most of the models give wrong prediction and affect the judgement of PCANet.

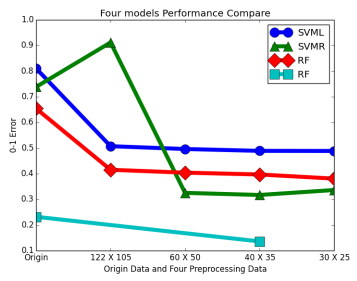

**Fig. 17:** Model Performances on Test with Different Preprocessing Data

## VI. Discussion

Though we have tried linear, kernel based, and tree based models with different preprocessing picture size, their performance still have a large gap (more than $0.1$ in terms of error rate) with deep learning based model. However, in practice, deep learning does not have such outstanding performance. The reason that cause such big differences is because of this topic – handwriting recognition – easily for human intuition. Furthermore, the process of deep learning is just explanable for handwriting recognition.

### A. *Best Model for Track 0*

We find the performance of PCANet is better than other models even with origin data and default parameters. After grid search for best parameters we get an error rate for about 0.136 on cross validation and 0.136 on public score.

### B. *Best Model for Track 1*

We try to utilize Cost-sensitive model to improve performance in Track 1. However, we does not obtain a better score.

## Acknowledgment

Thanks to the cooperation between teamates. Though our final performance does not reach the top 10, we do learn a lot in this final project.

## References

[1] Smola, Alex J., and Bernhard Scholkopf, *A tutorial on support vector regression*, 3rd ed. Statistics and computing 14.3 (2004): 199-222

[2] Cristianini, Nello, and John Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*, 3rd ed. Cambridge university press, 2000.

[3] Friedman, Jerome H, *Greedy function approximation: a gradient boosting machine*, 3rd ed. Annals of Statistics (2001): 1189-1232

[4] Liaw, Andy, and Matthew Wiener, *Classification and Regression by randomForest*, 3rd ed. R news 2.3 (2002): 18-22

[5] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma, *PCANet: A Simple Deep Learning Baseline for Image Classification?*, 3rd ed. arxiv (2014) 1404.3606

[6] Han-Hsing Tu, and Hsuan-Tien Lin, *One-sided Support Vector Regression for Multiclass Cost-sensitive Classification*, 3rd ed. Proceedings of the 27 th International Conference on Machine Learning, Haifa, Israel, 2010