CS-233(b) Introduction to Machine Learning (BA4) MS1 Report

Kanghyeon Kim, Eunae Lee, Sangwon Lee

1. Introduction

The recognition of handwritten mathematical symbols has numerous applications in education, scientific research, and human-computer interaction, being used to improve learning experiences, promote collaboration, and advance artificial intelligence. This project aims to implement and optimize the machine learning models for the classification of the hand-drawn mathematical symbols dataset, HASYv2 with K-Means, Logistic Regression, and Support Vector Machine.

2. Method

2.1 Dataset

The HASYv2 dataset contains 168,233 images (32x32 pixels) of LaTeX symbols drawn by hand and 369 unique symbol classes, including numbers, Latin and Greek letters, and various mathematical operators. Only a sub-sample of the 10 most frequent classes (**Fig.1**) (*\int: 350 sampels, \sum: 339, \infty: 291, \alpha: 259, \xi: 256, \equiv: 251, \partial: 240, \mathds{R}: 237, \in: 229, \square: 217*) were used here. The dataset was split into 2138 train images and 531 test images.

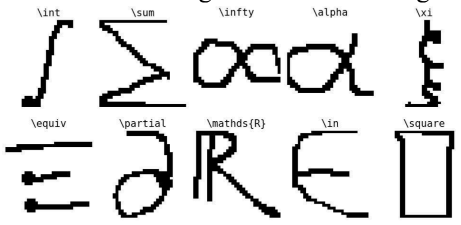

Figure 1. *The sub-sample of the HASYv2 dataset*

2.2 Preprocessing

Before applying the classification algorithms, the images were preprocessed to improve their quality and reduce the computational complexity through normalization.

2.3 Algorithms

We implemented K-means, Logistic Regression, and Support Vector Machine models with Python. For K-means, we made K (given number) clusters of data points based on their distance in the feature space and calculate the centers of these clusters. This process was repeated until the centers converged. The label which has the largest number of samples in the cluster was assigned to the cluster. For logistic regression, we fit a model that estimates a set of parameters that relate the predictors to the probability of each category using maximum likelihood estimation. In order to fit a model, the independent variables are assumed to have a linear relationship with the log-odds of the dependent variable. For support vector machine, we used the scikit-learn implementation, sklearn.svm.SVC.

2.3 Validation

For validation, the train images were randomly split into 80% for training and 20% for validation by the implemented function, *train_split()* in main.py. We used cross-validation with splitting into 5 groups (*k_fold_split()* function in main.py) to evaluate the performance of a model with different values of the hyperparameters and grid-search to find the optimized hyperparameters. We used the accuracy and F1 score for scoring metrices.

3. Validation Results

3.1 K-Means

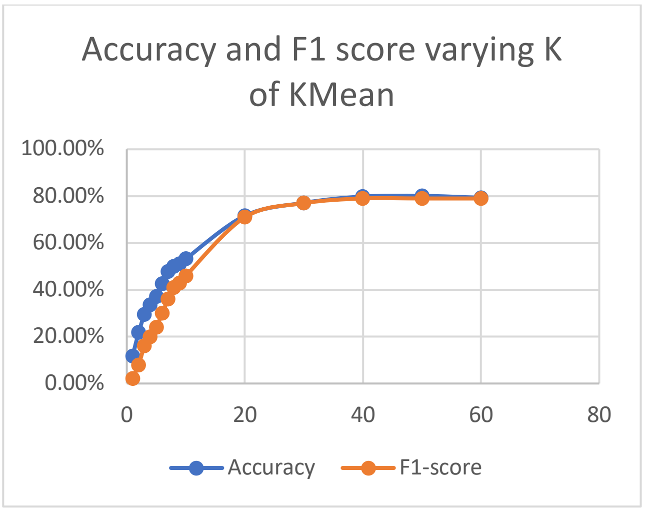

Figure 2. *Accuracy and F1 Score varying K (number of clusters) in K-Means method. The value of x-axis is K.*

3.2 Logistic Regression.

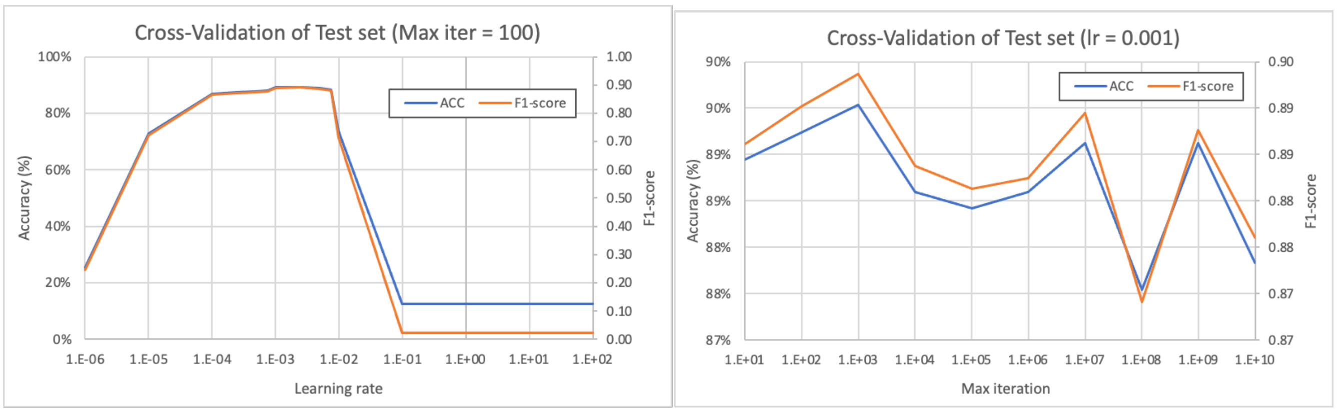

***Figure 3**. Cross-Validation of Test set varying the learning rate(lr) (A: left) and maximum number of iteration(Max-iter) (B: right) in the logistic regression method*

3.3 Support Vector Machine

The support vector machines with the linear model and models with rbf (radial basis function) kernel and polynomial kernel are compared under the default setting of parameters (C=1, gamma=1, degree=1, coef0=0).

	Linear	Rbf(Radial basis function) kernel	Polynomial kernel
Averaged Accuracy	92.63%	11.58	89.5
Averaged F1-score	0.92	0.02	0.90

***Table 1**. The average of accuracy and F1-score from the cross validation of linear, rbf(radial basis function) and polynomial SVM models*

C	1e-5	1e-4	1e-3	1e-2	1e-1	1	1e3
Accuracy	15.439%	87.602%	93.450%	92.632%	92.632%	92.632%	92.632%
F1-score	0.054882	0.874305	0.932950	0.924343	0.924343	0.924343	0.924343

***Table 2.** The average of accuracy and F1-score from the cross validation of linear SVM models varying the value of C, regularization parameter (penalty) of linear SVM model.*

4. Discussion and Test Results

Figure 2 shows the tendency of the accuracy and F1 score varying K, the number of clusters in K-Means method. The optimal K was initially expected to 10, the number of classes in the dataset. However, the accuracy and F1 score increased as K increased to 50. It means there are some samples which have significant differences in the feature matrix even though they are classified to same symbol in the dataset. Therefore, K was set to 50 for comparing with other methods.

Figure 3A shows the learning results of the model measured by fixing the maximum iterations to 100 and changing the learning rate. Until the learning rate was 0.1, the model was not trained at all with an accuracy of about 12%, but the performance of the model increased rapidly at lr=0.01. The learning rate increased until 1E-3, but there was no noticeable significant increase. It showed that the performance of the model also decreased as the learning rate decreased below 1E-3. Therefore, we judged that it is appropriate to set the learning rate to a value between 0.01 and 0.001 (1E-2 to 1E-3). **Figure 3B** is the performance result of the model measured by fixing the learning rate to a specific value (=0.001) and changing the max-iteration value. As the max-iter value increases to 1000, the performance of the model also increases. Over 1000, the results show that the performance of the model is similar or degraded rather than improved. As the Iteration increases, the learning time increases, so it is reasonable to choose as small as possible unless the performance increases significantly. Therefore, we judged that it was reasonable to set the max-iter to 1000.

Table 1 shows the accuracy and F1 score of the linear, rbf and polynomial SVM models with default values of parameters. The linear model showed the best performance and rbf kernel shows the low accuracy. It means rbf model was over-fit to the training set and the linear SVM is the appropriate SVM for this dataset. However, we cannot do the grid-search for optimization of hyperparameters (such as gamma, degree, etc.) of SVM models (rbf, polynomial) because of resources. Because the superiority and inferiority of the models may vary depending on the parameter value. Here, we chose the linear SVM for the best model among the support vector machines. (**Table 2**)We checked the change of averaged accuracy and F1 score from cross-validation varying the value of C, the regularization parameter (penalty) of linear SVM model. It shows the maximum value at C=0.01 and there is no change of the highest scores after this.

When we compare the test results and time of the K-Means(K=50), Logistic Regression (lr=0.001, max_iter=1000), linear SVM (C=0.01), we got the following results **Table 3**. We can check that the linear SVM outperforms both K-means and Logistic Regression in classifying hand-drawn mathematical symbols from the HASYv2 dataset. However, when we consider the runtime, the Logistic Regression model can be the attractive candidate for the large dataset.

	K-Means(K=50)	Logistic Regression (lr=0.001, max_iter=1000)	Linear SVM (C=0.01)
Accuracy	84.369%	90.584%	95.104%
F1-score	0.842207	0.904015	0.950515
Time(s)	0.71	0.76	1.63

Table 3. Accuracy, F1-score and Time for the optimized models; K-Means (K=50), Logistic Regression (lr=0.001, max_iter=1000), Linear SVM (C=0.01)