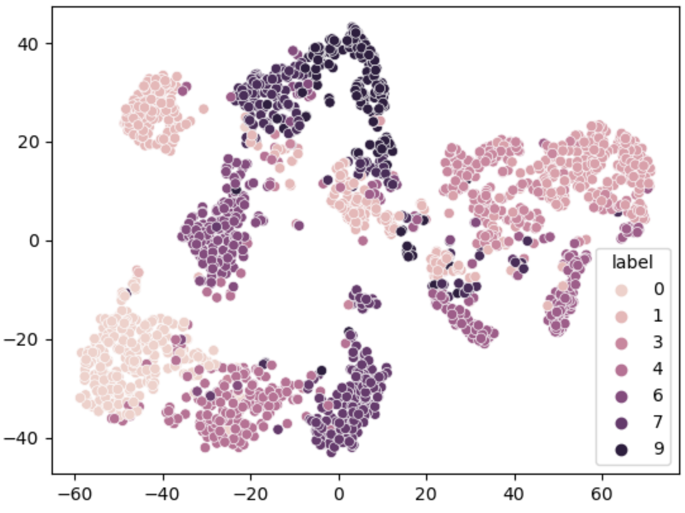
Handwritten Mathematical Symbol Recognition using Machine Learning

Mak Fazlic, Alexandre Marcel Mourot, Noé Macé
EPFL CS233

Abstract
In this paper, we present our approach to classifying handwritten mathematical symbols using various machine learning techniques. Our work is based on a course project for the CS-233(b) Introduction to Machine Learning at the EPFL. We implemented and evaluated various machine learning methods on a subset of the HASYv2 dataset, which consists of small 32x32 images of hand-drawn mathematical symbols. The three methods used in our study are K-Means, Logistic Regression, and Support Vector Machines. We evaluated the performance of these models using accuracy and F1-score as evaluation metrics.

1 Introduction

The data that was provided to us consisted of two parts. Primarily we got the images which were represented with a 32x32 pixel structure and in .png format. These images were also supported by a .csv file with labels from 0 to 9 representing 10 different classes these images fall under. Our job was to train a machine learning system where we can accurately classify the images based on the predefined labels. We represented images as a 1024 element vector where the values are pixel intensities from 0 to 1. Since our data is in 1024 dimensions, it is difficult to visualize, but using t-distributed stochastic neighbor embedding we were able to visualize such high dimensional data in 2d, as seen from the image hereafter. Such mapping allows us to see correlation between several types of characters. Before starting with the training of our three models (K-Means, Logistic Regression and Support Vector Machine), we split the data into a train and test segments 80% and 20% respectively. Afterwards, we split the training data to train and validation segments 80% and 20% respectively, when we needed a validation set to find the best hyperparameters.



2 Method

2.1 K-Means

The K-mean model is a clustering classification with hyper-parameter K the number of clusters. We uses a validation test method to find the best K to our model. We therefore have two main phases for the K-mean model. First we find the best K with our validation set, we test K from 1 to 60 to cover a lot of K possibility and take the best one for our final training. Then we train our model over the data sample, predict the test data and compare it with the test labels to get our accuracy. To find the best K we could have used the elbow method, however we choose to take advantage of the provided labeled dataset to use a validation set.

For the training of the model, as the starting clusters are chosen randomly, we ran the model several times and choose the best accuracy we get over 10 repetitions. With these several trials we can obtain a more accurate model in case of a bad first random choice of clusters.

2.2 Logistic Regression

Logistic regression is a supervised learning algorithm that predicts the probability of the output class given input features. The code implements the softmax function, which computes the probability of each class for a given input sample. The main components of this implementation include functions for computing the loss and gradient, as well as for training and predicting using the model. It uses gradient descent to update the model parameters and early stopping to improve training efficiency.

2.3 Support Vector Machine

The SVM was implemented using a SVC module provided, and to find the best hyperparameters we used a validation set and several iterations while changing hyperparameters. While doing so, we monitored improvements and recorded the best performing hyperparamenters. When it comes to different types of kernels, we tried three: Linear, Polynomial and RBF. For other parameters we used several numpy range functions to generate multiple parameters.

3 Results

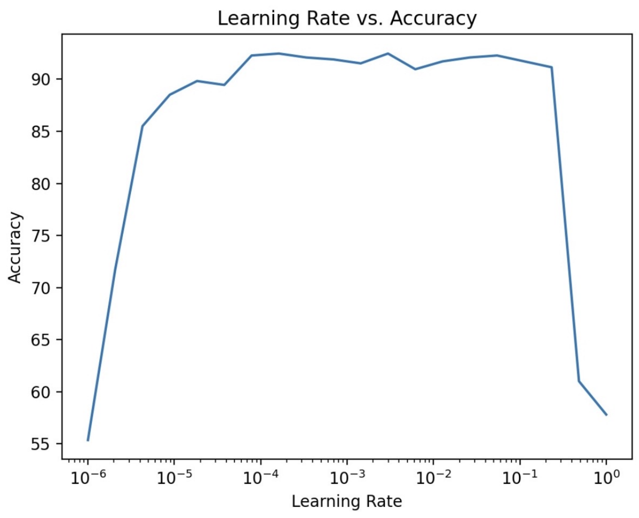
3.1 K-Means

After finding the best K among 60 we finally get 58 thanks to a validation set approach. Then training our model with K=58 we obtain an accuracy of 85 percent which is not bad but less than other methods. The limits of this model is that K is overfitting for large values of K (superior to 2000) after testing out K=1000 and K=2000 we observe a cap to 89 percent accuracy. With our computational capacity we were not able to find this perfect value of K. Then the optimal accuracy attainable is around 89 percent for our K-means model.

3.2 Logistic Regression

We encountered multiple errors due to numerical instability and had to modify the softmax function to subtracts the maximum score from each score before exponentiating them. This fix doesn't change the values of the scores following this proof : Using this axiom $a^{(b-c)} = \frac{a^b}{a^c}$: $\frac{e^{(x-\max(x))}}{\sum(e^{(x-\max(x))})} = \frac{e^x}{(e^{\max(x)}.\sum(e^x/e^{\max(x)}))} = \frac{e^x}{\sum(e^x)}$

We use hyper-parameter tuning techniques, specifically grid search, to evaluate the performance of various learning rates systematically. The most performing learning rate on average leads us to choosing : lr =0.00016.



3.3 Support Vector Machine

After running validate_linear, validate_poly and validate_rbf which are a part of our SVM module, we get the report of the best results on the validation data and the best hyperparameters to use. Taking these hyperparameters, training the SVM and testing against the test data yields these results with their respective input hyperparameters:

Kernel	C	Gamma	Degree	Coef0
Linear	0.01	0.005	-	-
RBF	1	0.01	-	-
Poly	0.01	0.005	4	0.215

Kernel	Accuracy	F1
Linear	94.73%	0.945437
RBF	94.73%	0.944764
Poly	96.23%	0.960709

4 Conclusion

Given the results presented in the report, we can claim that the SVM is the most accurate model to predict handwritten characters. It proved to be very robust and consistently gave more accurate results then the other two models. SVM which proved the best was the polynomial of degree 4. The reason why we could not achieve perfect (100% test accuracy) results, is most likely due to the problem not being completely linearly separable as indicated in the introduction with many points overlapping when mapped in 2d.