Handwritten Character Classification Using SVM

Jaber Aloufi, Arunabho Basu, Cheng-Han Ho and Arth Patel

Abstract—**This report describes the implementation of handwritten character classifier using Support Vector Machine and other techniques we experimented with to find best classifier. The training data is scanned images of handwritten characters (a, b, c, d, h, i, j, k and unknown) with their corresponding labels [1, 2, 3, 4, 5, 6, 7, 8 and -1]. The original dataset was provided with labels, so we are using supervised learning techniques to train the model. By extracting the features of each character and train the model with different parameters for different classifier, support vector machine has the highest accuracy score overall among the ones tested. The highest accuracy score is around 93 percent on SVM. The report describes the major steps that has been taken to implement SVM and also other algorithms we used to find best possible classifier.**

I. Introduction

We implemented handwritten characters classifier in four steps. The first step is collecting the dataset (the scanned images of handwritten letters). The second step includes gray scaling, centering the original images then converting them into binary images as a part of reprocessing our raw data. Then we extract the features and use sklearn SVM and KNN algorithms to train the model. The last step is testing the trained model using a small part of the dataset that has been split before the training. The algorithms that have been used in this project are SVM, and KNN. K-nearest-Neighbor is a discriminative supervised machine learning algorithm that uses labeled input data to learn a function that used to produce the correct output for the unlabeled testing data. The assumption in KNN is that all the data points that belong to the same class are near to each other. KNN works as follows: first, we load the data and initialize the K value (number of neighbors). Then for any testing data, we compute the distance between every data point we have and the testing point. We can use among others - Euclidean, Manhattan or Minkowski function to calculate the distance. After that we choose the closest K data points, get their labels and assign that label to the testing point. SVM is also a discriminative supervised machine learning algorithm. We implemented SVM as recommended in A Practical Guide to Support Vector Classification [1], which is :

1. Data is transformed to the format of an SVM package
2. Simple scaling is conducted on the data
3. Considered the RBF kernel $K(x, y) = e^{-\gamma||x-y||^2}$
4. Used cross-validation to find the best parameter C and γ
5. Used the best parameter C and γ to train the whole training set.
6. Performed the test [1].

In general given a training set of instance-label pairs (x_i, y_i) the support vector machines tries to solve the following optimization problem

$$\min_{w,b,\omega} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \omega_i$$

where

$$y_i(w^T \varphi(x_i) + b) \geq 1 - \omega_i$$

and $\omega_i \geq 0$

Here training vectors x_i are mapped into a higher dimensional space by the function φ [1]. The support-vector machine constructs a hyperplane or set of hyperplanes in a high dimensional space or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection[2]. The best separation between any two classes is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class, since in general the larger the margin, the generalization error of the classifier will be lower. The SVM algorithm is implemented using a kernel. The kernel transforms an input data space into the required form. Here, kernel takes a low-dimensional input space and transforms it into a higher dimensional space. In other words, we can say that it converts non separable problem to separable problems by adding more dimension to it. It is very useful in non-linear separation problem. Kernel helps us to build a more accurate classifier. There are many transformation methods, such as, linear, polynomial, radial basis function (RBF) and sigmoid. The different kernels have different uses and are used in different cases of classifying data. The linear kernel employs dot product between any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values. The polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish between linear and nonlinear input space. The radial basis function kernel is the default kernel function in sklearn.svm. RBF can map an input space in infinite dimensional space [3]. After transformation using kernel is achieved, we are able to draw the boundary decision line between the classes we have. Other than kernel, there are other parameters which we can tune to achieve better classification results. The regularization parameter C is the penalty parameter, which represents misclassification or error term. The error or misclassification term tells the SVM optimization that how much error is bearable. This is how we can control the trade-off between decision boundary and error/ misclassification. A smaller value of C creates a small-margin hyperplane and a larger value of C creates a larger-margin hyperplane. Another important parameter is Gamma. Value of Gamma is between 0 and 1. A higher value of gamma will exactly fit the training dataset which can cause overfitting whereas lower value of Gamma will loosely fit the training dataset and reduce model reliability [3]. We have to tune gamma accordingly to optimize classification accuracy. Here, we are using Support Vector Classification (SVC) of SVM algorithm which is used for multiclass classification.

II. Implementation

We have been provided with nd array of binary images with their corresponding labels. Those images needed to be processed into a useable dataset format. First, we checked the dataset and found that there were many images which were rotated, corrupted, had empty arrays or had wrong labels; so we removed all those images. All the image data of the training set were not of the same dimension. To deal with different dimension images, we removed the images that were larger than the dimension of 50 x 50 pixels. Then we found the case that some images had lower than 50 pixels on either rows or columns. If a row had less than 50 pixels, the image was centered and resized by padding each size with 0. Then stored those images into nd array with each image as 1d array because the sklearn.svm uses 1d array as input. The resulting nd array each image is 1d array of 2500 elements.

After pre-processing original training data into workable dataset we split the data into training and testing to cross validate our model. We split data into 80% for training dataset and 20% for testing.

We experimented with different supervised learning classifier to train our model. We mainly used SVM and KNN to train the model. We implemented KNN and SVM from scikit package's sklearn.neighbors.KNeighborsClassifier and sklearn.svm.SVC respectively. Our input was an array of one dimension with 2500 elements. So, our model had 2500 input feature vectors. There are different parameters for KNN like number of neighbors, weight function, algorithm, Leaf size for BallTree and KDTree algorithm etc. Likewise, SVM has parameters like penalty parameter, degree of polynomial kernel, kernel coefficient gamma, tolerance, probability etc. After fitting training data with our model, we cross validated our model and recorded accuracy of each model with their corresponding parameters and then found key parameters which affect our model the most. By tuning those parameters, we were able to optimize accuracy on test dataset. The final model was stored with trained parameters using pickle.

We created test function which uses the already trained model to predict labels of the input test dataset. As we also stored probability values for each class of each image, we were able to compute probabilities of the prediction of test data as well. So, if prediction probabilities of each class were less than 0.25 then we assumed that the result has low confidence, so we assigned it a label -1, i.e., unknown character. We also created test dataset for testing our model on characters 'a' and 'b' each with 50 samples. For 'a' the trained model gave 95% accuracy and for 'b' the accuracy was 100%.

III. Experiments

SVM function from sklearn provides different parameters for tuning the model to obtain better accuracy. After testing different kernels, we found that RBF provided the best result.

The different kernels use different functions to transform two-dimensional space into higher dimension.

1. Linear: $K(x_i, x_j) = x_i^T x_j$
2. Polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$
3. Radial Basis Function (RBF): $K(x, y) = e^{-\gamma||x-y||^2}$
4. Sigmoid: $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)^1$

Kernel	Accuracy
RBF	0.933
linear	0.805
polynomial	0.113
linearSVC	0.774

Table 1. Accuracy based on different kernels of SVM

Changing the Gamma parameter is crucial since it defines how far the influence of a single training example reaches, for low gamma, points far away from plausible separation line are considered in the calculation for the separation line [10]. Whereas high gamma means only the points close to plausible line are considered in the calculation.

Using RBF kernel, we found different accuracies for different gamma value. We noticed that we got best result for gamma=0.005.

Gamma	Accuracy
0.001	0.815
0.005	0.933
0.01	0.861
0.05	0.265
0.1	0.183
0.5	0.161

Table 2. Accuracy based on different gamma

Comparing the result with SVM and KNN we found that the KNN with different k values - all provide less accuracy score compared to SVM using RBF kernel. The best results we get is for k = 2 which is 79% accuracy on test set. As we increase value of k the accuracy of the model decreases.

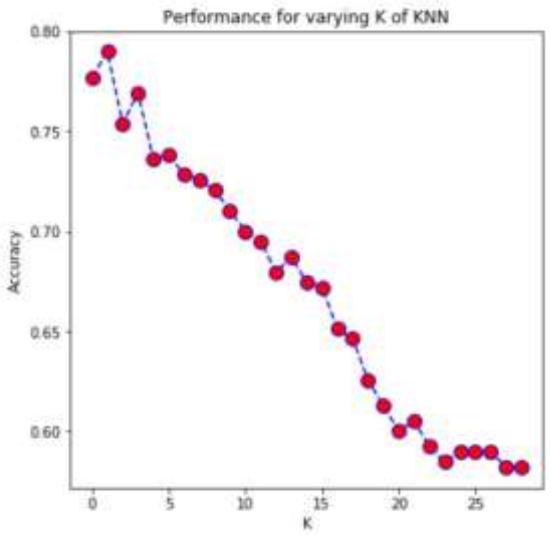

Figure 1. Accuracy based on different k values

After training and cross validating our model using SVM we tested our trained model on blind dataset for which we were getting around 97% accuracy. As our blind dataset was comparatively easier set, we achieved better accuracy.

What the model is successful in achieving is that it can easily classify input characters which are easy for us to recognize at one glance. If the test data set has very noisy images, very tiny characters or confusing handwriting then it will be difficult for the algorithm to classify those images correctly. For recognizing hard test data, we needed more training data and also if we have more data then we can experiment with other machine learning algorithms like CNN.

IV. Conclusions

Based on our experiments with the training data we found that the SVM model normally does better on given characters recognition task among others classifier. Also, we found that by pre-processing the input dataset we can significantly improve our model performance and also we can save time to train the model. As our project aims to identify characters, SVM did provide the best result in accuracy. Here due to limitation in training data SVM turned out to be the best classifier but if we have more data then we can design more complex model which can be trained to classify hard dataset.

References

[1] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, "A Practical Guide to Support Vector Classification," National Taiwan University, Taipei 106, Taiwan http://www.csie.ntu.edu.tw/~cjlin

[2] https://scikit-learn.org/stable/modules/svm.html

[3] A. Navlani, "Support Vector Machine with Scikit-Learn," Datacamp, 12-July 2018.[Online].Available:https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python

[4] https://scikitlearn.org/stable/auto_examples/svm/plot_rbf_parameters.html

[5] https://en.wikipedia.org/wiki/Radial_basis_function_kernel

[6] N. Shanthi and K. Duraiswamy, "A novel SVM-based handwritten Tamil character recognition system," Pattern Analysis and Applications, vol. 13, no. 2, pp. 173–180, 2009.

[7] M. B. Fraj, "In Depth: Parameter tuning for SVC," Medium, 05-Jan-2018. [Online]. Available: https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769. [Accessed: 30-Nov-2019].

[8] R. Jain, Startup, and Startup, "Simple Tutorial on SVM and Parameter Tuning in Python and R," HackerEarth Blog, 20-May-2019. [Online]. Available: https://www.hackerearth.com/blog/developers/simple-tutorial-svm-parameter-tuning-python-r/. [Accessed: 30-Nov-2019].

[9] O. Harrison, "Machine Learning Basics with the K-Nearest Neighbors Algorithm," Medium, 14-Jul-2019. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. [Accessed: 30-Nov-2019].

[10] Patel, Savan. "Chapter 2 : SVM (Support Vector Machine) - Theory." Medium, Machine Learning 101, 4 May 2017, https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72.